University of Alberta

Automated Planning and Scheduling for Industrial Construction Processes

by

Di Hu

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Construction Engineering and Management

Department of Civil and Environmental Engineering

Spring 2013
Edmonton, Alberta

Examining Committee

Dr. Yasser Mohamed, Civil and Environmental Engineering

Dr. Aminah Robinson Fayek, Civil and Environmental Engineering

Dr. Simaan AbouRizk, Civil and Environmental Engineering

Dr. Amy Kim, Civil and Environmental Engineering

Dr. Amit Kumar, Mechanical Engineering

Dr. Tarek Hegazy, Civil and Environmental Engineering, University of Waterloo

Dedication

This thesis is dedicated with love and respect

to my beloved parents Zhepian Xiong and Hantong Hu;

to my dear relatives;

to my best friends

Abstract

Cost overrun and schedule slippage are common problems for mega industrial construction projects. Lack of effective planning and scheduling tools is identified as a major contributing factor to poor project performance. Planning and scheduling tools should be custom designed to address the characteristics of mage industrial projects: unique components, modularized execution strategy and its extremely accelerated project delivery. The main objective of this research is to investigate and develop automated solutions for planning and scheduling two essential stages of mega industrial projects, shop fabrication and on-site construction.

This research explores use of discrete event simulation (DES) to automate scheduling of shop fabrication and on-site construction processes. For industrial fabrication shops, a new simulation structuring methodology is developed to address the complex routing issue. Following this methodology, a simulation model is developed for pipe spool fabrication shops, which performs scheduling for shop operations, and mainly evaluates the impact of fabrication sequence on the spool cycle time. For site construction, a time-stepped simulation framework is developed to address congestion and dynamic resource allocation issue. For a real-life industrial construction case, this framework returns a schedule that has 12% shorter duration than those generated from Microsoft Project and Primavera P6.

The research investigates use of domain-independent Artificial Intelligence (AI) planning to automate the sequence planing for pipe spool fabrication and on-site module installation. Experiment results show that AI planning is not suitable for

sequencing spool fabrication due to the limited parsing capability of existing AI planners. However, AI planning is efficient to identify feasible sequence plans for module installation based on the current module availability and installation status.

The research finds Dynamic Programming (DP) is suitable to sequence pipe spool fabrication. A DP algorithm is developed to automatically identify the optimal sequence in terms of the minimum position welds. Simulation experiments were conducted with 29 real-life spools to quantify the performance improvement obtained from the DP algorithm. Results showes that by using the DP algorithm, there is a 45% reduction in the number of position welds, which is translated to a reduction in the total cycle time, ranging from 4.8% to 12%.

Aknowledgement

First and foremost, I would like to thank my PhD supervisor, Dr. Yasser Mohamed for his immense support, visionary guidance and great patience throughout the course of my study. I owe my great gratitude to Dr. Al-Hussein for his insightful comments during my defense preparation process.

I would like to acknowledge Dr. Tarek Hegazy for serving as my external examiner and providing valuable comments and suggestions. I am also grateful to my candidacy and doctoral committee: Dr. Aminah Robinson Fayek, Dr. Simaan M. AbouRizk, Dr. Amit Kumar, and Dr. Amy Kim.

I would like to take this opportunity to thank Brenda Penner, Amy Carter, Maria Al-Hussein, Stephen Hague and other faculty members and staff and colleagues in the construction engineering and management group.

I would particularly like to thank PCL Industrial Management Inc. This thesis would not have been possible without generous support and close collaboration from Ulrich (Rick) Hermann, Manager of Construction Engineering, and Hosein Taghaddos, Construction Engineer of Construction Engineering, at PCL Industrial Management Inc. Their invaluable input to this thesis is deeply appreciated.

Especially, I would like to express my deepest gratitude to my family for their unconditional love and unfailing support. My parents and grandparents have taught me how to live, to dream, and to love. Although I have been far away from you, your love and support transcend the distance and always have and will encourage me to succeed.

Table of Contents

List of Tables

List of Figures

CHAPTER 1. Introduction

1.1 BACKGROUND AND PROBLEM STATEMENT

A surging global demand and soaring prices of oil and gas put unprecedented pressure on existing oil production infrastructure around the globe. This leads to a significant wave of capital investment into oil and gas projects. Thanks to fast growing oil sands projects, the province of Alberta recently experienced an economic boom reminiscent of the first oil boom in Canada in the 1970s. According to current government figures, about $193.5-billion in major projects are under way in Alberta, with 65% ($125.8) of those being oil sands related (Alberta Enterprise and Advanced Education 2012).

Oil sands production steadily ramped up in the late 1990s and early 2000s. It is spurred by high oil prices and by strong demands from the international community, which sees the oil sands as the next big source of oil (other than conventional crude oil), and as part of the solution to energy security issues. The oil sands currently represent 59% of western Canada's total crude oil production (CAPP 2012).

In order to produce oil from oil sands, facilities that extract and upgrade the bitumen to produce petroleum products are needed. Building such facilities is considered a special type of construction, "industrial construction." Generally speaking, industrial construction refers to constructing a wide range of facilities or plants, e.g. chemical plants, pharmaceutical plants, oil/gas production plants, petrochemical refineries, and nuclear power plants. Industrial projects vary both in size and complexity. Purposes of industrial projects include construction of a completely new facility, expansion of an existing facility, as well as demolition of an existing facility.

1.1.1 Complexity of oil sands projects

Many oil sands projects executed during the oil boom are defined as 'mega projects.' These projects continue to increase in size. The first mega-project during this period was Suncor's Millennium project, with a final cost of $3.4 billion. Shell Canada Ltd.'s Muskeg River project was next to follow the trend with a total around $5.7 billion. Syncrude Canada Ltd. completed its UE-1 in 2004 with a cost ballooning to $7.8 billion (Feuffel and Hanley 2009). A recent oil sands project, Kearl Inistial Development (KID), by Imperial Oil Ltd. is close to completion with a capital appropriation of $10.9 billion.

Cost is, however, merely one of characteristics that define a mega project. Mega projects often push beyond the limit of almost every aspect of a construction project. The Alberta Economic Development Authority (AEDA, 2004) summarized a number of facts about mega projects in the province of Alberta. For example, for a mega project of $2.5 billion, over 3.5 million engineering man-hours are usually required, which generate 40,000 to 50,000 design drawings and involve 10,000 to 20,000 vendor and shop drawings. A labor force of 6,000 to 8,000 is needed to perform construction and to provide 15 million construction man-hours. Work of a variety of disciplines, e.g. civil, piling, structural steel, piping, electrical, etc., is involved in a single project. Each discipline can be viewed as a supply chain which is closely interwoven with other disciplines at certain stages. A deviation in one discipline can cause a cascade of disruptive effects on other disciplines. Therefore, intensive effort is required for project coordination.

1.1.2 Pre-fabrication, pre-assembly and modularization

Since many oil sands projects are located in frontier areas where site conditions and weather problems are severe, and where local labor availability is not sufficient and a substantial

workforce needs to be relocated from somewhere else (i.e. on-site labor cost might be very expensive), pre-fabrication, pre-assembly and modularization are common strategies to perform these projects. Another major reason for this approach is that it allows site preparation to occur while some pre-fabrication and pre-assembly is being done on components required for a plant. This is especially relevant as many oil sands projects are executed in a reduced timeframe. For simplicity, pre-fabrication and pre-assembly are referred to as fabrication and assembly in the rest of this thesis.

Fabrication refers to the pre-production of some components of an industrial project, usually in a shop environment, which are then shipped to the construction site for final installation. Fabrication often involves only a trade (Tatum 1987), such as pipe spool fabrication or structural steel fabrication. Industrial fabrication does not fall under a typical manufacturing system in which the same components are used repetitively to produce standardized items. Pipe spools or structural steel pieces are usually unique and need to be custom built. The characteristic of uniqueness differentiates industrial fabrication shops from traditional mass production manufacturing shops. This point will be elaborated in detail in the following section.

After they are fabricated in shops, some components are shipped to an assembly yard (i.e. usually close to the fabrication shop) where they are synthesized with other fabricated components into a unit called a module. A module is a part of a pipe rack or plant that includes a steel structure, pipe spools, equipment, cable tray, heat tracing, instrumentation, electrical components, fireproofing and insulation. Different types of modules include pipe racks, processes, stair-towers, and buildings. Modules are usually sized within the limitations of transportation. They are usually shipped from the assembly yard to the construction site and joined together to form a completed portion of the industrial plant.

Modularization can account for as much as 30% to 40% of the overall project scope (Fossen and Kukkola 2009). The proportion of shop fabrication and module assembly used in industrial projects has significantly increased over the past 20 years (Hass et al 2000). This might be accredited to modularization's controlled factory environment and production conditions which can lead to higher productivity and better quality. Another advantage of fabrication and assembly is that both field construction duration and cost can be significantly reduced. Figure 1-1 shows the piping supply chain that consists of fabrication, assembly and site installation stages.

1.1.3 Fast tracking

In order to reduce the investment payback time and to reduce the period of risk exposure (e.g. fluctuation in oil price, uncertainty of inflation and interest costs), owners of oil sands projects tend to prefer an accelerated project delivery process, such as fast track. Fast-tracking forces overlap between design, procurement, fabrication, module assembly and site installation in an industrial project. As such, fabrication, assembly and even installation start before the design is completed. However, information available at the beginning of any project is scarce. Much of the initial design has to be made based on the 'best guesses' of engineers. As the project progresses, this initial design is prone to change. In industrial construction, incomplete designs usually result in a large number of drawing revisions, which in turn lead to late or out-of-sequence delivery of ISO drawings. Since ISO drawings are essential guidelines for fabrication, module assembly and site installation stages, their delay can generate significant disturbance to these operations. Sometimes, design changes occur after fabrication, assembly or site installation has begun. This usually means that work packages might be cancelled and all man-hours that have been spent prior to the cancellation might be completely wasted. Fast tracking also means that procurement proceeds concurrently with evolving project design. When purchases are based on poorly defined

requirements, the risk of procuring the wrong materials is high (Wyss 2009). The supply of raw materials is as unreliable as that of ISO drawings, and thus, disrupts subsequent fabrication, assembly and installation stages in a similar way. A study (Burati et al. 1992) on quality deviation of nine fast-tracking industrial projects showed that design deviation averaged 78% of the total number of deviations, 79% of the total deviation costs and 9.5% of the total project cost. Further, it showed that design change accounted for two-thirds of the design deviations. Another survey (Wang 2006) indicated that 62% of the total shop drawings needed revisions.

Fast tracking also implies that the duration of each project stage is shortened. During the construction stage, for example, a compressed schedule usually implies overlapping of work packages that would be sequentially performed in a normal schedule. This usually causes trade interference, work disruptions and consequently, productivity losses. This is especially the case during the site construction stage when a variety of disciplines (e.g. civil, architectural, piping, steel structure, mechanical, electrical, etc.) proceed with their work concurrently, in the same construction site. Interference between work packages usually manifests in the form of resource and space conflicts.

1.1.4 Uniqueness of products

Many oil sands facilities are one-of-a-kind, uniquely designed. This uniqueness is also reflected in the facilities' constituent components, e.g. pipe spools, steel structures, and modules. A petroleum refinery can require as many as 10,000 piping inventory codes, each of which represents a unique piping component (Wyss 2009). Pipe spools can vary in material, configuration, types of joints, and many other properties. Song and AbouRizk (2003) characterized the steel fabrication process by the high product mix and low production volume.

They found that many fabricated steel pieces are unique and vary in geometry and processing requirements. The unique design of steel pieces is mainly determined by unique functions and unique loads (each structure is different). Modules, which are assembled from these unique pipe spools, steel pieces and other specialty items, are thus unique in nature and each can be treated as its own individual project.

The unique nature of these industrial components means that they need to be custom built. Both pipe spools and structural steel pieces have unique routings in the shop. A routing is the sequence of shop operations necessary to fabricate a spool or a steel piece. This is very different from mass production manufacturing shops where identical or standard projects are produced and only a few typical routings are followed. Therefore, production planning is constantly required for each pipe spool or steel piece.

Unique industrial components are not interchangeable. Delay in delivery of such a unique component might hold back the assembly of a module or the installation of a processing unit on site. For industrial fabrication shops, those unique characteristics mean that these industrial components cannot be entirely or partially fabricated in advance, as standard products are in manufacturing shops. It also means that the fabricators are unable to use on-hand inventory to buffer against variability from within or outside the shop. This requires a high level of coordination between fabrication, assembly and site installation so that just-in-time supply of ISO drawings, materials and fabricated components can facilitate a smooth site construction process.

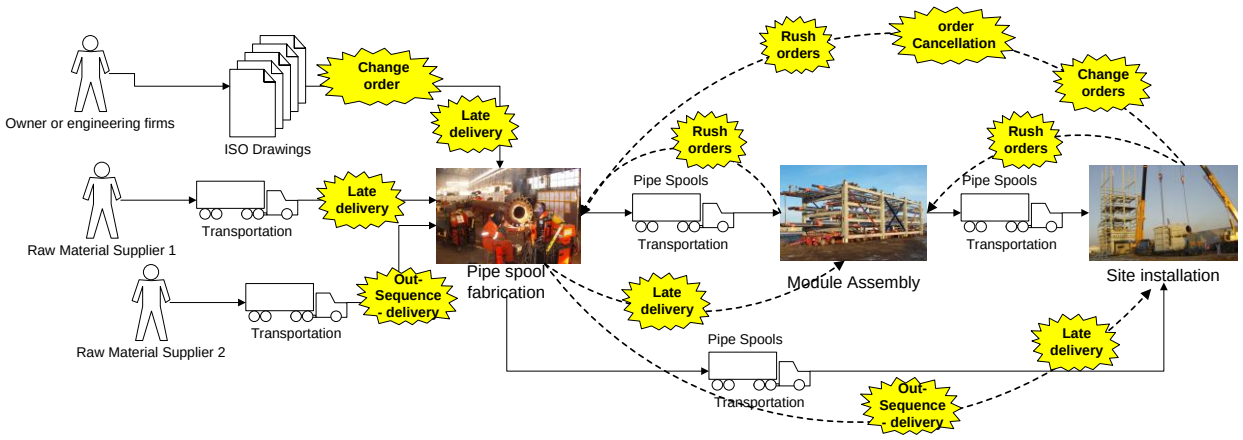


Figure 1-1 Piping supply chain in industrial projects

1.1.5 Supply chain disruptions

In a well-coordinated supply chain, on-site installation should drive module assembly, which in turn drives shop fabrication. For example, if there are two construction work areas (CWAs) (a construction work area is a spatial division of an industrial project that contains mechanical equipment, structural steel, piping and electrical scopes of work) and CWA 1 is scheduled to be prepared (completion of underground structural foundation) before CWA 2, then all the modules and components that compose CWA 1 should accordingly be produced and delivered prior to those of CWA 2. In an ideal situation, if the schedule of on-site installation remains consistent throughout the entire project, scheduling operations for the module assembly or for the fabrication shop should straight-forward. However, this is often not the case in reality. The schedule of on-site installation is prone to change, due to many factors such as scope/design changes, site conditions or constructability issues. Since this variability originates at the end of the supply chain, its influence can ripple back to all the preceding stages, e.g. module assembly and spool fabrication (Wang 2006). The variability manifests in the form of rush orders, change orders or order cancellations (Figure 1-1).

For fabrication shops, rush orders, change orders and order cancellations result in occurrence of rework, stoppage in fabrication and change in spool fabrication sequence, which disrupt shop operations and hamper shop productivity. Effects of these disruptions can be amplified when there is an unreliable supply of ISO drawings and raw materials at the same time. In addition, fabricators also suffer from the variability that originates within their own shops, such as machine breakdown, staff absenteeism, and rework. Due to the uniqueness of these industrial components, fabrication shops cannot entirely or partially fabricate these components in advance or use on-hand inventory to buffer against these disruptions. Many research studies (Howell and Ballard 1996, Tommelein 1998, Song and AbouRizk 2006, and Wang et al. 2009) identified that spool fabrication shops are often not operating at optimal productivity.

The major consequence of disrupted shop fabrication is the late or out-of-sequence delivery of fabricated industry components that, in turn, disrupts both the module assembly and the site installation stages (Figure 1-1). As many supply chains (e.g. piping, structural steel, equipment, instruments and other pre-ordered specialty items) merge just prior to final installation and the precondition for a successful installation is that all matching components be available, delay in any of these supply chains could hinder the site installation progress. Assume that there are 5 supply chains providing fabricated or assembled components necessary for the site installation and that each has 10% probability of having delayed or out-of-sequence delivery, then the probability of having timely successful installation is about 59% (= 0.95). Furthermore, site installation faces the same problem of unreliable supply of ISO drawings and raw materials and design revisions as shop fabrication does. As a result, schedule slippage and cost overrun are very common in today's industrial projects. A study commissioned by the Construction Owner's Association of Alberta (COAA) in 2004 focused on labor productivity of heavy industrial

projects. It is found that tool time, the hours actually spent working on the asset, is as low as 37% and much of the remaining time is spent waiting for material and equipment (COAA website).

1.1.6 Challenges for planning and scheduling

Under the combined effect of the aforementioned influencing factors (fast tracking, the unique design and consequently high level of uncertainty), performance of many industrial projects (including all fabrication, assembly and site installation phases) is often not satisfying. Cost overruns, schedule delays and loss of productivity are common characteristics of these large-scale industrial projects. Jergeas (2008) and Ruwanpura et al. (2006) reviewed the major oil and gas projects in Alberta, Canada and discovered that the costs of industrial projects could balloon to 200% of the original cost estimates. The Association of Professional Engineers, Geologists and Geophysicists of Alberta (APEGGA) also found that cost overruns ranging from 50% to 100% are present in almost all major oil and gas projects in Alberta (Smyth 2004).

COAA initiated a series of studies to identify underlying reasons for project cost overruns. One of the studies identified insufficient planning as one of possible contributing factors to low productivity and cost overrun. Ruwanpura et al. (2006) argued that effective planning can reduce project cost by 40%, while inadequate planning can result in cost overrun as high as 400%. Jergeas (2008) also identified causes for cost overrun and schedule slippage in mega-size oil sands projects, including overly optimistic initial schedules and inadequate project front-end planning.

Project planning and scheduling is at the core of project management and is essential to project success. Project planning is the process of deciding what to do and how to do it before the project is carried out. It is achieved by identifying the scope of work from the design of the

project (i.e. the "what") and selecting the proper construction methods for each piece of work (Fischer and Aalami 1996). The result of the planning process is a list of activities and their precedence dependencies. Project scheduling is a process of allocating limited resources among activities over time. It determines when activities are performed and how long it will take to complete each activity. Project planning and project scheduling are complementary, but they are not the same.

Current construction project planning and scheduling relies mainly on network-based approaches such as critical path method (CPM) and project evaluation review technique (PERT). Commercial software that is based on CPM, such as Primavera P6 or Microsoft Project, has also gained prevalence in the construction industry. These CPM-based tools are useful in presenting a schedule network and in performing CPM calculations, but they also require a complete project plan (all involved activities, their precedence dependencies and resource requirements) as input. This means that these tools are still heavily dependent on the manual formulation of project plans and schedules. Manual project planning and scheduling is a time-consuming and error-prone process, especially for mega-size industrial construction projects. It is an even more tedious job to update and maintain these project plans and schedules. Project planning and scheduling also require intensive knowledge and expertise. For example, construction sequencing involves identifying and applying various physical or technical aspects governing the sequence between different activities. To develop realistic project schedules, various constraints (e.g. dependency relationship, imposed dates, calendar and resource availability) need to be taken into consideration during the scheduling process. CPM-based tools generally lack the ability to capture and apply this knowledge. In other words, they provide little assistance to generate project plans and schedules. As such, project plans and schedules are primarily formulated based

on planners' or schedulers' personal experience and knowledge of the current status of construction (either in fabrication shops or on the construction site). Given the dynamic and uncertain project environments and the sheer size and complexity of industrial projects, it is quite challenging for human planners or schedulers to process all the necessary information to develop accurate project plans and schedules. As a result, the efficiency and effectiveness of project plans and schedules (from human formulation) are severely insufficient.

1.2 SCOPE OF RESEARCH

Manual project planning or project scheduling processes cannot meet the requirements of mega-size oil sands projects. Potential performance improvement can be achieved through automated solutions to both project planning and project scheduling. However, mega-size industrial projects usually span a long period of time and involve various stages. Operations that are involved in one stage might have a totally different nature from those of another stage. For example, fabrication shops run on a project-by-project basis where a factory facility manufactures components for several projects simultaneously. Fabrication shops usually involve one trade (i.e. a spool fabrication shop only fabricates pipe spools). On the other hand, site construction is a typical construction project environment where many different trades are involved to construct a facility (or facilities) that is (are) specific to that project. It is therefore almost infeasible to develop a universal project planning or scheduling tool that can serve all the stages of industrial projects. In other words, each stage should be equipped with a custom built project planning or scheduling tool that accommodates the characteristics of operations in that specific stage. This study focuses on developing automated planning and scheduling tools for fabrication shops and on-site construction.

1.3 RESEARCH OBJECTIVES

The primary objective of this study is to develop automated solutions for project planning as well as project scheduling in both the shop fabrication stage and site construction stage of industrial projects. To reach this goal, three auxiliary objectives have been identified:

I. To develop a new simulation methodology that addresses the uniqueness of the product family in industrial fabrication shops and facilitates simulation-based scheduling and schedule updating for fabrication shops.

II. To investigate the feasibility of automating sequence planning for both pipe spool fabrication and on-site module installation to provide added support to human planners to make informed decisions in an efficient manner.

III. To develop a time-stepped simulation scheduling tool for on-site construction of industrial projects that (1) complies with various constraints of work packages (precedence dependency, time dependent resource availability limit, calendar, time constraint), (2) dynamically allocates resources to work packages, and (3) accounts for jobsite congestion constraints of work areas.

1.4 RESEARCH METHODOLOGY

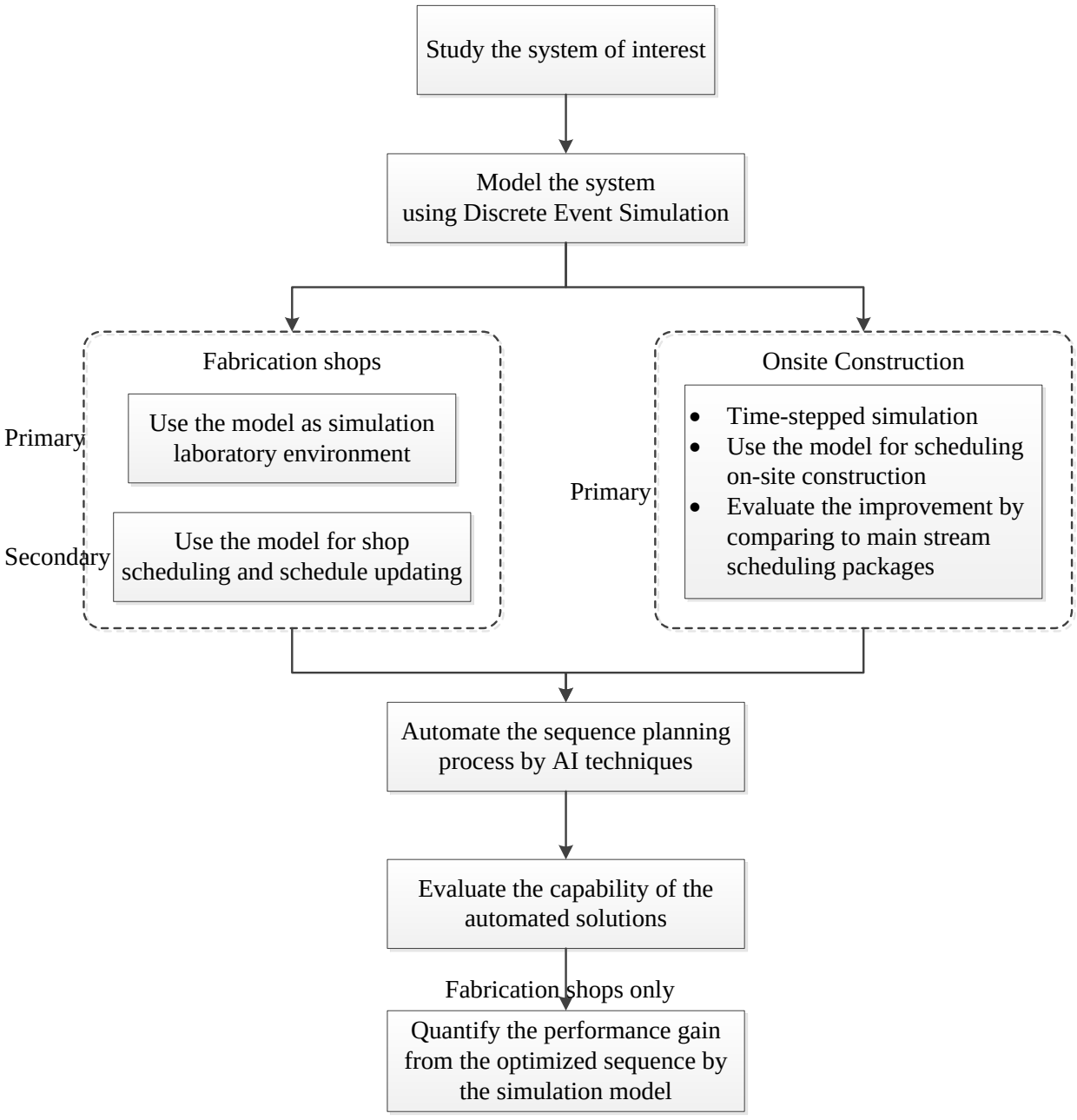


Figure 1-2 Research methodology

The above-mentioned objectives are achieved through the methodology shown in Figure 1-2. Each step in the flow chart is explained as follows:

Step 1: study the system of interest (either fabrication shops or on-site construction), identify primary characteristics of the system and understand the requirements for construction planning and scheduling.

Step 2: model the system at the appropriate level of detail using DES technology. For fabrication shops, the primary purpose is to offer a simulation laboratory environment where various alternative sequences can be evaluated in terms of performance. The simulation model also serves as a scheduling tool that is customized for the fabrication shop process. For on-site construction, a special type of DES simulation, time-stepped simulation (i.e. semi-continuous simulation), is employed to model the on-site construction process. The main purpose of the simulation model is scheduling on-site work packages such that their schedules comply with various constraints, e.g. dependency relationship, imposed dates, calendar, time dependent resource availability, as well as congestion. A dynamic resource allocation algorithm is implemented in the simulation model to resolve both the resource and the space conflict between overlapping work packages. This simulation-based scheduling system is also evaluated by comparing it to other mainstream scheduling packages on the basis of the same industrial construction case study.

Step 3: investigate domain independent AI planning techniques that are efficient and effective to automate the sequence planning process of the system, i.e. sequence for pipe spool fabrication and sequence for on-site module installation. Domain independent AI planning technique (also referred to as general purpose planning) is first selected to experiment on both sequence planning problems. It is discovered that this technique works efficiently to identify feasible sequences for module installation, but fails to do so for pipe spool fabrication due to the limited parsing capability of existing AI planners. This leads to a search for other problem solving techniques.

DP is found to be a good candidate. A DP-based algorithm is customized for identifying the optimal spool fabrication sequences.

Step 4: evaluate the effectiveness of automated solutions developed in Step 3. Nine experiments are conducted to test the capability of AI planning technique (i.e. PDDL and compliant AI planners) to model and to solve the module installation sequencing problem. These experiments use test cases from a real-life industrial project. On the other hand, 29 pipe spools are selected from a real-life industrial fabrication shop to test the capability of the DP-based algorithm to solve the spool fabrication sequencing problem.

Step5: quantify the performance improvement that is gained from the optimized fabrication sequences for 29 pipe spools. The simulation model developed in Step 2 is used to compare the total cycle time resulting from the original sequences (used in the fabrication shop) and the optimized sequences (generated by the DP algorithm).

1.5 THESIS ORGANIZATION

The remainder of this thesis is organized into the following chapters. Chapter 2 reviews current DES methodologies and highlights the problems of using these methodologies to model industrial fabrication shops. It then presents a new simulation structuring methodology that is based on a self-routing entity mechanism. Chapter 3 briefly discusses the spool fabrication sequencing problem and reviews past research on construction sequencing. The focus of this chapter is placed on the use of Dynamic Programming (DP) to formulate the spool fabrication problem and the development of a DP-based algorithm to identify optimal sequences for spool fabrication. It also presents the results of simulation experiments used to quantify the improvement gained from using the DP algorithm. Chapter 4 highlights the requirement for

dynamic resource allocation that is one of the main characteristics of the on-site construction of mega-size industrial projects. It then reviews prior research that is related to resource constrained scheduling and to jobsite congestion. It elaborates the time-stepped simulation framework and the dynamic resource allocation mechanism (i.e. implemented in the time-stepped framework). It also presents a real-life large-scale industrial construction case study and demonstrates the advantage of the proposed CDRASS framework by comparing a generated schedule with those achieved using mainstream scheduling packages. Chapter 5 briefly explains the domain independent AI planning technique, one of its standard planning languages—PDDL, and its applicability to industrial construction sequencing problems. It investigates the feasibility of using domain independent AI planning technique to sequence two industrial construction processes, pipe spool fabrication and on-site module installation. It also provides details of a series of experiments that are conducted for each process. Based on the results of those experiments, the conclusion is drawn.

1.6 REFERENCES

Alberta Enterprise and Advanced Education. (2012). "Inventory of Major Alberta Projects.", available online < http://www.albertacanada.com/sp_majoralbertaprojects.pdf> (Jan. 16, 2013).

Alberta Economic Development Authority (2004). "Mega project excellence: preparing for Alberta's legacy - an action plan." available online <https://aeda.alberta.ca/AEDA%20Public%20Document%20Library/MegaProjectExcel_Dec102004.pdf> (Jan. 16, 2013).

Burati, J. L., Farrington, J. J. and Ledbetter W. B. (1992). "Causes of Quality Deviations in Design and Construction." *J. Constr. Eng. Manage.*, 118(1), 34-49.

Canadian Association of Petroleum Producers (CAPP). (2012). "Crude Oil Forecast, Markets & Pipelines.", available online < http://www.capp.ca/getdoc.aspx?DocId=209546&DT=NTV> (Jan. 16, 2013).

Construction Owner Association of Alberta (COAA). (n.d.) Citing Websites. *Workface Planning*. Retrieved from http://www.coaa.ab.ca/Productivity/WorkFacePlanning.aspx (Mar. 12, 2013)

Fayek, A. Robinson. Invited speaker and workshop. (2004). "A Preliminary Study to Identify and Quantify Productivity Deviations on Heavy Industrial Construction Projects in Alberta", *Construction Owners Association of Alberta Best Practices XII Conference*, Edmonton, Alberta, May 19-20, 2004.

Feuffel, J. and Hanley, K. Jr. (2009). *Workface Planning*. Excellence in Construction, PCL College of Construction.

Fischer, M. A., and Aalami, F. (1996). "Scheduling with Computer-Interpretable Construction Method Models." *J. Constr. Eng. Manage.*, 122(4), 337–347.

Fossen, B and Kukkola, R. (2009). *Executing Industrial Module Projects Under Extreme Market Conditions*, PCL College of Construction.

Haas, C.T., O'Connor, J.T., Tucker, R.T., Eickmann, J.A. and Fagerlund, W.R. (2000*). Prefabrication and Preassembly Trends and Effects on the Construction Workforce*. Report No. 14, Center for Construction Industry Studies, Austin, TX.

Howell, G.A. and Ballard, H.G. (1996). *Managing Uncertaintyin the Piping Process*. RR 47-13, Constr. Industry Institute, Univ. of Texas, Austin, TX, September, 103 pp.

Jergeas, F. (2008). “Analysis of the front-end loading of Alberta mega oil sands projects.” *Project Management Journal*, 39 (4), 95–104.

Ruwanpura, J.Y., Ahmed, T.N., Karim Kaba, K. and Mulvany, G.P. (2006). “Project Planning and Scheduling and its Impact to Project Outcome: A study of EPC Projects in Canada.” *AACE International Transactions*, 20.1-20.9.

Smyth, M. (2004). APEGGA President Addresses Mega-project Overruns, Release February 12, 2004. Speech of the Association of Professional engineers, Geologists and Geophysicists of Alberta.

Song, L., and AbouRizk, S.M. (2003). “Building a Virtual Shop Model for Steel Fabrication.” *Proceedings of Winter Simulation Conference*, New Orleans, Louisiana, USA, pp. 1510-1517.

Song, L., and AbouRizk, S. M. (2006). "Virtual Shop Model for Experimental Planning of Steel Fabrication Projects." *J. Comput. Civ. Eng.*, 20(5), 308-316.

Tatum, C. B., Vanegas, J. A., and Williams, J. M. (1987). *Constructability improvement using prefabrication, preassembly, and modularization*, Construction Industry Institute, Austin, TX.

Tommelein, I.D. (1998). "Pull-driven Scheduling for Pipe-Spool Installation: Simulation of Lean Construction Technique." *J. Constr. Eng. Manage.*, 124 (4), 279-288.

Walsh, K.D., Hershauer, J.C., Walsh, T.A., Tommelein, I.D., and Sawhney, A. (2002). "Lead Time Reduction via Pre-Positioning of Inventory in an Industrial Construction Supply Chain." *Proceedings of Winter Simulation Conference Proceedings*, San Diego, CA, USA, pp. 1737-1744.

Wang, P. (2006). "Production-based Large Scale Construction Simulation Modeling." Ph.D. thesis, University of Alberta, Edmonton, Alberta, Canada.

Wang, P., Mohamed, Y., AbouRizk, S. M., and Rawa, A. R. T. (2009). "Flow Production of Pipe Spool Fabrication: Simulation to Support Implementation of Lean Technique." *J. Constr. Eng. Manage.*, 135(10), 1027-1038.

Wyss, Stephen. (2009). "Active Management of Pipe spool Fabricators." *Chemical Engineering*, 116(1), 40-45.

CHAPTER 2. Simulation Model Structuring Methodology for Industrial Fabrication Shops

2.1 BACKGROUND AND PROBLEM STATEMENT

In the construction domain, contractors are often faced with projects with varying degrees of uniqueness. Between two extremes on the spectrum—*one-of-a-kind* project and *repetitive* project, most construction projects are characterized by more or less degree of customization and repetitiveness. Industrial fabrication shops (e.g. pipe spool fabrication shops and structural steel fabrication shops) are an example, where the same set of operations is repetitively performed on different products (e.g. pipe spools and structural steel pieces) but the sequence of these operations varies considerably from one product to another, due to unique design and configuration. Industrial fabrication is described as a "low-volume and high product mix production process" (Song et al. 2006). It is also described as job shop environment (Karumanasseri and AbouRizk 2002). This differentiates industrial fabrication shops from manufacturing shops where identical or standard products are mass produced with limited variation . As a result, planning and scheduling is constantly required for each shop product.

Traditional-critical-path-method- (CPM) related approaches are not effective to model industrial fabrication shops, due to CPM's limitation or inability to model the repetition of operations, the interactions between resources, and what-if scenarios. Discrete event simulation (DES) has long been widely used to model and study real-world systems (Wang and Halpin, 2004), especially for processes that are repetitive in nature. Recently, simulation has increasingly been used for scheduling day-to-day operations (referred to as simulation-based scheduling). By simulating the behavior of a production system over time, DES is able to reproduce the process of how products and resources interact with each other. A schedule can then be viewed as a record of this

artificial history. DES also provides a cost-effective laboratory environment where various alternatives can be tested and compared and the best one can be selected without interrupting the real system.

Despite these advantages, DES has limitations in modeling industrial fabrication shops. The major challenge derives from the high product mix. Industrial shop products, though undergoing the same operations, differ considerably in the sequence of operations necessary to fabricate them. This sequence is referred to as routing for the rest of this chapter. Most current simulation methodologies 'hard code' this routing in the graphical representation using directional arrows and branching elements. To guide simulation entities (entities represent shop products) through a series of shop floor operations, each routing needs to be explicitly indicated in the model. This method of modeling works for identical products or standard products where only one or few typical routings need to be modeled. However, when facing complex, large-scale industrial projects with more than 10,000 unique pipe spools it becomes cumbersome and inefficient, for two reasons. First, almost every unique product requires a unique routing in the shop. When all these routings have to be tracked and explicitly mapped out in the simulation model, the graphical representation becomes cluttered and hard to read. Second, the simulation model is rather static in that once it is developed, it can only process routings that are already defined in the model. Whenever a new routing is encountered the model has to be modified. This means that the simulation model is very unlikely to be re-used from one project to another, which makes the development of the simulation model quite uneconomical.

This chapter proposes a new simulation model structuring methodology. It directly addresses the complex routing issue in industrial fabrication shops and significantly simplifies the simulation model development. Moreover, this structuring methodology also fully supports both shop

scheduling and schedule updating, the latter of which is essential for industrial fabrication shops that operate under uncertain dynamic project circumstances.

In the following sections of this chapter, a literature review of DES methodologies is provided first, which is followed by a generalization of the simulation model structuring paradigm that is common among many construction simulation environments. After this, the new model structuring methodology is introduced and two major components, entity object information model and state-based entity routing mechanism, are explained in detail. An example is provided to illustrate how this methodology supports shop scheduling and schedule updating. Finally, conclusions, limitations and future development are also discussed.

2.2 SIMULATION METHODOLOGIES

Simulation methodologies (also called 'simulation strategies') specify the world view that a simulation analyst applies towards modeling a real-world system, and therefore, significantly impact model development (Martinez and Ioannou 1999; Zhang et al. 2005). Different simulation methodologies vary mainly in the basic building blocks they use to model a system (Pidd 1998). Event Scheduling (ES) describes a system by a chronological list of unconditional events (Zhang et al. 2005), each of which is composed of a routine or a list of actions (Pidd 1998) and can trigger another event. Activity Scanning (AS) views a system from the viewpoint of activities. AS-based simulation models advance by repeatedly scanning start-up conditions of activities and activating those whose conditions are satisfied (Martinez and Ioannou 1999, Zhang et al. 2005). Due to repeated activity scanning and, consequently, the slow runtimes on computers, AS has been replaced by one of its modified forms—the three phase approach. Many construction simulation tools adopt the three phase approach, such as CYCLONE (Halpin 1977)

and STROBOSCOPE (Martinez 1996). Process Interaction (PI) decomposes a system into processes. In PI world view, entities transverse the model and trigger various activities by requesting resources, engaging resources to perform operations and then releasing them. Each entity has its own life cycle. A simulation model is thus defined in terms of entities' life cycles. Simulation entities are distinguishable in PI approach. There are moving entities, as described above, as well as static entities that are scarce resources. The quality of the simulation model largely depends on how simulation analysts select moving entities and scarce resources (Martinez and Ioannou 1999). PI is especially suitable to model systems in manufacturing and service industries (Martinez and Ioannou 1999), because it offers a natural metaphor for production systems in these industries. One of established applications of PI in the construction domain is Simphony.NET (AbouRizk and Mohamed 2000).

2.3 TRADITIONAL SIMULATION MODEL STRUCTURING PARADIGM

Process interaction and activity scanning are two of the most frequently used methodologies in construction. CYCLONE (Halpin 1977), COOPS (Liu and Ioannou 1992), and STROBOSCOPE (Martinez 1996) are based on AS methodologies, or specifically, based on modified forms of AS—three-phase, while SLAM II (Shi and AbouRizk 1997) and Simphony.Net (AbouRizk and Mohamed 2000) are based on PI methodologies. Lu (2003) further simplified the AS methodology and proposed Simplified Discrete Event Simulation Approach (SDESA). Zhang (2005) developed an object-oriented strategy (Activity Object-Oriented, AOO) to model construction activities.

Despite the apparent differences between these simulation environments, they all contain the same set of fundamental elements necessary to compose a simulation model (shown in Figure 2-

1), including moving *entities*, *operations* that transform entities, *resources* that perform operations, and *directional arrows* indicating the flow of entities or the sequence of operations. All of these elements are modeled in one way or another in these simulation environments. Another commonality is that these simulation environments are all associated with a graphical representation. Graphical method is used to develop simulation models since it reduces the requirement for familiarity of simulation programming languages. Except for moving entities that are often invisible, all the other elements are included in the graphical representation, in one way or another.

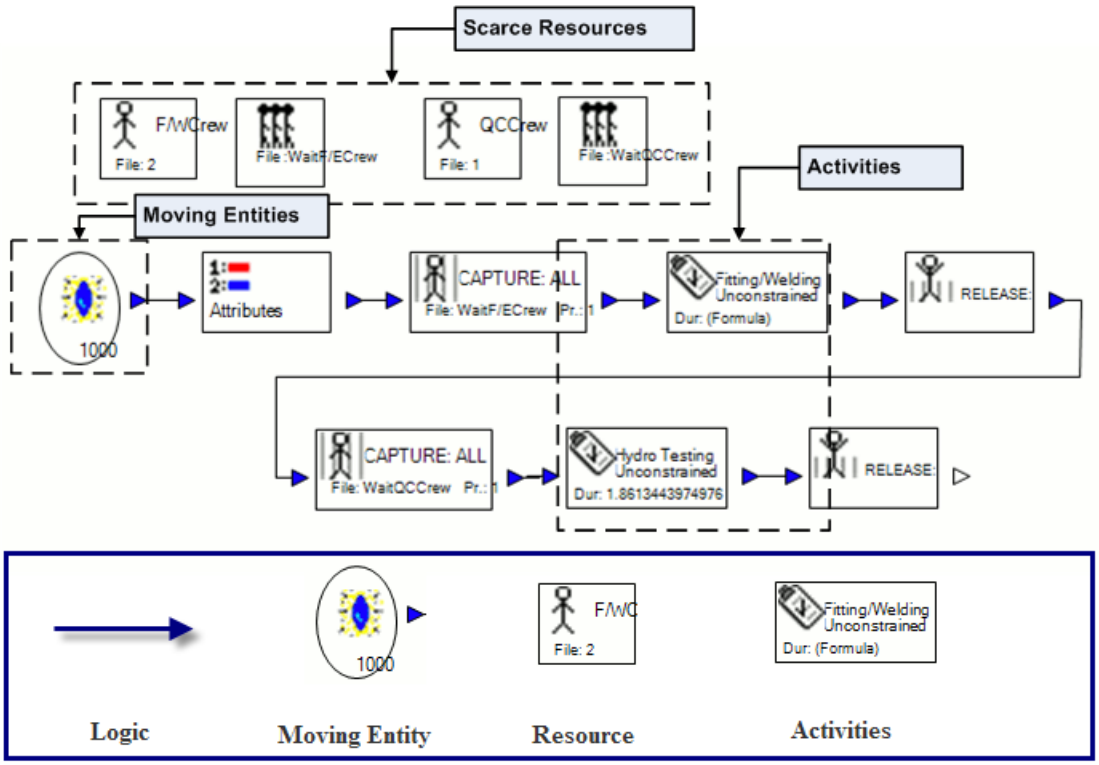


Figure 2-1 Four fundamental elements for a simulation model (in Simphony.Net)

Further observation finds that all these simulation environments require explicit mapping out of the routings of moving entities in the graphical representation, with directional arrows and

branching elements. Dubiel and Tsimhoni (2005) pointed out that "in all commercial DES packages, a path must be drawn from one point to another in order for an entity to move between those two points." This "generality of movement" (Dubiel and Tsimhoni 2005) of entities implies a hidden assumption that variation in routing is small (i.e. all entities follow the same path or a few typical paths). However, this assumption does not hold when it comes to modeling industrial fabrication shops where almost every product tends to have a unique routing. The graphical network might be cumbersome when there are a large number of distinct routings. Another alternative is to generalize these routings. However, certain unrealistic assumptions must be made as a result. Wang (2009) developed spool fabrication shop model (Figure 2-2) using the traditional way of modeling entity routing. In this model, it is easy to identify that pipe spools coming from RollFitter3 (i.e. a roll fitting table) have to go to either RollWelder4 or RollWelder5, instead of any other welding stations (e.g. RollWelder1, 2 or 3). But this is not the case in reality. In fact, pipe spools, after being fitted, should be able to choose any welding station, as long as it is available. Similar unrealistic assumptions can be identified in routing between other fitting tables (RollFitter2 and RollFitter1) and welding stations (RollWelder1, 2, 3, 4 and 5). These assumptions reduce the accuracy and fidelity of simulation models. Sadeghi and Fayek (2008) developed a simulation model for industrial fabrication shops in order to study long-term performance. However, its graphical representation is not described in the paper and therefore it is impossible to compare to this study. Rokni (2010) developed a simulation with high level of details to model the pipe spool fabrication shop. However, the directional arrows are still used to represent the flow of the entity.

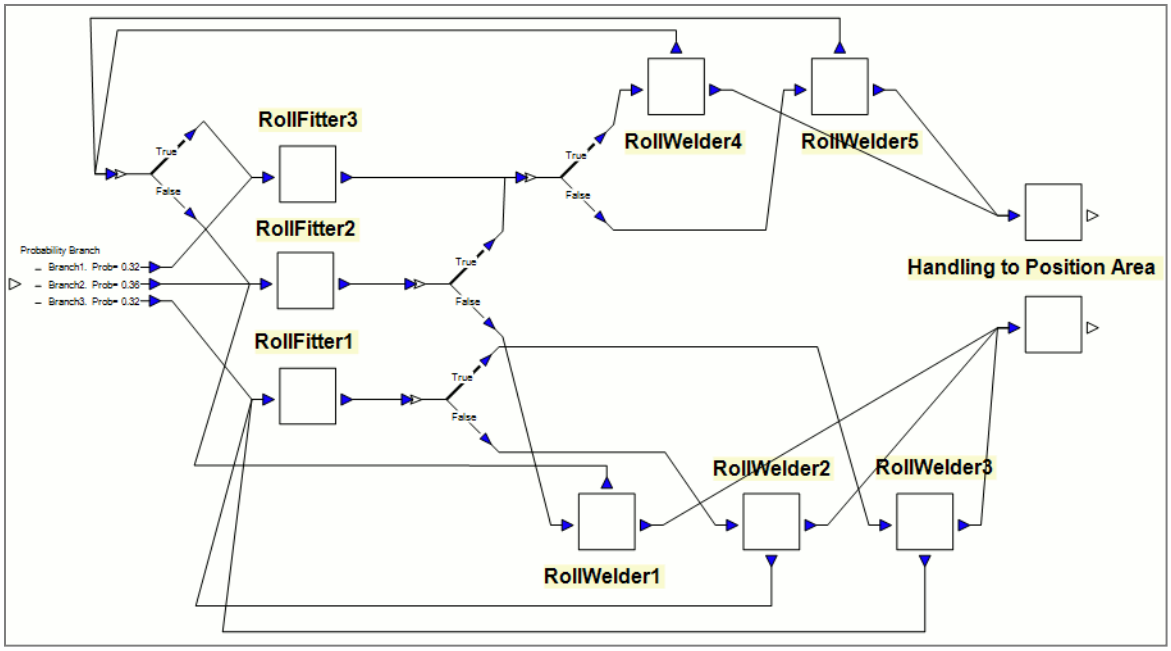


Figure 2-2 Example simulation model for a spool fabrication shop (Ping Wang et al. 2009)

2.4 PROPOSED SIMULATION MODEL STRUCTURING FOR INDUSTRIAL FABRICATION SHOPS

Within industrial fabrication shops, products (pipe spool or structural steel) arrive, undergo certain processing and exit the system. To model this system, it is natural to choose shop products as moving entities and to treat workstations (e.g. fitting tables or welding stations) as resources. As such, PI is a suitable method to model industrial fabrication shops.

However, problems still exist with PI-based simulation models. To address these problems, a new methodology for structuring a simulation model is developed. More specifically, all the routing (i.e. directional arrows and/or branching elements) that indicates the flow of entities is extracted from the graphical model and is carried by individual entities instead. This poses certain requirements for simulation entities, including (1) distinction between entities so that each represents a unique product, and (2) ability to autonomously route through the simulation model (i.e. according to their process plans). In the following sections, these points will be

addressed through detailed explanations of entity information model and the state-based entity routing mechanism.

2.4.1 Entity object information model

Moving entities in PI-based simulation tools are by default considered equal. In order to make them distinct and representative of individual products, some key product information needs to be carried by entities. This often refers to descriptive attributes such as product ID, name, product type and any other identification information that helps distinguish one product from another (e.g. a pipe spool always has an associated ISO number in addition to the pipe spool number). Other physical characteristics such as dimensions, size and weight are also necessary, since they may impact the product routing in the shop. For example, the diameter of a pipe usually determines the cutting station where the pipe can be cut.

Routing information (i.e. other interchangeable terms include process plan or fabrication sequence), like product information, is also highly product-dependent. Routing of a pipe spool or structural steel is largely decided by its physical configurations (Figure 2-3). Since industrial shop products tend to have unique configurations, the routing often differs greatly from one product to another. Figure 2-3 shows an example pipe spool fabrication sequence. Each fabrication step represents a pair of a fitting operation and a welding operation. Fabrication steps depicted in a vertical direction have to be performed in a sequential way (i.e. the final pipe spool cannot be produced before sub-assembly 7 and 8 are fabricated), while processes depicted on the same horizontal level can be performed concurrently (i.e. sub-assembly 7 and sub-assembly 8 can be performed independently). Each step must have all the components (e.g. raw material or intermediate sub-assemblies) ready before it can start.

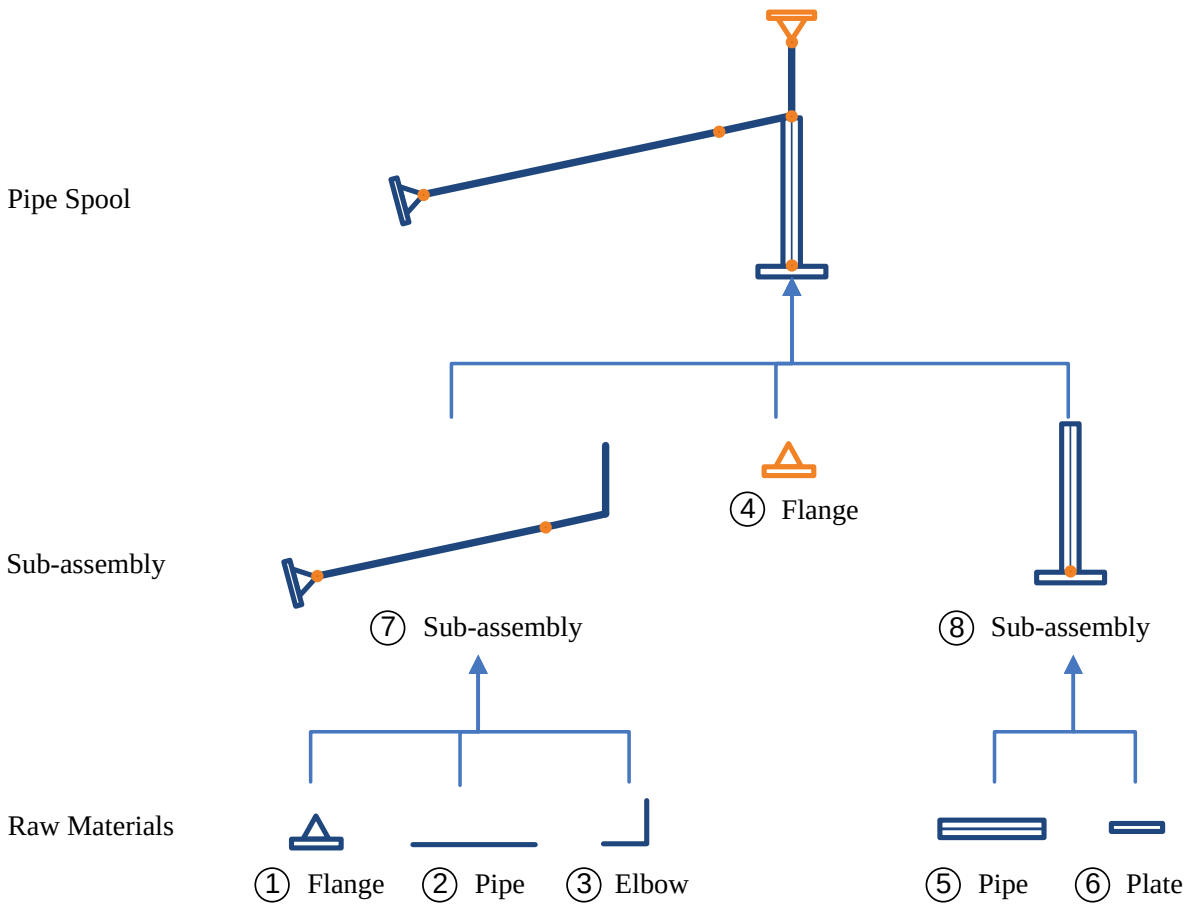


Figure 2-3 Fabrication sequence of an example pipe spool

To enable entities to carry both product and routing information, an entity information model is designed (Figure 2-4). An information model can be defined as "a collection of information that describes, represents, or abstracts something" (Fisher and Froese 1996). Its scope largely depends on the perspective one takes towards the object. The main purpose here is to enable entities to be distinguished from one another, and to route themselves through the simulation. Product information and routing information are therefore major pieces to be included. In addition, two other sets of information are also incorporated in the information model, including entity state information and schedule information. Compared to product and routing information,

these are output information generated during the simulation. The state information describes how far the fabrication has progressed. It helps the entity determine the next fabrication step(s) to perform. The details of how to use this information will be explained in the next section. The schedule information simply records details of operations that have been completed, such as its start time, duration, finish time and work stations that performed the operation.

In this information model, routing information is divided into two parts. The first part is a list of operations that a product needs to go through before becoming the final product. Each operation represents the transformation from raw material to an intermediate sub-assembly or the final product. For example, the fitting operation that assembles spool component 1, 2 and 3 produces sub-assembly 7 (Figure 2-3). The information model also specifies the type of workstation that can provide such service. For example, a fitting operation has to be performed at a fitting table and a welding operation engages a welding station. The second part specifies the precedence relationships between these operations. This dependency is determined by two underlying reasons: (1) technical reasons and (2) composition reasons. An example of a technical reason is that fitting operations for pipe spools always precede the welding operations since the former provides only temporary connections while the latter provides permanent connections between spool components. Figure 2-3 shows a good example of composition reasons. The fitting operation that assembles sub-assembly 7 and 8 into the final pipe spool can only take place after both sub-assemblies have been welded.

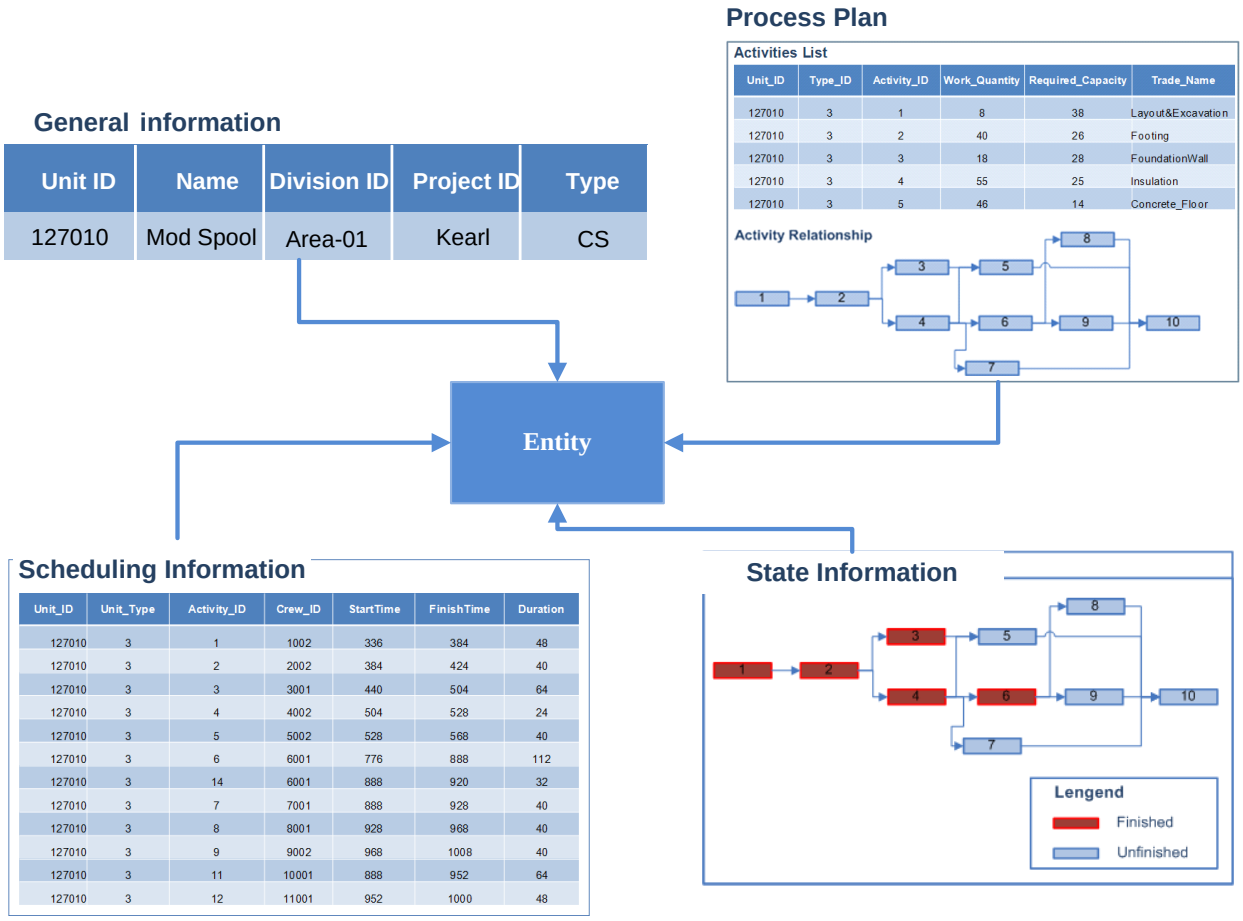


Figure 2-4 Entity object information model

2.4.2 State-based entity routing mechanism

The main objective is to enable moving entities to autonomously route through the simulation model. This means that moving entities should be able to determine what operations to perform next and where to have them performed. Such decisions are often made when entities enter the simulation model or whenever an operation is completed. The aforementioned integrated information model encapsulates all information necessary for entities to make such decisions. However, how the information is used to infer the next operation to perform has not been addressed.

A state-based entity routing algorithm (Figure 2-5) is developed. It mainly makes use of two information sets: (1) routing information and (2) state information. Routing information works as a road map that shows how an entity gradually evolves from the initial state (i.e. raw materials) to the goal state (i.e. the final product). State information, on the other hand, describes the current completion status of entities. Each entity is composed of distinct operations. The state of each operation can be simply described by its percent completed. The states of individual operations constitute the state of the moving entity. Figure 2-4 shows an example entity state. It indicates that the entity has completed fabrication operations 1, 2, 3, 4 and 6, but is yet to fulfill the operations 5, 7, 8, 9 and 10. At this moment, the algorithm will be activated since operation 6 has just been completed and the state of the entity has been updated. The algorithm is formulated as shown in Figure 2-5.

```
Start simulation
        Set current time to zero
        Set Start state to the initial state
        While not (final state) do
                Scanning current state information
                *Identifying operations that can be started based on precedence constraints
                Dispatching a copy of moving entity to corresponding workstations
                *If Entity copies from all predecessors have already been received
                        Performing the operation
                        Updating current state
                        Record all schedule times
                End If
        End while
End simulation
```

Figure 2-5 State-based entity routing algorithm

Key steps in this algorithm are marked with asterisks. The selection of operation to perform is controlled by the precedence constraints and the current entity state. If an operation is completed, a copy of the current entity will be dispatched to each of its succeeding operations. However, this does not necessarily lead to the start of any succeeding activities. If the succeeding operation has only one predecessor, it will be triggered immediately. However, if the succeeding operation has more than one predecessor, it will not be performed until all the copies of the entity have been received. This kind of checking is done whenever a dispatching event occurs. Suppose, for example, an entity with the same process plan as shown in Figure 2-4 has already finished operations 5, 6 and 7 at the same time. According to the algorithm, upon completion of operation 5, a copy of the entity will be sent to each of the workstations where operation 7 and 10 can be performed. A similar situation occurs when operation 6 is completed. Workstations that will perform operations 8 and 9 will receive a copy of the entity. In this case, both operation 8 and operation 9 can start right away. Operation 10, however, cannot start since it still needs another two copies of the entity from the workstations that perform operation 8 and 9. It should be noted that when operation 10 is eligible to be performed, all four copies will be consolidated into a single entity so that operation 10 will be performed just once. This algorithm will be repeatedly triggered until the entity goes through all the operations and exits the simulation model.

It should be noted that the state-based entity routing mechanism is only responsible for the process plan (or precedence relationship) while the resource constraint is still taken care of by the capture-engage-release routine already built into the simulation environment.

2.4.3 Simplified graphical representation

When the routing information is extracted from the graphical representation and carried directly by moving entities, the model is reduced to facility modeling of the industrial fabrication shop. Specifically, only workstations that perform shop floor operations will be modeled. In simulation environments where hierarchical simulation modeling is allowed, each workstation can be modeled with a composite simulation element (Figure 2-6), which has a lower level child model that defines all details (e.g. what resource is required and how the operation is carried out) related to this particular operation. Figure 2-6 shows such an example of hierarchical modeling in Simphony.NET (AbouRizk and Mohamed 2000). The simulation model now has flexibility to accommodate variations in entity routing. It is able to process any entity, as long as the model contains all the required workstations (or operations), regardless of what routing it chooses to traverse them.

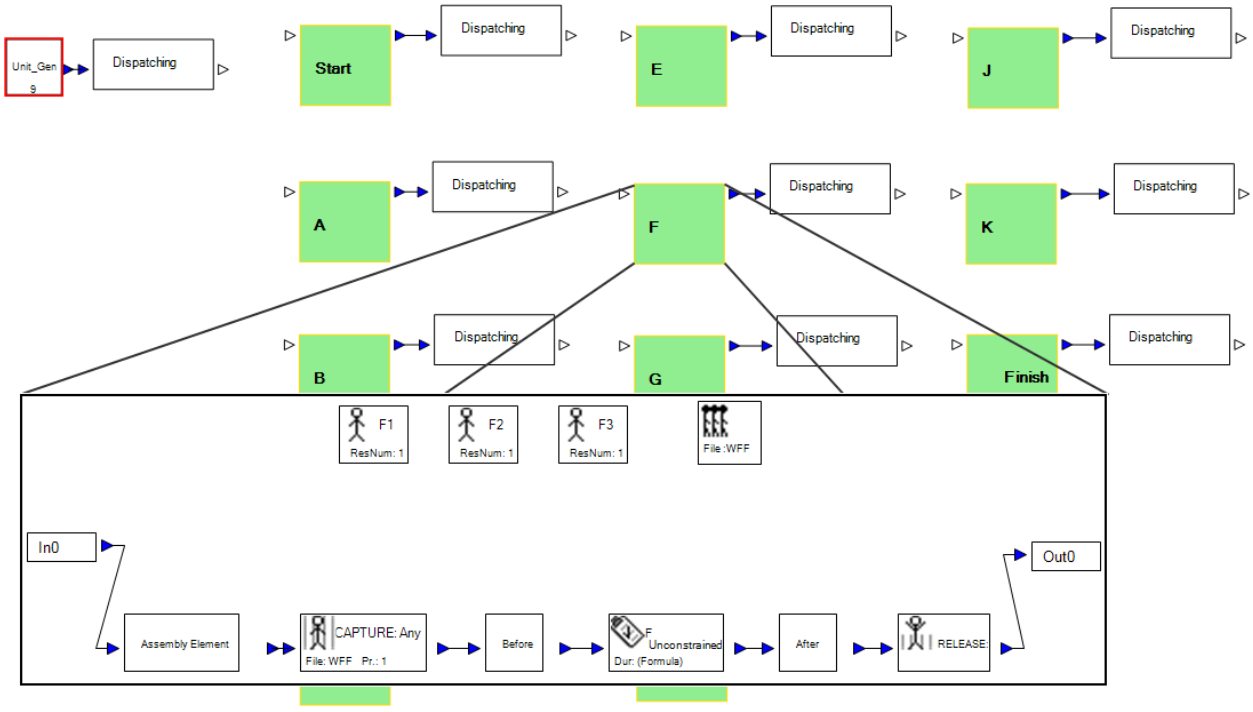


Figure 2-6 Simplified graphical representation of illustration example

2.4.4 Schedule updating

In this approach, the progression of a simulation entity along its life cycle is mainly driven by its current state. This means that the entity is able to start its simulation from any intermediate state throughout its life cycle, rather than only from the initial state (i.e. when no operation has been performed yet). If simulation users feed the simulation model with a group of entities with partially completed states, the simulation can run directly from those states and simulate the rest of their life cycles. This represents a departure from conventional simulation-based scheduling practice, which requires manipulating the simulation to run in a way that exactly imitates the as-built progress and to continue simulation from that point. Comparing these two, the new approach provides a more natural way to perform scheduling and schedule updating. In addition, the simulation also has the ability to reconstruct the current work load of an industrial fabrication shop. It is particularly useful to model a situation where a shop is already loaded with existing products when new orders enter the system.

2.5 METHODOLOGY IMPLEMENTION

The implementation of the proposed simulation structuring methodology in Simphony.NET results in two special purpose simulation (SPS) templates. The first template is a generic simulation tool that facilitates modeling any construction process that involves a high product mix and varied construction (or fabrication) sequences. A model developed from this template is shown in Figure 2-6. The second template is customized for modeling pipe spool fabrication shops. It consists of many simulation elements that are specific to pipe spool fabrication such as bay element and lay-down element. These elements represent the objects that are involved in real-life pipe spool fabrication shops. The detailed implementation of these simulation elements

in Simphony.Net is provided in Appendix A. The second template also has a number of built-in spool fabrication rules. For example, a simulation entity that represents a pipe spool is usually decomposed into a number of entities that represent its components. A simulation element is designed for this purpose, which tags each entity with a unique component ID and the original pipe spool ID (as a reference to retrieve information about the original pipe spool entity). A number of entities are flowing concurrently in the simulation and are gradually assembled together in the sequence specified in the spool process plan. Each assembly operation involves conversion of a number of spool components into a spool sub-assembly or the final assembly. Another simulation element is also designed to fulfill this purpose, which collects all entities (that represent spool components) that are involved in the assembly and generates a new entity that represents the resulting sub-assembly. Another rule is that all components that belong to the same pipe spool have to be fitted at the same fitting table, though they can be welded at different welding stations. For example, sub-assembly 7, sub-assembly 8 and the final assembly in Figure 2-3 should be fitted at the same fitting table and can be welded at any welding station depending on its availability. This rule is also implemented in the pipe spool fabrication simulation template.

2.6 SYSTEM ARCHITECTURE

Since a large amount of input information is required for the model development and a great deal of output information (i.e. scheduling information) is generated after the simulation run, the simulation works closely with a database. Figure 2-7 shows the architecture of the system that is composed by (1) Microsoft Access®, (2) Microsoft Project® and (3) a simulation model.

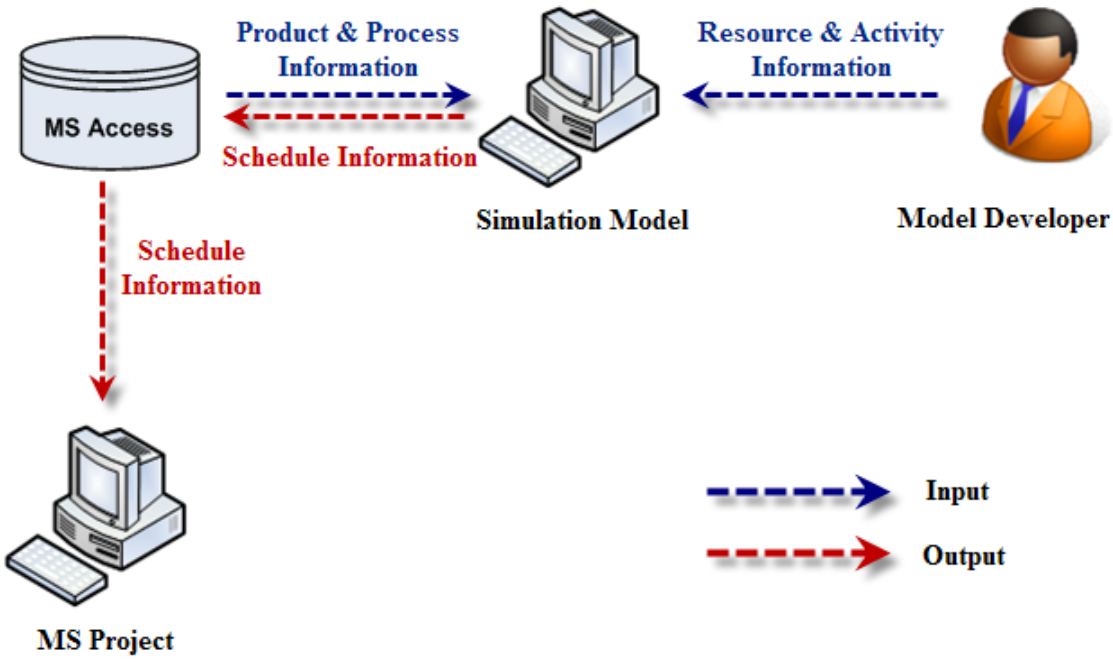


Figure 2-7 Architecture of the prototype system

The system starts with simulation developers modeling existing workstations on the shop floor using elements defined in the simulation template. The simulation model then reads both product and routing information from a Microsoft Access® database and assigns it to entities so that each of them represents a shop product. The simulation model is then ready to run. Schedule information is collected for each entity during the simulation run and is sent back to the database after simulation ends. The resulting schedule is then translated by a user-defined function in Microsoft Access® from simulation times to real calendar dates. Finally, the schedule is presented in Gantt chart style when it is sent from the database to Microsoft Project®.

2.7 AN ILLUSTRATION EXAMPLE

A simple case study is used to illustrate this concept. Assume the shop receives an order that involves three different types of shop products. The process plan for each type (Figure 2-8) differs from each other. For example, Type 2 varies from Type 1 mainly in operation duration

(e.g. for operations D, G, I and J) and it does not include operation E, while Type 3 varies in both process plan and duration, which results in a completely different network than for Type 1. The order involves producing nine units (i.e. three units per type). Assume these product units are dispatched to the shop floor at the same time and are processed in the order of their identity numbers, e.g. unit 1 has higher priority than unit 2. Each workstation has two or three resources available to perform the operation. For simplification purpose, durations of all the operations are assumed to be deterministic, but stochastic durations are also natively allowed in Simphony.NET—users can select a certain type of probability distribution and specify the associated parameters.

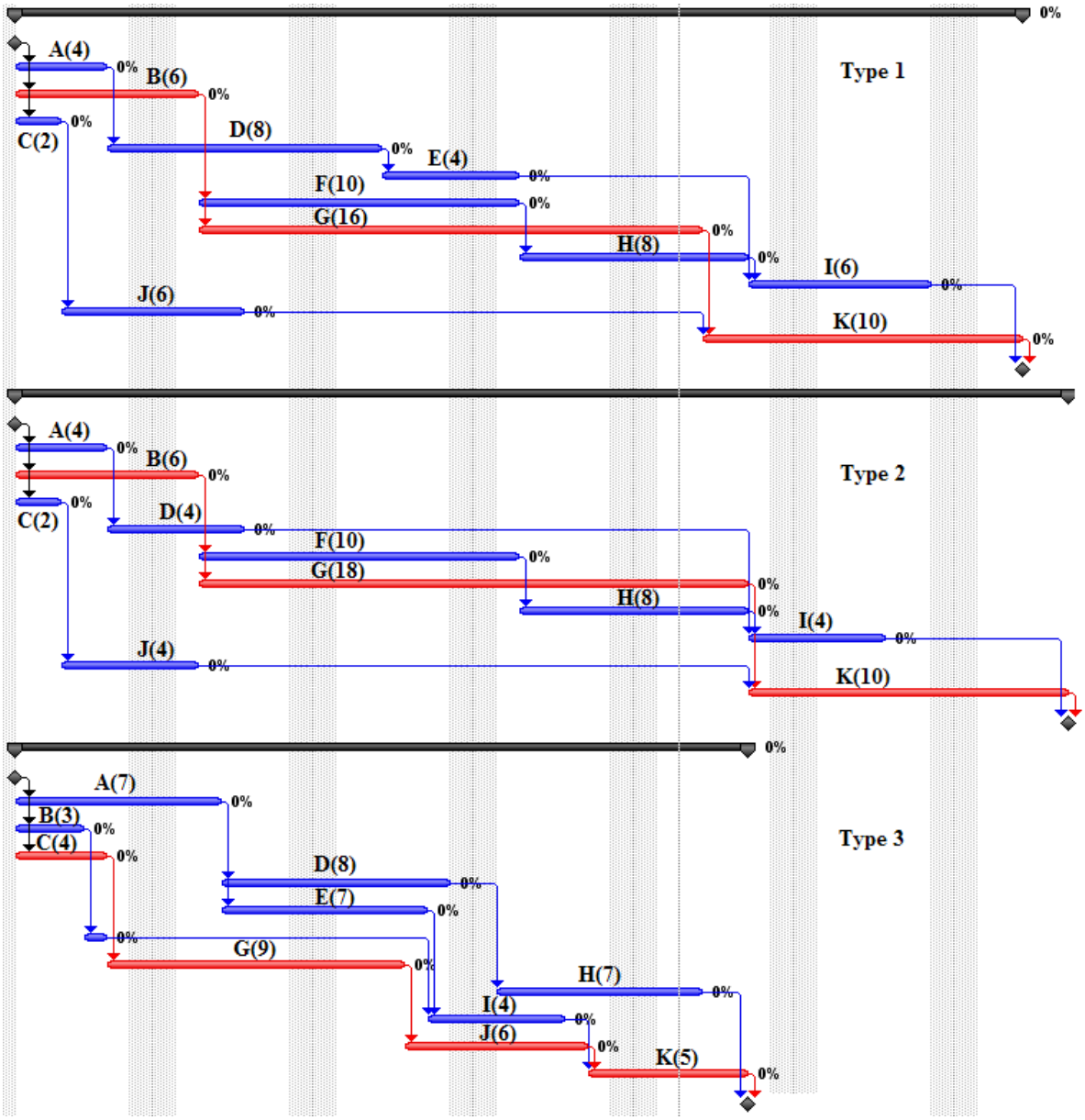


Figure 2-8 Process plans for three product types

Figure 2-6 shows the graphical representation developed by the proposed methodology while Figure 2-9 shows the graphical representation developed by the traditional methodology. It is easy to identify that Figure 2-6 is much less cluttered than Figure 2-9. Note that only three different types of routings are considered in this example. The difference could be much more when thousands of unique products need to be modeled (e.g. for an industrial construction project). In addition, whenever a new routing is encountered, the model in Figure 2-9 has to be

modified (i.e. adding more arrows and branching elements). On the other hand, the model in Figure 2-6 does not require any modification, if the model already contains all the required operations.

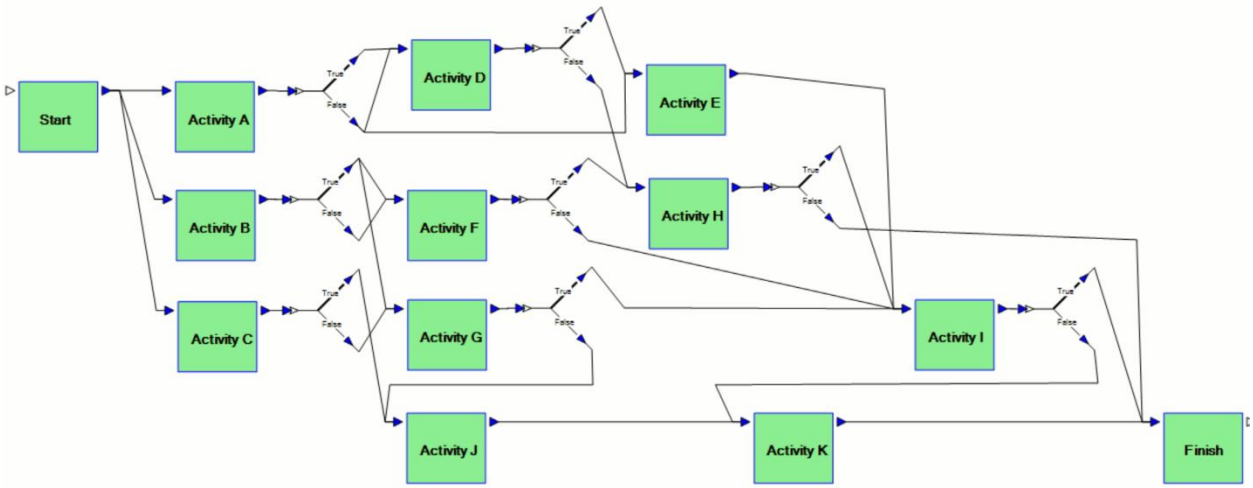


Figure 2-9 Graphical representation using traditional simulation methodology

To illustrate the schedule updating, a virtual depiction of as-built progress is also developed, which highlights its deviations from the initial schedule. Assume at the end of the fifth day some schedule slippage is identified. Table 2-1 lists all the units with deviations (i.e. the units that are not listed here are not started yet). After it is fed with this as-built progress, the simulation model generates a schedule update. A comparison between the initial schedule and the updated one can then be made by Microsoft Project® (light grey bars in Figure 2-10).

Table 2-1 As-built progress of the illustrative example

Unit	Operation ID	Operation Name	Original Schedule	Actual Progress	Schedule deviations
Unit 1	1	A	100.00%	100.00%	on time
	2	B	83.30%	50.00%	2 days behind
	3	C	100.00%	100.00%	3 days behind
	4	D	12.50%	12.50%	on time
Unit2	1	A	100.00%	100.00%	on time

	2	B	83.30%	50.00%	2 days behind
	3	C	100.00%	100.00%	3 days behind
	4	D	25.00%	25.00%	on time
Unit 3	1	A	14.30%	14.30%	on time
	2	B	100.00%	100.00%	2 days behind
	3	C	75.00%	0.00%	3 days behind
Unit4	1	A	14.30%	14.30%	on time
	2	B	66.70%	0.00%	2 days behind
	3	C	75.00%	0.00%	3 days behind

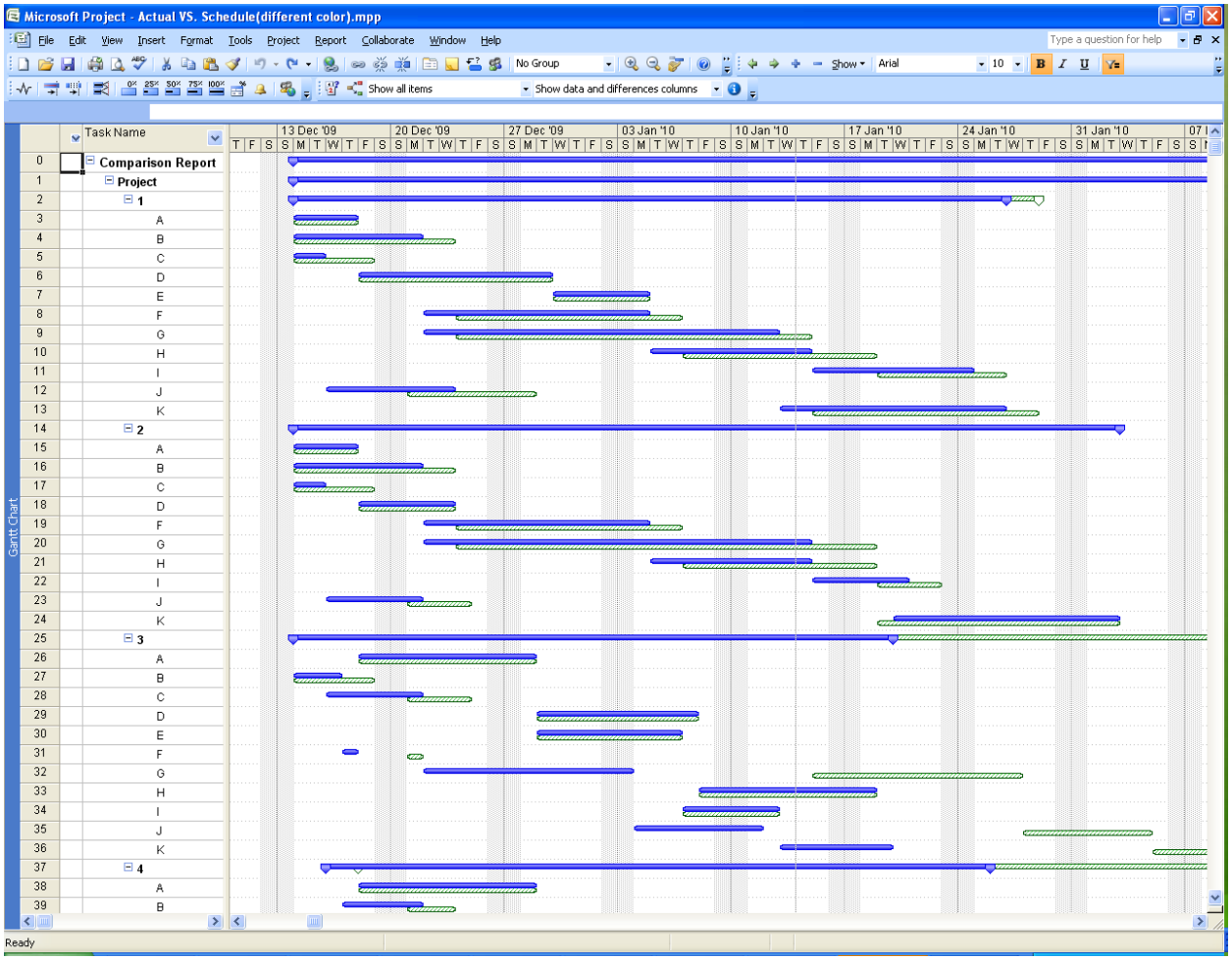


Figure 2-10 Initial schedule and updated schedule

2.8 CONCLUSION

The proposed methodology for structuring a simulation model stresses that when modeling a construction system where products have highly varied processes, it is better to extract the logic

information from the graphical model and to enable each moving entity to carry it directly. To achieve this, an entity object information model and a state-based entity routing mechanism are developed, which track the state of each entity along its life cycle and use precedence constraints and the current entity state to figure out how to transverse various operations in the simulation model. Advantages include a less cluttered graphical representation, reusability of the simulation model, and a natural way to perform both scheduling and schedule updating.

There are also limitations for the proposed methodology. The major limitation is that graphical representations of simulation models no longer contain the routing information (how entities flow in the simulation). Simulation users have to define all routing patterns in the database before the simulation can run. Another limitation is that simulation models heavily depend on reading a large amount of data from the database. If these data are scattered in various existing information systems or some of them are contained in paper documents, the consequential manual input could be very cumbersome. These limitations will be addressed in the future research section.

2.9 REFERENCES

AbouRizk, S., and Mohamed, Y. (2000). “Simphony-an integrated environment for construction simulation.” *Winter Simulation Conference Proceedings, 2000*, Orlando, FL, USA, 2: 1907-1914.

Chang, D. Y. (1986). "RESQUE: A resource based simulation system for construction process planning." PhD dissertation, University of Michigan, Ann Arbor, Michigan.

Dubiel, B. and Tsimhoni, O. (2005). "Integrating Agent-based Modeling into a Discrete Event Simulation." *Winter Simulation Conference Proceedings, 2005*, Orlando, FL, USA, 1029-1037.

Fisher, M. and Froese, T. (1996). "Examples and Characteristics of Shared Project Models." *Journal of Computing In Civil Engineering*, ASCE, 10(3), 174-182.

Halpin, D. W. (1977). "CYCLONE—Method for modeling job site processes." *Journal of Construction Division*, ASCE, 103(3), 489–499.

Harrell, C. R., and Tumay, K. (1990). "ProModelPC tutorial." *Winter Simulation Conference Proceedings, 1990*, IEEE, New Orleans, LA, USA, 128-131.

Karumanasseri, G. and AbouRizk, S. (2002). "Decision Support System for Scheduling Steel Fabrication Projects. " *J. Constr. Eng. Manage.*, 128(5), 392–399.

Liu, L. Y., and Ioannou, P. G. (1992). "Graphical object-oriented discrete-event simulation system." *Winter Simulation Conference Proceedings, 2008,* Piscataway, N.J., USA, 1285–1291.

Martinez, J.C. (1996). "STROBOSCOPE - State and Resource Based Simulation of Construction Process." PhD dissertation, University of Michigan, Ann Arbor, Michigan.

Martinez, J. C. and Ioannou, P. G. (1999). "General-Purpose System for Effective Construction Simulation". *Journal of Construction Engineering and Management*. ASCE, 125(4), 265-276

Pidd, M. (1998). *Computer simulation in management science*, Wiley, Chichester, U.K.

Rokni, S. (2009). "Optimization of Industrial Shop Scheduling Using Simulation and Fuzzy Logic". PhD dissertation, University of Alberta, Edmonton, Alberta.

Sadeghi, N., and Fayek, A. R. (2008). "A framework for simulating industrial construction processes." *Winter Simulation Conference*, 2396-2401.

Song, L., and AbouRizk, S. M. (2006). “Virtual shop model for experimental planning of steel fabrication projects.” *J. Comput. Civ. Eng.*, 20 (5), 308–316.

Wang, P., Mohamed, Y., AbouRizk, S.M., Rawa, A. R. T. (2009). "Flow Production of Pipe Spool Fabrication: Simulation to Support Implementation of Lean Technique." *Journal of Construction Engineering and Management*, ASCE, 135(10), 1027-1038.

Wang, Shihyi and Halpin, Daniel W. (2004). “Simulation experiment for improving construction processes.” *Proceedings of the 2004 Winter Simulation Conference*, 1252 – 1259.

Wyss, Stephen. (2009). “Active Management of Pipespool Fabricators.” *Chemical Engineering*, 116(1), 40-45.

Zhang, H., Tam, C. M., Li, H. (2005). "Activity Object-Oriented Simulation Strategy for Modeling Construction Operations." *Journal of Computing In Civil Engineering*, ASCE, 19(3), 313-322.

CHAPTER 3. A Dynamic Programming Solution to Automate Fabrication Sequencing of Pipe Spools

3.1 PROBLEM STATEMENT

A surging global demand and soaring prices of natural oil and gas put unprecedented pressure on existing oil production infrastructure in Alberta. This leads to a significant wave of capital investment into oil and gas projects. Many of these projects are referred to as 'Mega' industrial projects meaning that the capital investment required per project will exceed $1 billion (ECC 2007). Cost, however, is only one of the characteristics to describe the mega-ness of a petro-chemical project. Mega-projects usually imply that they are also technology-complex and have very long project execution period. For example, a typical project involves many trades and disciplines (e.g. civil, steel structure, piping, electrical, mechanical, HVAC, etc.), has extremely high level of engineering and construction activities, e.g. 3.5 million engineering man-hours and 15 million construction man-hours are required for a $2.5 billion project in Alberta (AEDA 2004). In addition, these activities are interwoven and tightly coupled with each other; a change or deviation in one activity can produce a cascade of effects on others that depend on it.

Piping is always the single largest job (Kim and Ibbs 1995) and is a critical and costly process for industrial construction projects (BRT 1982). Piping is usually broken up into three stages—shop fabrication, module assembly and site installation. This is mainly due to a fact that, considering the remote location, severe weather conditions and congestion issues, it is more efficient and effective to pre-fabricate and pre-assemble parts of project in a controlled environment (e.g. fabrication shops and module assembly yards) than on site. It has been reported that the proportion of shop fabrication has significantly increased over the past 20 years (Haas et al 2000). Figure 3-1 depicts different stages of the piping supply chain.

Pipe spools are building blocks to the piping system of industrial construction projects. They are fabricated in fabrication shops and are shipped to either module assembly yard (i.e. on-module piping) or construction site (i.e. off-module piping). Timely delivery of pipe spools has great impact on the successful execution of the overall industrial projects. The performance of pipe spool fabrication shops is, however, found not quite satisfactory (Howell and Ballard 1996, Tommelein 1998, and Wang et al. 2009). This is because pipe spool fabrication shops are very susceptible to various disruptions from within or outside the shop (Figure 3-1). For example, the supply of ISO drawings or raw materials could be late or out-of-sequence. Change orders from the owner or general contractor could generate a huge amount of rework. Rush orders from the assembly yard or the site can also cause deviations from the original shop schedules. In addition, most pipe spools have unique designs and need to be custom built (Wyss 2009). Pipe spools can be unique in material, shape, configuration, type of joints, and many other properties. As such, pipe spools cannot be entirely or partially fabricated in advance, which makes the fabricators unable to use on-hand inventory to buffer against the variability from within or outside the shop.

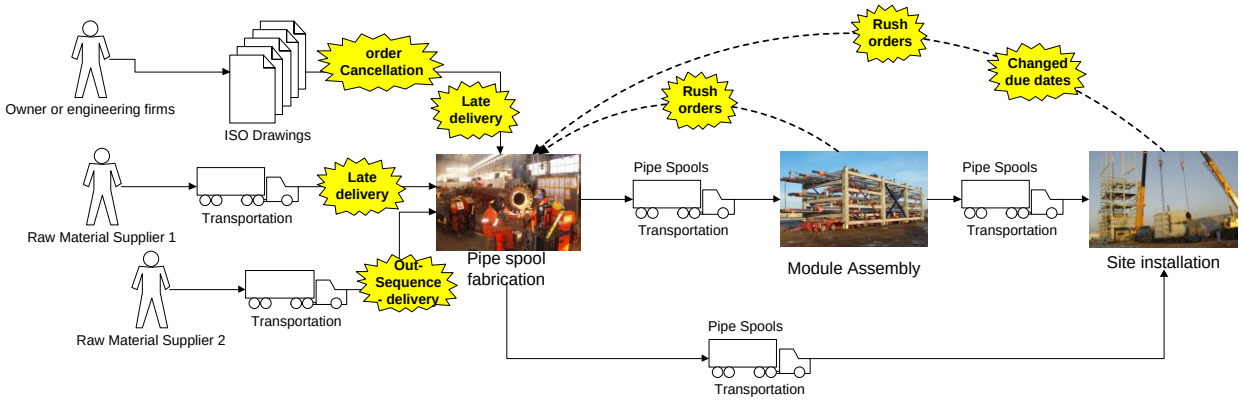


Figure 3-1 Disturbances from piping supply chain of an industrial construction project

To improve pipe spool fabrication shop performance, a number of innovative attempts (Tommelein 1998, Tommelein 2006, Walsh et al 2002, Wang et al 2009) have been made and

many factors have been investigated. These include shop layout, dispatching rules, buffer location, and standardized products. However, there is one factor that is overlooked, which is the sequence of pipe spool fabrication. The fabrication sequence determines the process that pipe spools go through from raw materials (i.e. raw pipes and pipe fittings) to the final product. Unique configurations cause the fabrication sequence to vary from one pipe spool to another. This is different from manufacturing where the majority of products has similar configurations and follows a few typical fabrication steps. Pipe spool fabrication sequence is usually determined by shop foremen based on personal experience and intuition. Interviews with shop foremen and superintendents show that the fabrication sequence for the same pipe spool could vary with human planners, because there is no standard way of sequencing in the industry. In fact, many pipe spools can be fabricated in several alternative sequences. However, it is rare for these alternative sequences to get compared and evaluated.

The magnitude of the impact of fabrication sequence could have on the fabrication performance has been investigated by Hu and Mohamed (2011). A simulation model is developed to represent the shop operations and pipe spools are with two alternative fabrication sequences (a good scenario and a bad scenario, but not necessarily the best and the worst). The total cycle time for all pipe spools are collected and compared (Figure 3-2). The results show there is 10.09% reduction in the total cycle time for all pipe spools and 16.88% decrease in the number of handlings during the fabrication.

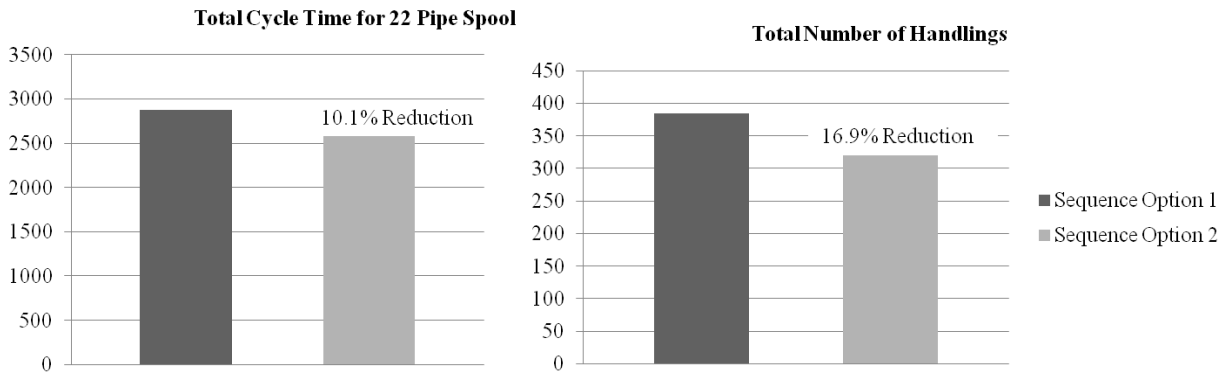


Figure 3-2 Simulation experiment result (Hu and Mohamed 2011)

To sequence pipe spool fabrication and especially to identify the optimal fabrication sequence is an issue that needs to be addressed. A review of literature on construction sequencing finds that many of the past research only focus on sequencing rationales of building projects. However, the building blocks for industrial construction projects (e.g. steel structures, equipment, pipe works, and cable trays) are very different from those for building projects (e.g. columns, beams, walls, slabs). This makes it hard to apply the existing sequencing rationales or planning systems to industrial construction projects.

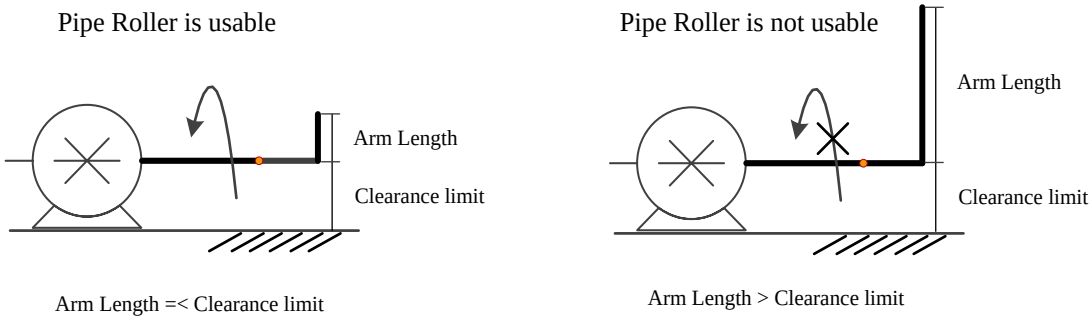
A search for problem solving techniques in computing science has found that artificial intelligence (AI) planning and dynamic programming (DP) seem to be good candidates for solving the pipe spool fabrication sequencing problem. The use of AI planning has been investigated in one of previous papers (Hu and Mohamed 2012). Conclusions include (1) AI planning suffices in handling all the logic of pipe spool fabrication; but (2) existing AI planners have limited parsing capability (to parse the system description with both numerical calculations and conditional effects which are required by the pipe spool fabrication problem) and make it difficult to solve real-life pipe spool problems. This chapter mainly focuses on exploring the applicability of DP.

In this chapter, a DP-based algorithm to identify the optimal fabrication sequences of pipe spools is presented. The chapter starts with description of pipe spool fabrication sequencing problem. Review of previous research on construction sequencing is provided in the next section and the gap in the body of knowledge is pointed out. A brief discussion of AI planning technique and its application on pipe spool fabrication problem. The main focus of the chapter is placed on the use of DP. First the pipe spool fabrication problem is formulated in DP manner. A DP-based algorithm is then proposed and simulation experiments are conducted to test its effectiveness. Finally, limitations and future research directions are also discussed.

3.2 PIPE SPOOL FABRICATION

Pipe spools are fabricated from a number of raw pipes and pipe fittings (e.g. elbows, flanges, tees, etc.) in fabrication shops. These raw pipe spool parts need to go through three major types of operations to become the final product, basically cutting, fitting and welding. Cutting always occurs at the beginning of the fabrication process and is only applied to raw pipes (Not pipe fittings). Generally, fitting precedes welding since fitting offers merely a temporary connection while welding forms a permanent one. However, more often than not, fitting and welding occurs in alternate manner. Specifically, some of pipe spool parts are fitted and welded first, which results in a sub-assembly (in-progress assembly, part of the final pipe spool). After welded, the sub-assembly is moved back to the original fitting table and gets fitted with other pipe spool parts. It then proceeds with welding to become another sub-assembly or the final product. This fitting-welding cycle ends when all pipe spool parts are welded. Pipe spool fitting and welding can be grouped into two types: (1) roll fitting and roll welding (Figure 3-3(a)) and; (2) position fitting and position welding (Figure 3-3(b)). Roll fitting and welding means the main pipe can be turned by a pipe roller and the fitter or the welder does not have to change his or her position to

perform the operation, whereas position fitting and welding occur when one or more branches of the main pipe exceed the clearance limit (Figure 3-3(b)). In such case, the fitter or the welder has to move around the main pipe to perform fitting or welding. As a result, position fitting and position welding usually takes more time to finish than roll fitting and roll welding. To minimize number of position fitting and welding is one of the objectives of pipe spool fabrication sequencing.



(a) Roll welding (b) Position welding

Figure 3-3 Roll welding and position welding (Hu and Mohamed 2012)

3.3 PIPE SPOOL FABRICATION SEQUENCING

The fabrication sequence defines the process of how a pipe spool will be fabricated gradually from raw materials (e.g. pipes and fittings), to intermediate sub-assemblies, and eventually to the final product. A pipe spool, in many cases, can be fabricated through a number of alternative sequences. Figure 3-4 shows such an example. The pipe spool can be fabricated at least by two different sequences from the same raw materials. Fabrication sequence1 necessitates two position welding when it fabricates sub-assembly 7, sub-assembly 8 and spool part 4 since the rolling axis would be axis Z and the longest arm length would be L (2.7m) which is larger than rolling clearance (1.5m) (Figure 3-4). Fabrication sequence 2, on the other hand, requires no

position welding (all spool parts can be roll welded). Another difference between these two sequences is that sequence 1 takes three fabrication steps to finish the fabrication while sequence 1 only needs two steps. The reason is that although the fabrication in step 2 and in step 3 are on the same axis Z they require different type of welding and thus cannot be completed in the same step. More fabrication steps also imply more handling needed between these steps, which further deteriorate the shop performance. It can be concluded, without further analysis, that sequence 2 will outperform sequence 1.

Likewise, most of real-life pipe spools have a number of alternative fabrication sequences. However, sequence is usually determined by shop foremen in very heuristic manner and these alternative sequences seldom have a chance to be compared and evaluated. As a result, opportunities of productivity improvement slip away. The root cause is attributed to the fact that currently there is no standard, structured way to identify sequence for pipe spools in the industry, and neither has any academic research on this specific topic been found.

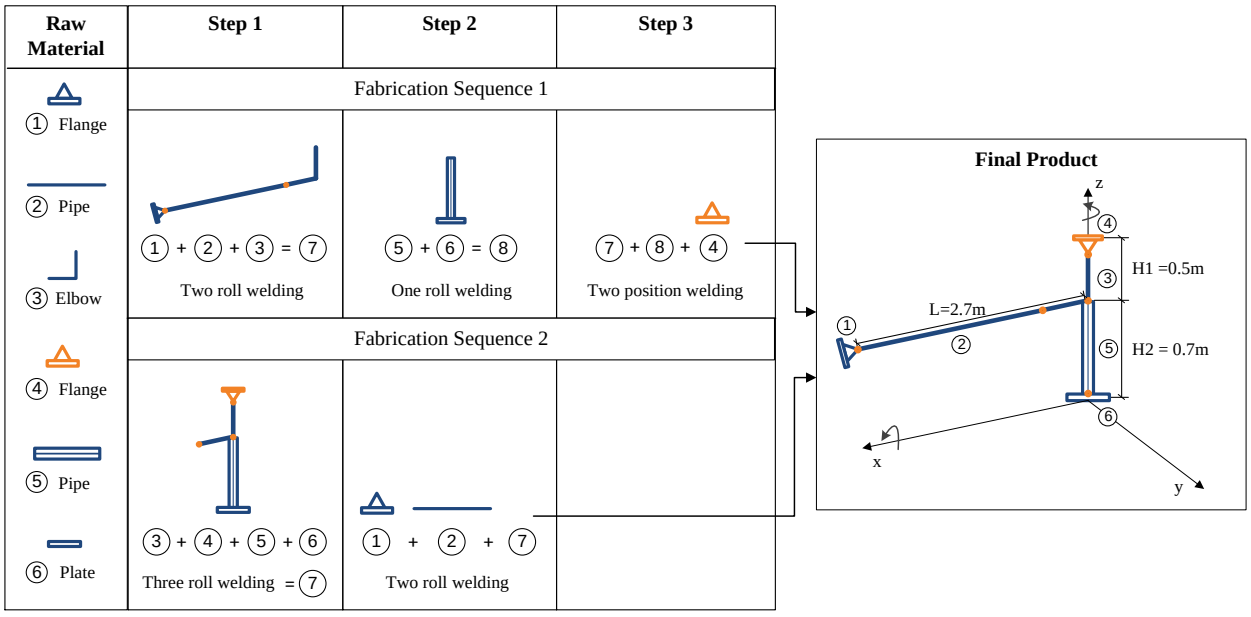


Figure 3-4 Alternative pipe spool fabrication sequences (Hu and Mohamed 2012)

3.4 PREVIOUS RESEARCH ON CONSTRUCTION SEQUENCING

Construction sequencing in this chapter refers to logic dependency between processes by considering the geometric and technological requirements. It should be distinguished from scheduling problems where process are sequenced and prioritized under limited resource availability in scheduling problems. A review of the previous relevant researches reveals that two major topics are: (1) identify and formalize construction sequence rationales (2) automate generation of construction sequences. Gray (1986) identified sequence rationales such as "covered by" or "weather protected by other components" that are generalized from different schedules generated by contractors. Kartam and Levitt (1989) consider "gravity support" and "enclosure" relationships as "general principles." Navinchandra et al. (1988) identifies similar dependencies as "supported by" and "connected to." Most of these sequence rationales are derived from the physical relationships between building components. Echeverry et al (1991) enrich the body of knowledge by adding three more factors that might govern the sequence of construction. Basically, they identify, in addition to "physical relationships among building components," "trade interaction," "path interference," and "code regulations." Other researchers have attempted to develop AI planning systems that capitalize on the existing construction sequencing rationales and automatically generate sequential dependencies between construction activities, for example, CONSTRUCTION-PLANEX (Hendrickson 1987), GHOST (Navinchandra et al. 1988), OARPLAN (Darwiche et al. 1989) and Construction Method Modeler (Aalami et al. 1998). Recent work by Koo et al. (2007) pointed out that much research on domain specific AI planning systems is focused more on identifying a correct construction sequence, than on discovering a number of possible sequence alternatives. They introduced a prototype system named "constraint-loaded CPM (CLCPM)," that makes use of constraint

ontology, and a classification mechanism to automatically assign "role" and "status" to relevant activities (i.e. to a target activity that needs re-sequencing) in CPM. These activities exert their "impact" on the target activity through "multiple network chains." Their "role" and "status" will be analyzed in the context of each related network chain and a final decision (whether or not can change sequence with the target activity) will be made based on pre-defined inference rules. Since all relevant activities will be evaluated, multiple re-sequencing alternatives might be generated.

The first observation from the previous research is that most of sequencing rationales are focused on building projects. These sequencing rationales are derived directly from the physical relationships between building components (e.g. columns, beams, walls and slabs). This makes it difficult to apply them to industrial construction projects, i.e. the building blocks are pipe spools and modules which most likely need to be pre-fabricated or pre-assembled before the final installation on site. The sequence constraints between these components are significantly different from those between building components. The second observation is that existing planning systems are mostly knowledge-based systems (KBS) which are highly domain-specific AI planning systems (e.g. building construction). Finally, the CLCPM system (Koo et al. 2007) even needs an existing schedule to infer re-sequence options.

Rokni (2009) developed heuristic search algorithm to identify the optimal sequence for pipe spools. However, the algorithm is a brute force search algorithm which needs to search through all the entire search space to find the optimal sequence. For example, if a pipe spool has *n* welding points, the number of possible solutions in the search space is n! (i.e. for a spool with 13 welds, it has 6,227,020,800 possible sequences). Although heuristics such as, determination of the type of welding by certain type of joint or assembly of small spool components first has

been built to reduce the search space, it is never proved in the paper that how effective these heuristics are for this purpose (i.e. the reduction rate in the search space). In addition, the benchmark to evaluate the algorithm is set up as how close the algorithm-returned sequence is to human planner's sequence, rather than how to further improve it.

3.5 ARTIFICIAL INTELLIGENCE (AI) PLANNING

Artificial Intelligence is a broad topic that consists of a variety of topcis such as knowledge representation, planning, machine learning, natural language processing, fuzzy logic, etc. Planning has been one of major research topics in artificial intelligence since 1960s (Newell and Simon 1972). It should be noted that AI planning in this chapter only refers to domain independent planning. AI planning is a process of selecting and sequencing a set of actions that change the system under study from an initial state to a desired goal state. This process starts with a general description of the system: (1) objects and states, a set of simple literals are used to describe the state of the objects that constitute the system; (2) actions, a finite number of actions are available to act upon the objects and to change their states. Each action comes with its own preconditions (must be true before applying the action) and effects (the resulting states after perform the action); (3) an initial state and a goal state, the main principle of selecting actions to perform is to reduce the difference between the initial state and the goal state so that eventually the system can achieve the goal from the initial state; (4) evaluation criteria, which is used to compare alternative plans and to find the optimal one. This description is then fed to a generic planning engine (called planner) which returns a sequence of actions (or plan) to achieve the goal. Particular modeling languages (e.g. STRIPS or ADL) are used to generate such a description. Planning Domain Description Language (PDDL) is very popular recently. It was first developed by Drew McDermott (1998). Since then, it has evolved and refined through several versions. AI

planning has been successfully implemented in many domains such as robot navigation, manufacturability of machined parts, and emergency evacuation (Ghallab et al. 2004).

AI planning could be a good candidate to solve the pipe spool fabrication sequencing problem. First, each pipe spool deals with limited number of pipes, pipe fittings and welding points. Pipes and fittings can be all viewed as the same object type—pipe spool part and welding points are the other object type. Second, there are only four actions available to change the state of a pipe spool or its parts—roll fitting, roll welding, position fitting as well as position welding. This differs greatly from the other types of construction where a wide variety of objects (e.g. columns, beams, walls, footings, etc.) are involved and hundreds of actions are required to build these components. Third, the preconditions and effects of pipe spool fabrication are quite straightforward. For fitting, it only requires two spool parts with an un-fitted (not fitted yet) welding point and the result is that spool parts are connected (becomes a sub-assembly) with a fitted welding point. For welding, it needs a sub-assembly with a fitted welding point and the result is a finished sub-assembly and a welded welding point. This is a unique characteristic. Researchers (Levitt and Kunz 1987, Navinchandra 1988, Darwiche et al. 1989) found that it is quite difficult to formulate preconditions and effects for operations in other construction projects (e.g. building construction).

A number of experiments have been conducted to test the capability of AI planning to solve pipe spool fabrication problem. The details of these experiments can be referenced in a previous paper (Hu and Mohamed 2012). Experiments selected three popular planners: (1) Metric-FF (Hoffmann 2002); (2) LPRPG (Coles et al. 2008); (3) LPG (Gerevini and Serina 2002). Results, however, showed that these planners lack adequate parsing capability when the system description contains both numerical calculations and conditional effects which are required by

the pipe spool fabrication problem. A grounding process is needed in this case to release the conditions from the effects of actions but requires enumerating all possible situations in the action description. A side effect of the grounding process is that the number of actions defined in the domain file grows exponentially with the number of welds in the pipe spool. For example, a pipe spool has N welds and then 2N-1 actions need to be explicitly formulated (e.g. a pipe spool with 13 welds requires 4096 actions defined). It poses huge computation challenge to AI planners and makes it almost impossible to solve some moderately complex pipe spool problems. This finding led to a search for other problem solving techniques. Dynamic programming is found to be a fit candidate.

3.6 DYNAMIC PROGRAMMING

Dynamic programming (DP) is a generic, efficient approach to solve optimization problems that usually involves a succession of decision making process (Chinneck 2010). To implement DP, two key Characteristics are required for the target problem: (1) Overlapping sub-problems and (2) Optimal substructure. When a problem can be broken down into sub-problems whose solutions will be reused for a number of times, the problem is said to have overlapping sub-problems. On the other hand, optimal substructure means that the optimal solution of a problem can be constructed from the optimal solutions of its sub-problems (Farmer 2008).

DP solves a problem by decomposing it. The decomposition requires that the problem and all its sub-problem are expressed in a recursive form. Decomposition process stops when the sub-problem becomes so simple that its solution is rather easy to be found (Dreyfuse and Law 1977). From this point, DP adds to this simple problem a small part at a time so that the new sub-problem is more complicate than the previous sub-problem. The optimal solution to the new sub-

problem is usually constructed from the optimal solution to the previous sub-problem. This process continues until the new sub-problem is the original problem (Chinneck 2010). The sequence of this process is the reverse to the sequence of decomposition used in the first step. At the end, both the sequence of solving the problem and the utility of this solution can be returned as part of the optimal solution.

DP improves the computation efficiency by two reasons. Unlike brute force approaches of total enumeration, DP memorizes the solutions of sub-problems and avoids re-calculating them when they are needed again. Second, it rules out many inferior solutions as it progress through a succession of decision making. This often considerably reduces the search space required to reach the optimal solution.

DP differs from many other optimization algorithms which contain a universal algorithm that applies to every problem and user only need to feed in the specific numbers of a problem. To use DP, every problem needs to be custom formulated and involves innovative thinking.

3.6.1 Problem formulation in DP perspective

To use DP, it is necessary to formulate the pipe spool fabrication sequencing problem in such a way that it matches the important characteristics of DP. Assume that the cost for roll-welding is 1 and the cost for position-welding is 2. The problem of finding the minimum number of position welding sequence for spool fabrication can be converted to the problem of discovering the minimum cost sequence for spool fabrication. The numbers used here are merely to distinguish the position welding from roll welding in the algorithm and are not the actual representation of costs for roll welding and position welding.

Stages: The problem can be divided into stages. In the pipe spool problem, a pipe spool is composed of several pipe spool parts and welds. The minimum cost fabrication sequence for the whole pipe spool should also be the minimum cost sequence for any of its sub-assemblies. Thus, the problem at stage t is to find the minimum cost sequence to assemble a sub-assembly (could be the original pipe spool). Figure 3-5 shows a number of such stages for an example pipe spool. At stage 1, the objective is to find the minimum cost sequence for the entire pipe spool. To do so, it is necessary to know the minimum cost sequence to assemble sub-assembly 5, 6, 7, and 8 (1 and 4 are basic pipe spool parts and need no fabrication), since the cost to assemble the pipe spool is dependent on the cost for these sub-assemblies. At stage 2, the objective is to find the minimum cost sequence to weld sub-assembly 5, which in turn leads to finding the minimum costs to assemble sub-assembly 7 and sub-assembly 9. This process continues until a stage is reached (e.g. stage 3) where the sub-assembly is composed from two basic pipe spool parts and one welding point, where the cost can be immediately determined.

States: Each stage has a number of states. Figure 3-5 shows that the pipe spool can be assembled by welding spool part 1 and sub-assembly 5, or by welding sub-assembly 6 and sub-assembly 7, or by welding sub-assembly 8 and spool part 4. Likewise, stage 2 and stage 3 also have alternative fabrication sequences to choose from.

Decision at a stage: The decision refers to choosing the optimum fabrication sequence of all alternatives at one stage. Since it is impossible to make a decision at an early stage (i.e. the cost of many sub-assemblies are unknown), decisions are made from the last stage (stage 3 in Figure 3-5). It is easy to make a decision since there is only one way to fabricate the sub-assembly, and cost can be immediately determined. From this stage, one can work backwards to make decisions for previous stages.

Recursive value relationship: The first few functions and variables need to be defined:

- C(*k*): minimum cost to assemble sub-assembly *k*,
- C(*i*): minimum cost to assemble child sub-assembly *i* of sub-assembly *k*,
- C(*j*): minimum cost to assemble child sub-assembly *j* of sub-assembly *k*,
- W (*i*, *j*): cost to assemble *i* and *j* to produce sub-assembly *k*,
- The recursive value relationship is then: $C(k) = \min\{C(i) + C(j) + W(i, j)\}$.

Note that C(·) appears on both sides of the recursive relationship. The optimum at stage t depends on two things: the value of current fabrication decision W(i, j), and the value of a previously found optima C(i) and C(j).

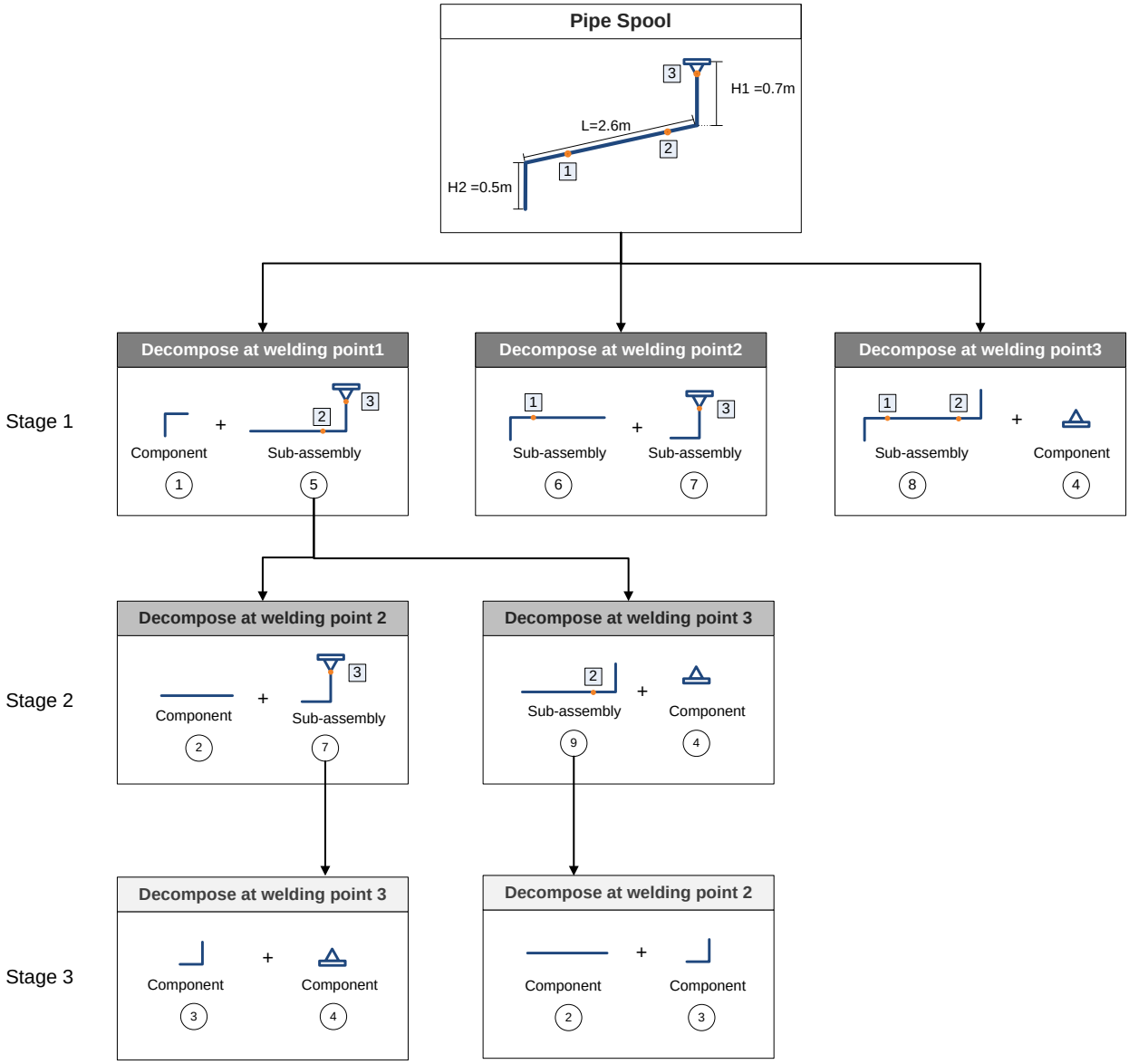


Figure 3-5 Decomposition stages of an example pipe spool

Use the pipe spool in Figure 3-5 as an example to clarify these concepts. Before analysis, the clearance limit is assumed to be 1.5 meters.

To determine the stages, the pipe spool should be subdivided until the sub-assemblies become so simple that their fabrication costs can be determined immediately. In other words, the pipe spool

should be sub-divided into sub-assemblies that only involve two pipe spool components and one welding point. Figure 3-5 gives an example to decompose the pipe spool.

Stage 1: Three alternatives are available since decomposition at each welding point can result in a new pair of sub-assemblies. All three alternatives require further decomposition since either one or two resulting sub-assemblies are not simple enough. Suppose C_0 represents the total cost to assemble the pipe spool while C_i represents the cost to assemble the *i*th (=5,…,12) sub-assembly. For basic pipe spool parts (e.g. 1, 2, 3 and 4 in round bracket of Figure 3-5), the cost to assemble them is considered 0.

$$C_0 = \text{Min}\begin{pmatrix} 0+C5+W1,5 \\ C6+C7+W6,7 \\ C8+0+W8,4 \end{pmatrix}$$

Stage 2: For space limit, take alternative 1 as an example (the same process should be applied to alternative 2 and 3). Only sub-assembly 5 needs further decomposition. It has two alternatives: decomposed at welding point 2, which results in spool part 2 and sub-assembly 7, or decomposed at welding point 3, which results in spool part 4 and sub-assembly 9. Either alternative needs further decomposition which leads to stage 3.

$$C_5 = \text{Min}\begin{pmatrix} 0+C7+W2,7 \\ C9+0+W9,4 \end{pmatrix}$$

Stage 3: This is the final stage where there is only way to fabricate both sub-assemblies (7 and 9), and the fabrication cost can be determined right away.

Sub-assembly 7 is composed from spool part 3 and spool part 4 (both have zero cost as they require no fabrication).

$$C_7 = C_3+C_4+W_{3,4} = W_{3,4}$$

Since the arm length of spool part 3 is less than 1.5 meters, sub-assembly 7 can be roll-welded. Thus,

$W_{3,4} = 1$

$C_7 = 1$

Sub-assembly 9 is composed from spool part 2 and part 3.

$C_9 = C_2 + C_3 + W_{2,3} = W_{2,3}$

Since H1 (0.7 m, the arm length of the elbow shown in Figure 3-5) is less than the clearance limit (1.5 m), sub-assembly 9 can also be roll-welded.

$W_{2,3} = 1$

$C_9 = 1$

Since the fabrication cost of lower level sub-assemblies have been determined, it is now possible to construct the solution in the bottom-up manner. The sequence is the reverse of the decomposition order.

Back to stage 2, sub-assembly 5 could be assembled from either spool part 2 and sub-assembly 7 or sub-assembly 9 and spool part 4.

$C_5 = \text{Min}\begin{pmatrix} 0+C7+W2,7 \\ C9+0+W9,4 \end{pmatrix}$

$W_{2,7}$ and $W_{9,4}$ should be determined by the dimensions of their constituent components and their relative position to the welding point. As Figure 3-6(a) shows, if sub-assembly 5 is assembled from spool part 2 and sub-assembly 7, it can be roll-welded, since H1 (0.7 m) is less than clearance limit (1.5 m).

$W_{2,7} = 1$

However, when sub-assembly 5 is assembled from sub-assembly 9 and spool part 4 (Figure 3-6(b)), it has to be position-welded, since L (2.1 m, the length of pipe shown in Figure 3-5) is more than the clearance limit (1.5 m).

$W_{9,4} = 2$

Therefore, the minimum fabrication cost for sub-assembly 5 can be determined and the optimal sequence for sub-assembly 5 is to fabricate pipe spool part 2 and sub-assembly 7.

$C_5 = \text{Min}\binom{0+1+1}{1+0+2} = \text{Min}\binom{2}{3} = 2$

Stage 1: In order to determine the minimum fabrication cost for the whole pipe spool, the same decomposition and solution construction processes should apply to the other two alternatives (i.e. from sub-assembly 6 and sub-assembly 7, or from sub-assembly 8 and spool part 4). The details are omitted here and results are shown below.

$C_6 = C_1 + C_2 + W_{1,2} = W_{1,2} = 1$

$C_7 = 1$ (has already been calculated above)

$C_8 = \text{Min}\binom{C6+0+W6,3}{0+C9+W3,9}$, Since $C_6 = C_9 = 1$ and $W_{6,9} = W_{3,9} = 1$

$C_8 = 2$

Finally, for the entire pipe spool,

$C_0 = \text{Min}\begin{pmatrix} 0+C5+W1,5 \\ C6+C7+W6,7 \\ C8+0+W8,4 \end{pmatrix}$

If the pipe spool is assembled from spool part 1 and sub-assembly 5 (alternative 1 shown in the Figure 3-7(a)), the rolling axis will be the axis X and the maximum rolling arm will be H1 (0.7 m). Since H1 (0.7 m) is less than the clearance limit (1.5 m), it can be performed by roll-welding. Thus,

$W_{1,5} = 1$

$C_5 + W_{1,5} = 2 + 1 = 3$

If the pipe spool is assembled from sub-assembly 6 and sub-assembly 7 (alternative 2 shown in the Figure 3-7(b)), the rolling axis will be the axis X again. The maximum rolling arm is still H1 (0.7 m < clearance limit 1.5 m). Thus,

$W_{6,7} = 1$

$C_6 + C_7 + W_{6,7} = 1 + 1 + 1 = 3$

If the pipe spool is assembled from sub-assembly 8 and spool part 4 (alternative 3 shown in the Figure 3-7(c)), the rolling axis will be the axis Z again. The maximum rolling arm will be L (2.6 m), which is longer than the clearance limit (1.5 m). A position-weld has to be performed in this case. Thus,

$W_{8,4} = 2$

$C_8 + W_{8,4} = 2 + 2 = 4$

Consequently,

$C_0 = \text{Min}\begin{pmatrix}3\\3\\4\end{pmatrix} = 3$

Therefore, there are two optimal fabrication sequences (alternative 1 and 2 in Figure 3-5) for the pipe spool, both of which incur the minimum fabrication cost.

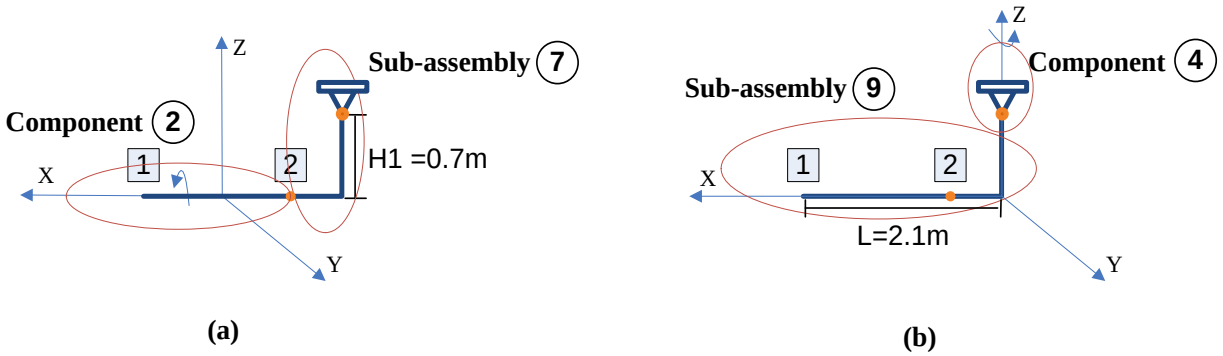
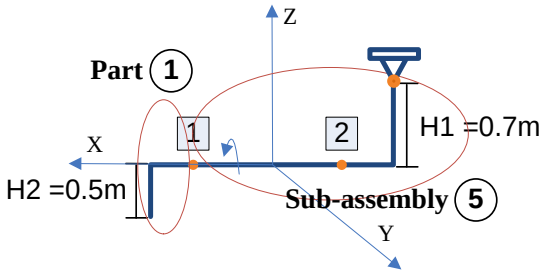
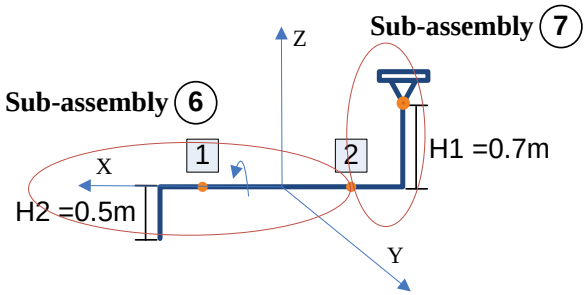


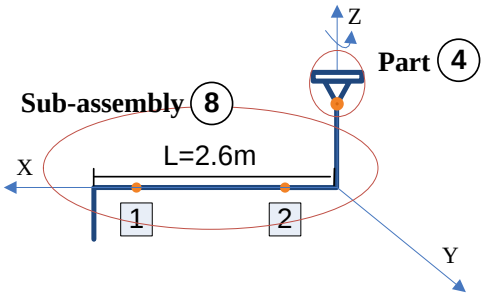
Figure 3-6 Two alternative ways of fabricating sub-assembly 5



(a) Alternative 1: Part 1 + Sub-assembly 5



(b) Alternative 2 :
Sub-assembly 6 + Sub-assembly 7



(c) Alternative 3 : Sub-assembly 8 + Part 4

Figure 3-7 Three alternative ways of fabricating the pipe spool

3.6.2 DP algorithm

Based on the DP formulation of the pipe spool fabrication sequencing problem, an algorithm (abstracted pseudo code shown in Figure 3-8) is developed to automatically identify the optimal sequence. The detailed implementation of the algorithm in Python is provided in appendix B.

1) ***Initialization***

- Initialize *coordinates* and *dimensions* of each pipe spool part ***P***
- Initialize *coordinates* of each welding point ***W*** and the *axis* that each of them sits on
- Initialize the *configuration* of the pipe spool (i.e. relationship between ***P*** and ***W***)
- Initialize the clearance limit (for roll welding)

2) find a ***Minimum Fabrication Cost*** for an sub-assembly ***A***

- If ***A*** is a pipe spool part
  - ***Minimum fabrication cost*** = 0
- If ***A*** is **simple** assembly (consists of only one welding point and two pipe parts)
  - ***Minimum fabrication cost*** = cost for fabricating p_i and p_j together (i.e. if roll-welding, cost = 1; if position welding, cost = 2)
- If ***A*** is **complex** (i.e. contains more than one welding points and more than two pipe spool parts)
- Initialize ***minimum fabrication cost (MFC)*** for ***A*** = the Maximum Integer Value)
  - For each welding point w_k in ***A***
    - Decompose ***A*** at w_k and results in two sub-assemblies $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$
    - Calculate the ***minimum fabrication cost*** for both $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$ (recursive call)
    - $\boldsymbol{mwc_k}$ = ***minimum fabrication cost*** for $\boldsymbol{A_1}$ +
      minimum fabrication cost for $\boldsymbol{A_2}$ +
      cost for fabricating $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$ together
    - If the $\boldsymbol{mwc_k}$ < ***MFC*** (i.e. obtained from all the previous decomposition)
      MFC = $\boldsymbol{mwc_k}$
      Record the welding point w_k
      Record the optimal fabrication sequence for $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$
    - Next welding point
- Record the welding point *w* that achieves ***MFC***
- Record the optimal fabrication sequence for ***A***

Figure 3-8 DP-based pipe spool fabrication sequencing algorithm

3.7 SIMULATION EXPERIMENTS

In order to test the effectiveness of the DP algorithm, two simulation experiments are conducted to prove if the fabrication sequences generated from the DP algorithm can really improve the shop performance. Unlike the previous simulation experiment [12], the fabrication sequences used here are the ones generated from the DP program, and ones that are generated by shop foremen which were actually used in one of Edmonton-based KBR fabrication shops. Figure 3-9 shows the general process of the simulation experiments. Two sets of sequences are input into

the same simulation model and results are collected and compared. Major performance metrics include: (1) the number of position welds, (2) the total cycle time, and (3) the number of handlings. The difference between the two experiments is that experiment 1 focuses on all 29 pipe spools, while experiment 2 only focuses on 19 of the total 29 pipe spools. These are the ones that have variations between the original sequences and the DP sequences.

3.7.1 Pipe spool set

29 pipe spools are used in the simulation experiment. They are real-life pipe spools that have been fabricated in one of KBR fabrication shop. They are selected because their fabrications have been tracked from the beginning (i.e. when they are issued to the shop) to the end (i.e. when the fabrication is completed). All processes (cutting, fitting, and welding) are recorded in time study sheets, including their sequences. These sequences are referred to as Original Sequences in Figure 3-9. In addition, they were all collected in the same quarter of the year and thus are quite representative of real-life work load of the fabrication shop. Figure 3-10 shows the composition of this pipe spool set in terms of pipe spool complexity.

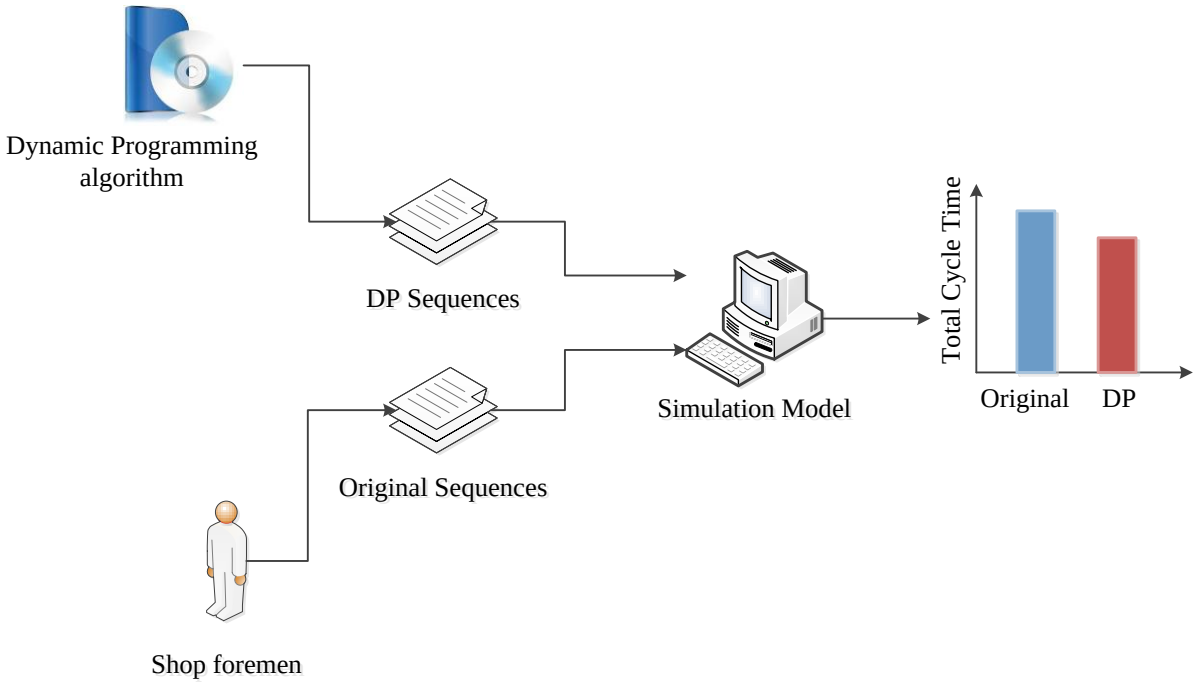


Figure 3-9 Simulation experiment

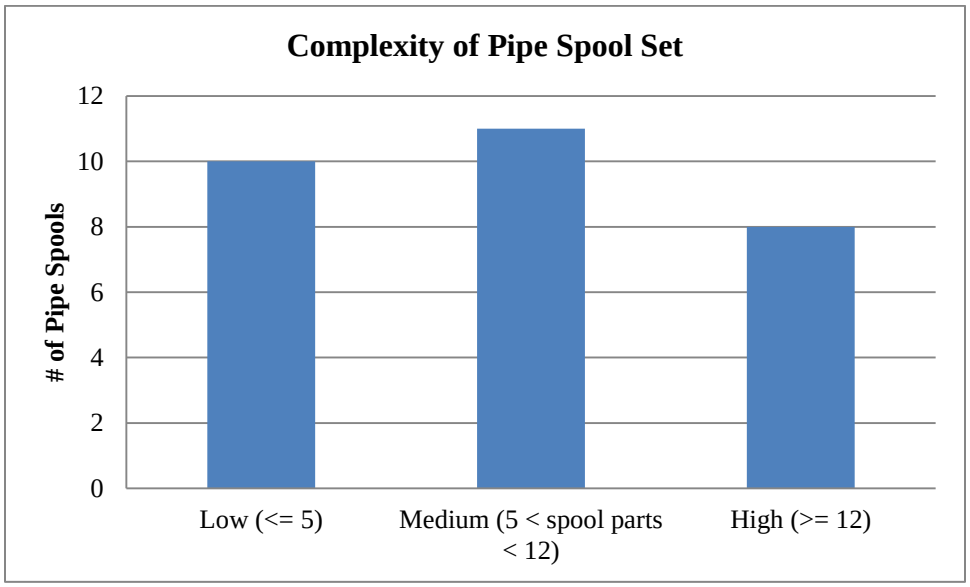


Figure 3-10 Complexity of pipe spool set

3.7.2 Experiment 1 results

The results of experiment 1 show that the total number of position welds has been reduced from 20 to 11 (Figure 3-11(b)). Considering that the total number of welding points for all 29 pipe

spools is 203, the improvement percentage is around 4.4%. This generally matches the total cycle time result, which is about 4.8% reduction (Figure 3-11(a)). The total number of handlings, however, actually increased by 2.5% (Figure 3-11 (c)). This can be explained in that the number of handlings is not considered as one of the optimization objectives in the DP algorithm. The increased number of handlings definitely offsets the improvement that is gained from the reduction in position welding.

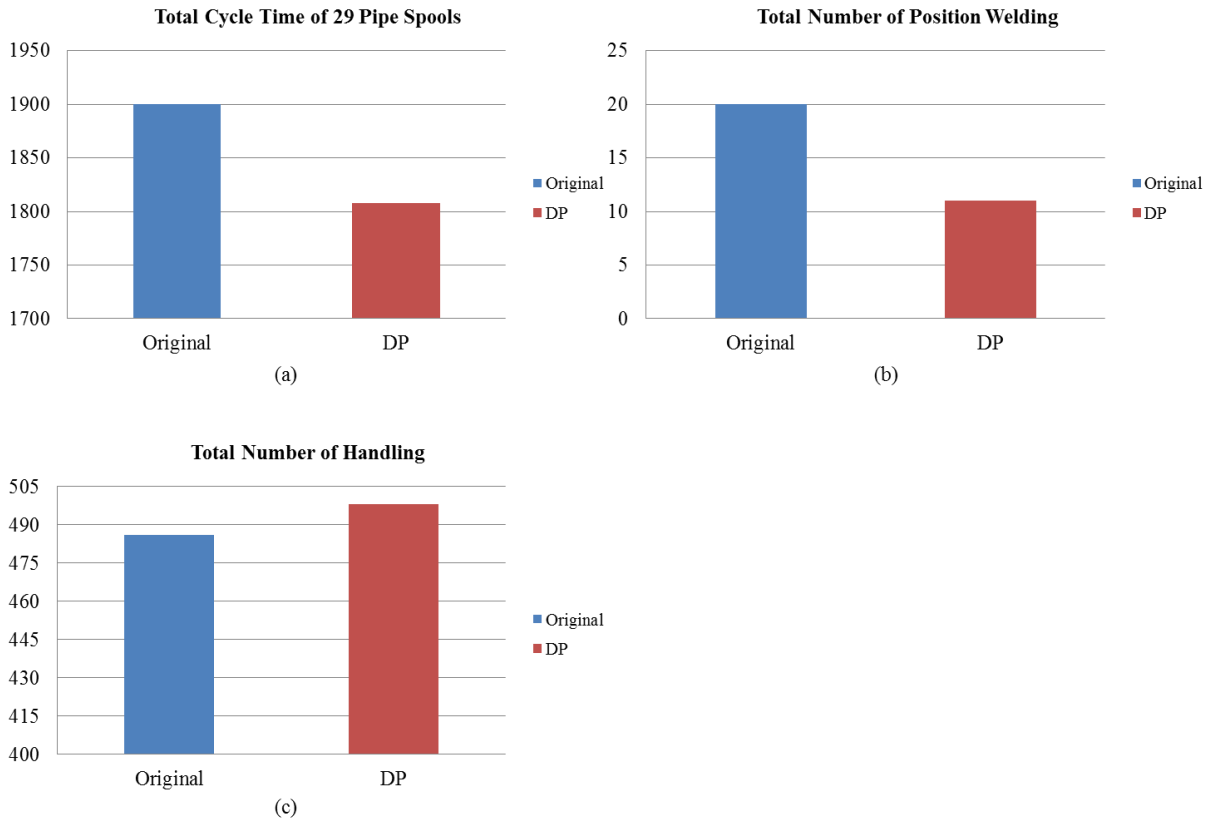


Figure 3-11 Results of experiment 1 (including 29 pipe spools)

3.7.3 Experiment 2 results

Experiment 2 focuses on the pipe spools that have different sequences (from the original sequences) after using the DP algorithm. Figure 3-12 (b) shows that the decrease in position welding is still 9. This is the same as in experiment 1. However, the improvement in terms of the

total cycle time is increased to 12% (Figure 3-12 (a)). The total number of handlings is increased 3.1% compared to the original sequences (Figure 3-12(c)).

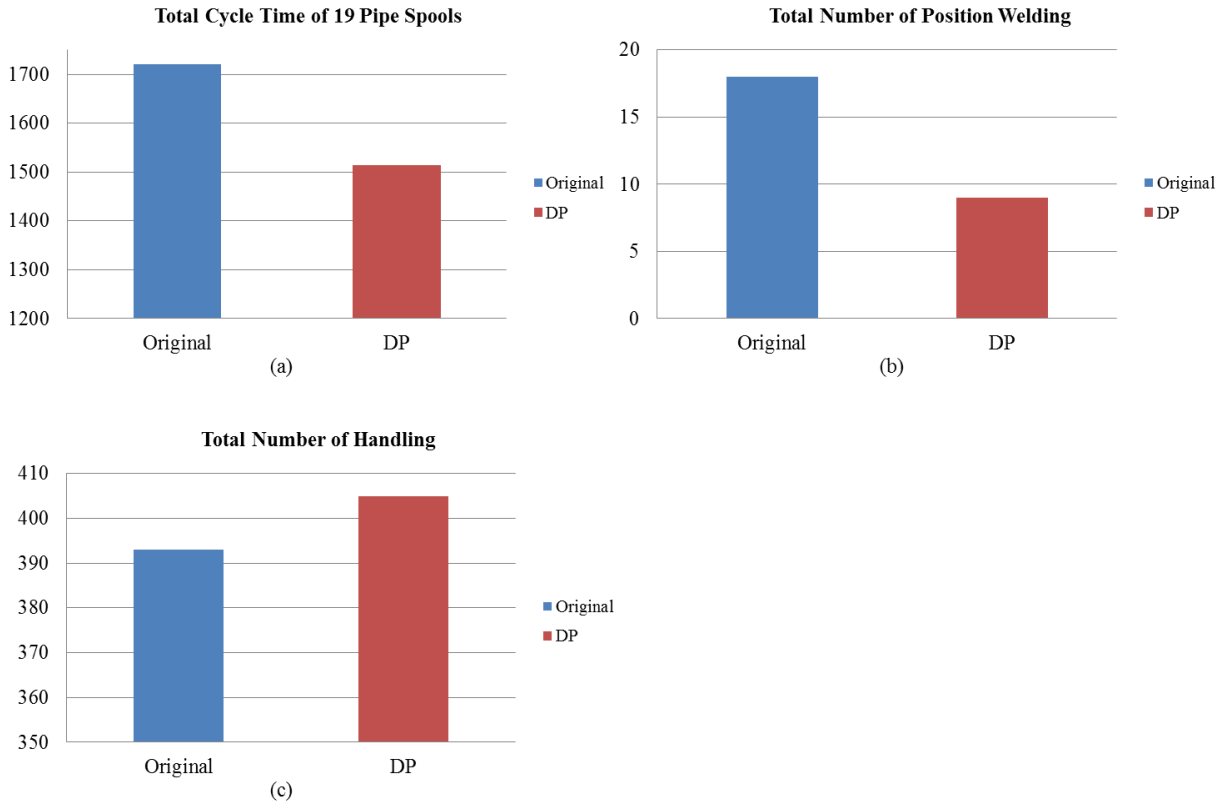


Figure 3-12 Results of experiment 2 (including 19 pipe spools)

3.7.4 Discussion of experiment results

Both experiments show that the DP algorithm is effective in minimizing the position welding (or position fitting), which results in a significant reduction in the cycle time of fabricating pipe spools. Given the enormous number of pipe spools involved in an industrial project and the fast-tracking nature of the project, it will not only save the fabrication shop a considerable amount in dollar value, but will also enhance its responsiveness to the changing project environment. However, there are several issues that need to be addressed in future research. First, the number of handlings should be incorporated as part of optimization objectives in the DP algorithm.

Handling is a type of non-value-adding activity and should be minimized. It is also observed that although in most cases including 29 pipe spools, the program returns the sequence in a few seconds; it took the program more than one hour to calculate the optimal sequence for an extremely complicated pipe spool which has 17 spool parts and 16 welding points. This means that the program itself needs to be optimized so that it meets the efficiency requirement for real-life fabrication shops. Integration is another issue that needs to be addressed. Since the DP program needs intensive information about the dimensions of pipe spool parts, the coordinates of welding points, as well as the configurations of pipe spools, it is critical for the program to be able to pull this information from existing drafting software. Manual input should be minimized so that input errors can be minimized.

3.8 CONCLUSION

Fabrication sequence is one of the key factors that impact pipe spool shop performance. Currently, this sequence is determined by human planners in a very heuristic manner, and thus, is not guaranteed to be optimal. Potential performance enhancement can be obtained through automation of this decision-making process through advanced problem solving techniques. This chapter investigated the use of dynamic programming. DP offers a way of decomposing and solving the problem, and leaves great flexibility to build in pipe spool fabrication logic and to perform any type of numerical calculations. A DP algorithm is customized for the pipe spool sequencing problem. Simulation is used to test the effectiveness of this DP algorithm and results show that the program is able to reduce the number of position welds as well as to shorten the pipe spool cycle time. Future work is focused on embedding more pipe spool fabrication heuristics into the DP algorithm to optimize more objectives in addition to the minimum number of postion welding.

3.9 REFERENCES

Aalami, F., Kunz, J., and Fischer, M. (1998). "Model-based sequencing mechanisms used to automate activity sequencing." *Working Paper No. 50*, CIFE, Stanford Univ., Stanford, Calif.

Alberta Economic Development Authority (2004). "Mega Project Excellence: Preparing for Alberta's Legacy - An Action Plan." available online <https://aeda.alberta.ca/AEDA%20Public%20Document%20Library/MegaProjectExcel_Dec102004.pdf> (Aug. 29, 2012).

BRT (1982). *Construction technologies needs and practices*. The Business Roundtable, Construction Industry Cost Effectiveness (CICE) Project Report B-3, New York, N.Y.

Chinneck, J. W. (2010), *Practical Optimization: A Gentle Introduction*, Carleton University, Ottawa, Canada, available online <http://www.sce.carleton.ca/faculty/chinneck/po.html> (Aug. 29, 2012).

Coles, A.I., Fox, M., Long, D. and Smith, A.J. (2008). "A Hybrid Relaxed Planning Graph-LP Heuristic for Numeric Planning Domains." *Proc., Eighteenth Int. Conf. on Automated Planning and Scheduling (ICAPS 08)*, Sydney, Australia, September.

Darwiche, A., Levitt, R., and Hayes-Roth, B. (1989). "OARPLAN: Generating Project Plans by Reasoning about Objects, Actions and Resources." *AI EDAM*, 2(3), 169-181.

Dreyfus, S. E. and Law, A. M. (1977), *The Art and Theory of Dynamic Programming*, Academic Press, Inc. New York.

Echeverry, D., Ibbs, C. W., and Kim, S. (1991). "Sequencing knowledge for construction scheduling." *J. Constr. Engrg. and Mgmt.*, ASCE, 117(1), 118–130.

Engineering and Construction Contracting (2007). "Tackling the Unique Challenges of Mega Projects." Panel Discussion, 39th Engineering and Construction Contracting (ECC) Conference, Colorado Springs, Colorado.

Farmer, J. (2008). "Introduction to Dynamic Programming." available online <http://20bits.com/article/introduction-to-dynamic-programming> (Aug. 29, 2012).

Gerevini, A. and Serina, I. (2002). "LPG: a Planner based on Local Search for Planning Graphs." *Proc., Sixth Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS'02)*, AAAI Press, Toulouse, France.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory and Practice*, Elsevier Inc. San Francisco.

Gray, C. (1986). "Intelligent construction time and cost analysis." *Journal of Construction Mgmt and Economics*, 4(2), 135–150.

Hendrickson, C, Zozaya-Gorostiza, C., Rehak, D., Baracco-Miller, E., and Lim, P. (1987). "Expert system for construction planning", *J. Comp. in Civ. Engrg.*, ASCE, 1(4), 253-269.

Hoffmann, J. (2002). "Extending FF to Numerical State Variables." *Proc., 15th European Conf. on Artificial Intelligence*, Lyon, France, July.

Howell, G.A. and Ballard, H.G. (1996). "Managing Uncertainty in the Piping Process." *RR 47-13*, Constr. Industry Institute, Univ. of Texas, Austin, TX, September, 103 pp.

Hu, D. and Mohamed, Y. (2011). "Effect of pipe spool sequence in industrial construction processes." *Proc., 3rd International/9th Construction Specialty Conf.*, CSCE, Ottawa, Ont., June.

Hu, D. and Mohamed, Y. (2012). “Pipe spool fabrication sequencing by automated planning.” *Proc., Construction Research Congress,* ASCE, West Lafayette, Indiana, USA, May.

Haas, C.T., O'Connor, J.T., Tucker, R.T., Eickmann, J.A. and Fagerlund, W.R. (2000*). Prefabrication and Preassembly Trends and Effects on the Construction Workforce*. Report No. 14, Center for Construction Industry Studies, Austin, TX.

Kartam, N., and Levitt, R. E. (1990) “Intelligent planning of construction projects”, *J. Computing in Civ. Engrg.*, ASCE, 4(2), 155–175.

Kim, J.J., and Ibbs, C.W. (1995). “Work-Package-Process Model for Piping Construction.” *Journal of Construction Engineering and Management*. 121(4), 381-387.

Koo, B., Fischer, M., and Kunz, J. (2007). “Formalization of construction sequencing rationale and classification mechanism to support rapid generation of sequencing alternatives.” *J. Computing in Civ. Engrg.*, 21(6), 423–433.

Levitt, R.E. and Kunz, J.C. (1987). “Using Artificial Intelligence techniques to Support Project Management.” *The Journal of Artificial Intelligence in Engineering, Design, Analysis and Manufacturing*, Vol. 1, No. 1, 3-24.

Navinchandra, D., Sriram, D., and Logcher, R. D. (1988). "GHOST: Project Network Generator." *J. Computing in Civ. Engrg.*, ASCE, 2(3), 239-254.

Newell, A., and Simon, H. (1972). *Human problem solving*. Prentice Hall, Englewood Cliffs, N.J.

Rokni, S. (2009). "Optimization of Industrial Shop Scheduling Using Simulation and Fuzzy Logic". PhD dissertation, University of Alberta, Edmonton, Alberta.

Tommelein, I.D. (1998). “Pull-driven Scheduling for Pipe-Spool Installation: Simulation of Lean Construction Technique.” ASCE, *Journal of Construction Engineering and Management*, 124 (4) 279-288.

Tommelein, I.D. (2006). “Process Benefits from Use of Standard Products - Simulation Experiments using the Pipe Spool Model.” *Proc., 14th Conference of the International Group for Lean Construction (IGLC14),* Santiago, Chile, 177-189.

Walsh, K.D., Hershauer, J.C., Walsh, T.A., Tommelein, I.D., and Sawhney, A. (2002). “Lead Time Reduction via Pre-Positioning of Inventory in an Industrial Construction Supply Chain.” *Proc., of the 2002 Winter Simulation Conference,* San Diego, CA, USA, 1737-1744.

Wang, P., Mohamed, Y., AbouRizk, S. M., and Rawa, A. R. T. (2009). “Flow Production of Pipe Spool Fabrication: Simulation to Support Implementation of Lean Technique.” *J. Constr. Engrg. and Mgmt.,* ASCE, 135(10), 1027-1038.

Wyss, Stephen. (2009). “Active Management of Pipe spool Fabricators.” *Chemical Engineering*, 116(1): 40-45.

Haas, C.T., O'Connor, J.T., Tucker, R.T., Eickmann, J.A. and Fagerlund, W.R. (2000*). Prefabrication and Preassembly Trends and Effects on the Construction Workforce*. Report No. 14, Center for Construction Industry Studies, Austin, TX.

CHAPTER 4. Congestion-Constrained Dynamic Resource Allocation Scheduling Tool for Industrial Construction Projects

4.1 PROBLEM STATEMENT

Industrial construction includes a wide range of projects such as petroleum refineries, oil and gas production facilities, petrochemical plants and power plants. Recently, industrial construction projects have seen a significant increase in both tighter schedules and fast-tracked engineering and construction, which leads to increased interference among work packages on-site. Schedules are too tight to allow one discipline to end their work prior to another discipline beginning theirs. Work packages often overlap rather than proceeding consecutively. This increases occurrences of resource over-allocation, where resource requirement exceeds availability. Decisions have to be made about how to allocate limited resources in order to best serve the overall project. Prioritizing work packages is common practice to solve resource over-allocation problems. Sometimes, planners allocate the resource to higher-priority work packages and delay lower-priority ones until the resource becomes available again. Other times, resources are released from work packages that are in progress and re-allocated to newly started work packages, if early completion of the new work packages is considered critical for the entire project. In this case, work packages with higher priority receive more resources than what they normally require, while work packages with lower priority end up with a reduced resource level. As a result, their durations are either shortened or extended. In reality, resource level assigned to work packages might be any value within a range, which is described by three (minimum, normal and maximum) levels. Furthermore, this resource level can fluctuate as the work package progresses.

Overlapped work packages also give rise to spatial interference and congestion. A petro-chemical project is literally a ‘steel maze’ that consists of pipe works, steel structures, cable trays,

vessels, tanks, pumps and a variety of equipment (Hammed 2009), all of which are compacted into a relatively small area. This dense jobsite could incorporate several million direct hours of work and several thousand skilled workers. Congestion is aggravated when two or more work packages are performed concurrently in the same area (due to overlap between work packages). Trade stacking is the congestion of two or more trades within a work area. It is quite common when the schedule is compressed. Since congestion diminishes productivity and increases the risk of incidents, it is necessary to impose congestion constraint on these areas so that the congestion situation can be controlled (i.e. defining the maximum number of people that can be present concurrently in a work area). This congestion constraint, in turn, affects the amount of resources (especially skilled workers) that could be allocated to work packages in the area (i.e. might be less than the amount that it normally requires). On the other hand, the congestion status of a work area changes dynamically over time while work packages start or are completed in that area.

In addition, resource limit is often not fixed at the same level throughout project duration. Resources employed in the on-site construction of industrial projects are often expensive. For example, cranes for lifting various pre-assembled modules, or skilled workers, due to the remote locations of these projects. Resource level (i.e. especially manpower) usually starts at a relatively low level, peaks somewhere in the middle of the project and gradually declines towards the end. The availability of resources can vary from one period to another and be at different levels during the project life cycle. In the industry, this is usually referred to as Time-dependent Resource Availability.

These aforementioned facts illustrate the issue that both resource availability level and resource utilization level can be variable over time, even in the middle of execution of work packages.

This variation definitely has impact on the execution of work packages as well as on their schedules. A challenge is posed then to project planners as to how to reflect the dynamic characteristics of resource allocation in the project schedule.

This chapter presents a congestion-constrained, dynamic resource allocation scheduling system (CDRASS) using time-stepped simulation technology. The objective is to develop a work package schedule with the shortest project duration, which is generated under a dynamic resource allocation mechanism. This is achieved by using time-stepped simulation to reflect possible variations in either resource availability or resource allocation in every time unit (i.e. could be hour, day or week depending on the users' requirement). The feasibility of the schedule is established by satisfying various hard constraints on work packages (e.g. dependency relationship, imposed dates, calendar and resource availability), as well as congestion constraint. Work packages can have durations that differ from the initial estimates, while keeping their dependency relationships valid.

The next section reviews the literature that is relevant to this research. Limitations of existing techniques are examined. The relevance of parallel scheduling scheme and time-stepped simulation is then discussed. Details of the simulation mechanism are elaborated afterwards. A case study from a real industrial project is used to illustrate the practicability of CDRASS. Results are also compared to those achieved by main stream commercial scheduling packages, and analysis of results is provided. Finally, conclusions and future improvements are given at the end.

4.2 LITERATURE REVIEW

This research draws on literature in two areas: (1) resource-constrained project scheduling, and (2) construction jobsite congestion.

Traditional critical-path-method- (CPM) based project scheduling techniques (e.g. PERT, ANO, AOA and PDM) are widely criticized for their unrealistic assumption that resources are unlimited. Recognition of this limitation motivated intensive research in regards to the resource-constrained project scheduling problem (RCPSP). There are two major topics in this domain: resource allocation and resource leveling. The former aims to find the shortest project duration within the resource availability constraints, while the latter seeks to reduce the fluctuation in resource usage with assumptions such as unlimited resource availability and fixed project duration (Hegazy 1999). Since this research is more closely related to the former topic, only literature related to resource allocation is reviewed herein. Many researchers have attempted to formulate the RCPSP as a mathematical programming problem using various optimization techniques such as linear programming, branch-and-bound, and enumerative branch-and-cut (Karshenas and Haber 1990, Demeulemeester and Herroelen 2002, Jiang and Shi 2005). These techniques are able to find a global optimal solution if the RCPSP problem is solvable. However, for most real-life projects, these techniques are computationally impractical (Moselhi and Lorterapong 1992, Hegazy 1999, Kim and Garza 2003, Lu and Li 2003). Another way to tackle the RCPSP problem is through heuristic techniques. Heuristics provide criteria for prioritizing concurrent work packages that are competing for the same resource. Commonly used heuristics include the least total float (LTF), the minimum latest finish (LFT), and resource scheduling method (RSM) (Davis and Patterson 2001). Shanmuganayagam (1990) proposed current float (CF) as a variation of LTF. Moselhi and Lorterapong (1992) devised another heuristic technique,

'least impact', which allocates resources to a set of activities (out of all feasible sets at a certain time), rather than to an individual activity, as in most heuristic techniques. Lu and Li (2003) proposed a new heuristic called 'work content' and claimed it has comparable performance with LTF in terms of finding the shortest project duration. A common feature of heuristic techniques is that they are easy to apply, involve less computational efforts, and thus are suitable for real-life projects. However, they can only offer a near-optimal solution and their effectiveness varies with different problems. Meta-heuristic-based project scheduling techniques become popular recently. These algorithms perform stochastic searches on populations of solutions which evolve over a number of iterations (Elbeltagi et al. 2005, Lu et al. 2008). Genetic algorithm (GA) has also been adopted to solve RCPSP problems (Chan et al. 1996, Hegazy 1999, Kandil and EI-Rayes 2006). Unlike mathematic and heuristic techniques, these algorithms are usually designed to achieve more than one objective simultaneously (e.g. the minimum project duration, the least resource utilization variation and the least cost). Kandil and EI-Rayes (2006) develop a GA-based algorithm to optimize time and cost as well as quality. GA has been used widely to optimize various construction problems, such as fleet configuration for earthmoving (Marzouk and Moselhi 2004), and location selection for crane lifting (Al-Hussein 2005). GA limitations have also been identified, e.g. long processing time and tendency to be trapped in local optima (Elbeltagi et al. 2005, Ng and Zhang 2008). This motivated researchers to explore other meta-heuristic techniques. Particle swarm optimization (PSO) and ant colony optimization (ACO) are the two algorithms recently introduced to the construction scheduling domain. Christodoulou (2007) first developed an ACO algorithm to address RCPSP problems and claimed that ACO has a structure and representation similar to traditional CPM networks, and thus, has less complexity to model RCPSP problems than GA and PSO. Ng and Zhang (2008) adopted ant colony system

(ACS), a variation of traditional ACO algorithm, to solve time and cost trade-off problems. By applying the ACS algorithm to an 18-activity construction project, they found that ACS significantly improves computational efficiency compared to the traditional ACO-based algorithm. The applicability of PSO to RCPSP problems was also investigated (Zhang et al. 2005 and Lu et al. 2008). In comparison to other meta-heuristic algorithms such as GA, it is found that PSO outperforms with less average deviations from the optimal solution. A similar conclusion was also reached by Elbeltagi et al. (2005).

An observation can be made that most previous RCPSP research is focused on finding more efficient and more effective scheduling optimization techniques. Scheduling optimization aside, they all assume that the work package can only start when all required resources are available throughout its duration and that, once captured, the resource levels stay constant for the duration of the work package. For example, if a piping work package requires a crew of 10 pipe fitters to perform the work, it is allowed to start only when there are 10 or more pipe fitters available. Otherwise, it would be simply delayed or interrupted until the precondition could be met. In theory, the resource level should remain at 10 pipe fitters throughout the work package's performance. In practice, however, the work package might be carried out even if there are only 8 pipe fitters available. The resource level allocated to a work package could be any value within the range (minimum, normal, and maximum). In addition, resource level could also change during the execution of the work package. For example, when a work area is congested, the number of skilled workers assigned to work packages that occur in this area might be reduced to alleviate the congestion. Or, skilled workers might be diverted from lower priority, in-progress work packages to higher priority ones, so that those work packages can be started. The dynamic characteristics of resource allocation, due to changing site conditions, are ignored in most

previous RCPSP research, although it may have substantial impact on work packages' performance and must be considered in the scheduling process.

Hegazy and Menesi (2010) proposed a new critical path analysis method called Critical Path Segment (CPS), which decompose activities into a group of segments based on days. This method converts the complicated precedence relationships (such as Start-to-Start and Finish-to-Finish) into Finish-to-Start. However, as for resource allocation, it holds the same deterministic view point (as mentioned above). Dynamic resource allocation is therefore not fully addressed.

The jobsite congestion issue has been approached mainly in two different ways: (1) space scheduling, and (2) productivity loss due to site congestion. The former takes space as one of the pre-conditions (as a resource constraint) to perform a work package and returns a space-loaded construction schedule (Thabet and Beliveau 1994, Thabet and Beliveau 1997, Riley and Sanvido 1997, Zouein and Tommelein 2001 and Akinci and Fischer 2002). A main goal of this line of research is to discover the mapping between space requirements of work packages and the physical space in 2D or 3D format, and then to detect and resolve the space conflict between work packages in close proximity. However, there is another type of congestion—overcrowding. It only reflects a degree of how crowded a work area is but not necessarily amounts to a space conflict. Jobsite crowding is believed to be one of the major causes of productivity loss and safety hazards (Ahuja and Nandakumar 1986, Dozzi and AbouRizk 1993, Thabet and Beliveau 1994, Ovararin and Popescu 2001). Many researchers attempted to quantify the impact of site congestion on crew productivity. Ovararin and Popescu (2001) used the frequency of having more than one crew working concurrently in the same area to represent the severity of congestion. Thomas and Smith (1990) reported that a skilled worker normally needs 19m^2 to perform a task, and productivity plunges to half when area per person shrinks to 10.4 m^2. Horner

and Talhouni (1995) suggested that productivity starts to reduce when area per person is lower than 28.3 m^2. Thebet and Beliveau (1994) created a productivity-space-capacity-factor (SCF) curve, which depicts how productivity declines as work space becomes increasingly congested. A similar curve was used in Dozzi and AbouRizk (1993) to illustrate loss in efficiency with percentage of crowding.

Few researchers attempted to incorporate the impact of jobsite overcrowding into the schedules of work packages. Zouein and Tommelein (2001) suggested the space congestion issue can be solved, in addition to delaying the start of a work package, by lowering the resource level of work packages, believing that the space requirement can be reduced as its resource level declines. Although recognizing the relationship between the resource level and the degree of congestion, they did not fully make use of dynamic resource allocation. Instead, they still assumed that the resource level, whether it is decreased or at the normal level, is determined at the start of a work package and stays constant throughout its duration. Thabet and Beliveau (1994) also recognized that work space crowding lengthens the duration of work packages and attempted to reflect this in the schedules of work packages. However, they suggested that instead of lowering resource level (as Zouein and Tommelein 2001), the production rate should be reduced, due to multiple work packages taking place concurrently in the same work area, and consequently, increased congestion. This means that work packages can still hold the normal amount of resources, regardless of how crowded the work area is, and only get penalized by decreased production rates. In reality, however, this is not the case, since overcrowding not only causes reduction in productivity, but also brings about safety hazards and should be restricted with a maximum limit.

A gap exists between the project-scheduling-related research and industry practice. Little attention has been paid to the variability of resource availability, resource need and resource

level of work packages. Instead, a deterministic point of view dominates most previous research. Meanwhile, jobsite overcrowding is seldom considered and integrated during the scheduling process. Zouein and Tommelein (2001) investigated the alleviation of congestion by varying the crew size required by work packages. However, they still held the similar deterministic view that crew size be decided before the start of a work package and should remain the same throughout the work packages' performance.

4.3 PARALLEL SCHEDULING SCHEME

The CDRASS system was developed using time-stepped discrete event simulation (DES) technology. Since scheduling optimization is not the focus of this chapter, heuristic technique is selected to implement in DES. Many meta-heuristic optimization techniques require extensive computing time for real-life problems, and thus, are not appropriate for simulations.

Heuristic-optimization-based resource constrained project scheduling has two components: a scheduling scheme and a priority rule (Kolisch 1996). Scheduling scheme can be categorized into two different modes: serial scheduling scheme and parallel scheduling scheme. Serial scheduling scheme determines a sequence of activities, ordered by their priority (based on whatever priority rule is used). This sequence of activities is then scheduled one at a time and at the earliest time when both dependency and resource availability constraints can be met. Parallel scheduling scheme, on the other hand, assigns and releases resources at every time unit. Likewise, at the beginning of each time unit, a list of eligible activities (those whose predecessors have been completed) is updated and ordered by priority. Activities that create no resource over-allocation are scheduled and others are delayed or interrupted. This procedure repeats until all activities are scheduled. The major difference between the two scheduling

schemes is that the serial mode releases resources only at the completion of activities while the parallel mode releases resources at the end of every time unit. This means that in parallel mode, activities are assigned with resources for the current time only. In the next time unit, the same activities may or may not be able to capture the same resources (or the same amount of resources) (Ahuja et al. 1994 and Lu and Li 2003).

If the serial scheduling scheme is adopted, it implies that resources must be available throughout the duration of an activity, and also, that the resource level must stay constant during this period. In contrast, the parallel scheduling scheme has no such implication. It allows resource level of activities to vary from time-to-time. Considering the flexibility that is required by dynamic resource allocation in this study, the parallel scheduling scheme is preferred over the serial scheduling scheme.

4.4 TIME-STEPPED DISCRETE EVENT SIMULATION

The heuristic resource allocation approach can be implemented and automated through the use of DES. Simulation has long been used to model and analyze various construction processes, with the objective of improving long-term performance. For the last two decades, it has increasingly been used for planning and scheduling day-to-day operations. It is usually referred to as simulation-based scheduling. Compared to traditional CPM method, DES is able to explicitly model resource interactions such that the resulting schedule is automatically leveled to the availability of resources. Another major advantage of using DES is that it provides a cost-effective laboratory environment where various alternatives can be tested and compared and the best one can be selected, without interrupting the real system. Many researchers have investigated the use of DES to solve construction planning and scheduling problems (Senior and

Halpin 1998, Martinez and Iaonnou 1997, Zhang et al 2002, Song and AbouRizk 2006, Mohamed et al. 2007, Hu and Mohamed 2010, and Taghaddos et al. 2012). Sadeghi and Fayek (2010) developed fuzzy discrete event simulation to develop a 'robust' schedule (insert time buffer proactively into the schedule) for construction projects. Sadeghi and Fayek (2012) further investigates the use of stochastic events to model start and finish of activities and to calculate critical path with highest expected value.

However, previous research depends heavily on the event-driven type of discrete event simulation (DES). Event-driven DES uses two events (start event and/or end event) to represent the beginning and the completion of a work package (Figure 4-1). Once the duration is sampled from a statistic distribution, the execution of the work package (including resource level) is determined and will stay the same throughout its duration. It then skips the interval between these two events. Time-stepped simulation is another type of DES where time advances in equal increments (Figure 4-1). At each step, the event list is checked to see if whether an event is scheduled to occur. If yes, the system state will be updated accordingly; otherwise, the simulation advances to the next time step and the system state remains unchanged. Future events are also scheduled in response to events occurring at the current time. This procedure continues until either there are no more events in the event list or a pre-defined time limit has been reached.

Generally speaking, event-driven simulation is much more efficient, from a computing-time point of view, than time-stepped simulation, as it skips the time intervals between events without checking the details. However, it also loses the capability of capturing changes that might happen during these intervals. Event-driven simulation is best suited for modeling a system with an accurate forecast of future events, and no occurrences between two scheduled events. The time-stepped DES is preferred over the event-driven DES to develop the CDRASS system because it

dissects the execution of a work package into subsequent segments and carries them out individually (one at each time unit). This level of granularity allows incorporation of schedule changes during work package execution and analysis of its possible effect on other work packages. Meanwhile, with a properly designed resource allocation algorithm, the adoption of the parallel scheduling scheme does not necessarily create any stoppage in the work packages' execution. Figure 4-1 offers a direct comparison between event-driven simulation and time-stepped simulation.

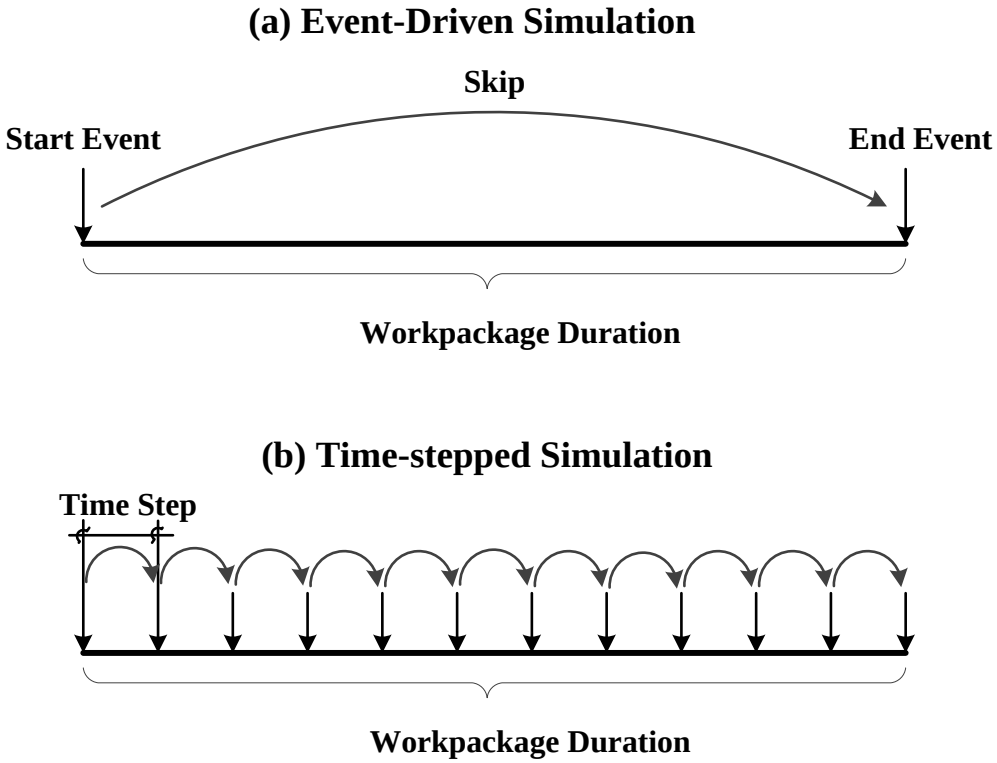


Figure 4-1 Event-driven simulation and time-stepped simulation

In the CDRASS system, simulation is designed to advance time in a 24/7 manner (i.e. 24 hours per day and 7 days per week). Each time step represents an hour, but could easily be scaled up or down (i.e. time step could be minutes, days, or weeks). The major reason to choose to advance time in a 24/7 manner is because calendars might vary from one work package to another (e.g.

piling work packages use the 10/5 calendar, i.e. 10 hours per day and 5 days per week, while steel structure work packages use the 8/6 calendar, i.e. 8 hours per day and 6 days per week). The 24/7 calendar offers a common foundation on which different calendars can be incorporated in the simulation model (Figure 4-2).

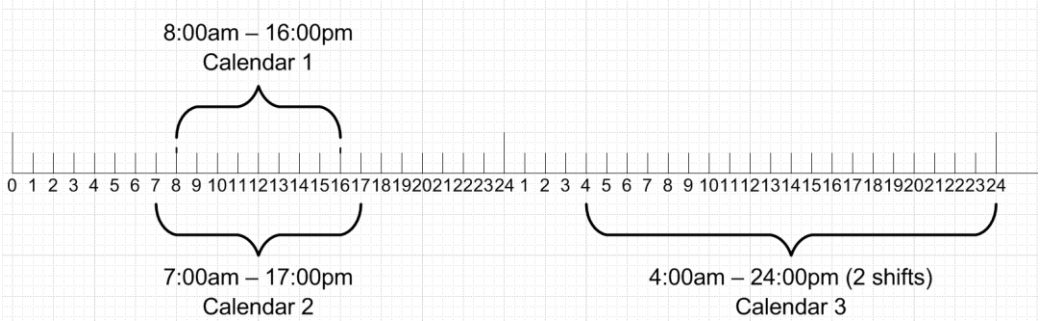


Figure 4-2 Simulation time advance in 24/7 manner

4.5 WORK-AREA-WORK-PACKAGE AS MOVING ENTITY

Project managers tend to overstaff the jobsite based on the notion that increased manpower levels can accelerate the schedule and increase productivity. However, after a certain optimum level, further increased manpower would do nothing but cause physical interference, as overlapping work packages also gives rise to the occurrence of jobsite overcrowding, which in turn causes productivity loss.

Congestion issues are associated with ‘work area,’ a concept that does not directly correspond to work packages. This is especially the case in industrial projects where many work packages are linear and cross a number of work areas, while each work area might involve more than one work package. Figure 4-3 shows a situation where several work packages take place concurrently in the same work area (e.g. WorkArea101) and a work package (e.g. WP2) can extend across

several work areas (e.g. WorkArea100, 101 and 102). As a result, the execution of WP2 should comply with both resource limit and congestion constraints of all three work areas that it crosses.

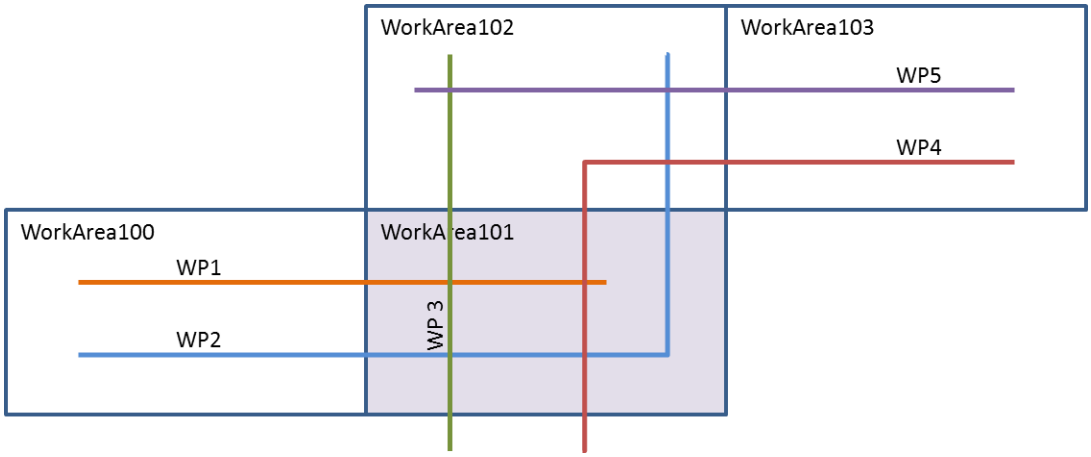


Figure 4-3 Work areas and congestion limit

The congestion constraint can be treated as a type of resource that is expressed as how many skilled workers can be present in the same work area. For example, assume that WP2 normally requires 10 skilled workers. Due to the congestion status of WorkArea101, however, it could only get work space to accommodate 8 skilled workers. This can be modeled as it captures 8 skilled workers from a certain trade and meanwhile takes work space for 8 skilled workers from WorkArea101. A dilemma arises when WorkArea100 and WorkArea102 differ from WorkArea101. For example, WorkArea100 allows 10 skilled workers while WorkArea102 can only accommodate 7 skilled workers, since WP2 also crosses these two work areas and should respect congestion constraints imposed in these areas. It is then inaccurate to assign 8 skilled workers to the entire work package WP2 (Figure 4-4a). When WP2 is being performed in WorkArea101, it does not involve congestion limits in WorkArea100 and WorkArea102, and vice versa.

Figure 4-4b presents a new way of implementing congestion constraint. Basically, work packages are further broken down into smaller elements called Work-Area-Work-Packages (WAWP). Each WAWP represents a portion of the work package in a specific work area. In this way, congestion constraint can act respectively on individual WAWPs instead of on the whole work package. In the simulation, WAWPs are treated as moving entities (not resource entities). Meanwhile, skilled workers represent limited and expensive resources, the use of which needs to be effectively planned. In this study, they are considered the resource entities.

Since work packages are already at the lowest level of project planning and scheduling, WAWPs need to be automatically created before the simulation commences. The creation of WAWPs can be viewed as part of the pre-simulation data processing. The details and assumptions about the creation process are described as follows:

(1) Definition of WAWP

In order to define WAWPs, it is necessary to know in which work area(s) a work package is going to take place. Each work package/work area pair defines a WAWP. A link is established between WAWP and its parent work package. In this way, WAWPs have access to the properties of their parent work packages, e.g. discipline, total man-hours, total quantity, unit of measurement, etc.

(2) Size of WAWP

It is assumed that the size of WAWPs is roughly derived from dividing the total man-hours (or the total quantity) of the parent work package by the number of work areas it crosses. This is merely an approximation, but accurate enough for the work-area-work-package level. For

example, sizes of piping WAWPs can be determined by the length of pipe spools or the number of ISOs in each work area.

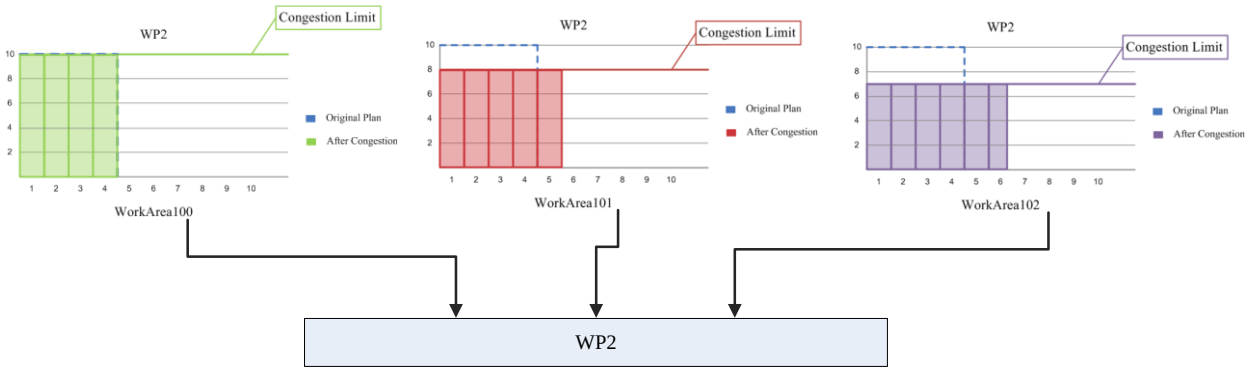
(3) Sequence of WAWP

WAWPs inherit all precedence relationships from parent work packages. For example, if work package A is a predecessor to work package B, all WAWPs of A (A1, A2, …, An) are predecessors to all WAWPs of B (B1, B2, …, Bn). It should be noted that excessive precedence dependency should be eliminated at this step. This will be elaborated in the case study section. Both the type of precedence relationship and the time lag remain the same. No sequence exists between WAWPs from the same work packages.

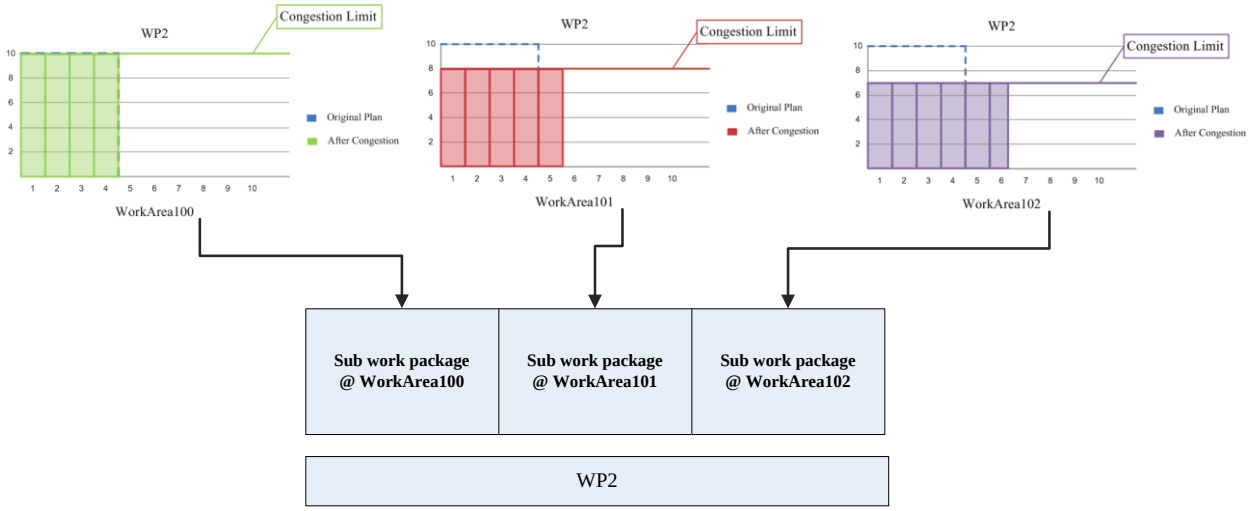
(4) Continuity of WAWP

It is assumed that each WAWP should be performed continuously once it has been started. However, on the work package level, children WAWPs are not necessarily performed in a continuous fashion. This assumption makes sense especially in the context of industrial projects, since stoppage between WAWPs can be interpreted as time that may be required to set up scaffolding in each work area when the work takes place in elevation.

Congestion constraint mainly affects the amount of resources that can be allocated to the work packages. A work package could be delayed if the congestion constraint cannot allow the minimum number of skilled workers (i.e. minimum manpower level), even though there are sufficient labor resources. Meanwhile, congestion constraint is a dynamic constraint that keeps changing over time as work packages start and finish in the work area. A work package might be allocated fewer resources when the area becomes increasingly congested, or vice versa. The resource allocation to a work package might change during its execution due to congestion status.



(a) Traditional way of implementing congestion limit



(b) WorkAreaWorkPackage

Figure 4-4 Work-area-work-package and congestion constraint

4.6 SIMULATION ALGORITHM

Time-stepped simulation needs to go through a routine procedure in every time step, which includes these typical steps, as shown in Figure 4-5: (1) if this is the beginning of the simulation, move to step 2 immediately; otherwise, update the progress for WAWPs that successfully captured resources in the last time step, release all the resources and update the resource availability limit at the current time; (2) identify eligible WAWPs (i.e. those that satisfy all prerequisites, e.g. drawings, materials, time constraints, as well as precedence dependency); (3)

prioritize eligible WAWPs and allocate resources to them considering the limited availability and congestion conditions in each work area; (4) advance the simulation to the next time step unless all WAWPs have been completed already or a pre-defined time limit has been reached.

Figure 4-5 Routine procedure for every time step

4.6.1 Identify eligible work packages

Work packages should not be released to the work face until all the preconditions have been met. It is necessary to check the eligibility of every work package before it can be scheduled. This also applies to WAWPs, each of which represents a portion of its parent work package. Various constraint checks are built into the simulation, including predecessor status check, time constraints check, and prerequisites check, as defined below.

(1) Predecessor WAWPs Check

It is the precedence relationships that drive the schedule to move forward. As predecessor WAWPs progress or are completed, the succeeding WAWPs become eligible to be executed. The CDRASS system incorporates all four types of precedence relationships: (1) start-to-start (SS); (2) start-to-finish (SF); (3) finish-to-start (FS); and (4) finish-to-finish (FF). Both positive and negative time lags are allowed to complement the four relationship types in order to model the logic relationship between WAWPs. More than one relationship can exist between a pair of WAWPs. A common example would be a WAWP that has both SS and FF relationships with its succeeding WAWPs.

Two predecessor WAWP checks are built into the simulation based on the type of precedence relationships. If it is related to the start of predecessors, such as SS and SF, then the WAWP is eligible if all predecessors have already been started (or have a start time), and the current time satisfies the lag or the lead. If it is related to the finish of predecessors, such as FS and FF, then the WAWP is eligible if all predecessors already have an estimated finish time and the current time satisfies the lag or the lead.

(2) Time Constraint Check

A work package would have certain time constraints which somewhat govern its start time or/and finish time. For example, early start constraint (cannot be started before a certain time) or late finish constraint (cannot be finished later than a certain time). This constraint also applies to the children WAWPs of this work package. Simulation converts the current simulation time into date-time format and compares with these constraints to determine the eligibility of WAWPs.

(3) Prerequisite Check

Material and drawing availability are the most frequent reasons to delay work packages. In fast-tracked industrial projects, construction and procurement starts way before the design is completed. Since scope is either undetermined or prone to change, the delivery of material and drawings becomes the major issue to hamper project progress. The start of WAWPs should be assigned to match delivery dates of required material and drawings.

Only WAWPs that have successfully passed all three checks are schedulable or eligible. Others will be sent back to the WAWP pool, and checked in future time units (Figure 4-6).

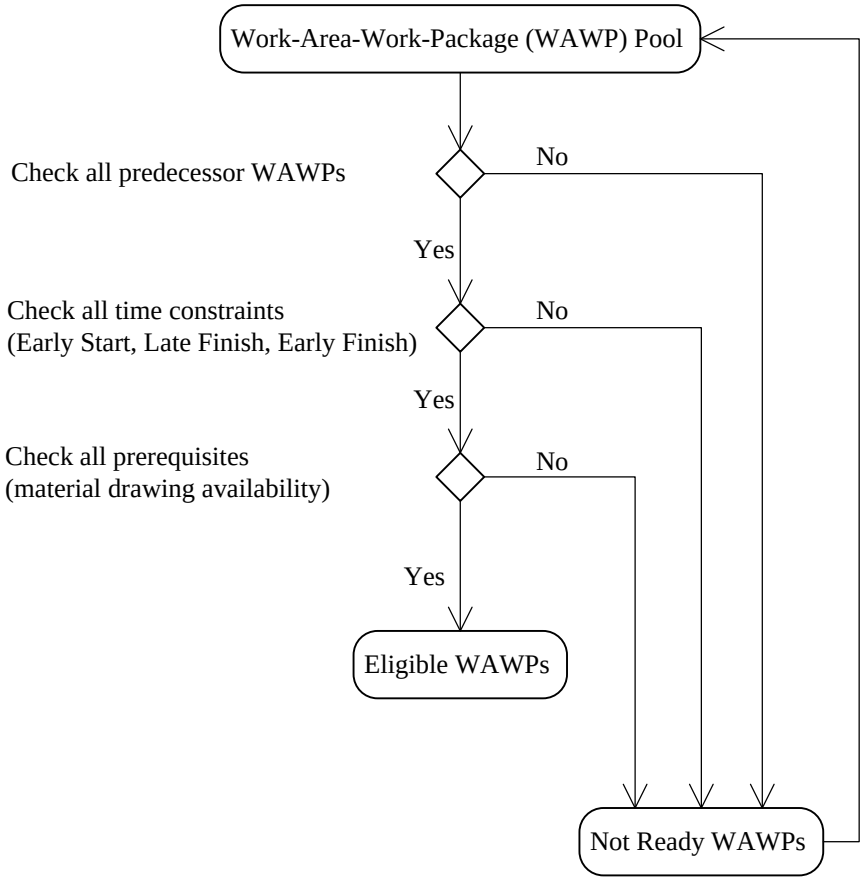


Figure 4-6 Various built-in constraint checks

4.6.2 Dynamic resource allocation algorithm

A two-round resource allocation algorithm is designed to allow for dynamic resource allocation while maintaining the continuity of WAWPs (shown in Figure 4-7).

First, all eligible WAWPs are divided into two categories: (1) in-progress WAWPs, and (2) newly-included WAWPs. The first round of resource allocation is active if the number of in-progress WAWPs is not equal to zero. It allocates resources to all in-progress WAWPs up to their respective minimum-manpower-level requirements. The purpose is to guarantee that all in-progress WAWPs can be continuously performed. However, the resource level may or may not be the same as in the previous time units.

After the first round is done, all WAWPs (in-progress and newly-added) are combined in a single list and sorted by their priority, given a particular priority rule (e.g. least total float, earliest late finish or current float). The second round of resource allocation is active if resources are not depleted in the first round. The second round adopts greedy first-fit criteria when allocating resources to work packages. This means that it attempts to fulfill the resource requirements of higher-priority WAWPs. The allocation is successful when the sum of resources that a WAWP has obtained from both rounds is more than the minimum manpower level. Otherwise, the WAWP has to be delayed. If resources are abundant, the resources allocated to a WAWP could sum up to the WAWP's normal resource level. In this case, the amount of resources captured in the second round is equal to the normal resource level minus the amount captured in the first round. For example, assume that a piping WAWP normally requires 10 pipe fitters and has already captured 8 pipe fitters in the first round of resource allocation. It can capture another two pipe fitters in the second round of resource allocation if two or more than two pipe fitters are still available. If resources are insufficient, all the remaining resources are allocated to this WAWP. For example, assume that there is only one pipe fitter left in the second round. The piping work package will still capture this pipe fitter and the total manpower level is 9 (less than the normal required amount of 10, but more than the minimum required amount of 8).

It should be noted that resource allocation is also constrained by congestion conditions of related work areas. The amount of resources that can be allocated to a WAWP is the minimum of both the availability of the resource it requires and the number of people that can be present in the work area. For example, if there are more than 10 pipe fitters available but the work area can only accommodate 8 people, then the maximum amount of resources that can be allocated to the piping WAWP is 8 (=min(10, 8)).

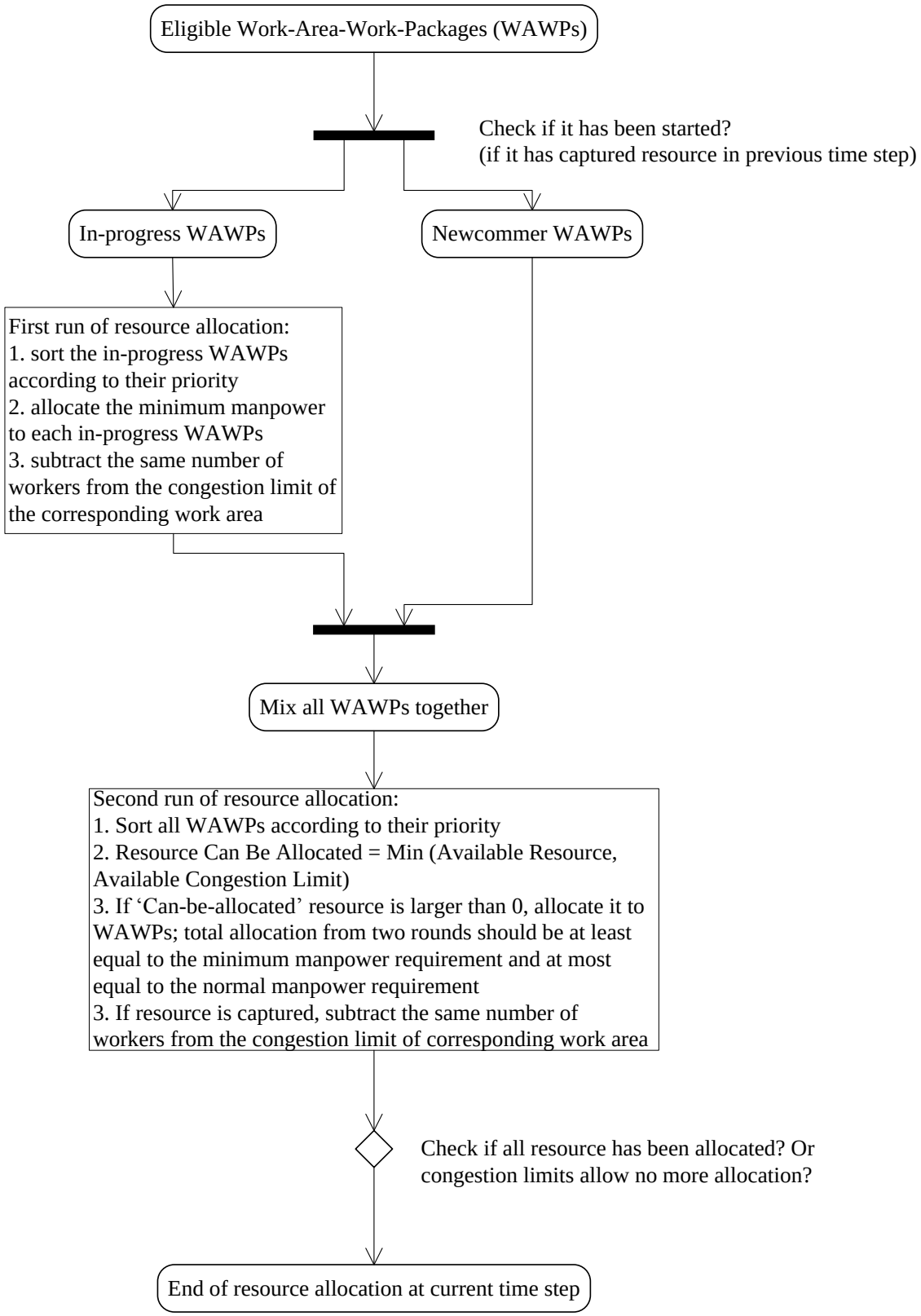
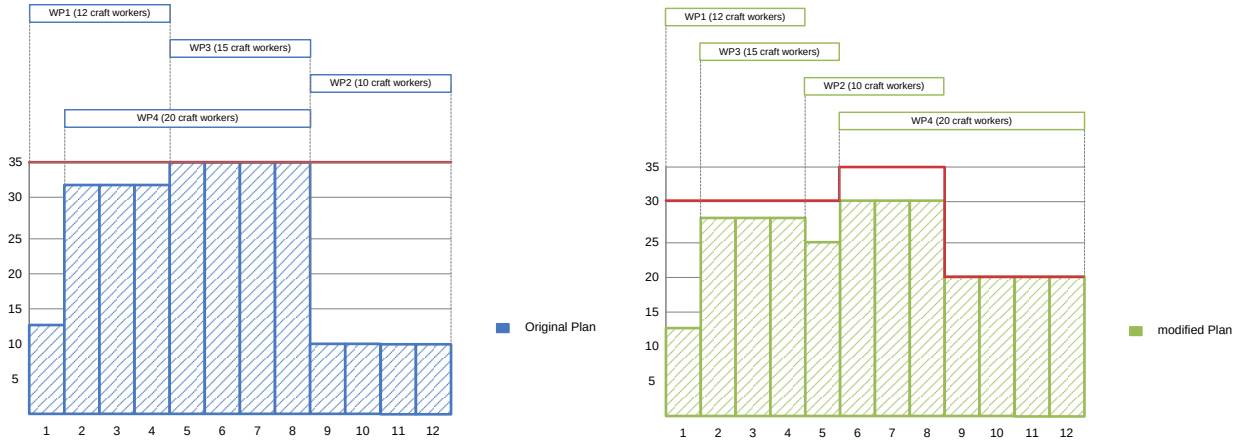


Figure 4-7 Dynamic resource allocation algorithm

4.6.3 Variable resource level and variable durations

Although WAWPs that are in progress are guaranteed to be performed continuously (no stoppage), their resource levels are not necessarily constant. Resource levels of a WAWP might change under a few circumstances. First, in cases where a higher-priority WAWP that requests the same type of resource is added to the eligible list, and it successfully captures resources in the second round of resource allocation, the resource level of the in-progress WAWP might be reduced. Likewise, in cases where a higher-priority WAWP that takes place in the same work area becomes schedulable and it manages to take some congestion resources in the second round, the resource level of the in-progress WAWP might as well be decreased. Another reason for change in WAWP resource levels could simply be a decrease in the availability of the resource (due to time-dependent resource limit, as shown in Figure 4-8). The resource level could also be increased when all aforementioned situations are inversed (i.e. completion of higher priority WAWP that requires the same resources or occurs in the same work area, or availability of the resource is increased). It should be noted that traditional scheduling tools such as MS Project also allow for variations in resource allocation, but the variation is restricted to either the normal resource requirement, or nothing. However, completely stopping work on a WAWP that is already started is very rare in real-life projects. Possible extra costs for protecting work-in-process, mobilizing equipment, double-handling materials and associated set-up time and moving time make complete interruption a very expensive option.



(a) Constant Resource Limit (b) Time dependent Resource Limit

Figure 4-8 Constant resource limit vs. time-dependent resource limit

Given the fixed work quantity of a WAWP, its duration is bound to change as the resource level changes. Traditionally, durations of work packages are treated as a constant number. Even in simulation-based scheduling, the duration of a work package is generated once a number is sampled from a probability distribution. This duration will not change at all as the work package progresses. However, this is no longer the case when resource level is allowed to vary during the execution of work packages or their children WAWPs. Thus, the original duration of a work package or a WAWP is merely an estimate and the actual simulation duration might differ.

Variable durations pose a challenge to maintaining the precedence dependency between WAWPs. This is the case especially when the dependency is related to the completion time of a predecessor WAWP. For example, assume that two WAWPs (WAWP1 and WAWP2) have a FF relationship. WAWP1 is the predecessor. If the duration of WAWP1 is somehow extended, the completion time for WAWP1 will of course be postponed. This means that the completion time for WAWP2 should be delayed accordingly, which, in turn, means the resource level of WAWP2

should be reduced. This adjustment mechanism is already built into the dynamic resource allocation algorithm.

4.6.4 Update the progress of work packages

All WAWPs that have successfully captured resources in the current simulation time are collected in a list. The progress of these WAWPs will be updated accordingly at the end of the current simulation time. For example, if a piping WAWP captures 8 pipe fitters, the man-hours that it has gained are equal to 8 times the time unit. If the time unit is an hour, 8 man-hours are gained. If the time unit is a day, then the total man-hours obtained depend on the calendar of the work package. Assuming that it uses the 10/5 calendar (10 hours per day and 5 days in a week), 80 man-hours are then obtained. If a WAWP has fulfilled the total man-hours, it will be added to the completed work package list. This means that the WAWP can no longer be included in the eligible WAWP list.

The end of the current simulation time is also the beginning of the next simulation time. Two important additional tasks need to be performed. First, all resources captured in the current simulation time should be released to replenish the resource pool. Second, the availability limit of each type of resource should be updated according to the pre-defined, time-dependent value.

4.7 SIMULATION EFFICIENCY AND CALENDAR CONSTRAINT

Since the time-stepped simulation advances time in a 24/7 manner, it will definitely require much more time than other event-driven simulations. The simulation efficiency is then an issue that has to be addressed. Most work packages or WAWPs will not be performed 24 hours per day or 7 days per week. There must be certain time units that are non-working time for all work packages.

A check is then performed at the beginning of each time unit to see if it is non-working time for all eligible WAWPs. If yes, then a standard procedure will be carried out (identify eligible work packages→allocate resources→advance simulation time→update the progress). Otherwise, this procedure is skipped and the simulation time is advanced one time unit forward. This will significantly reduce the computation required by the simulation, and thus, reduce the required time to run the simulation model.

4.8 SYSTEM ARCHITECTURE

The CDRASS system requires a variety of information from different information sources. The design is based on the idea that simulation runs as a scheduling engine in a behind-the-scene manner. It automatically reads data from the database, constructs and runs the simulation model, and eventually writes the generated schedule back to the database. Users do not have to get involved in model development. The simulation tool also interfaces with other information systems at a database level (Figure 4-9). For example, material and drawing availability information can be exchanged via a database between a material management system and a simulation database. As-built information is pulled from a progress tracking system so that simulation can start right from the current time instead of from the beginning of the project. Meanwhile, some, if not all, of the work packages can be extracted directly from the 3D model, depending on how much information is embedded. For example, various modules (e.g. pipe rack, building, substation, steel structure, etc.) can be easily identified in the 3D model and their physical location information (e.g. coordinates, dimensions) can also be obtained. It should be noted that not all data can be pulled from existing information systems and manual input is still required to populate some data tables (e.g. dependency relationships between work packages, man-hours of work packages, or time dependent availability limits of resources) in the simulation

database. This system has been developed to implement the proposed algorithms using Visual Basic.Net and Simphony.Net 4.0 (detailed implementation is provided in Appendix C). Simphony is a discrete-event simulation environment for construction projects that facilitates graphical, hierarchical and modular modeling (AbouRizk and Mohamed 2000). It has evolved through several versions concurrent with upgrades of the Microsoft.NET framework. The most recent version is Simphony.Net 4.0, which is based on the .NET 4.0 framework. Visual Studio 2010 is the main programming environment for Simphony.Net 4.0 and Microsoft Access 2007 is used as the main data base management system (DBMS).

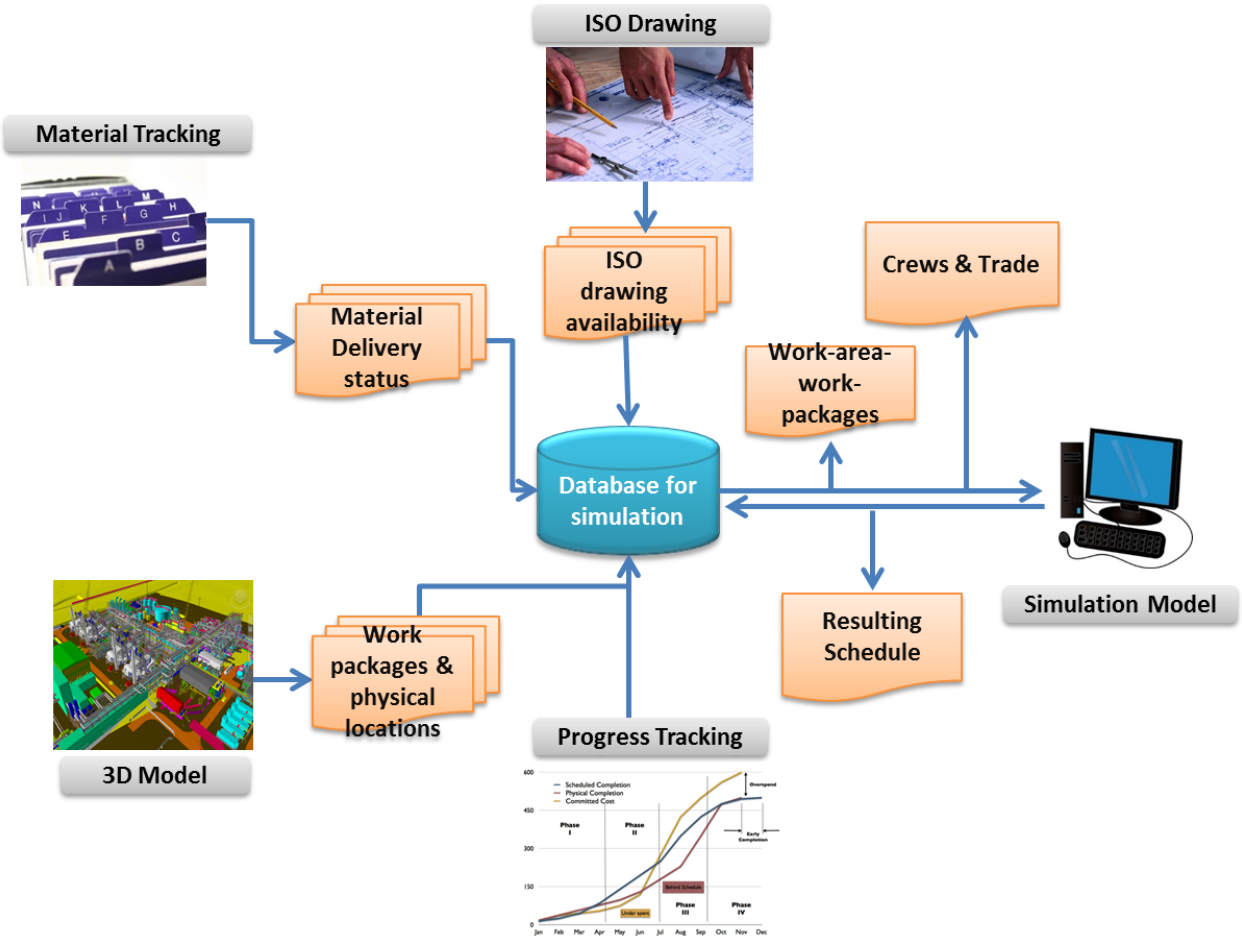


Figure 4-9 CDRASS system architecture

4.9 CASE STUDY

In order to demonstrate the practicality of the CDRASS system, a real case from PCL, a construction leader in Canada and the United States, is used. PCL industrial management Inc. is closely involved in industrial construction work and provides services from pipe spool pre-fabrication, module pre-assembly, to field construction. The Kearl Initial Development (KID) project was started by Imperial Oil Limited in 2009 and PCL is the general contractor on this project. This is the first building block within phased development, offering 110,000 barrels of bitumen processing capacity per day. The capital investment is approximately $10.9 billion. The project is a typical mega oil-sand project that is being constructed in the province of Alberta, Canada. The sheer size of the project is beyond illustrating for the purpose of the case study. As such, only a portion of pipe rack area is focused on here (the part that is marked by yellow left-right arrows in Figure 4-10).

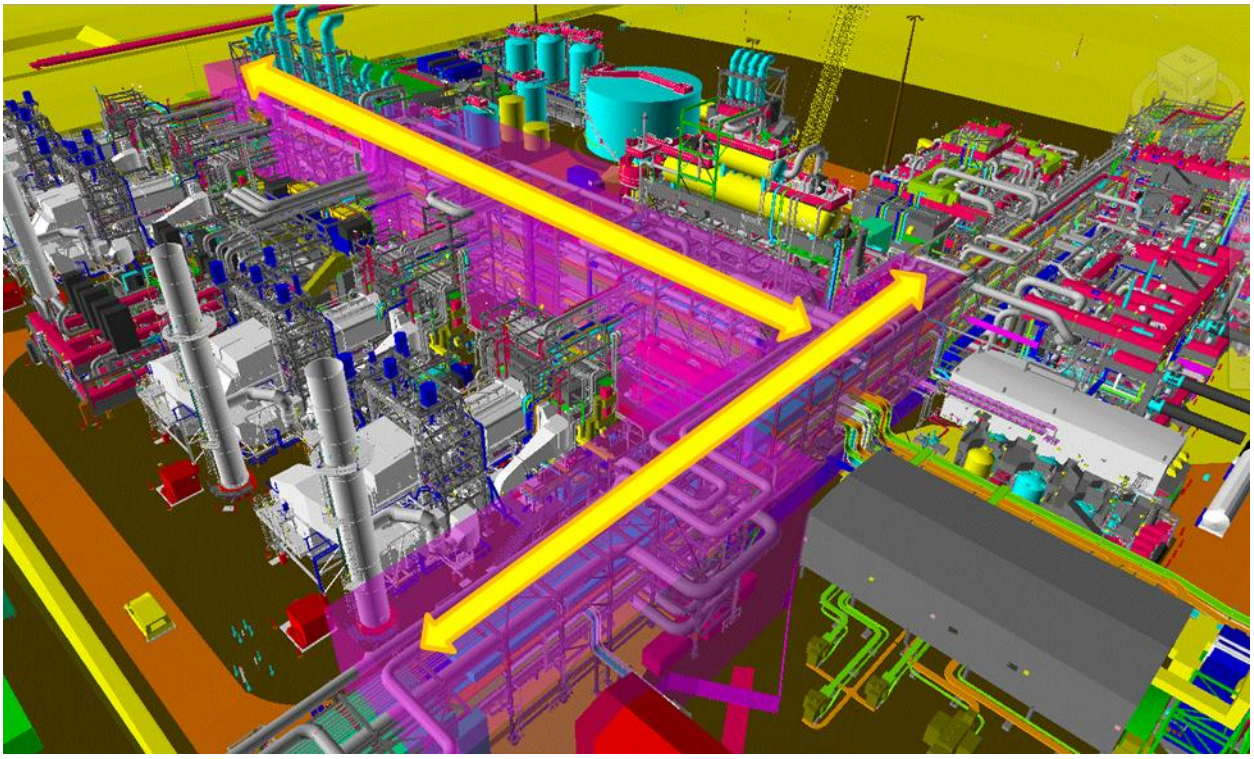


Figure 4-10 Pipe-rack area of Kearl Initial Development (KID) project

The major work in this area is to install a number of pipe-rack modules in a stacked configuration. These modules are pre-assembled in a module yard and shipped to the construction site for final installation. Other disciplines involved here include piling, support structure, piping (for interconnecting modules), silencer, hydro-testing, insulation, as well as electrical cable tray. 51 work packages are identified with quantity, work areas, dependency relationships, and resource requirements data, part of which is shown in Table 4-1 (the full list of work package data is included in Appendix D). As many petro-chemical projects are fast tracked, schedules are too tight to allow one discipline to end their work prior to another discipline beginning theirs. This means that overlap between work packages (e.g. work packages 38 to 42 and work packages 44 to 49 in Table 4-1) is quite common, which has significant implications for the execution of work packages. Five trades are required to perform these work packages. Daily availability limit for each trade is shown in Table 4-2.

Table 4-1 Case study data

Work Package No. (1)	Description (2)	Quantity (man hours) (3)	Work Areas (4)	Predecessors (FS) (5)	Craft Personnel Requirements		
					Trade (6)	Normal Crew Size (7)	Min Crew Size (8)
38	HydrotesingBetw011AB012ABC	30	011AB	29(-2 days)	PF	10	8
39	HydrotesingBetw012ABC005AB	30	012ABC	30(-2 days)	PF	10	8
40	HydrotesingBetw005AB006AB	30	005AB	31(-2 days)	PF	10	8
41	HydrotesingBetw006AB007ABC	30	006AB	32(-2 days)	PF	10	8
42	HydrotesingBetw007ABC014AB	30	007ABC	33(-2 days)	PF	10	8
43	HydrotestingSilencerOnTopOf007ABC014AB	80	007ABC, 014AB	36,37	PF	10	8
44	InsulationBetw011AB012ABC	20	011AB	38(-1 day)	INS	10	8
45	InsulationBetw012ABC005AB	20	012ABC	39(-1 day)	INS	10	8
46	InsulationBetw005AB006AB	20	005AB	40(-1 day)	INS	10	8
47	InsulationBetw006AB007ABC	20	006AB	41(-1 day)	INS	10	8
48	InsulationBetw007ABC014AB	20	007ABC	42(-1 day)	INS	10	8
49	InsulationSilencerOntopOf007ABC014AB	40	014AB	43(-1 day)	INS	10	8

Table 4-2 Daily trade availability limit

Trade (1)	Start Date (2)	End Date (3)	Available Amount (4)

EL (Electrician)	01-Sep-12	31-Jan-13	20
INS (Insulation)	01-Sep-12	31-Jan-13	20
IW (Iron Worker)	01-Sep-12	31-Jan-13	20
PF (Pipe Fitter)	01-Sep-12	31-Jan-13	16
PIL (Piling)	01-Sep-12	31-Jan-13	20

Multi-tier pipe rack areas are usually highly congested with various pipes and cable trays running (horizontally or vertically) on the steel structures. Given that many work packages have to be performed concurrently in these areas (i.e. due to the overlaps between work packages) and the work space is quite limited, the congestion issue is unavoidable. Since congestion diminishes productivity and increases the risk of incidents, it is necessary to impose congestion constraint on these areas. Table 3 shows the congestion constraint for each work area. Congestion constraint is imposed on the work area level, not directly related to individual work packages. For example, Figure 4-10 shows that many work packages (e.g. piping, cable tray) are linear and extend across two or more work areas. When executed, these work packages should respect the congestion constraint in each work area. To associate work packages with work areas, a work package (i.e. that occurs in more than one work area) is broken down into a number of elements—WAWPs. For example, work package *Piling@011AB012ABC* (shown in Table 1) take places in two work areas and thus can be decomposed into two WAWPs, *Piling@011AB* and *Piling@012ABC*.

Table 4-3 Congestion constraint of each work area

Work Area (1)	Max Craft Persons (2)

011AB	16
012ABC	16
005AB	16
006AB	16
007ABC	16
014AB	16

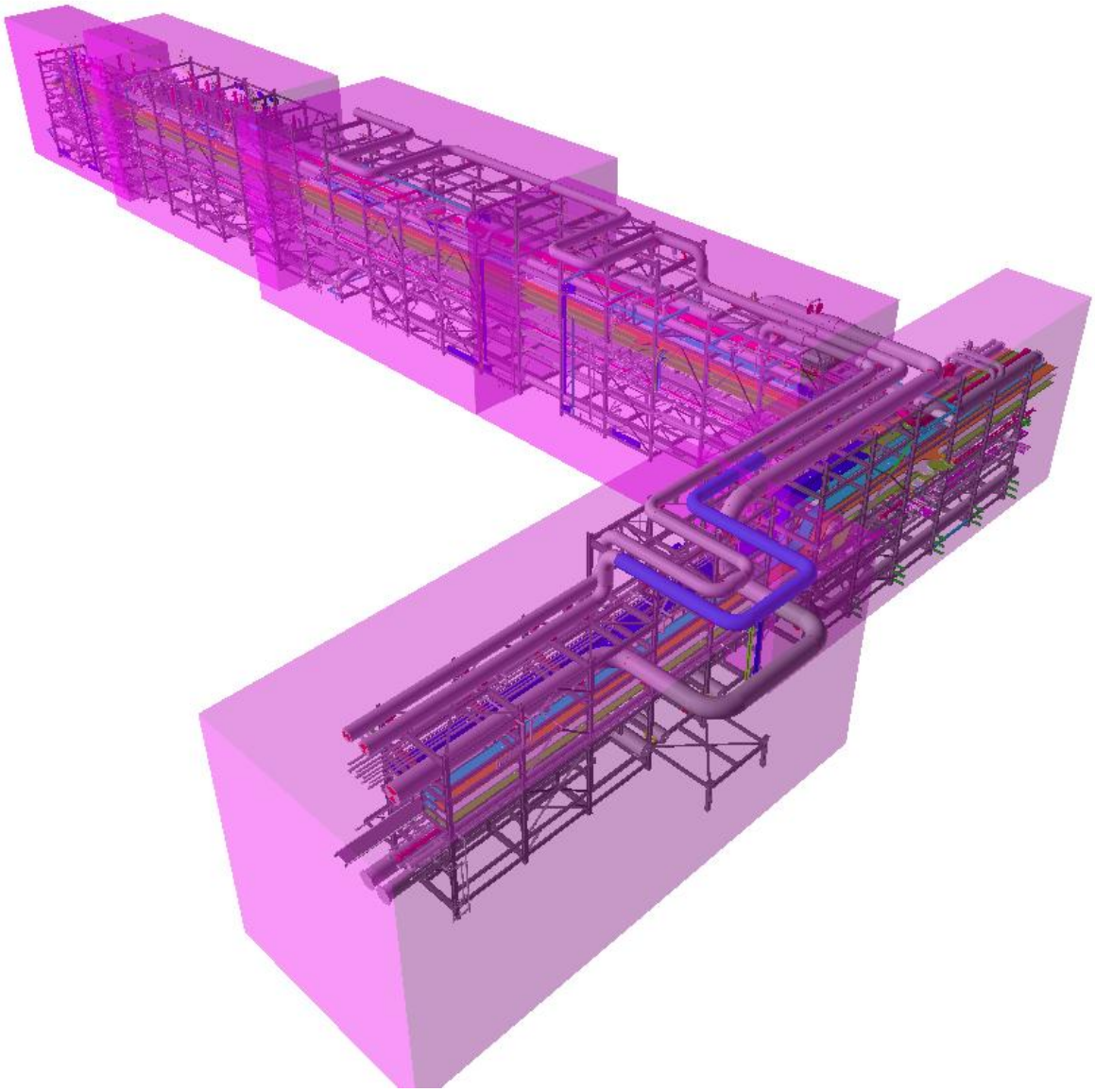


Figure 4-11 Work packages and work areas

4.9.1 Pre-simulation calculations

After reading information from the database, the simulation goes through a pre-simulation stage. The main task of this stage is to translate the information to the WAWP level. For work-quantity-related information (e.g. total man hours), it is simply divided by the number of work areas that the work package crosses. This is merely an approximation, but accurate enough for

the WAWP level. WAWPs can directly copy resource requirement information (e.g. trade, crew size) from their parent work packages. WAWPs can inherit all logic-dependency relationships from their parent work packages, but excessive constraints should be removed too. For example, Figure 4-12 shows a situation where work package A is one of predecessors of work package B, and both take place in work area 1 and work area 2. One of predecessors to B1 (the portion of work package B in work area 1) is A1 (the portion of work package A in work area 1), but not A2 (the portion of work package A in work area 2). Likewise, there is no dependency relationship between B2 and A1.

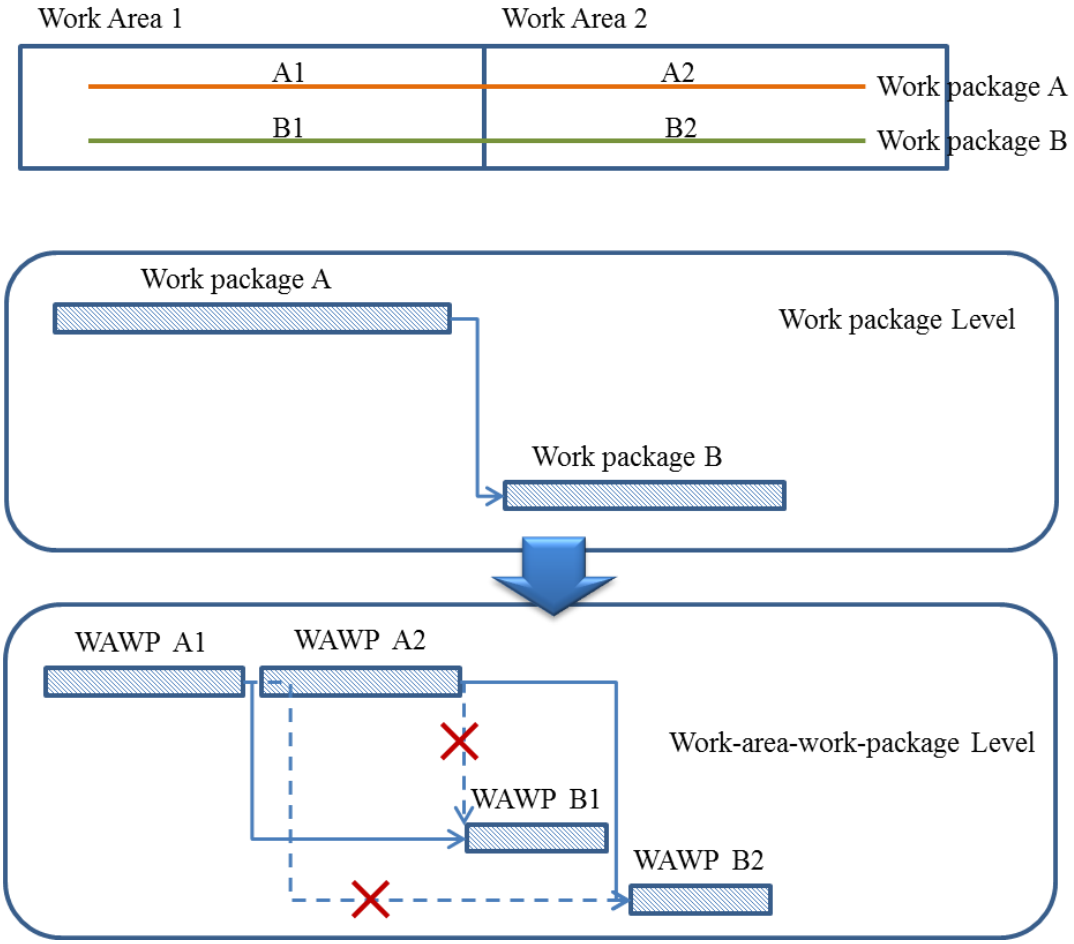


Figure 4-12 Remove excessive dependency relationship

After all the basic information is determined for WAWPs, a CPM calculation is carried out before the simulation starts. This calculation does not take resource availability constraints into account, since the major objective is to calculate various CPM values (e.g. total float, current float, early start or early finish), which can be used to prioritize WAWPs. There is no need to calculate the actual early/late start/finish times and actual floats.

4.9.2 Simulation run and results

In this case, the least total float is selected as the main priority rule. When running, the simulation takes all the constraints (e.g. dependency, resource availability, calendar, time constraint, and congestion) into consideration and dynamically assigns resource to WAWPs. At the end of simulation, the start times and finish times of WAWPs are rolled up to the work package level, i.e. the earliest start time of children WAWPs becomes the start of their parent work package, while the latest finish time of children WAWPs defines the finish of the work package.

It is assumed that the entire sub-project starts September 10th, 2012. The simulation result shows that it can be completed by January 7th, 2013.

Congestion issues exists in every work area (as in Figure 4-13, as the darker bars indicate that the congestion constraint has been exceeded) when the congestion constraint is not imposed. After running the simulation, the resulting schedule shows that all congestion situations have been resolved (Figure 4-14).

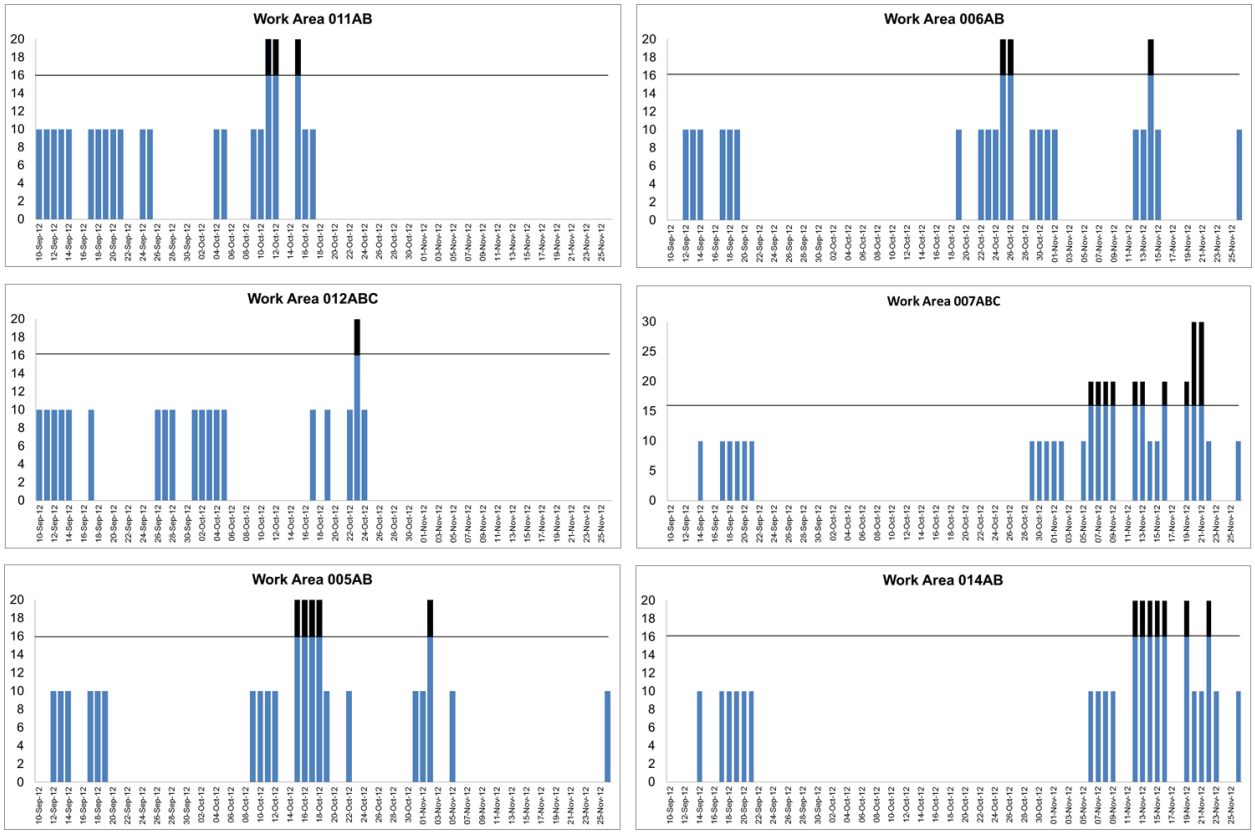


Figure 4-13 Congestion status when congestion constraint is not active

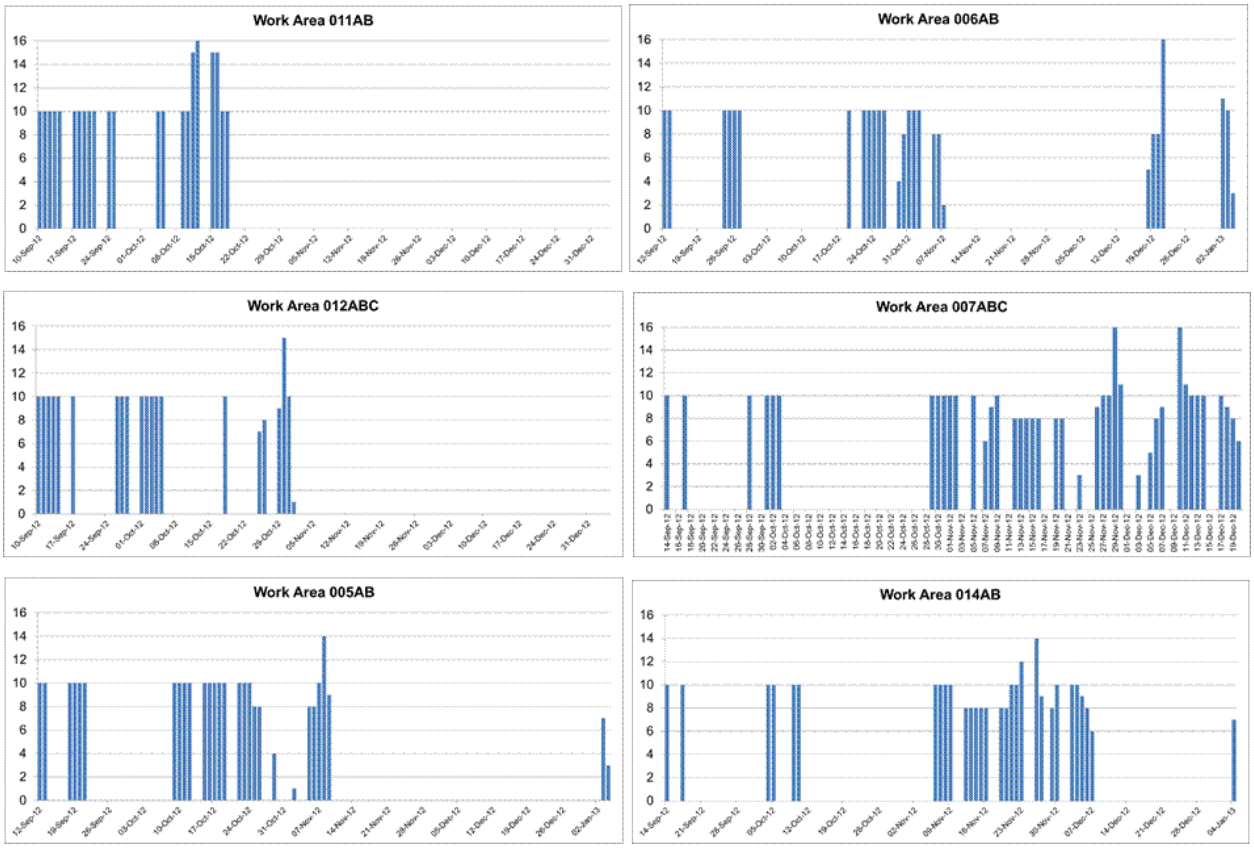


Figure 4-14 Congestion status after the simulation run

4.9.3 Comparison to MS Project and Primavera

The same case study is also implemented manually in MS Project and Primavera 6. The experiment has two steps: (1) schedule all WAWPs without any constraints (i.e. only with precedence dependency and calendar constraints); (2) schedule under both resource availability limit and congestion constraint.

When only considering precedence dependency and calendar constraint, all programs, including the simulation tool, result in the same project completion date—November 26th, 2012. All three generated schedules are identical.

The simulation tool sets a departure from MS Project and Primavera 6 when both resource limit (Table 4-2) and congestion constraint (Table 4-3) come into play. It returns a project completion date of January 7th, 2013. MS Project, after resource leveling, reaches a project finish date of January 24th, 2013. Since MS Project is using its own proprietary algorithm, there is no other resource allocation algorithm to choose. The only variation that can be made is to allow the program to create splits in activities (i.e. allow interruption). Although quite impractical in reality (as mentioned before), this option was experimented with, and the result was still January 24th, 2013. Primavera 6, on the other hand, allows users to experiment with various heuristics (early start/finish, late start/finish, total float, free float, etc.). When using total float as the main heuristic (i.e. the same as used in the simulation) to allocate resources, the program results in a project finish date of January 24th, 2013. Many other heuristics have been tried and most of them return an even later project completion date, except for free float heuristic. When adopting free float as the main heuristic, the program manages to return a project finish date of January 22th, 2013. However, this result is still not as desirable as the one also returned by the simulation—January 7th, 2013.

4.9.4 Discussion of results

It was found that the major difference between the simulation tool and other scheduling programs stems from the way that resources are allocated. CDRASS allows adjustment of resource allocation at any point during the execution of work packages. However, MS Project or Primavera 6 sticks to the same resource allocation throughout the execution of work packages. If allowed to create splits in work packages, MS Project can completely interrupt the work package and allocate no resources to it. The simulation tool, on the other hand, can assign work packages with any amount of resource within the normal and the minimal range mentioned previously.

This is best illustrated by Figure 4-15. It focuses on four work packages, work package 37 to 40. Work packages 37 and 38 are to install the pipes and pipe spools that are used to connect silencers on the top of pipe rack modules, while work package 39 and 40 are to install the silencers (i.e. a type of module equipment). Though two different disciplines (piping and equipment) are involved, all these work packages require the same type of skilled workers—pipe fitters. The background bar chart shows the original schedule when there is no resource limit or congestion constraint. Overlap between these work packages exists (as shown in Figure 4-15). This leads to a situation where a higher-priority work package starts during the execution of a lower-priority work package. For example, work package 38 and 40 both have overlap with work package 39. The resource limit for pipe fitters is 16 and each work package normally requires a crew of 10 pipe fitters. During the overlap, the total resource requirement could surge to 20, which exceeds the resource limit of 16. Figure 4-15 also shows the total float of each work package (i.e. the number of days on the right of the bars), e.g. work package 37 and 39 have 1 day total float, while work packages 38 and 40 have zero total float, and thus, have higher priority than the former two work packages. When work package 39 starts before work packages 38 and 40, a decision needs to be made as to whether the resources should be re-allocated.

MS Project or Primavera only assign either the full amount of required resources or nothing to work packages. Therefore, if the aforementioned situation occurs, it completely postpones one work package or the other. Figure 4-15 shows that MS Project and Primavera can only perform one work package at a time, though the sequence might be different. However, the CDRASS system allows these work packages to take place concurrently. This is due to the dynamic resource allocation mechanism built into the simulation. When work package 38 starts, it can distract some resources from work package 39, even though 39 is still in progress. The two-

round resource allocation algorithm allocates 8 pipe fitters to work package 39 (i.e. as the minimal manpower level is 8) in the first round. In the second round of resource allocation, it allocates the remaining 8 pipe fitters to work package 38 (i.e. the total number of pipe fitters is 16), since it has higher priority to work package 39. In this way, the continuity of work package 39 is maintained. Meanwhile, work package 38 can be started immediately without delay. The same process happens when work package 40 starts.

The effect of this dynamic resource allocation is quite straightforward, as shown in Figure 4-15. In the case of the simulation tool, all four work packages can be completed within 10 working days (i.e. from Nov. 7th to Nov. 20th, 2012). The duration work package 39 is also extended from 6 days to 7.5 days. In contrast, it takes 15 working days and 26 working days to complete these work packages in MS Project and Primavera, respectively. This is just a glimpse of what happened in the entire project schedule, but pinpoints why the simulation tool returns much shorter project duration than the other two scheduling programs.

4.10 CONCLUSIONS

This chapter presents a time-stepped simulation-based scheduling system that (1) complies with various constraints of work packages (precedence dependency, time dependent resource limit, calendar, as well as time constraint), (2) dynamically allocates resources to work packages, and (3) accounts for jobsite congestion constraints of work areas. This scheduling system is implemented such that it can automatically read the information from a database, construct and run the simulation model, and return the generated schedule to the database. It integrates with existing information systems on the database level and works in a behind-the-scene manner so that users do not have to get involved in model development. A real industrial construction case

was used to test the practicality of the tool. The generated schedule was also compared with those from popular project scheduling software Microsoft (MS) and Project, and Primavera (P6). It is observed that the schedule generated from CDRASS is 13 working days shorter than those returned by MS Project and P6.

4.11 LIMITATIONS AND FUTURE WORK

There a number of limitations of this CDRASS system that need to be addressed in future work. First, the resources considered in this development are restricted to various skilled workers. Another important type of resource on the sites of industrial construction projects is cranes. The availability of cranes, if required, is essential to start the execution of work packages, and therefore, should be considered as another constraint to work packages. The dynamic resource allocation algorithm can also be improved by incorporating advanced resource allocation algorithms (e.g. simulation-based auction protocol, SBAP, by Taghaddos et al. 2012). Meanwhile, time unit can be expanded from hour to day, so as to strike a balance between dynamic variation and stability in resource allocation. The change of time unit should not change the main simulation mechanism and resource allocation algorithm. Simphony 4.0 is by nature an event-driven DES environment. When it is used to implement time-stepped simulation, it could impose overhead on processing time. It might be more efficient if the CDRASS system could be implemented in a pure time-stepped simulation engine. Finally, part of the precedence dependency information can be derived automatically from 3D models due to the physical constraints between project components. This could reduce the manual input that is required to populate the simulation database.

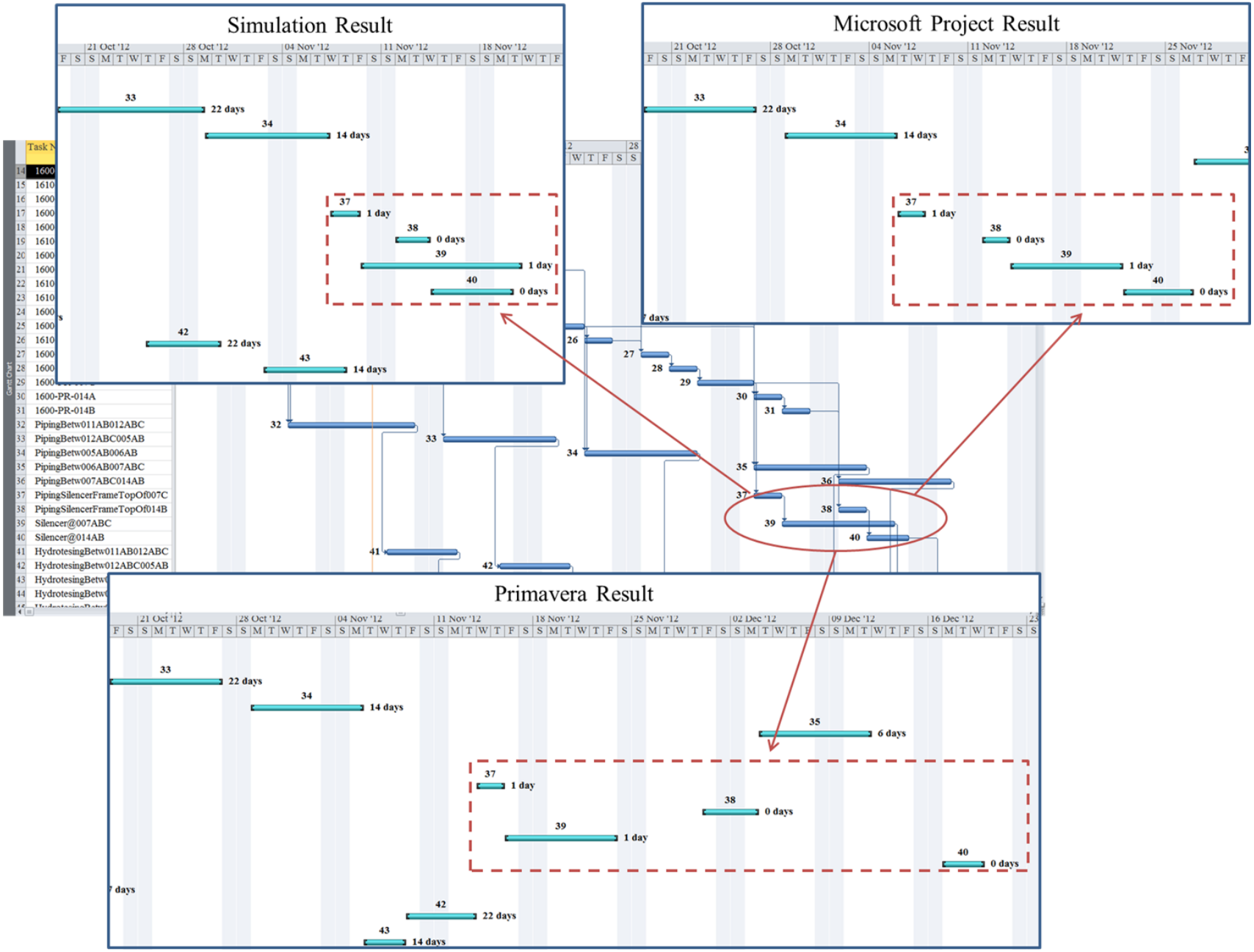


Figure 4-15 Comparison between the simulation tool, MS Project and Primavera

4.12 REFERENCES

AbouRizk, S., and Mohamed, Y. (2000). “Simphony-an integrated environment for construction simulation.” *Proceedings of the 2000 Winter Simulation Conference*, Orlando, FL, USA, 2: 1907-1914.

Ahuja, H. N., and Nandakumar, V. (1985). “Simulation model to forecast project completion time.” *J. Constr. Eng. Manage.* , 111 (4), 325–342.

Ahuja, H., Dozzi, S. P., and AbouRizk, S. M. (1994). *Project management techniques in planning and controlling construction projects*, 2nd Ed., Wiley, New York.

Akinci, B., Fischer, M., and Kunz, J. (2002). “Automated generation of work spaces required by construction activities.” *J. Constr. Eng. Manage.* , 128 (4), 306–315.

Al-Hussein, M., Alkass, S., and Moselhi, O. (2005). “Optimization algorithm for selection and on-site location of mobile cranes.” *J. Constr. Eng. Manage.* , 131 (5), 579–590.

Chan, W., Chua, D., and Kannan, G. (1996). "Construction resource scheduling with genetic algorithms." *J. Constr. Eng. Manage.*, 112(2), 125–132.

Christodoulou, S. (2010). “Scheduling resource-constrained projects with ant colony optimization artificial agents.” *J. Comput. Civ. Eng.* , 24 (1), 45–55.

Davis E. W., Patterson J. H. (1975). “A comparison of heuristic and optimum solutions in resource-constrained project scheduling.” *Management Science*, 21(8), 944-955.

Demeulemeester, E. L., and Herroelen, W. S. (2002). *Project Scheduling : A Research Handbook*, Kluwer Academic Publishers, Boston, Mass, USA.

Dozzi, S. P., and AbouRizk, S. M. (1993). *Productivity in Construction*, Institute for Research in Construction, National Research Council, Ottawa, ON, Canada.

Elbeltagi, E., Hegazy, T., and Grierson, D. (2005). "Comparison among five evolutionary-based optimization algorithms." *J. Adv. Engng. Informatics*, 19(1), 43 – 53.

Hammad, A. (2009). "An Integrated Framework for Managing Labour Resources Data in Industrial Construction Projects: A Knowledge Discovery in Data (KDD) Approach." University of Alberta, Edmonton, Canada

Hegazy, T. (1999). "Optimization of resource allocation and leveling using genetic algorithm." *J. Constr. Eng. Manage.*, 125(3), 167–175.

Hegazy, T. and Menesi, W. (2010). "Critical Path Segments Scheduling Technique." J. Constr. Eng. Manage., 136(10), 1078–1085.

Hu, D., and Mohamed, Y. (2010). "State-Based Simulation Mechanism for Facilitating Project Schedule Updating." Construction Research Congress 2010, Banff, AB, Canada, ASCE, 369-378.

Kandil, A., and El-Rayes, K. (2006a). "MACROS: Multi-objective automated construction resource optimization system." *J. Manage. Eng.*, 22(3), 126–134.

Karshenas, S., and Haber, D. (1990). "Economic optimization of construction project scheduling." *Journal of Construction Management and Economics*, Vol. 8, No. 2, 135-146.

Kim, K., and de la Garza, J. M. (2005). “Evaluation of the resource constrained critical path method algorithms.” *J. Constr. Eng. Manage.*, 131(5), 522–532.

Horner, R. M. W., and Talhouni, B. T. (1995). *Effects of accelerated working, delays and disruption on labour productivity*, Chartered Institute of Building, Ascot, U.K.

Hu, D., and Mohamed, Y. (2010). “State-Based Simulation Mechanism for Facilitating Project Schedule Updating.” Construction Research Congress 2010, Banff, AB, Canada, ASCE, 369-378.

Jiang G., and Shi J (2005). “Exact algorithm for solving project scheduling problems under multi resource constrains.” *J. Constr. Eng. Manage.*, 131(9), 986–992.

Kolisch R. (1996). “Efficient priority rules for the resource-constrained project scheduling problem.” *Journal of Operations Management*, 14(3), 179-192.

Lu, M., and Li, H. (2003). “Resource-activity critical-path method for construction planning.” *J. Constr. Eng. Manage.*, 129(4), 412–420.

Lu, M., Lam, H.C., Dai, F., (2008). “Resource-constrained critical path analysis based on discrete event simulation and particle swarm optimization.” *Automation in Construction*, 17(6), 670–681.

Martinez, J. C., and Ioannou, P. G. (1997). "State-based probabilistic scheduling using STROBOSCOPE's CPM add-on." Construction Congress V, Minneapolis, MN, ASCE, 438–45.

Marzouk, M. , and Moselhi, O. (2004). "Multiobjective optimization of earthmoving operations." J. Constr. Eng. Manage. , 130 (1), 105–113.

Mohamed, Y., Borrego, D., Francisco, L., Al-Hussein, M., Abourizk, S., and Hermann, U. (2007). "Simulation-based scheduling of module assembly yards: Case study." *Engineering, Construction and Architectural Management*, 14(3), 293-311.

Moselhi, O., and Lorterapong, P. (1993). "Near optimal solution for resource-constrained scheduling problems." *Construction Management and Economics*, 11, 293-303.

Ng, S. T., and Zhang, Y. (2008). "Optimizing construction time and cost using ant colony optimization approach." *J. Constr. Eng. Manage.* , 134 (9), 721–728.

Ovararin, N., and Popescu, C. (2001). "Field factors affecting masonry productivity." *AACE International Transactions*, ES91-ES97.

Riley, D. R., and Sanvido, V. E. (1997). "Space planning method for multistory building construction." *J. Constr. Eng. Manage.*, 123 (2), 171–180.

Sadeghi, N., and Fayek, A. Robinson. (2011). “A fuzzy-based approach for proactive scheduling of construction projects.” *Proceedings, 3rd International/9th Construction Specialty Conference*, CSCE, Ottawa, Ont., June 14-17: CN-007-1-CN-007-11.

Sadeghi, N., Fayek, A., and Ingolfsson, A. (2012). ”Simulation-Based Approach for Estimating Project Completion Time of Stochastic Resource–Constrained Project Networks.” *J. Comput. Civ. Eng.*, 26(4), 558–560.

Senior, B. A., and Halpin, D. W. (1998). “Simplified simulation system for construction projects.” *J. Constr. Eng. Manage.* , 124 (1), 72–81.

Song, L., and AbouRizk, S. M. (2006). “Virtual shop model for experimental planning of steel fabrication projects.” *J. Comput. Civ. Eng.*, 20 (5), 308–316.

Taghaddos, H., AbouRizk, S., Mohamed, Y., and Hermann, U. (2012). ”Simulation-Based Auction Protocol for Resource Scheduling Problems.” *J. Constr. Eng. Manage.*, 138(1), 31–42.

Thabet, W. Y., and Beliveau, Y. J. (1994). “Modeling work space to schedule repetitive floors in multistory buildings.” *J. Constr. Eng. Manage.* , 120 (1), 96–116.

Thomas, H. R., Jr., and Smith, G. R. (1990). “Loss of construction labor productivity due to inefficiencies and disruptions: The weight of expert opinion.” *PTI Rep. No. 9019*, Pennsylvania Transportation Institute, Pennsylvania State University.

Zhang, H., Tam, C.M., and Shi J. (2002) "Simulation-based methodology for project scheduling." *Construction Management and Economics*, 20 (8), 667 – 678.

Zhang, H., Li, X., Li, H. and Huang, F. (2005). "Particle swarm optimization-based schemes for resource-constrained project scheduling." *Automation in Construction*, 14 (3), 393–404.

Zouein, P. P., and Tommelein, I. D. (2001). "Improvement algorithm for limited space scheduling." *J. Constr. Eng. Manage.* , 127 (2), 116–124.

CHAPTER 5. Automating Sequence Planning for Industrial Construction Processes Using Domain Independent Artificial Intelligence Planning

5.1 PROBLEM STATEMENT

Project planning is one of the core project management functions, which "links the design of a facility to its construction" (Fischer and Aalami 1996). It involves two major tasks: (1) identify a list of activities necessary to deliver the designed facility, and (2) generate proper sequential relationships between them. The latter is also referred to as construction sequencing. Construction sequencing involves intensive knowledge and expertise, since each sequential relationship has its underlying rationale. For example, the physical relationship between building components, e.g. construction of columns should precede beams due to the support relationship between them. Others reasons could be technical relationships which depend on the technical method, e.g. pipe spool components should be fitted (temporary connection) before they can be welded (permanent connection).

In practice, many construction planners rely on critical-path–method- (CPM) based tools to generate project plans. As useful as these tools may be in displaying and analyzing project plans, they generally lack the ability to capture and use aforementioned knowledge to sequence various activities. In other words, they provide little assistance, if any, to generate project plans. As such, current project planning is mainly performed by human planners in a manual and heuristic manner. This becomes a challenging task as planners face the growing complexity, increasing uncertainty as well as ever compressed schedule that have become the common characteristics of large-scale construction projects, such as those within industrial construction.

Automating this construction sequencing process is beneficial to the overall project success. However, limited attention has been paid to automated construction sequence. In addition,

previous research has been focused on developing solutions for building construction projects using domain specific artificial intelligence (AI) planning techniques (e.g. knowledge based expert system). Most sequencing rationales are derived from the physical relationships between building components (e.g. columns, beams and slabs), which make it difficult to apply them to other types of construction, e.g. industrial construction projects.

This chapter investigated the use of domain-independent AI planning technique to solve construction sequencing problems for two industrial construction processes. Industrial construction refers to building facilities such as petro-chemical refineries or oil/gas production plants. The construction processes studied in this chapter are (1) pipe spool fabrication and (2) module installation. A standard AI planning language called Planning Domain Definition Language (PDDL) is used to model these processes and several AI planners that can handle PDDL representations (e.g. Metric-FF by Hoffmann 2002, LPRPG by Coles et al. 2008, and LPG by Gerevini and Serina 2002) are employed to identify feasible plans to accomplish the construction process without violating any physical or technical constraints. A number of experiments are conducted to test the effectiveness of PDDL on each of these processes and results show that PDDL technique is more suitable to generate plans for module installation than pipe spool fabrication.

The rest of chapter is organized as follows. Reviews of previous research work related to automated planning and scheduling for construction projects are provided in the next section. This is followed by a brief explanation of domain-independent AI planning technique (also referred to as general purpose planning), a standard planning language PDDL and its applicability to construction sequencing problems. Two industrial construction processes, namely pipe spool fabrication and pipe-rack module installation, are investigated to see if

domain-independent AI planning is capable of generating feasible sequences or plans for real-life construction processes. Brief descriptions of experiments and results are provided for each case. Finally, limitations and some issues for future research are discussed.

5.2 LITERATURE REVIEW

A review of previous research finds that limited attention has been paid to the topic of automated project planning and scheduling in the construction domain and that most relevant research was conducted in the 1980s and the early 1990s. Many researchers realized that development of automated planning and scheduling systems should start with formalization of sequencing knowledge between activities. Gray (1986) introduced sequencing rationales such as "fixing base provided by," "flexibility of material," "covered by," "service provided by," and "protected by" that are generalized from different contractors' schedules. Darwiche et al. (1989) made use of similar dependencies, such as "supported-by," "adjacent-to," "enclosed-by," and "in-same-floor." Most of the aforementioned sequence rationales are derived from the physical relationships between building components. Echeverry et al. (1991) enriched the body of knowledge by adding three more types of sequencing factors in addition to physical relationships, specifically "trade interaction," "path interference," and "code regulation." Detailed sequencing rationales are listed under each factor type. For example, under "trade interaction" category, they identified "space competition," "resource limitation," "unsafe environment effects," etc. These sequencing rationales possess different levels of flexibility, i.e. some of them are hard constraints that cannot be violated and result in only one feasible sequence, while others are soft constraints that can be satisfied by a variety of sequence options. To effectively apply these sequencing rationales to construction activities and to automatically generate realistic construction plans or schedules, many researchers developed automated planning or scheduling systems. Gray (1986) developed

the TIME system, a rule-based knowledge-based system (KBS), to analyze a design, to generate the network of all required activities (i.e. project schedule) and to perform critical path method (CPM) calculations. Hendrickson (1988) developed a frame-based KBS system, called CONSTRUCTION PLANEX, for construction planning and scheduling of modular high-rise buildings. The system takes a detailed description of the building design as input, such as design elements (building components), site information as well as resource availability. Based on this input, it automatically selects activities that are necessary to build these design elements and aggregates them into project activities (i.e. bottom up manner) for which precedence dependency estimate durations are added. PLANEX then performs CPM calculations and resource allocation. The output is a project schedule and a cost estimate. Darwiche et al. (1989) developed a prototype planning system called OARPLAN. Like PLANEX, it needs users to input a description of the designed building, e.g. a set of physical building components with their specifications and relationships. Similarly, OARPLAN starts with including activities that are necessary for the building components and keeps elaborating them to a level of detail that is appropriate for project estimate or control (i.e. top-down manner). Meanwhile, it also assigns precedence dependencies to sub-activities (generated during elaboration), if necessary. Different knowledge sources are triggered to automate this activity elaboration and sequencing process. Aalami et al. (1998) characterized OARPLAN as a component-based reasoning system as opposed to a process-based reasoning system such as GHOST (Navinchandra et al. 1988). GHOST starts with an unrealistically optimum schedule where all activities are in parallel. It then uses “critic knowledge sources” (CKSs) to modify the schedule by introducing sequences between activities when necessary. GHOST also includes an activity elaboration process in which new sub-activities are introduced and sequenced by CKSs. GHOST generates a project

schedule as output, at appropriate levels of detail with precedence dependencies among activities. Fischer and Aalami (1996) argued that construction method knowledge is the missing link between a design and a construction plan. They formalized this knowledge by creating construction method templates that consist of five elements: domain, constituting activities, activity sequencing, constituting objects, and resource requirements. Fischer and Aalami (1996) developed a system where all these construction method templates are organized in a hierarchical structure and are used to elaborate a high level seed activity to a desired level of detail. Once the elaboration process is done, the system performs duration estimate and CPM calculations. User participation is required when selecting a method to apply to an activity. Therefore, it is not a fully automatic system. Koo et al. (2007) pointed out that most aforementioned automatic planning and scheduling systems aim at identifying a correct construction sequence rather than providing possible sequence alternatives. They prototyped a system called CLCPM, which automatically analyzes the "role" and "status" of activities of interest (i.e. need to be re-sequenced) using a constraint ontology and a classification mechanism. A decision is made afterwards with respect to whether these activities can be re-sequenced.

An observation from the previous research is that most of the developed automatic systems are knowledge-based expert systems which use only domain specific knowledge and rules, and that almost all of them focus on building projects. In other words, these systems can only be used to generate plans for building projects. For example, many sequencing rationales are derived from the physical relationships between building components (e.g. columns, beams, walls and slabs). They cannot be readily applied to industrial construction processes where the building blocks are pipe spools, equipment and modules. The sequence rationales between these components are fundamentally different from those applied in building construction projects. Kartam and Levitt

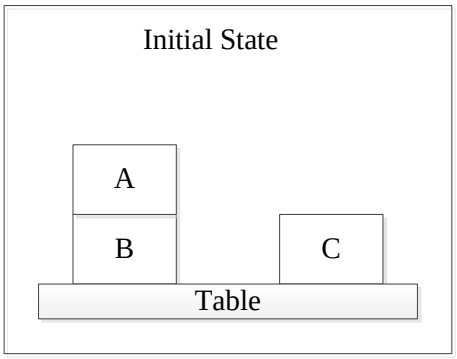
(1990) experimented with domain-independent AI planning technique and customized a planner SIPE to identify feasible plans for modular building projects. They, like many other researchers, used physical relationships between building components, e.g. supported-by and enclosed-by, to propagate sequence constraints among activities. It was found that the system was only able to handle simplified construction processes and was inefficient for real-life planning situations (Echeverry et al. 1991).

Another observation is that many of these automatic systems (e.g. TIME, CONSTRUCTION PLANEX, GHOST, etc.) aim at handling the complete scheduling process. This means that systems are designed to perform all the functions: activity identification and inclusion, activity sequencing, activity duration estimation, CPM calculation and/or cost estimation. As useful as they are, they all suffer from the complication which stems from squeezing all these planning and scheduling functions into one single system. For example, individual knowledge sources are required for each of these functions and a sophisticated mechanism is needed to manage the interactions among the knowledge sources so that no conflict occurs. An error in any of the components could propagate and consequently compromise the quality of generated schedules. It is therefore believed by the authors that it might be more efficient and effective to develop a dedicated tool to automate only one part of the planning and scheduling process, e.g. construction sequencing. Reasons for this is that today, many commercial software packages are already equipped with the capability of identifying and creating work packages (or activities) directly from a 3D model, for example ConstructSim V8i by Bentley. Meanwhile, many sophisticated scheduling packages (e.g. Microsoft Project, Primavera P6) or simulation-based scheduling systems (Hu and Mohamed 2010, Taghaddos et al. 2012) have been developed to perform various scheduling functions (e.g. resource allocation and leveling, CPM calculations,

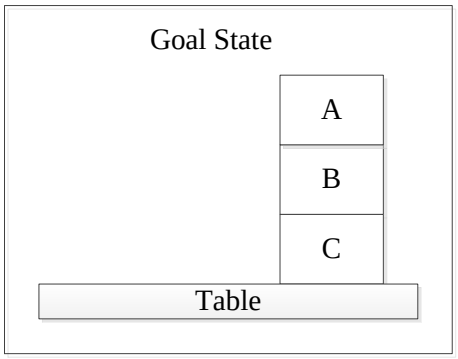
and even schedule optimization). A gap exists between them with regard to how to generate logical sequences between activities. As mentioned before, sequencing is still largely manually performed by human planners. This chapter then focuses on investigating automated solutions to the construction sequencing process using domain-independent AI planning technique.

5.3 DOMAIN-INDEPENDENT AI PLANNING

Planning has been one of the major AI research areas since the 1960s (Newell and Simon 1972). Domain-independent planning (also referred to as general purpose planning or classic planning) is the process of identifying a sequence of actions that gradually change a system (i.e. the block system in Figure 5-1) from the initial state to a desired goal state. It assumes a state representation of the system and defines a set of literals or predicates to describe the state (i.e. block A is on the top of block B or block C is on the table, as shown in Figure 5-1a). One or more actions (also called operators) are defined to change the state of the system (i.e. the 'Place-on-table' action in Figure 5-1c). Each action is defined with preconditions and effects (Figure 5-1c). An action is applicable when all the preconditions are satisfied. Effects are true when the action is accomplished, which usually means that a change has occurred in the state of the system. The AI planning system is fed with an initial state (Figure 5-1a) of the system and a goal state (Figure 5-1b). It then selects appropriate actions to bridge the gap between the current state and the goal state. When the goal state is achieved, a feasible plan (a sequence of actions) is found.



a



b

Domain Definition File (PDDL)

```
(define (domain block)
  (:requirements :conditional-effects :equality :negative-preconditions)
  (:type block table)
  (:predicates
    (ontopof ?b ?b0 – block)
    (ontable ?b – block ?t – table)
  )
  (:action place-on-table
          :parameters (?b ?b1 – block ?t – table)
          :precondition (and (not exists (?b0 - block) (ontopof ?b0 ?b))
                        (and (not (=?b ?b1) (ontopof ?b ?b1)) )
          :effect (and (ontable ?b ?t)
                  (not (ontopof ?b ?b1)) )
  )
  … …
)
```

c

Problem Definition File (PDDL)

```
(define (problem blockplacing)
  (:domain block)
  (:requirements :conditional-effects :equality :negative-preconditions)
  (:object A B C – block T – table)
  (:init
    (ontopof A B)
    (ontable B T)
    (ontable C T)
  )
  (:goal (and (ontopof B C) (ontopof A B) (ontable C T)) )
)
```

d

Figure 5-1 The block system and the description in PDDL

Planning Domain Description Language (PDDL) is one of the standard languages (e.g. STRIPS) for domain-independent planners. It has evolved through a number of versions since it was first proposed by Drew McDermott (1998). The representation of a target system and its associated planning problem in PDDL consists of two pieces of description. First, there is a general description of the system under study called Domain Definition File (Figure 5-1c). It defines which object classes are of interest in the system, what possible states they could have, and what actions are available to change their states. Another piece of description called Problem Definition File (Figure 5-1d) is used to describe a specific planning problem. It defines the actual instances for each object class and depicts their initial states and goal states. The domain definition file can be re-used as long as it is a valid representation of the system while the problem definition file could vary with each specific planning problem (i.e. different instances could be involved and/or different initial or goal states). These two descriptions are eventually input into a PDDL compliant planner which might be able to identify a feasible plan using a variety of search methods.

Domain-independent AI planning has been successfully employed in several areas, such as robot navigation, machined parts manufacturing sequencing, and emergency evacuation (Ghallab et al. 2004). It could also be a good candidate technique to plan construction processes when planning problems are properly abstracted and scoped.

5.4 APPLICABILITY OF DOMAIN-INDEPENDENT AI PLANNING TO CONSTRUCTION SEQUENCING

In order to make the best use of domain-independent AI planning technique as a plan generator, the domain in which planning problems arise should satisfy a number of features. First, domain-independent AI planning assumes a state representation of the system under study (Darwiche et al. 1989). States are abstract descriptions of the system's status that must serve for two main purposes: (1) to distinguish the difference between the initial state and the goal state for the planning problem, and (2) to determine what actions could be taken under each possible state of the system. Since it is impossible to represent infinite knowledge, the states of the system must be finite and the required knowledge should be limited. Second, actions are used in domain-independent AI planning to change the states of the system gradually and to lead it to reach the goal state. Actions should be unique (distinct from each other) and the total number of actions should be limited. In addition, preconditions and effects of actions should also have the ability to be precisely formulated (i.e. a precondition is a specific state of the system that when satisfied, allows the action to become applicable, while an effect is the resulting state of the system after the action is performed). Finally, actions could be repeatedly used in the plan. For example, action 'place-on-table' in the block stacking system (Figure 5-1c) could be applied to different blocks.

Darwiche et al. (1989) argued that domain-independent AI planning is not quite suitable for planning and sequencing construction projects. The major reasons they cited include: (1) in the construction domain, there are huge number of unique actions such that it is almost impossible to have a complete enumeration of them, and (2) construction actions are usually not clearly defined with preconditions and effects. For example, in order to install pre-assembled equipment

in a room of a building, it might require not only that the floor that supports it should be completed and service like water or power supply is in place, but also that the exterior walls of the floor (where the room is located) should not be completed so that the equipment can be lifted and transported by a crane to the final location.

It is important to put this argument in context. Many researchers, including Darwiche, attempted to develop a planning system that is capable of producing plans or schedules for an entire project, which usually implies planning or scheduling at a work-package level or at a master-plan level (i.e. highest planning level). The scope of planning leads to the fact that numerous types of objects (e.g. walls, columns, beams, floors, windows, doors, etc. in buildings) and their possible states should be considered and that all available actions should be enumerated. This means that a large amount of case-specific knowledge will need to be formulated in domain and problem definition files (Figure 5-1c and Figure 5-1d), which becomes a very cumbersome task to fulfill. It might also make the planning problem too complex to be solved by any AI planner. In addition, actions (or activities) at this level of detail also tend to be unique and have less repetition. Levitt and Kunz (1987) discovered that, at this level, the precondition to start an action is usually the completion of another action and that is the sequence. This leads to a situation where the sequence is already implicitly contained in preconditions and effects of actions and if the same information is input into a CPM tool (with no searching capability), the same sequence or plan would be generated. Thus, no new knowledge is created during the AI planning process. Levitt and Kunz (1987) concluded that domain-independent AI planning is suitable to plan on the construction operation level where fewer objects and actions are involved and more repetition of actions occurs. This point of view is echoed in this study. By experimenting with domain-independent AI planning technique on two industrial construction

processes, it has been proven that domain-independent planning is a competent plan generator for problems which are properly abstracted and scoped.

Most previous research on automatic planning for construction projects was conducted during the 1980s and early 1990s. On the other hand, research on domain-independent AI planning has continued to advance significantly over time. Domain-independent AI planning technique evolved from classical planning that is only capable of handling logic inference, to more advanced planning that can address various issues encountered in real-life settings such as numeric-valued variables, time constraints, or a non-deterministic environment. The computing capacity of personal computers has also improved drastically. It is then worth revisiting the opportunity of using newly developed domain-independent AI planning technique to automatically plan construction processes.

5.5 PIPE SPOOL FABRICATION SEQUENCING PROBLEM

Pipe spools are building blocks for industrial construction projects and are fabricated from a group of raw pipes and pipe fittings. Cutting, fitting and welding are three major steps that a pipe spool goes through during fabrication. Among them, cutting always occurs at the beginning and only applies to raw pipes (i.e. cut to required sizes). This is followed by fitting (i.e. temporarily connecting) some of the cut pipes and pipe fittings together, which results in a sub-assembly (e.g. Assembly 7 at step 1 in Figure 5-2). This sub-assembly is then transported to welding stations where it is welded (i.e. permanently connected). Fitting and welding take place alternately, rather than simply in sequence. For example, after welding, the sub-assembly would be sent back to the original fitting table where it is fitted with more pipes and/or pipe fittings (e.g. sub-assembly 7

and sub-assembly 8 are connected at step 3 in Figure 5-2). This back and forth between the fitting table and welding stations continues until all the components are welded.

Pipe spools can be fabricated through a number of alternative sequences. Figure 5-2 shows an example of a pipe spool with a simple configuration that can be fabricated in at least two different sequences. One difference between these sequences is that sequence 1 requires one more fabrication step than sequence 2. Another difference is that sequence 1 involves two position welds (at step 3) while sequence 2 only involves two roll welds (Figure 5-3a). Position welding (Figure 5-3b) occurs when one or more than one branch of the main pipe exceeds the clearance limit of the rolling machine. Compared to roll welding, position welding requires more manual work and therefore takes more time to complete. This means that sequence 2 can result in better performance (e.g. cycle time) than sequence 1. One of the major objectives of sequencing pipe spool fabrication is to identify a sequence that requires the minimum number of position welds.

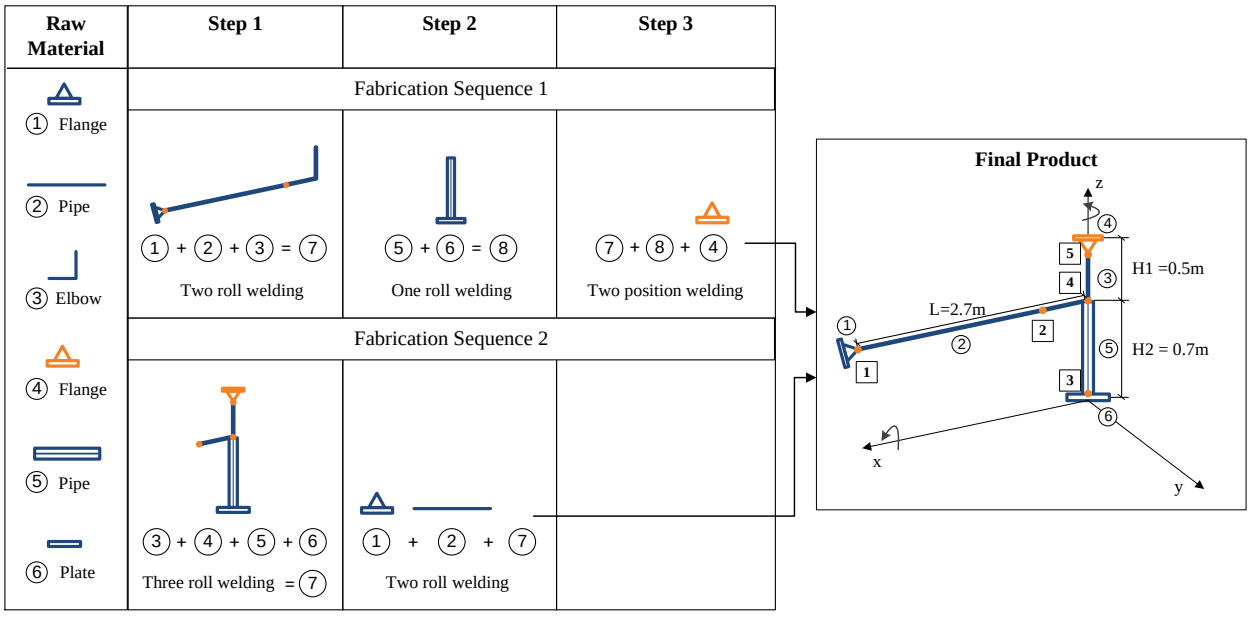
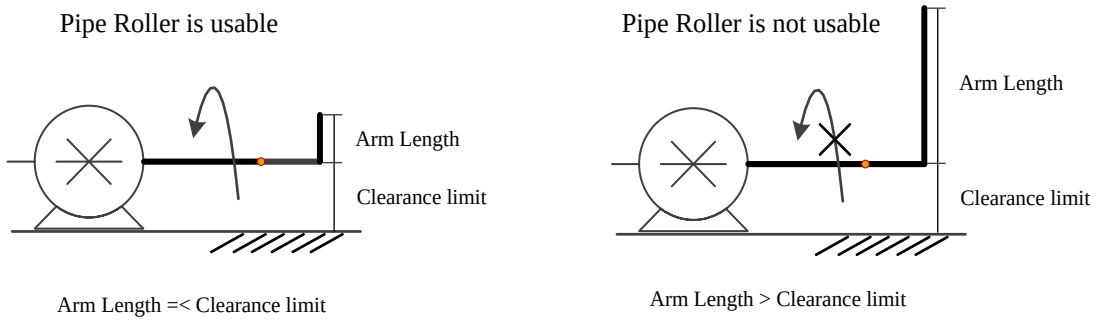


Figure 5-2 Pipe spool fabrication sequences



(a) Roll welding (b) Position welding

Figure 5-3 Roll welding and position welding

5.5.1 Problem abstraction

Since it is impossible to model every aspect of pipe spool fabrication, an appropriate level of abstraction is necessary for PDDL representations. First, cutting always occurs at the beginning of the fabrication process. It is the invariable part of the fabrication sequence and therefore, does not need to be considered here. Every fitting operation is followed by a subsequent welding operation which means the sequence of fitting is identical to the sequence of welding. It is then appropriate to use the sequence of welding operation to represent the sequence of the whole pipe spool fabrication. Two types of welding operations, roll welding and position welding, are considered, and their selection is determined by the maximum arm length from the rolling axis where the welding is performed and the clearance limit of the rolling machine.

Two types of objects are considered: (1) pipe spool assemblies (represent all raw components and in-progress subassemblies), and (2) welding points (indicated by numbers enclosed in squares in Figure 5-2). For assemblies, there are only two states: active and not active (i.e. only active assemblies can participate in the next welding). For welding points, three predicates are used to describe their states. A welding point should belong to an assembly (e.g. welding point 1

belongs to Assembly 7 in Figure 5-2). It is also located on a specific axis (e.g. welding point 1 is on axis X in Figure 5-2). To indicate progress of fabrication, a welding point should be described as welded or not welded.

In order to determine if a welding operation is a position welding or a roll welding, coordinates and dimensions of an assembly should also be included. Each pipe spool assembly is treated as a 3D rectangular box. Assembly 5 (i.e. an elbow) in Figure 5-4, for example, is described by a base point (0, 0, 920) which happens to be a welding point and its dimensions (0, 89, 89), respectively on the X, Y, and Z axes. Assembly 4 can be described in a similar manner. When fabricating Assembly 4 and Assembly 5 (in Figure 5-4), the axis Z will be the rolling axis if a roll welding is to be performed. The maximum branch length is either on the X axis or on the Y axis. In this case, the maximum branch length is 89 (i.e. on Y axis). If it is less than the clearance limit, then a roll weld is feasible. Otherwise, a position weld is required.

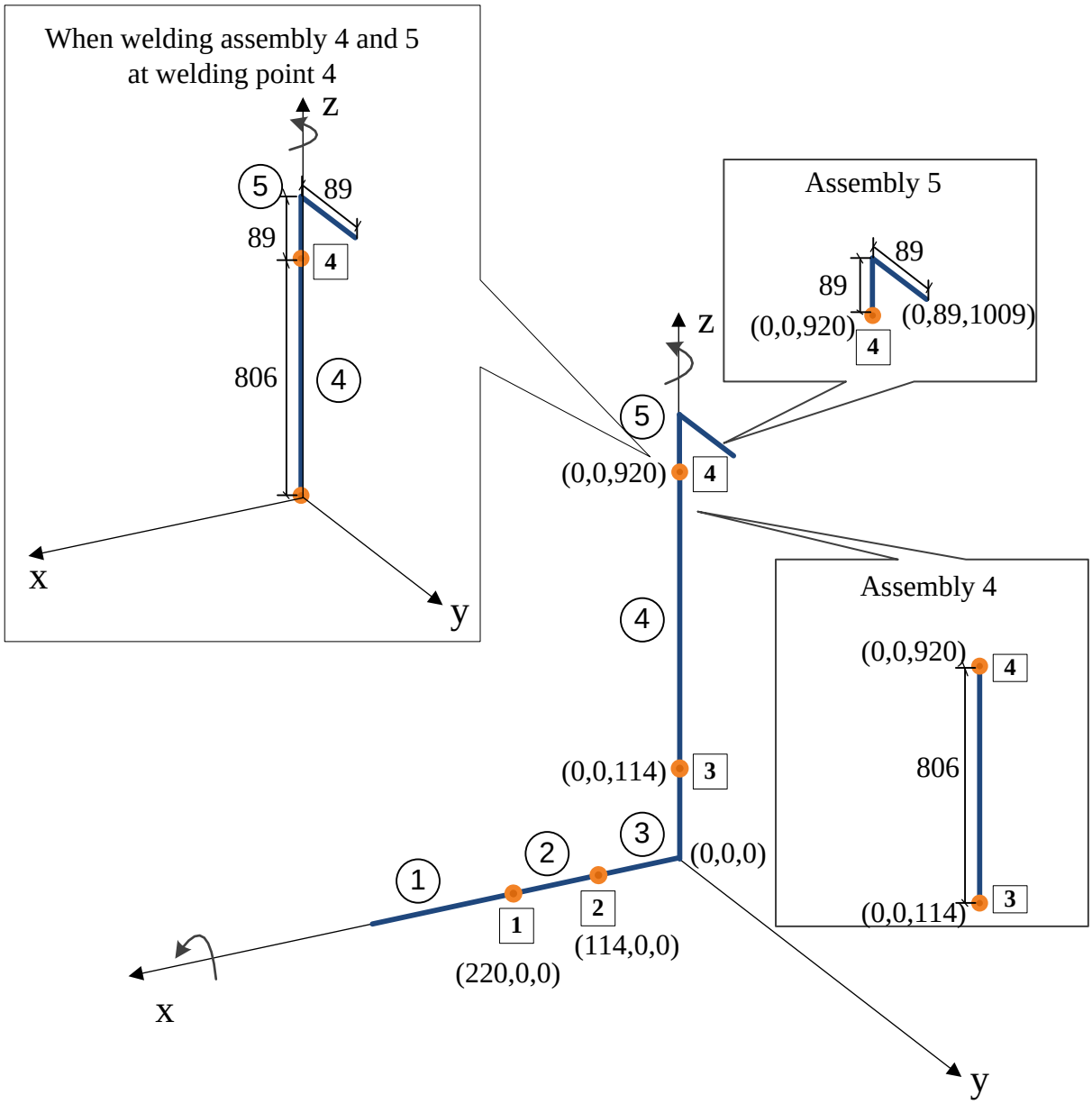


Figure 5-4 Example pipe spool components and their geometries

5.5.2 PDDL domain and problem definition representation

Domain definition file is intended to describe general knowledge about the system under study. On the other hand, problem definition file provides specific knowledge about a planning problem. Object types and possible states as well as all available actions are modeled in the domain

definition file, while instances of object types with their initial states and goal states are included in problem definition file. Figure 5-5 and Figure 5-6 show an example domain definition file and an example problem definition file, respectively.

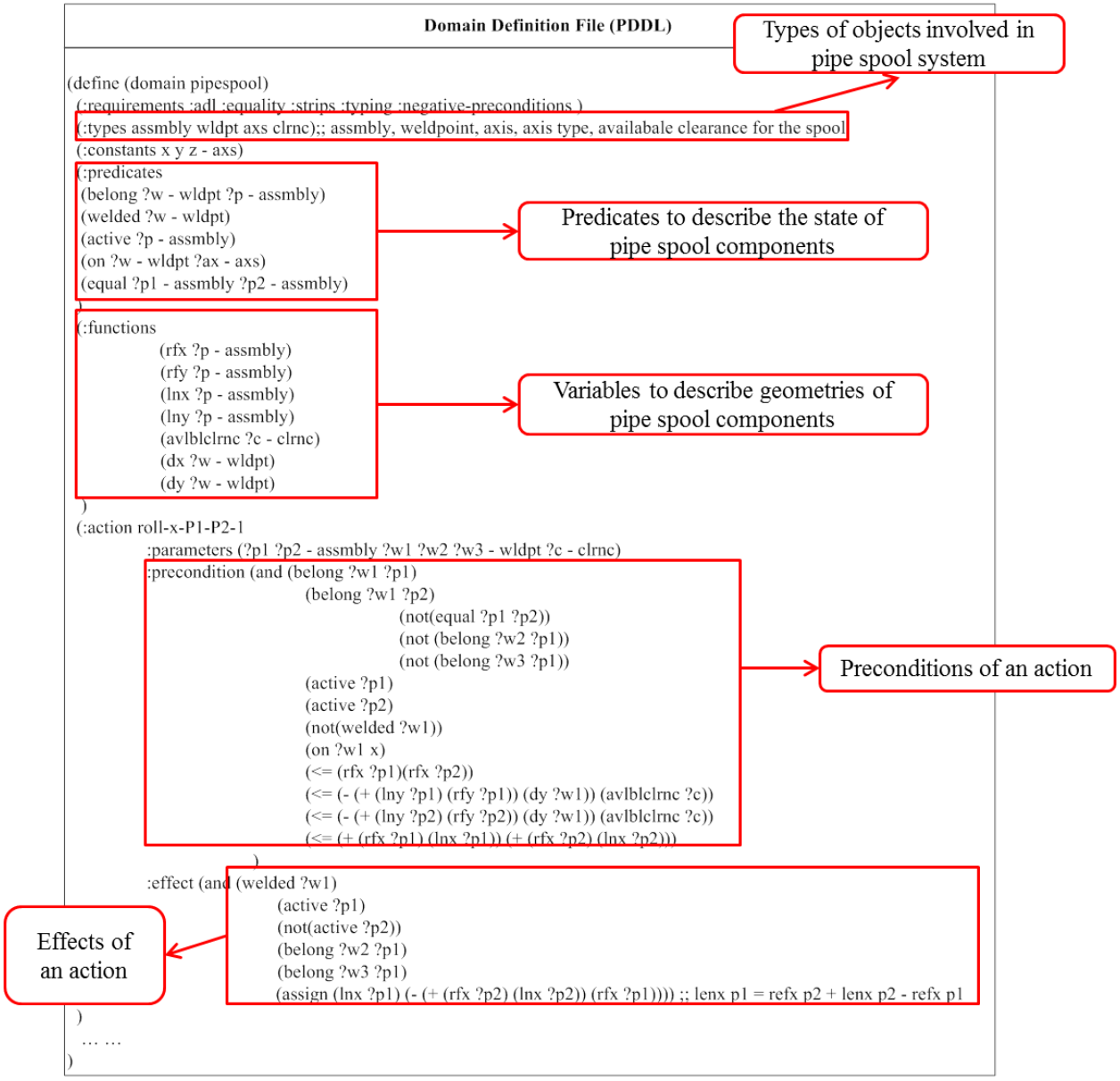


Figure 5-5 An example PDDL domain definition file for pipe spool fabrication

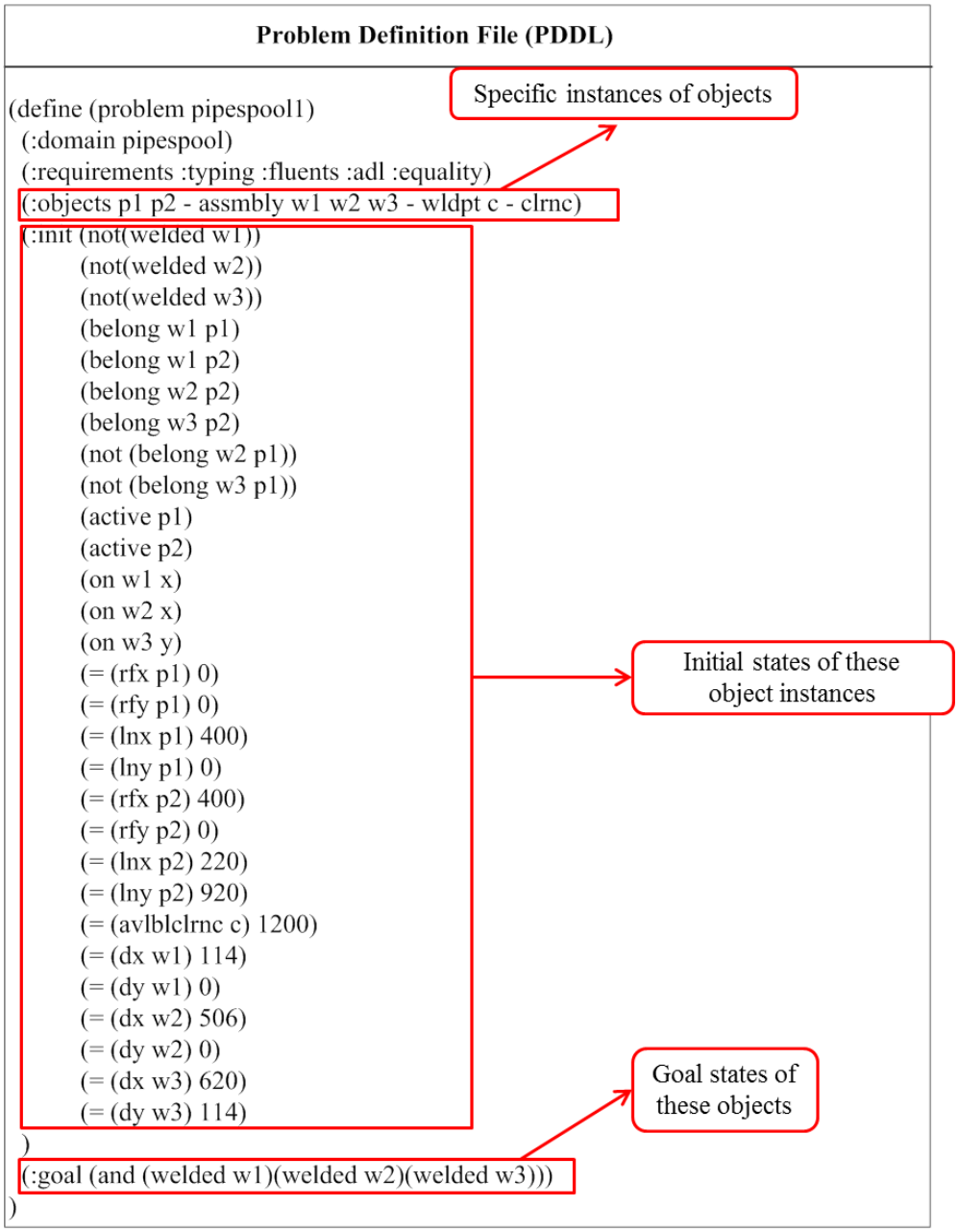


Figure 5-6 An example PDDL problem definition file for pipe spool fabrication

5.5.3 Experiments and results

A series of experiments (Figure 5-7) were conducted to test the capability of PDDL to model and solve pipe spool fabrication sequencing problems. The experiments begin with very simple pipe spools and then gradually move to more complex and more realistic configurations. Three

popular planners (domain-independent) were used in the experiments, namely Metric-FF (Hoffmann 2002), LPRPG (Coles et al. 2008) and LPG (Gerevini and Serina 2002). Metric-FF searches on a state space while the other two search on a plan space.

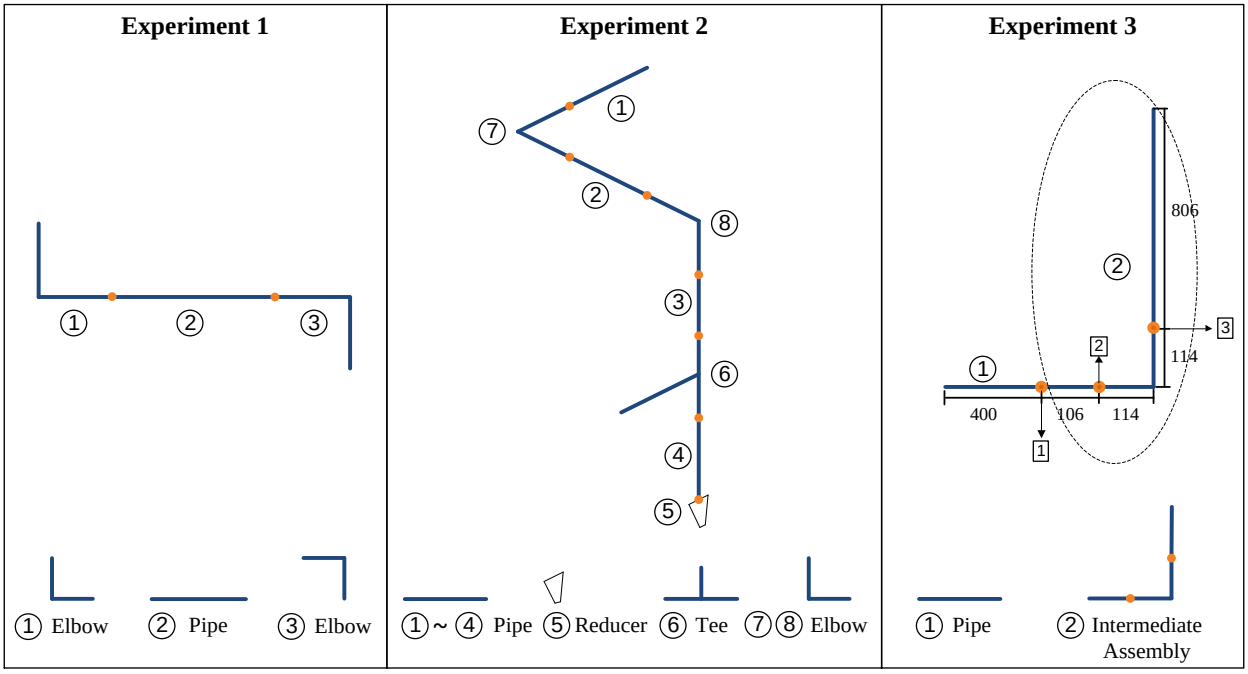


Figure 5-7 Experiments of using PDDL to model and plan sequence for pipe spools

In Experiment 1, a simple pipe spool is designed (Figure 5-7). No numerical values are included. It simply tests if the AI planners can handle the logic aspect of the pipe spool fabrication sequencing problem. Experiment 2 uses a pipe spool with more complex configuration (Figure 5-7). Again, no numerical values are considered in Experiment 2. Although the pipe spool in Experiment 3 seems to have a simpler configuration than that in Experiment 2, it involves the major challenge that coordinates and dimensions of assemblies are considered. The objective is that position welding can be distinguished by the AI planners from roll welding. Coordinates and dimensions of assemblies also need to be updated after each welding operation that creates a new

in-progress assembly. PDDL files for each experiment are attached in Appendix E. Results of these experiments are compiled in Table 1.

Table 5-1 Resulting plans from each AI planner

	Experiment1	Experiment2	Experiment3
LPG	Unsolvable	Unsolvable	Step1: ROLL-X-P1-P2-1 (P1 P2 W1 W3 W2 C)
LPRPG	Unsolvable	Unsolvable	Unsolvable
Metric-FF	Step1: ROLL-FITTING P1 P2 W1 Step2: ROLL-FITTING P1 P3 W2	Step1: ROLL-FITTING-X P1 P7 W1 Step2: ROLL-FITTING-Y P1 P2 W2 Step3: ROLL-FITTING-Y P1 P8 W3 Step4: ROLL-FITTING-Z P1 P3 W4 Step5: ROLL-FITTING-Z P1 P6 W5 Step6: ROLL-FITTING-Z P1 P4 W6 Step7: ROLL-FITTING-Z P1 P5 W7	Step1: ROLL-X-P1-P2-4 (P1 P2 W1 W1 W1 C)

The results obtained from these three experiments indicate that Metric-FF is more capable than the other two AI planners in terms of handling pipe spool fabrication logic. However, Metric-FF has its limitation too. In Experiment 3 where numerical calculation and assignment is involved, Metric-FF could not handle the combination of conditional effects (certain effect of an action is preconditioned) and numerical calculations. One way to get around this limitation is to break down the conditional effects by moving the condition part to the preconditions of the whole action. This requires elaborating original actions to more specific sub-actions. In PDDL, or Lisp language, this is called a "grounding" process. After converting the conditional effect, LPG planner is able to return a solution which is shown in Table1. Metric-FF seems to be able to do the same but a closer check finds that it returns an illogical solution. The challenge regarding the grounding process is that the number of actions defined in the domain file will grow exponentially with the number of welds in the pipe spool. If a pipe spool has N welds, then 2N-1 actions need to be explicitly formulated in the domain file (e.g. a pipe spool with 13 welds

requires 4096 actions to be defined). For extremely complicated pipe spools, it could be computationally prohibitive to find a solution.

5.6 MODULE INSTALLATION SEQUENCING PROBLEM

After the pipe spools are fabricated, some of them are shipped to the assembly yard to build modules while others are sent directly to the construction site for final installation. In the module assembly yard, pipe spools are mounted with other module components (equipment, instruments, electrical cable trays, etc.) on the steel frames. They are assembled together to form various modules, each of which is a basic building block during the on-site installation stage. Every module is sized so that it can be hauled by trucks to the construction site where the industrial facility is constructed.

After being delivered to the construction site, modules are ready to be placed in their final locations. Due to their fast tracking nature (i.e. construction commences before design and procurement is complete), change orders and rush orders are quite common in industrial construction projects. This usually leads to out-of-sequence delivery of modules. Given the limited availability of cranes and the installation constraints, some of these out-of-sequence modules cannot be installed immediately after delivery. Modules that are not to be installed within 24 hours of delivery are stored in a staging area, which will incur additional cost for storing and moving the modules.

Generally, module installation should comply with two basic constraints. The first rule is that modules in elevation cannot be installed unless the module that provides physical support has been installed (Figure 5-8a). The second rule is for modules that sit right next to each other. A situation to be avoided is when modules on both sides are already installed with the one in the

middle still to be installed (Figure 5-8b). This makes it very difficult to maneuver the crane and to complete module lifting and placing.

Current industry practice requires human planners to plan the module installation sequence and to update it based on the current situation of the construction site. However, fast changing conditions on the construction site and unreliable delivery of modules make it difficult for human planners to come up with or to update plans with both efficiency and quality. This chapter investigates the use of domain-independent AI planning to automate this decision-making process and provide human planners with decision support.

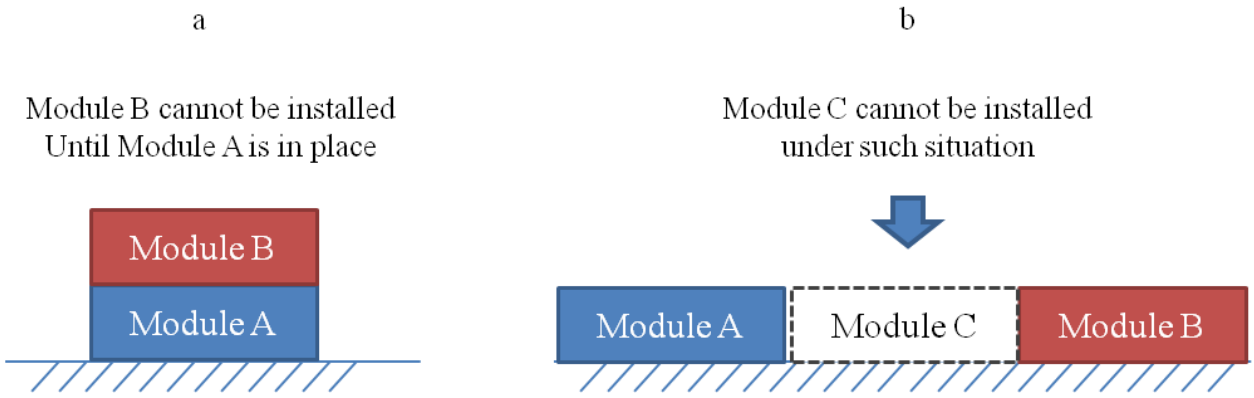


Figure 5-8 Constraints for module installation

5.6.1 Problem abstraction

The module installation sequencing problem can be abstracted to a level suitable to generate PDDL representations and to be solved by AI planners. First, module is the only object type that is involved in the planning problem. A number of predicates are used to describe its states. Modules that are placed on ground level are tagged as *base*. This predicates is intended to differentiate modules on the ground level and those on elevation, since they require different physical support. Three predicates are employed to model the adjacency between modules,

including *AdjX*, *AdjY*, and *AdjZ*. Modules that can be described by these predicates are sitting next to each other on the left-to-right direction, the front-to-end direction, and the vertical direction, respectively. Relative positions of modules are enough to impose the constraints on the action formulation. Therefore, actual geometries and locations of modules are not needed here. To indicate the installation progress, modules are also described as placed or not placed.

The only type of action involved here is Place. However, preconditions might vary with different modules at different locations. For example, modules on the ground level (or base modules) do not need physical support from other modules. Modules that have adjacent modules only on one side would never encounter similar situations, as in Figure 5-8b. Modules that have four adjacent modules around them (i.e. two on the left-to-right direction as well as on the front-to-back direction) require at least one end on each direction to be open. The number of actions increases when more adjacency situations are considered.

5.6.2 PDDL domain and problem definition representation

All aforementioned information should be expressed in the PDDL language. Figure 5-9 and Figure 5-10 show an example domain definition file and an example problem definition file for the module installation sequencing problem.

Domain Definition File (PDDL)

```
(define (domain modulesequencing)
 (:requirements :conditional-effects :equality :strips :typing :negative-preconditions :fluents :disjunctive-preconditions)
 (:types module) ;; module
 (:constants x y z - axs)
 (:predicates
  (placed ?m - module)     ;; module is placed in its final location
  (base ?m - module)       ;; base module (bottom one)
  (AdjZ ?m ?m0 - module)  ;; module m is on top of m0
  (AdjX ?m ?m0 - module) ;; module m is to the Left of m0
  (AdjY ?m ?m0 - module);; module m is in front of m0
  )

 (:action place_1
            :parameters (?m ?m2 - module)
            :precondition (and (base ?m)  ;; m is a base module
                                          (and(not (=?m ?m2))(AdjX ?m ?m2))
                                          (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                                          (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                                          (not (exists (?m4 - module)(AdjY ?m ?m4)))
                                    (not(placed ?m))  ;; m is not placed
                                )

            :effect (and (placed ?m))
 )
  … …
)
```

Figure 5-9 An example PDDL domain definition file for module installation

Problem Definition File (PDDL)

```
(define (problem cubeOf9modules)
 (:domain modulesequencing)
 (:requirements :typing :fluents)
 ;; m z,x,y z(AdjZ), x(AdjX), y(AdjY)
 (:objects m014A m014B m007A m007B - module)
 (:init (base m014A)
        (base m007A)
        (AdjZ m014B m014A)
        (AdjZ m007B m007A)
        (AdjX m014A m007A)
        (AdjX m014B m007B)
  )

 (:goal (and (placed m014A)(placed m014B)(placed m007A)(placed m007B)))
)
```

Figure 5-10 An example PDDL problem definition file for module installation

5.6.3 Experiments and results

Likewise, a series of experiments are conducted to test the capability of PDDL to model and to solve module installation sequencing problems. Experiments are also designed with increasingly complexity (i.e. increasing number of different adjacency situations). Test cases are extracted from a the same Kearl oil sands project as used in Chapter 4. The entire pipe rack area is focused in this chapter (Figure 5-11). They are also categorized into scenarios by the complexity of adjacency situations. For each scenario, two or three experiments are performed. Since no numerical calculation is involved in the module installation planning problems, Metric-FF is selected to be used in experiments based on its performance in pipe spool fabrication sequencing experiments.

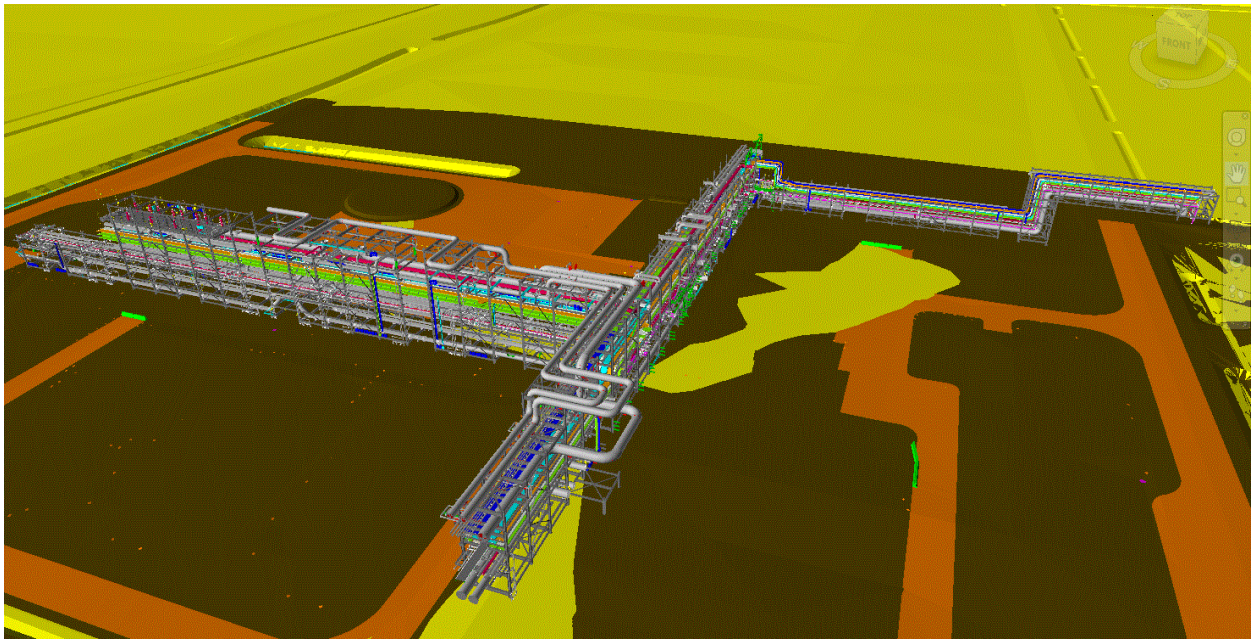


Figure 5-11 Pipe-rack modules 3D model from Kearl Initial Development (KID)

(1) Module Installation Scenario 1

Adjacent modules only happen on the left-to-right direction and on the vertical direction. Figure 5-12 shows the simplified graphical model of modules for Scenario 1. It is assumed in Experiment 1 that no modules have been installed. Experiment 2 has the same layout of modules but assumes that modules 006A and 006B have been installed. Generated module installation sequences are presented in Table 2. PDDL files for Experiment 1 and 2 are attached in Appendix E.

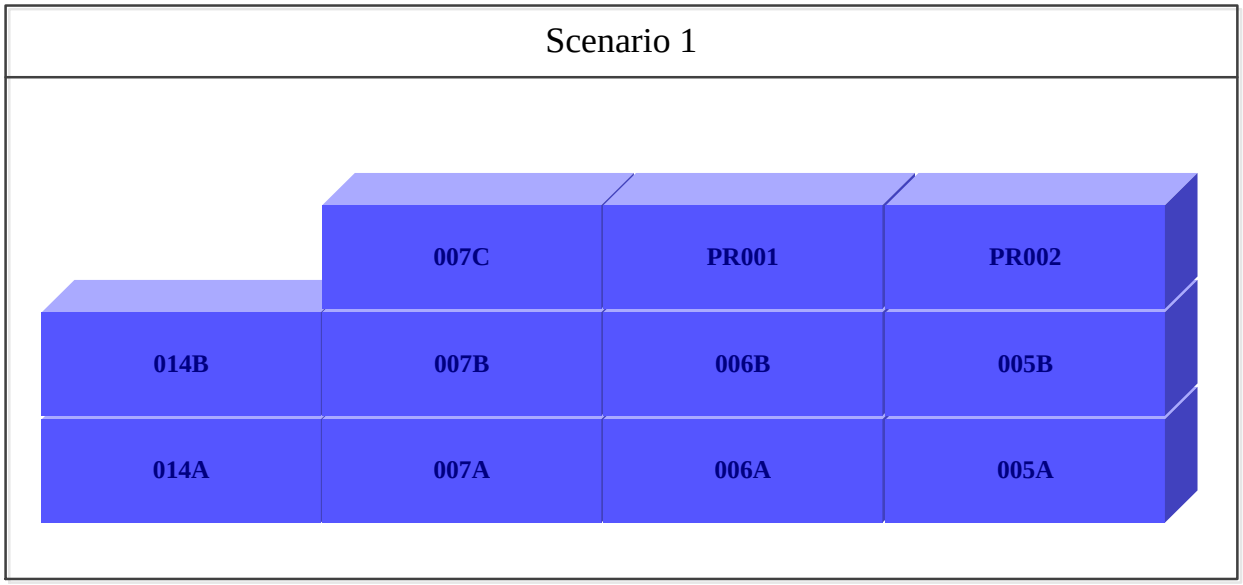


Figure 5-12 Simplified model for modules in scenario 1

Table 5-2 Generated plans for module installation scenario 1

	Experiment 1	Experiment 2
Metric-FF	Step 0: PLACE_5 M014A 1: PLACE_1 M014B 2: PLACE_1 M007A 3: PLACE_1 M007B 4: PLACE_1 M007C 5: PLACE_1 M006A 6: PLACE_1 M006B 7: PLACE_1 PR001 8: PLACE_1 M005A 9: PLACE_1 M005B 10: PLACE_1 PR002	Step 0: PLACE_5 M007A 1: PLACE_1 M014A 2: PLACE_1 M007B 3: PLACE_1 M014B 4: PLACE_1 M007C 5: PLACE_1 PR001 6: PLACE_1 M005A 7: PLACE_1 M005B 8: PLACE_1 PR002

(2) Module Installation Scenario 2

More modules are added in this scenario. Adjacent modules occur not only in the left-to-right direction and the vertical direction but also in the front-to-back direction (Figure 5-13). Three

experiments are conducted in Scenario 2. Experiment 3 assumes that no module has been installed. Experiment 4 assumes that module 006A and module 006B have already been placed while Experiment 5 assumes that module 006A, 006B, 013A and 013B have all been installed. Generated plans are presented in Table 3. PDDL files for Experiment 3, 4 and 5 are attached in Appendix E.

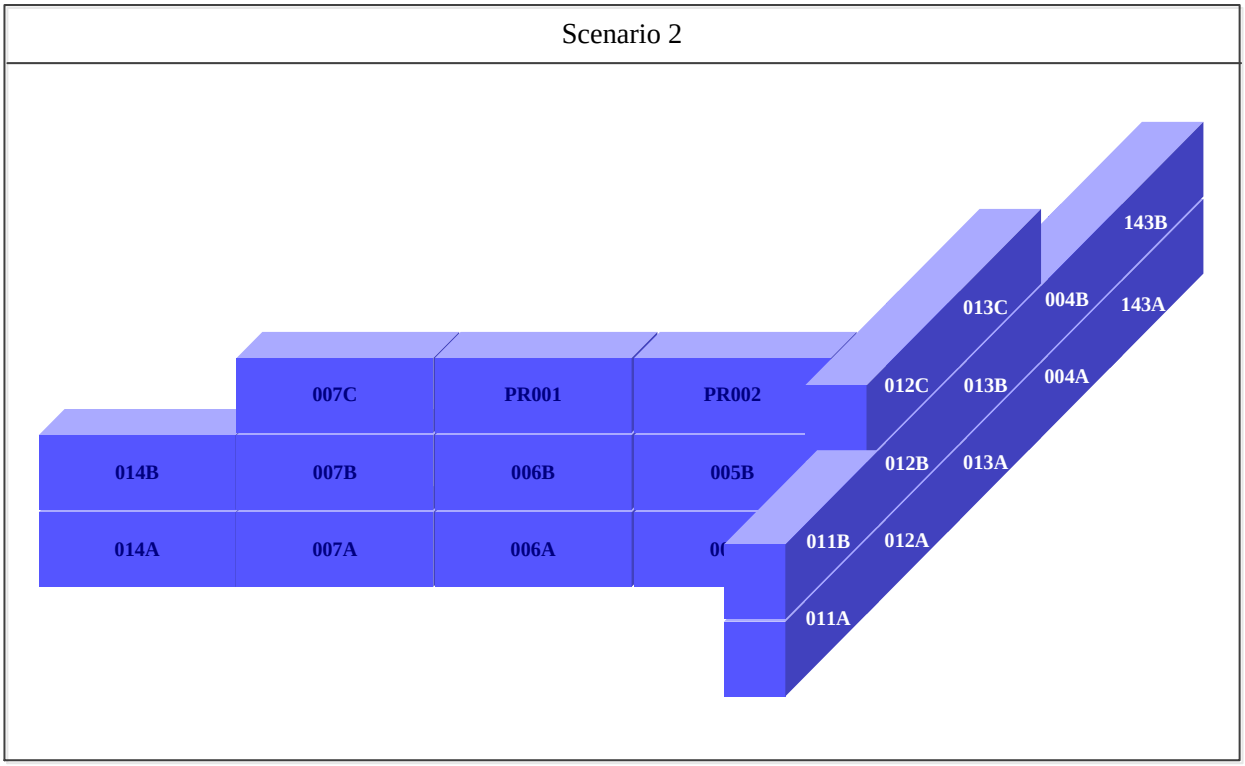


Figure 5-13 Simplified model for modules in scenario 2

Table 5-3 Generated plans for module installation scenario 2

	Experiment 3	Experiment 4	Experiment 5
Metric-FF	step 0: PLACE_1 M014A 1: PLACE_2 M014B 2: PLACE_5 M007A 3: PLACE_6 M007B 4: PLACE_2 M007C	step 0: PLACE_5 M007A 1: PLACE_1 M014A 2: PLACE_6 M007B 3: PLACE_2 M014B 4: PLACE_2 M007C	step 0: PLACE_5 M007A 1: PLACE_1 M014A 2: PLACE_6 M007B 3: PLACE_2 M014B 4: PLACE_2 M007C

	5: PLACE_5 M006A 6: PLACE_6 M006B 7: PLACE_6 PR001 8: PLACE_5 M005A 9: PLACE_6 M005B 10: PLACE_6 PR002 11: PLACE_7 M011A 12: PLACE_8 M011B 13: PLACE_13 M012A 14: PLACE_14 M012B 15: PLACE_15 M012C 16: PLACE_11 M013A 17: PLACE_12 M013B 18: PLACE_10 M013C 19: PLACE_11 M004A 20: PLACE_12 M004B 21: PLACE_9 M143A 22: PLACE_10 M143B	5: PLACE_6 PR001 6: PLACE_5 M005A 7: PLACE_6 M005B 8: PLACE_6 PR002 9: PLACE_7 M011A 10: PLACE_8 M011B 11: PLACE_13 M012A 12: PLACE_14 M012B 13: PLACE_15 M012C 14: PLACE_11 M013A 15: PLACE_12 M013B 16: PLACE_10 M013C 17: PLACE_11 M004A 18: PLACE_12 M004B 19: PLACE_9 M143A 20: PLACE_10 M143B	5: PLACE_6 PR001 6: PLACE_5 M005A 7: PLACE_6 M005B 8: PLACE_6 PR002 9: PLACE_13 M012A 10: PLACE_7 M011A 11: PLACE_14 M012B 12: PLACE_8 M011B 13: PLACE_15 M012C 14: PLACE_10 M013C 15: PLACE_11 M004A 16: PLACE_12 M004B 17: PLACE_9 M143A 18: PLACE_10 M143B

(3) Module Installation Scenario 3

This is the original planning problem as shown in Figure 5-11. The simplified graphical model is shown in Figure 5-14. Three experiments are conducted in this scenario. Experiment 6 assumes that no module has been installed. Experiment 7 assumes that modules 005A, 005B, 004A and 004B have been installed. Experiment 8 assumes that 005A, 005B, 004A, 004B, and PR113 have already been installed. PDDL files for Experiment 6, 7 and 8 are attached in Appendix E.

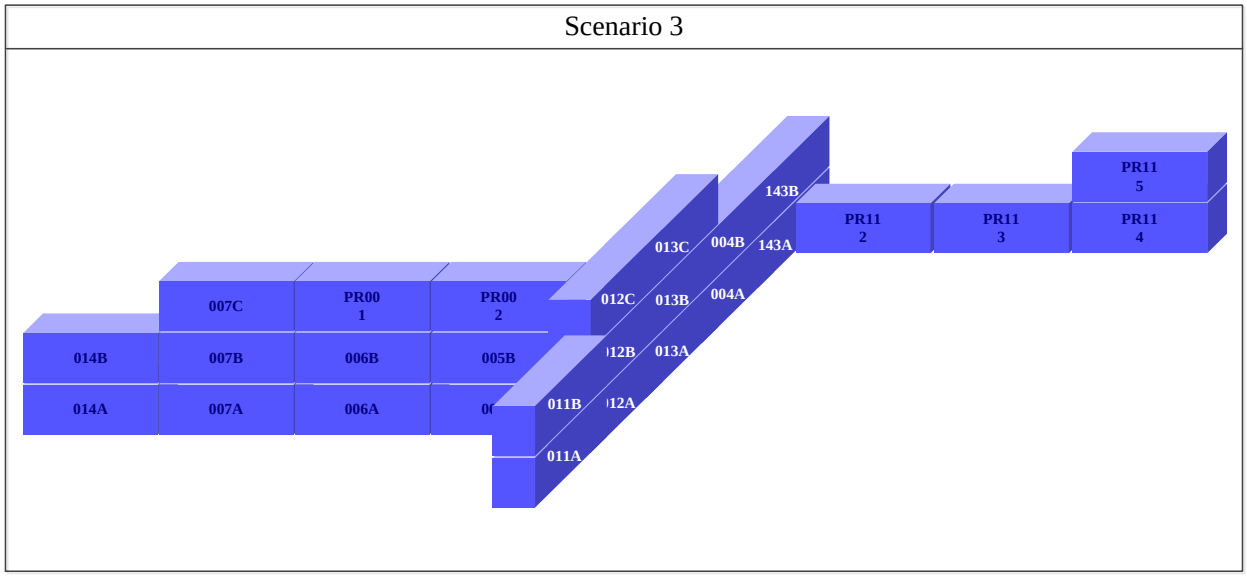


Figure 5-14 Simplified model for modules in scenario 3

Table 5-4 Generated plans for module installation scenario 3

	Experiment 6	Experiment 7	Experiment 8
Metric-FF	step 0: PLACE_1 M014A	step 0: PLACE_3 PR114	step 0: PLACE_3 PR114
	1: PLACE_2 M014B	1: PLACE_0 PR115	1: PLACE_0 PR115
	2: PLACE_5 M007A	2: PLACE_5 PR113	2: PLACE_5 PR112
	3: PLACE_6 M007B	3: PLACE_5 PR112	3: PLACE_16 M143A
	4: PLACE_2 M007C	4: PLACE_16 M143A	4: PLACE_10 M143B
	5: PLACE_5 M006A	5: PLACE_10 M143B	5: PLACE_11 M013A
	6: PLACE_6 M006B	6: PLACE_11 M013A	6: PLACE_12 M013B
	7: PLACE_6 PR001	7: PLACE_12 M013B	7: PLACE_10 M013C
	8: PLACE_5 M005A	8: PLACE_10 M013C	8: PLACE_13 M012A
	9: PLACE_6 M005B	9: PLACE_13 M012A	9: PLACE_14 M012B
	10: PLACE_6 PR002	10: PLACE_14 M012B	10: PLACE_15 M012C
	11: PLACE_7 M011A	11: PLACE_15 M012C	11: PLACE_7 M011A
	12: PLACE_8 M011B	12: PLACE_7 M011A	12: PLACE_8 M011B
	13: PLACE_13 M012A	13: PLACE_8 M011B	13: PLACE_6 PR002
	14: PLACE_14 M012B	14: PLACE_6 PR002	14: PLACE_5 M006A
	15: PLACE_15 M012C	15: PLACE_5 M006A	15: PLACE_6 M006B
	16: PLACE_11 M013A	16: PLACE_6 M006B	16: PLACE_6 PR001
	17: PLACE_12 M013B	17: PLACE_6 PR001	17: PLACE_5 M007A
	18: PLACE_10 M013C	18: PLACE_5 M007A	18: PLACE_6 M007B

	19: PLACE_11 M004A 20: PLACE_12 M004B 21: PLACE_16 M143A 22: PLACE_10 M143B 23: PLACE_5 PR112 24: PLACE_5 PR113 25: PLACE_3 PR114 26: PLACE_0 PR115	19: PLACE_6 M007B 20: PLACE_2 M007C 21: PLACE_1 M014A 22: PLACE_2 M014B	19: PLACE_2 M007C 20: PLACE_1 M014A 21: PLACE_2 M014B

The longest processing time takes place in Experiment 6. It takes about 57 seconds to find the plan. Processing times for other experiments are all less than 45 seconds.

5.7 CONCLUSIONS

This chapter investigates the use of domain-independent AI planning technique to plan two industrial construction processes, pipe spool fabrication and module installation. PDDL and three PDDL compliant AI planners (Metric-FF, LPRPG, LPG) are selected due to their expressiveness and efficiency in searching capability. A number of experiments have been conducted to test the effectiveness of PDDL technique in terms of modeling and solving the industrial construction planning problems. A number of experiments are conducted for each domain. Results show that PDDL is sufficiently expressive to model the logical and numerical parts of both construction processes. However, the parsing capability of existing AI planners is somewhat limited, particularly in presence of both numeric calculations and conditional effects. This makes PDDL not quite suitable to solve pipe spool fabrication sequencing problems. On the other hand, however, it has been observed that PDDL technique is both effective and efficient to solve module installation sequencing problems where only logic calculation is involved.

5.8 REFERENCES

Aalami, F., Kunz, J., and Fischer, M. (1998). "Model-based sequencing mechanisms used to automate activity sequencing." *Working Paper No. 50*, CIFE, Stanford Univ., Stanford, Calif.

Coles, A.I., Fox, M., Long, D. and Smith, A.J. (2008). "A Hybrid Relaxed Planning Graph-LP Heuristic for Numeric Planning Domains." *Proc., Eighteenth Int. Conf. on Automated Planning and Scheduling (ICAPS 08)*, Sydney, Australia, September.

Darwiche, A., Levitt, R., and Hayes-Roth, B. (1988). "OARPLAN: Generating Project Plans by Reasoning about Objects, Actions and Resources." *AI EDAM*, 2(3), 169-181.

Echeverry, D., Ibbs, C. W., and Kim, S. (1991). "Sequencing knowledge for construction scheduling." *J. Constr. Engrg. and Mgmt.*, ASCE, 117(1), 118–130.

Fischer, M. A., and Aalami, F. (1996). "Scheduling with Computer-Interpretable Construction Method Models." *J. Constr. Eng. Manage.*, 122(4), 337–347.

Gerevini, A. and Serina, I. (2002). "LPG: a Planner based on Local Search for Planning Graphs." *Proc., Sixth Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS'02)*, AAAI Press, Toulouse, France.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory and Practice*, Elsevier Inc. San Francisco.

Gray, C. (1986). "Intelligent construction time and cost analysis." *Journal of Construction Mgmt and Economics*, 4(2), 135–150.

Hendrickson, C., Zozaya-Gorostiza, C., Rehak, D., Baracco-Miller, E., and Lim, P. (1987). "Expert System for Construction Planning." *J. Comput. Civ. Eng.*, 1(4), 253–269.

Hoffmann, J. (2002). "Extending FF to Numerical State Variables." *Proc., 15th European Conf. on Artificial Intelligence*, Lyon, France, July.

Hu, D., and Mohamed, Y. (2010). "State-Based Simulation Mechanism for Facilitating Project Schedule Updating." *Proc., Construction Research Congress 2010*, ASCE, Banff, Alberta, Canada, 369–378.

Kartam, N. and Levitt, R. (1990). "Intelligent Planning of Construction Projects." *J. Comput. Civ. Eng.*, 4(2), 155–176.

Koo, B., Fischer, M., and Kunz, J. (2007). "Formalization of construction sequencing rationale and classification mechanism to support rapid generation of sequencing alternatives." *J. Computing in Civ. Engrg.*, 21(6), 423–433.

Levitt, R. E. , and Kunz, J. C. (1987). "Using artificial intelligence techniques to support project management." *Journal of Artificial Intelligence in Engrg. Design, Analysis and Manufacturing* , 1 (1), 3–24.

McDermott, D., and the AIPS-98 Planning Competition Committee (1998)."PDDL–the planning domain definition language." *Technical report,* < http://www.cs.washington.edu/education/courses/cse473/06sp/pddl.pdf > (Dec., 2012).

Navinchandra, D., Sriram, D., and Logcher, R. D. (1988). "GHOST: Project Network Generator." *J. Computing in Civ. Eng.*, ASCE, 2(3), 239-254.

Newell, A., and Simon, H., (1972). *Human problem solving*. Prentice Hall, Englewood Cliffs, N.J.

Taghaddos, H., AbouRizk, S., Mohamed, Y., and Hermann, U. (2012). ”Simulation-Based Auction Protocol for Resource Scheduling Problems.” J. Constr. Eng. Manage., 138(1), 31–42.

CHAPTER 6. Conclusions

6.1 CONCLUSIONS

Mega industrial projects, such as those involved in the oil sands, are frequently plagued by cost overruns and schedule slippages. Insufficient project planning is identified as one of major contributing factors to poor project performance. The main objective of this research is to explore and to develop automated solutions for planning and scheduling two major industrial construction stages: shop fabrication and on-site construction. Planning here refers to planning the sequences of construction processes that meet relevant technical or physical constraints and/or optimize certain performance metrics. Major conclusions of this research are presented as follows.

First, this research develops a new simulation model structuring methodology that accommodates the characteristics of industrial fabrication shops—shop floor operations are repetitive, but the routing (or the sequence of operations) varies with shop products. This is achieved through an integrated entity information model and a state-based entity routing mechanism, which enable entities to autonomously route through the simulation model. Advantages of the new simulation model structuring methodology include: (1) less cluster in the graphical representation of simulation models and (2) the facilitation of both scheduling and schedule updating for shop fabrication.

Following this simulation structuring methodology, a detailed simulation model is developed for pipe spool fabrication shops. This model is first used in a simulation experiment, the result of which shows that the sequence of pipe spool fabrication could have significant impact on the spool cycle time. A search for problem solving techniques in computing science finds that

Dynamic Programming (DP) is a good candidate to automate the sequencing of spool fabrication. A DP-based algorithm is customized for the pipe spool sequencing problem. A set of real-life pipe spools are used to evaluate the performance of the DP algorithm. For all pipe spools, the algorithm successfully returns an optimal sequence in terms of the minimum number of position welds. To quantify the productivity improvement, two simulation experiments are conducted. The results indicate 45% reduction in the total number of position welds, which translates into a reduction in the total cycle time by a range of 4.8% to 12%.

Highly compressed construction stages, frequent interference between trades and congested jobsites require a specialized scheduling tool for on-site construction of mega industrial projects. To meet these requirements, this research presents a time-stepped simulation-based scheduling framework that (1) complies with various constraints of work packages (precedence dependency, time dependent resource limit, calendar, as well as time constraint), (2) dynamically allocates resources to work packages, and (3) accounts for jobsite congestion constraints of work areas. A real industrial construction case was used to evaluate the effectiveness of the scheduling framework. The generated schedule was also compared with those obtained from popular project scheduling software Microsoft Project 2010, and Primavera P6 (R 8.2). Results show that the schedule generated from the simulation-based scheduling framework (86 working days) is 13 working days shorter than those returned by MS Project and P6 (99 working days) and the duration reduction is about 13%.

This research also investigates the use of domain independent AI planning technique to automatically plan operation sequences for two industrial construction processes, namely (1) pipe spool fabrication and (2) on-site module installation. A standard AI planning language called Planning Domain Definition Language (PDDL) is used to model these processes and

several AI planners that can handle PDDL representations are employed to identify feasible sequences to accomplish the construction processes without violating any physical or technical constraints. For pipe spool fabrication, a number of experiments are conducted and results show that the combination of conditional effects and numeric calculations in the PDDL representation pose a parsing challenge for existing planners, which makes AI planning unsuitable to solve pipe spool fabrication sequencing problems. For on-site module installation, a set of experiments are also conducted. Only one of AI planners, Metric-FF, is selected to be used in experiments based on its performance in pipe spool fabrication sequencing experiments. Results show that, in each experiment, Metric-FF successfully returns a feasible sequence plan that satisfies all the module installation constraints. For the most complex experiment, it takes about 57 seconds to find the plan. Processing time for other experiments are all less than 45 seconds.

6.2 MAJOR CONTRIBUTIONS

The main contributions of this thesis can be summarized as follows:

(1) Introducing a new simulation model structuring methodology that accounts for complex routing issues in industrial fabrication shops.

(2) Developing a Special Purpose Simulation template based on the above methodology for pipe spool fabrication.

(3) Customizing a DP-based algorithm to automatically identify sequences for spool fabrication that require the minimum number of position welds.

(4) Developing a time-stepped simulation-based scheduling framework for on-site construction of industrial projects, which 1) complies with various constraints of work packages (precedence dependence, time-dependent resource limit, calendar, as well as

time constraint), 2) dynamically allocates resources to work packages, and 3) accounts for jobsite congestion constraints of work areas. This development is currently being put into use by PCL industrial Management Inc.

(5) Investigating the use of domain independent AI planning to automate the sequencing process for both spool fabrication and on-site module installation.

(6) All above mentioned developments, though relevant to company practice, can be implemented in other industrial fabrications and industrial construction sites.

6.3 LIMITATIONS AND FUTURE WORK

This research also exposes a number of areas that have potential for improvement. Further research efforts could be invested in following areas:

(1) Dynamic programming algorithm should be tested with more real-life pipe spools to validate its contribution to the total cycle time reduction. Durations of operations use average numbers (from time study of real life fabrication shop) and should be modeled by probability distributions to add stochastic nature into simulation. The order of entering pipe spools into the simulation model will be randomized and the results of experiments will be compared. Pipe spool fabrication heuristics will be added into the DP algorithm to optimize more objectives in addition to postion welding.

(2) the use of genetic algorithm to sequence spool fabrication will be investigated and compared to the dynamic programming algorithm.

(3) Domain independent AI planning is one of the major research topics in artificial intelligence that keeps evolving. The expressiveness of its modeling languages and the searching capability of AI planners have improved significantly and will continue to do

so in the future. This research draws a tentative conclusion that AI planning is not quite suitable to solve the pipe spool fabrication sequencing problems due to the limited parsing capability of existing planners. However, this limitation is very likely to be overcome in the future. It is also worth exploring opportunities of applying domain independent AI planning to more construction processes.

(4) The time-stepped simulation based scheduling framework for site construction of industrial projects can be enhanced in a number of areas. First, the framework only focuses on various skilled workers on the construction site. It can be enhanced by considering major equipment (e.g. cranes) as one of the resource constraints to work packages. The dynamic resource allocation algorithm is currently a heuristic optimization process which can also be enhanced by incorporating other scheduling optimization techniques, e.g. meta-heuristic techniques, Multi-Agent Resource Allocation (MARA). The time window in which the resource allocation decision is made can be extended from a single time step to several time steps. For example, if a simulation time step represents a day in the real world, then resource allocation could be made not just for the current day but rather for the next 10 days.

(5) The time-stepped simulation is being implemented in PCL Industrial Management Inc. the improvement of scheduling process will be further tracked and evaluated.

(6) Simulation models developed for industrial fabrication shops and site construction can be integrated through a distributed simulation technique called High Level Architecture (HLA) into a single distributed simulation model called a federation. This aggregated simulation model provides a cost-effective virtual environment where industrial construction professionals can study the interactions between different stages in the

industrial project and develop efficient strategies to enhance the coordination and to mitigate the disruptive effects.

Appendix A

This is the VB.NET code for the 'Spool_Generation' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go
Imports System.Data
Imports System.Data.OleDb

Namespace SimphonyScript
Public Module Script
'Your functions here


Public Function Spool_Generation_OnCreate(ob As
CFCSim_ModelingElementInstance, x As Double, y As Double) As Boolean handles
Scripting.OnCreateEvent

      'call to the base constructor to create the element
      ob.OnCreate(x,y,True)

      'Add Attribues
      'to define Database file address
      ob.AddAttribute("SpoolDBFileAddress", "Database file
address",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRe
presentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      'to specify the 'Unit' table name
      ob.AddAttribute("SpoolDBTableName", "Spool Table
Name",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepre
sentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("SpoolItemsDBQueryName", "Spool Items Query
Name",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepre
sentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("SpoolComponentsDBQueryName", "Spool Components Query
Name",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepre
sentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("Spool_Component_Item_Relationship", "relationship
between Spool Items and
```

```
Components",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("ActCuttingDBQueryName", "Cutting activity query
name",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepre
sentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("ActHandlingDBQueryName", "Handling activity query
name",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepre
sentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("ActFittingDBQueryName", "Fitting activity query
name",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepre
sentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("ActWeldingDBQueryName", "Welding activity query
name",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepre
sentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      'to create a spool table
      ob.AddAttribute("Spool",
"Spool",CFC_AttributeInternalRepresentation.CFC_Array,CFC_AttributeExternalRe
presentation.CFC_Table,CFC_AttributeAccess.CFC_ReadWrite)

      'how many entities should the element produce in total
      ob.AddAttribute("NA", "Number of Arrivals",
CFC_AttributeInternalRepresentation.CFC_Numeric,
CFC_AttributeExternalRepresentation.CFC_Singular,
CFC_AttributeAccess.CFC_ReadWrite,1, 100000000000)

      'the time at which the first entity will be created.
      ob.AddAttribute("TFA", "Time of First Arrival",
CFC_AttributeInternalRepresentation.CFC_Numeric,
CFC_AttributeExternalRepresentation.CFC_Singular,
CFC_AttributeAccess.CFC_ReadWrite, 0, 100000000000)
      ob("TFA").Value=0

      'the time interval between the creations of any two entities. The
      'user can set this time to a constant or a distribution.
      ob.AddAttribute("TBA", "Time Between Arrivals",
CFC_AttributeInternalRepresentation.CFC_Distribution,
CFC_AttributeExternalRepresentation.CFC_Singular,
CFC_AttributeAccess.CFC_ReadWrite, 0,100000000000 )
      ob("TBA").Value=0

      ob.AddAttribute("Fired", "the number of fired
entities",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExtern
alRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_Hidden)

     ob.AddConnectionPoint("Out", x + 70, y + 25, TConnectionType.COutput, 5)
      return true
End Function



Public Function Spool_Generation_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle
```

```
      r.Size = new SizeF(65, 55)
      'To add an image you have to add the image to the templates "bitmap"
collection

      Dim image As new GoImage
     image.Image  =  ob.ModelingElement.Template.RetrieveImage("CreateEnt.bmp")
      'image.Position = new PointF(12.5f,5)
      image.Position = new PointF(2, 2)
      image.Size = new SizeF(30, 30)


      'creates a text with the quantity to create
      Dim text1 As new GoText
      If ob("NA").Calculation=CFC_AttributeCalculation.CFC_Simple Then
            text1.Text = Convert.ToString(ob("NA").Value)
      Else
            text1.Text = "Qty: (Formula)"
      End If
      text1.FontSize = 8
      'text.Position = new PointF(25, 45)
      text1.Position = new PointF(25, 43)

      Dim text2 As New GoText
      text2.Text = "Spool_Gen"
      text2.FontSize = 8
      text2.Position = new PointF(6,20)


      'adds all elements to a GoGroup and returns it
      Dim g As new GoGroup
      'g.Add(shape)
      g.Add(r)
      g.Add(text1)
      g.Add(text2)
      g.Add(image)
      return g

End Function


'Public Function Spool_Generation_OnCheckIntegrity(ob As
CFCSim_ModelingElementInstance) As Boolean handles
Scripting.OnCheckIntegrityEvent


'End Function


'Public Sub Spool_Generation_OnAttributeChanged(ob As
CFCSim_ModelingElementInstance, attr As CFCSim_Attribute) handles
Scripting.OnAttributeChangedEvent

'End Sub


Public Sub Spool_Generation_OnSimulationInitialize(ob As
CFCSim_ModelingElementInstance) handles Scripting.OnSimulationInitializeEvent
```

```
        'declares the entity firing event for use later
        ob.AddEvent("FireEntity")

        Dim Address As String
        Dim TableName As String

        Address = ob("SpoolDBFileAddress").Value
        TableName = ob("SpoolDBTableName").Value

        'To get the spool records from Database
        Dim cn As system.Data.OleDb.OleDbConnection
        Dim MydataAdapter As System.Data.OleDb.OleDbDataAdapter
        Dim Query As String = "SELECT * FROM " & TableName & " ORDER BY
Priority DESC"

        'Have to create an instance of dataset!!!
        cn = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & Address)

        'Fill Data
        Try
        cn.Open()
        MyDataAdapter = New OleDbDataAdapter(Query, cn)
        Dim MyDataTable = New DataTable()
        MyDataAdapter.Fill(MyDataTable)
        ob("Spool").SetDataTable(MyDataTable)
        Catch exp As Exception
        MessageBox.Show(exp.Message)
        finally
        cn.close()
        End Try

        'Update the arrtibute of "NA" (Number of Arrivals)
        ob("NA").Value = ob("Spool").RowCount

        ob.Invalidate()


End Sub


Public Sub Spool_Generation_OnSimulationInitializeRun(ob As
CFCSim_ModelingElementInstance, runNum As Int32) handles
Scripting.OnSimulationInitializeRunEvent

        'scheduals the fire entity event on a new entity at the time the first
entity is to be made
        ob.ScheduleEvent(ob.AddEntity(),"FireEntity",ob("TFA").Value)
        ob("Fired").Value=0
        ob.Invalidate()

End Sub
```

```
Public Sub Spool_Generation_OnSimulationProcessEvent(ob As
CFCSim_ModelingElementInstance, myEvent As String, entity As CFCSim_Entity)
handles Scripting.OnSimulationProcessEventEvent

      'quit if we have fired enough entities
      'MessageBox.Show("OK!")

      Select Case MyEvent

      Case "FireEntity"

      If ob("Fired").Value < cint(ob("NA").Value) Then

            Dim newEntity As CFCSim_Entity

            ob("Fired").Value = ob("Fired").Value + 1

            newEntity = ob.AddEntity()

            newEntity("EntityType") = "Spool"
            'MessageBox.Show("OK!")

            'set attributes to each spool entity according to ob("Spool")
            Dim i as Integer

            For i = 0 to ob("Spool").ColumnCount - 1

                  newEntity(ob("Spool").ColumnLabel(i)) =
ob("Spool").GetValueRC(ob("Fired").Value-1, i)
                  If ob("Spool").ColumnLabel(i)= "SpoolState" Then
                        newEntity(ob("Spool").ColumnLabel(i)) = "Issued"

                  End If

            Next

            'Add referrence of corresponding 'spool_element' as one of the
attributes of the entity
            Dim MySpoolElement As CFCSim_ModelingElementInstance
            For Each MySpoolElement In SimEnvironment.Elements.Values

                  If MySpoolElement.ElementType = "Spool_Element" Then

                        If MySpoolElement("JobControlNumber").Value =
newEntity("JobControlNumber") Then
                              newEntity("SpoolElement") = MySpoolElement
                                    Exit for
                        End If
                  End If

            Next

            'transfer the entity out
            ob.TransferOut(newEntity)

            'To inform users what happened in the simulation
```

```
            Trace.WriteLine("Entity: " & newEntity.ID & " Created",
"Simulation")

      End If

      End Select

      'To schedule the next firing of an entity
      If ob("Fired").Value < cint(ob("NA").Value) Then
            ob.ScheduleEvent(Entity, "FireEntity", ob("TBA").Value)
      End If

End Sub




End Module
End Namespace
```

This is the VB.NET code for the 'Spool_Element' Element

```
 'Simphony.NET Template Code

'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go

Namespace SimphonyScript
Public Module Script
'Your functions here




Public Function Spool_Element_OnCreate(ob As CFCSim_ModelingElementInstance,
x As Double, y As Double) As Boolean handles Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)
      ob.SetNumCoordinates(2)
      ob.Coordinates(0) = new System.Drawing.PointF(x, y)
      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)
```

```
      ob.AddAttribute("JobControlNumber", "ID of
Spool",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepr
esentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      ob.AddAttribute("Priority", "Priority of
Spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      ob.AddAttribute("Weight", "Weight of
Spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      ob.AddAttribute("DiameterInches", "DiameterInches of
Spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      ob.AddAttribute("Length", "Length of
Spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      ob.AddAttribute("IssuanceDate", "Issuance Date of
Spool",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepr
esentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      ob.AddAttribute("BayID", "ID of bay in which the spool is
processed",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExter
nalRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      ob.AddAttribute("Duration", "Duration of
Spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      ob.AddAttribute("SpoolState", "the states of a
spool",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepr
esentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      ob.AddAttribute("NumOfSpoolItems", "Number of spool items belonging to
the
spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadOnly)
      ob("NumOfSpoolItems").Value = 1

      ob.AddAttribute("NumOfSpoolComponents", "Number of spool components
belonging to the
spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadOnly)
      ob("NumOfSpoolComponents").Value = 1

      ob.AddAttribute("NumOfCutting", "Number of cutting for the
spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadOnly)
      ob("NumOfCutting").Value = 1

      ob.AddAttribute("NumOfHandling", "Number of handling for the
spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadOnly)
      ob("NumOfHandling").Value = 1
```

```
      ob.AddAttribute("NumOfFitting", "Number of fitting for the
spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadOnly)
      ob("NumOfFitting").Value = 1

      ob.AddAttribute("NumOfWelding", "Number of welding for the
spool",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadOnly)
      ob("NumOfWelding").Value = 1

      'Spool items refer to cut pipes and fittings
      ob.AddAttribute("SpoolItems", "Spool items of the
spool",CFC_AttributeInternalRepresentation.CFC_Array,CFC_AttributeExternalRep
resentation.CFC_Table,CFC_AttributeAccess.CFC_ReadWrite)
      ob("SpoolItems").SetRC(1,11)
      For i As Integer =0 to ob("NumOfSpoolItems").Value - 1
            For j As Integer = 0 to 10
                  ob("SpoolItems").SetValueRC(i,j,0)
            Next
      Next

      ob("SpoolItems").DataTable.Columns(0).ColumnName = "JobControlNumber"
      ob("SpoolItems").DataTable.Columns(1).ColumnName = "SpoolPartID"
      ob("SpoolItems").DataTable.Columns(2).ColumnName = "Priority"
      ob("SpoolItems").DataTable.Columns(3).ColumnName = "PartTypeID"
      ob("SpoolItems").DataTable.Columns(4).ColumnName = "Weight"
      ob("SpoolItems").DataTable.Columns(5).ColumnName = "Size"
      ob("SpoolItems").DataTable.Columns(6).ColumnName = "MaterialTypeID"
      ob("SpoolItems").DataTable.Columns(7).ColumnName = "Length"
      ob("SpoolItems").DataTable.Columns(8).ColumnName = "SpoolPartState"
      ob("SpoolItems").DataTable.Columns(9).ColumnName = "SpoolLocation_X"
      ob("SpoolItems").DataTable.Columns(10).ColumnName = "SpoolLocation_Y"

      'Spool components refer to parts that are composed by cut pipes and/or
fittings
      ob.AddAttribute("SpoolComponents", "Composite parts of the
spool",CFC_AttributeInternalRepresentation.CFC_Array,CFC_AttributeExternalRep
resentation.CFC_Table,CFC_AttributeAccess.CFC_ReadWrite)
      ob("SpoolComponents").SetRC(1,14)
      'For i As Integer = 0 to  ob("NumOfSpoolComponents").Value- 1
            'For j As Integer = 0 to 12
                  'ob("SpoolComponents").SetValueRC(i,j,0)
            'Next
      'Next

      ob("SpoolComponents").DataTable.Columns(0).ColumnName =
"JobControlNumber"
      ob("SpoolComponents").DataTable.Columns(1).ColumnName = "SpoolPartID"
      ob("SpoolComponents").DataTable.Columns(2).ColumnName = "Priority"
      ob("SpoolComponents").DataTable.Columns(3).ColumnName = "PartTypeID"
      ob("SpoolComponents").DataTable.Columns(4).ColumnName = "WeldID"
      ob("SpoolComponents").DataTable.Columns(5).ColumnName = "Sequence"
      ob("SpoolComponents").DataTable.Columns(6).ColumnName = "Stage"
      ob("SpoolComponents").DataTable.Columns(7).ColumnName = "Weight"
      ob("SpoolComponents").DataTable.Columns(8).ColumnName = "Size"
```

```
      ob("SpoolComponents").DataTable.Columns(9).ColumnName =
"MaterialTypeID"
      ob("SpoolComponents").DataTable.Columns(10).ColumnName = "Length"
      ob("SpoolComponents").DataTable.Columns(11).ColumnName =
"SpoolPartState"
      ob("SpoolComponents").DataTable.Columns(12).ColumnName =
"SpoolLocation_X"
      ob("SpoolComponents").DataTable.Columns(13).ColumnName =
"SpoolLocation_Y"

      'Relationship between Spool items and components
      ob.AddAttribute("SpoolComponentsItemsRelationship",
"",CFC_AttributeInternalRepresentation.CFC_Array,CFC_AttributeExternalRepresentation.CFC_Table,CFC_AttributeAccess.CFC_ReadWrite)
      ob("SpoolComponentsItemsRelationship").SetRC(1, 8)
      'For i As Integer = 0 to ob("NumOfSpoolComponents").Value- 1
            'For j As Integer = 0 to 3
                  'ob("SpoolComponentsItemsRelationship").SetValueRC(i,j,0)
            'Next
      'Next

      ob("SpoolComponentsItemsRelationship").DataTable.Columns(0).ColumnName
= "JobControlNumber"
      ob("SpoolComponentsItemsRelationship").DataTable.Columns(1).ColumnName
= "ComponentID"
      ob("SpoolComponentsItemsRelationship").DataTable.Columns(2).ColumnName
= "WeldID"
      ob("SpoolComponentsItemsRelationship").DataTable.Columns(3).ColumnName
= "PartID"
      ob("SpoolComponentsItemsRelationship").DataTable.Columns(4).ColumnName
= "Sequence"
      ob("SpoolComponentsItemsRelationship").DataTable.Columns(5).ColumnName
= "Stage"
      ob("SpoolComponentsItemsRelationship").DataTable.Columns(6).ColumnName
= "RollOrFixed"
      ob("SpoolComponentsItemsRelationship").DataTable.Columns(7).ColumnName
= "State"

      'Cutting activity for the spool
      ob.AddAttribute("CuttingActivities","Cutting activities for the
spool",CFC_AttributeInternalRepresentation.CFC_Array,CFC_AttributeExternalRepresentation.CFC_Table,CFC_AttributeAccess.CFC_ReadWrite)
      ob("CuttingActivities").SetRC(1,12)
      'For i As Integer = 0 to ob("NumOfCutting").Value - 1
            'For j As Integer = 0 to  11
                  'ob("CuttingActivities").SetValueRC(i,j,0)
            'Next
      'Next

      ob("CuttingActivities").DataTable.Columns(0).ColumnName =
"JobControlNumber"
      ob("CuttingActivities").DataTable.Columns(1).ColumnName = "SpoolPartID"
      ob("CuttingActivities").DataTable.Columns(2).ColumnName = "CuttingID"
      ob("CuttingActivities").DataTable.Columns(3).ColumnName =
"CuttingStationID"
      ob("CuttingActivities").DataTable.Columns(4).ColumnName =
"CuttingConfiguration"
```

```
      ob("CuttingActivities").DataTable.Columns(5).ColumnName =
"CuttingMethod"
      ob("CuttingActivities").DataTable.Columns(6).ColumnName =
"PipeSchedule"
      ob("CuttingActivities").DataTable.Columns(7).ColumnName =
"SingleOrDouble"
      ob("CuttingActivities").DataTable.Columns(8).ColumnName =
"NumberOfPersonnel"
      ob("CuttingActivities").DataTable.Columns(9).ColumnName = "StartTime"
      ob("CuttingActivities").DataTable.Columns(10).ColumnName = "Duration"
      ob("CuttingActivities").DataTable.Columns(11).ColumnName = "FinishTime"

      'Handling activity for the spool
      ob.AddAttribute("HandlingActivities", "Handling acitvities for the
spool",CFC_AttributeInternalRepresentation.CFC_Array,CFC_AttributeExternalRep
resentation.CFC_Table,CFC_AttributeAccess.CFC_ReadWrite)
      ob("HandlingActivities").SetRC(1,11)
      'For i As Integer = 0 to ob("NumOfHandling").Value - 1
            'For j As Integer = 0 to 8
                  'ob("HandlingActivities").SetValueRC(i,j,0)
            'Next
      'Next

      ob("HandlingActivities").DataTable.Columns(0).ColumnName =
"JobControlNumber"
      ob("HandlingActivities").DataTable.Columns(1).ColumnName =
"SpoolPartID"
      ob("HandlingActivities").DataTable.Columns(2).ColumnName = "HandlingID"
      ob("HandlingActivities").DataTable.Columns(3).ColumnName = "Type"
      ob("HandlingActivities").DataTable.Columns(4).ColumnName = "StartTime"
      ob("HandlingActivities").DataTable.Columns(5).ColumnName = "Duration"
      ob("HandlingActivities").DataTable.Columns(6).ColumnName = "FinishTime"
      ob("HandlingActivities").DataTable.Columns(7).ColumnName = "FromArea"
      ob("HandlingActivities").DataTable.Columns(8).ColumnName = "ToArea"
      ob("HandlingActivities").DataTable.Columns(9).ColumnName =
"NumberofPersonnel"
      ob("HandlingActivities").DataTable.Columns(10).ColumnName =
"NumberOfCrane"

      'Fitting activity for the spool
      ob.AddAttribute("FittingActivities", "Fitting acitivities for the
spool",CFC_AttributeInternalRepresentation.CFC_Array,CFC_AttributeExternalRep
resentation.CFC_Table,CFC_AttributeAccess.CFC_ReadWrite)
      ob("FittingActivities").SetRC(1, 14)
      'For i As Integer = 0 to ob("NumOfFitting").Value - 1
            'For j As Integer = 0 to 15
                  'ob("FittingActivities").SetValueRC(i,j,0)
            'Next
      'Next

      ob("FittingActivities").DataTable.Columns(0).ColumnName =
"JobControlNumber"
      ob("FittingActivities").DataTable.Columns(1).ColumnName = "ComponentID"
      ob("FittingActivities").DataTable.Columns(2).ColumnName = "WeldID"
      ob("FittingActivities").DataTable.Columns(3).ColumnName = "FittingID"
      ob("FittingActivities").DataTable.Columns(4).ColumnName = "JointType"
      ob("FittingActivities").DataTable.Columns(5).ColumnName = "Size"
```

```
      ob("FittingActivities").DataTable.Columns(6).ColumnName = "SCH/Rating"
      ob("FittingActivities").DataTable.Columns(7).ColumnName =
"WeldingVolume"
      ob("FittingActivities").DataTable.Columns(8).ColumnName = "RollOrFixed"
      ob("FittingActivities").DataTable.Columns(9).ColumnName = "StartTime"
      ob("FittingActivities").DataTable.Columns(10).ColumnName = "Duration"
      ob("FittingActivities").DataTable.Columns(11).ColumnName = "FinishTime"
      ob("FittingActivities").DataTable.Columns(12).ColumnName =
"FittingTableID"
      ob("FittingActivities").DataTable.Columns(13).ColumnName = "Rework"

      'Welding activity for the spool
      ob.AddAttribute("WeldingActivities", "Welding activities for the
spool",CFC_AttributeInternalRepresentation.CFC_Array,CFC_AttributeExternalRep
resentation.CFC_Table,CFC_AttributeAccess.CFC_ReadWrite)
      ob("WeldingActivities").SetRC(1, 15)
      'For i As Integer = 0 to ob("NumOfWelding").Value - 1
            'For j As Integer = 0 to 16
                  'ob("WeldingActivities").SetValueRC(i,j,0)
            'Next
      'Next

      ob("WeldingActivities").DataTable.Columns(0).ColumnName =
"JobControlNumber"
      ob("WeldingActivities").DataTable.Columns(1).ColumnName = "ComponentID"
      ob("WeldingActivities").DataTable.Columns(2).ColumnName = "WeldID"
      ob("WeldingActivities").DataTable.Columns(3).ColumnName = "WeldingID"
      ob("WeldingActivities").DataTable.Columns(4).ColumnName = "JointType"
      ob("WeldingActivities").DataTable.Columns(5).ColumnName = "Size"
      ob("WeldingActivities").DataTable.Columns(6).ColumnName = "SCH/Rating"
      ob("WeldingActivities").DataTable.Columns(7).ColumnName =
"WeldingVolume"
      ob("WeldingActivities").DataTable.Columns(8).ColumnName = "RollOrFixed"
      ob("WeldingActivities").DataTable.Columns(9).ColumnName = "StartTime"
      ob("WeldingActivities").DataTable.Columns(10).ColumnName = "Duration"
      ob("WeldingActivities").DataTable.Columns(11).ColumnName = "FinishTime"
      ob("WeldingActivities").DataTable.Columns(12).ColumnName =
"WeldingMachineID"
      ob("WeldingActivities").DataTable.Columns(13).ColumnName = "Rework"
      ob("WeldingActivities").DataTable.Columns(14).ColumnName =
"WeldingMethod"

      Return True

End Function



Public Function Spool_Element_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle
      r.Size = new SizeF(120, 50)

      Dim text1 As new GoText
```

```
        text1.Text = "Spool ID = " & ob("JobControlNumber").Value
        text1.FontSize = 7
        text1.Position = new PointF(15, 20)


        Dim g As new GoGroup
        g.Add(r)
        g.Add(text1)

        return g

End Function

End Module
End Namespace
```

This is the VB.NET code for the 'Composite_Element' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go

Namespace SimphonyScript
Public Module Script
'Your functions here

Public NumOfSpool As Integer
Public SpoolFileAddress As String
Public SpoolTableName As String
Public SpoolItemsQueryName As String
Public SpoolComponentsQueryName As String
Public Spool_Components_Items_Relationship As String
Public ActCutting As String
Public ActHandling As String
Public ActFitting As String
Public ActWelding As String

'Public Structure NumOfWeldsForComp
      'Dim ComponentID As Integer
      'Dim NumOfWelds As Integer
'End Structure
```

```
Public Function Composite_Element_OnCreate(ob As
CFCSim_ModelingElementInstance, x As Double, y As Double) As Boolean handles
Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)
      ob.SetNumCoordinates(2)
      ob.Coordinates(0) = new System.Drawing.PointF(x, y)
      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)

      ob.AddAttribute("NumOfSpools", "Number of
Spools",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternal
Representation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob("NumOfSpools").Value = 1
      Return True

End Function



Public Function Composite_Element_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle
      r.Size = new SizeF(120, 50)

      Dim text1 As new GoText
      text1.Text = "Composite Element"
      text1.FontSize = 7
      text1.Position = new PointF(15, 20)


      Dim g As new GoGroup
      g.Add(r)
      g.Add(text1)
      return g

End Function

Public Sub Composite_Element_OnSimulationInitializeRun(ob As
CFCSim_ModelingElementInstance, runNum As Int32) handles
Scripting.OnSimulationInitializeRunEvent

      Dim Child As CFCSim_ModelingElementInstance

      For each Child In ob.ChildElements.Values

            If Child.ElementType = "Spool_Generation" Then
                  NumOfSpool = Child("NA").Value

                  SpoolFileAddress = Child("SpoolDBFileAddress").Value
                  SpoolTableName = Child("SpoolDBTableName").Value
                  SpoolItemsQueryName = Child("SpoolItemsDBQueryName").Value
                  SpoolComponentsQueryName =
Child("SpoolComponentsDBQueryName").Value
```

```
                  Spool_Components_Items_Relationship =
Child("Spool_Component_Item_Relationship").Value
                  ActCutting = Child("ActCuttingDBQueryName").Value
                  ActHandling = Child("ActHandlingDBQueryName").Value
                  ActFitting = Child("ActFittingDBQueryName").Value
                  ActWelding = Child("ActWeldingDBQueryName").Value

                  ob("NumOfSpools").Value = NumOfSpool

                  For i As Integer = 0 to NumOfSpool - 1
                        'MessageBox.Show("OK!")
                        Dim AddElement As CFCSim_ModelingElementInstance
                        AddElement = ob.AddElement("Spool_Element", 50,
100+i*100)
                        AddElement("JobControlNumber").Value =
Child("Spool").GetValueRC(i, 0)
                        AddElement("Priority").Value =
Child("Spool").GetValueRC(i, 2)
                        AddElement("Weight").Value =
Child("Spool").GetValueRC(i, 3)
                        AddElement("DiameterInches").Value =
Child("Spool").GetValueRC(i, 4)
                        AddElement("Length").Value =
Child("Spool").GetValueRC(i, 5)
                        AddElement("IssuanceDate").Value =
Child("Spool").GetValueRC(i, 6)
                        AddElement("BayID").Value =
Child("Spool").GetValueRC(i, 8)
                        AddElement("Duration").Value =
Child("Spool").GetValueRC(i, 9)
                        AddElement("SpoolState").Value = "Issued"
                  Next
                  Exit For

            End If

      Next

      For each Child in ob.ChildElements.Values

            If Child.ElementType = "Spool_Element" Then
            'populate each "Spool_element"
            Dim SpoolID As String
            SpoolID = Child("JobControlNumber").Value.ToString


            Dim cn As System.Data.OleDb.OleDbConnection
            Dim MyDataAdapter1 As System.Data.OleDb.OleDbDataAdapter
            Dim MyDataAdapter2 As System.Data.OleDb.OleDbDataAdapter
            Dim MyDataAdapter3 As System.Data.OleDb.OleDbDataAdapter
            Dim MyDataAdapter4 As System.Data.OleDb.OleDbDataAdapter
            Dim MyDataAdapter5 As System.Data.OleDb.OleDbDataAdapter
            Dim MyDataAdapter6 As System.Data.OleDb.OleDbDataAdapter
            Dim MyDataAdapter7 As System.Data.OleDb.OleDbDataAdapter

            cn = New System.Data.OleDb.OleDbConnection("Provider =
Microsoft.Jet.OLEDB.4.0; Data Source = " & SpoolFileAddress & ";")
```

```
            'Fill Data
            Try
                  cn.Open()
                  Dim MyDataTable1 = New System.Data.DataTable()
                  Dim MyDataTable2 = New System.Data.DataTable()
                  Dim MyDataTable3 = New System.Data.DataTable()
                  Dim MyDataTable4 = New System.Data.DataTable()
                  Dim MyDataTable5 = New System.Data.DataTable()
                  Dim MyDataTable6 = New System.Data.DataTable()
                  Dim MyDataTable7 = New System.Data.DataTable()


                  'Spool Items
                  MyDataAdapter1 = New
System.Data.OleDb.OleDbDataAdapter("SELECT * FROM " & SpoolItemsQueryName & "
WHERE " & SpoolItemsQueryName & ".JobControlNumber = " & "'" & SpoolID & "'",
cn)
                  'Spool Components
                  MyDataAdapter2 = New
System.Data.OleDb.OleDbDataAdapter("SELECT * FROM " &
SpoolComponentsQueryName & " WHERE " & SpoolComponentsQueryName &
".JobControlNumber = " & "'" & SpoolID & "'", cn)
                  'Spool Components and Items Relationships
                  MyDataAdapter3 = New
System.Data.OleDb.OleDbDataAdapter("SELECT * FROM " &
Spool_Components_Items_Relationship & " WHERE " &
Spool_Components_Items_Relationship & ".JobControlNumber = " & "'" & SpoolID
& "'", cn)
                  'Spool Cutting Activities
                  MyDataAdapter4 = New
System.Data.OleDb.OleDbDataAdapter("SELECT * FROM " & ActCutting & " WHERE "
& ActCutting & ".JobControlNumber = " & "'" & SpoolID & "'", cn)
                  'Spool Handling Activities
                  MyDataAdapter5 = New
System.Data.OleDb.OleDbDataAdapter("SELECT * FROM " & ActHandling & " WHERE "
& ActHandling & ".JobControlNumber = " & "'" & SpoolID & "'", cn)
                  'Spool Fitting Acitivities
                  MyDataAdapter6 = New
System.Data.OleDb.OleDbDataAdapter("SELECT * FROM " & ActFitting & " WHERE "
& ActFitting & ".JobControlNumber = " & "'" & SpoolID & "'", cn)
                  'Spool Welding Activities
                  MyDataAdapter7 = New
System.Data.OleDb.OleDbDataAdapter("SELECT * FROM " & ActWelding & " WHERE "
& ActWelding & ".JobControlNumber = " & "'" & SpoolID & "'", cn)

                  'To get the spool items information
                  MyDataAdapter1.Fill(MyDataTable1)
                  'Fill the 'SpoolItems'Atrribute
                  Child("SpoolItems").SetDataTable(MyDataTable1)

                  'To get the spool components information
                  MyDataAdapter2.Fill(MyDataTable2)
                  'Fill the 'SpoolComponents'Atrribute
                  Child("SpoolComponents").SetDataTable(MyDataTable2)
```

```
                  'To get the spool components and items relationship
information
                  MyDataAdapter3.Fill(MyDataTable3)
                  'Fill the 'SpoolItems'Atrribute

      Child("SpoolComponentsItemsRelationship").SetDataTable(MyDataTable3)

                  'To get the spool cutting information
                  MyDataAdapter4.Fill(MyDataTable4)
                  'Fill the 'SpoolItems'Atrribute
                  Child("CuttingActivities").SetDataTable(MyDataTable4)

                  'To get the spool handling information
                  MyDataAdapter5.Fill(MyDataTable5)
                  'Fill the 'SpoolItems'Atrribute
                  Child("HandlingActivities").SetDataTable(MyDataTable5)

                  'To get the spool fitting information
                  MyDataAdapter6.Fill(MyDataTable6)
                  'Fill the 'SpoolItems'Atrribute
                  Child("FittingActivities").SetDataTable(MyDataTable6)

                  'To get the spool welding information
                  MyDataAdapter7.Fill(MyDataTable7)
                  'Fill the 'SpoolItems'Atrribute
                  Child("WeldingActivities").SetDataTable(MyDataTable7)

                  Catch exp As Exception
                        MessageBox.Show(exp.Message)
                  Finally
                  'MessageBox.Show("OK!")
                  cn.Close()

            End Try

            'Calculate Number of welds for each spool component
            'first to initialize the # of welds as 1 for each component
            Dim SpoolCompID As Integer
            Dim NumOfSpoolComp As Integer
            NumOfSpoolComp = Child("SpoolComponents").RowCount
            Dim SpoolComponentsWelds(NumOfSpoolComp, 1) As Integer
            For i As Integer = 0 to NumOfSpoolComp - 1
                  SpoolCompID = Child("SpoolComponents").GetValueRC(i, 1)
                  SpoolComponentsWelds(i,0) = SpoolCompID
                  SpoolComponentsWelds(i,1) = 1
            Next

            For i As Integer = 0 to Child("SpoolComponents").RowCount - 1
                  SpoolCompID = Child("SpoolComponents").GetValueRC(i, 1)
                  'To find its assembly stage
                  Dim CompStage As Integer
                  'To find constituent parts for this component
                  Dim CompParts As New ArrayList
                  For j As Integer = 0 to
Child("SpoolComponentsItemsRelationship").RowCount - 1
                        If
Child("SpoolComponentsItemsRelationship").GetValueRC(j, 1) = SpoolCompID Then
```

```
                              CompStage =
Child("SpoolComponentsItemsRelationship").GetValueRC(j,5)

      CompParts.Add(Child("SpoolComponentsItemsRelationship").GetValueRC(j,3))
                        End If
                  Next
                  'To find all the spool components that are in the same
stage
                  Dim CompAtSameStage As New ArrayList
                  For j As Integer = 0 to
Child("SpoolComponentsItemsRelationship").RowCount - 1
                        If
Child("SpoolComponentsItemsRelationship").GetValueRC(j,5) = CompStage Then

      CompAtSameStage.Add(Child("SpoolComponentsItemsRelationship").GetValueR
C(j,1))
                        End If
                  Next

                  'To check any of its part is also included in Arraylist
'CompAtSameStage'
                  Dim NumOfweldingsForComp As Integer = 0
                  For j As Integer = 0 to CompParts.Count - 1
                        Dim PartID As Integer
                        PartID = CompParts(j)
                        Dim NumOfWeldingsForPart As Integer
                        If CompAtSameStage.Contains(PartID) Then
                              For k As Integer = 0 to NumOfSpoolComp - 1
                                    If SpoolComponentsWelds(k,0) = PartID
Then
                                          NumOfWeldingsForPart =
SpoolComponentsWelds(k,1) + 1
                                    End If
                              Next
                        Else
                              NumOfWeldingsForPart = 1
                        End If
                        NumOfweldingsForComp = NumOfweldingsForComp +
NumOfWeldingsForPart
                  Next
                  'To calculate number of welds for this component
                  NumOfweldingsForComp = NumOfweldingsForComp - 1

                  'Update the records in SpoolComponentsWelds
                  For j As Integer = 0 to NumOfSpoolComp - 1
                        If SpoolComponentsWelds(j,0) = SpoolCompID Then
                              SpoolComponentsWelds(j, 1) =
NumOfweldingsForComp
                              Exit For
                        End If
                  Next

            Next

            'Add this number of welds for each component to the
"SpoolComponents" table
```

```
      Child("SpoolComponents").DataTable.Columns.Add("NumOfWelds",GetType(Int
eger))
            For i As Integer = 0 to Child("SpoolComponents").RowCount - 1
                  Dim ComponentID As Integer
                  ComponentID = Child("SpoolComponents").GetValueRC(i, 1)
                  For j As Integer = 0 to NumOfSpoolComp - 1
                        If SpoolComponentsWelds(j,0) = ComponentID Then

      Child("SpoolComponents").SetValueRC(i,14,SpoolComponentsWelds(j,1))
                        End If
                  Next
            Next

            Dim RowNum As Integer

            RowNum = Child("SpoolItems").RowCount
            Child("NumOfSpoolItems").Value = RowNum

            RowNum = Child("SpoolComponents").RowCount
            Child("NumOfSpoolComponents").Value = RowNum

            RowNum = Child("CuttingActivities").RowCount
            Child("NumOfCutting").Value = RowNum

            RowNum = Child("HandlingActivities").RowCount
            Child("NumOfHandling").Value = RowNum

            RowNum = Child("FittingActivities").RowCount
            Child("NumOfFitting").Value = RowNum

            RowNum = Child("WeldingActivities").RowCount
            Child("NumOfWelding").Value = RowNum

            Child.Invalidate()

            End If

      Next
End Sub

End Module
End Namespace
```

This is the VB.NET code for the 'BayElement' Element

```
'Simphony.NET Template Code

'Force explicit variable declaration and have automatic conversion of data
types

Option Explicit
```

```
Option Strict Off


'Imports for commonly used namespaces

Imports System

Imports System.Collections

Imports System.Diagnostics

Imports System.Math

Imports Simphony.NET

Imports System.Drawing

Imports System.Drawing.Drawing2D

Imports System.Windows.Forms

Imports Northwoods.Go


Namespace SimphonyScript

Public Module Script

'Your functions here


Public Function BayElement_OnCreate(ob As CFCSim_ModelingElementInstance, x
As Double, y As Double) As Boolean handles Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)

      ob.SetNumCoordinates(2)

      ob.Coordinates(0) = new System.Drawing.PointF(x, y)

      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)

      ob.AddAttribute("BayID", "The ID of the
bay",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalRep
resentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      Return True

End Function
```

```
Public Function BayElement_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle

      r.Size = new SizeF(300, 80)

      Dim text1 As new GoText

      text1.Text = "Bay ID = " & ob("BayID").Value

      text1.FontSize = 8

      text1.Position = new PointF(30, 32)

      Dim g As new GoGroup

      g.Add(r)

      g.Add(text1)

      return g

End Function

End Module

End Namespace
```

This is the VB.NET code for the 'LayDownAreaElement' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go

Namespace SimphonyScript
Public Module Script
'Your functions here
```

```
Public SpoolHaveAssigned As New ArrayList


Public Function LayDownAreaElement_OnCreate(ob As
CFCSim_ModelingElementInstance, x As Double, y As Double) As Boolean handles
Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)
      ob.SetNumCoordinates(2)
      ob.Coordinates(0) = new System.Drawing.PointF(x, y)
      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)

      ob.AddAttribute("LayDownAreaID", "The ID of the lay_down
area",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepre
sentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("BayID", "The ID of the Bay where the lay_down area is
located",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("PhysicalLocation_X", "X coordinate of the physical
location of the
station",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob("PhysicalLocation_X").Value = 0
      ob.AddAttribute("PhysicalLocation_Y", "Y coordinate of the physical
location of the
station",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob("PhysicalLocation_Y").Value = 0

      ob.AddAttribute("LayDownAreaType", "the type of laydown
area",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepre
sentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      'add waiting file for the lay-down area
      ob.AddFile("LayDownFile",CFC_FileType.QUEUE)

      ob.AddConnectionPoint("In", x - 15, y + 32,TConnectionType.CInput,5)
      ob.AddConnectionPoint("Out", x + 155, y + 32,TConnectionType.COutput,5)

      Return True

End Function


Public Function LayDownAreaElement_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle
      r.Size = new SizeF(150, 80)

      Dim text1 As new GoText
      text1.Text = "LayDown Element"
      text1.FontSize = 7
      text1.Position = new PointF(15, 40)

      Dim text2 As New GoText
```

```
      text2.Text = "LayDown Area ID = " & ob("LayDownAreaID").Value
      text2.FontSize = 7
      text2.Position = new PointF(15, 20)


      Dim g As new GoGroup
      g.Add(r)
      g.Add(text1)
      g.Add(text2)

      return g

End Function


Public Function LayDownAreaElement_OnListBoxInitialize(ob As
CFCSim_ModelingElementInstance, attr As CFCSim_Attribute) As ArrayList
handles Scripting.OnListBoxInitializeEvent

      Dim AttrList As New ArrayList
      If attr.Name = "LayDownAreaType" Then
            AttrList.Add("Raw Cut Pipes Or Partial Pipe Components")
            AttrList.Add("Raw Fittings")
      End If

      Return AttrList
End Function


Public Sub LayDownAreaElement_OnSimulationInitialize(ob As
CFCSim_ModelingElementInstance) handles Scripting.OnSimulationInitializeEvent

      ob.AddEvent("EnterIn")

End Sub


Public Sub LayDownAreaElement_OnSimulationTransferIn(ob As
CFCSim_ModelingElementInstance, entity As CFCSim_Entity, srcPt As
CFCSim_ConnectionPoint, dstPt As CFCSim_ConnectionPoint) handles
Scripting.OnSimulationTransferInEvent

      ob.ScheduleEvent(Entity,"EnterIn", 0)


End Sub


Public Sub LayDownAreaElement_OnSimulationProcessEvent(ob As
CFCSim_ModelingElementInstance, myEvent As String, entity As CFCSim_Entity)
handles Scripting.OnSimulationProcessEventEvent

      Select Case MyEvent

            Case "EnterIn"

                  If entity("EntityType") = "SpoolItem" Then
```

```
                        'MessageBox.Show("Spool Items Enter In!")

                        Dim SpoolID As String
                        Dim AvailableSpoolItemsInFile As New ArrayList
                        Dim AllSpoolItems As New ArrayList
                        Dim Priority As Integer
                        Dim MySpoolElement As CFCSim_ModelingElementInstance

                        'Determine which spool the incoming spool item
belongs to
                              SpoolID = entity("JobControlNumber")
                              MySpoolElement = entity("SpoolElement")
                              'Find all spool items that belongs to this
spool
                              For i As Integer = 0 to
MySpoolElement("SpoolItems").RowCount - 1
                                    Dim MySpoolItemID As String
                                    MySpoolItemID =
MySpoolElement("SpoolItems").GetValueRC(i,1)
                                    AllSpoolItems.Add(MySpoolItemID)
                              Next

                        'Add the entity to the file
                              Priority = entity("Priority")
                              ob.File("LayDownFile").Add(entity, Priority)

                        'Find spool items that already arrive in the File
                              Dim Ent As CFCSim_Entity
                              With ob.File("LayDownFile")
                                    If .Length <> 0 Then
                                          .MoveFirst()
                                          While (.EOF =False And .Length > 0)
                                                Ent = .Entity
                                                If Ent("JobControlNumber") =
SpoolID Then

      AvailableSpoolItemsInFile.Add(Ent("SpoolPartID"))
                                                End If
                                                .MoveNext()
                                          End While
                                    End If
                              End With

                        'Compare the 'AvailableSpoolItemsInFile' with
'AllSpoolItems' to find if all the spools item already being moved to the
lay-down area
                              If AllSpoolItems.Count =
AvailableSpoolItemsInFile.Count Then

                                    'If this is true, it means all spool
items are ready to be processed
                                    'Send Dummy entity to all the roll
fitting station to test any of them are available
                                    Dim MyFittingStation As
CFCSim_ModelingElementInstance
                                    For Each MyFittingStation in
SimEnvironment.Elements.Values
```

```
                                        'Only send entity to the roll
fitting station
                                        If MyFittingStation.ElementType =
"StationElement" Then
                                              If
MyFittingStation("StationType").Value = "FittingStation" Then
                                                    If
MyFittingStation("BayID").Value = ob("BayID").Value and
MyFittingStation("RollOrFixed").Value = "Roll" Then
                                                          Dim DummyEntity
As CFCSim_Entity
                                                          DummyEntity =
ob.AddEntity

      DummyEntity("EntityType") = "Dummy"

      DummyEntity("RequestLayDownArea") = ob("LayDownAreaID").Value

      DummyEntity("RequestFittingStation") =
MyFittingStation("StationID").Value

      DummyEntity("AvailableOrNot") = "NotAvail"
                                                          'Send the dummy
entity to this fitting station
                                                          Dim Incp As
CFCSim_ModelingElementInstance
                                                          For Each Incp In
MyFittingStation.ChildElements.Values
                                                                If
Incp.ElementType = "InPort"  Then

      Incp.OnSimulationTransferIn(DummyEntity,ob.ConnectionPoints("Out"),Incp
.ConnectionPoints("In"),True)
                                                                End If
                                                          Next
                                                    End If
                                              End If
                                        End If
                                  Next
                            End If

                  Else If entity("EntityType") = "Dummy"    Then
                        'If the fitting station is available
                        If entity("AvailableOrNot") = "Avail"
                              'First to check if a complete set of spool
parts is in the AssemblyFile
                              Dim CompleteSetOrNot As Boolean
                              CompleteSetOrNot =False
                              Dim SpoolID As String
                              Dim AvailableSpoolItemsInFile As New ArrayList
                              Dim Ent As CFCSim_Entity

                              'Get the Spool Id of the first entity in the
file
                              With ob.File("LayDownFile")
                                    If .Length <> 0 Then
                                          .MoveFirst()
```

```
                                        If .EOF =False Then
                                              Ent = .Entity
                                              SpoolID =
Ent("JobControlNumber")
                                        End If
                                  End If
                            End With

                            Dim MySpoolElement As
CFCSim_ModelingElementInstance
                            Dim AllSpoolItems As New ArrayList
                            For Each MySpoolElement In
SimEnvironment.Elements.Values
                                  If MySpoolElement.ElementType =
"Spool_Element" Then
                                        If
MySpoolElement("JobControlNumber").Value = SpoolID Then
                                              For j As Integer = 0 to
MySpoolElement("SpoolItems").RowCount - 1

      AllSpoolItems.Add(MySpoolElement("SpoolItems").GetValueRC(j,1))

                                              Next
                                              Exit For
                                        End If
                                  End If
                            Next


                            'Find spool items that already arrive in the
File
                            With ob.File("LayDownFile")
                                  If .Length <> 0 Then
                                        .MoveFirst()
                                        While (.EOF =False And .Length > 0)
                                              Ent = .Entity
                                              If Ent("JobControlNumber") =
SpoolID Then

      AvailableSpoolItemsInFile.Add(Ent("SpoolPartID"))
                                              End If
                                              .MoveNext()
                                        End While
                                  End If
                            End With

                            If AllSpoolItems.Count =
AvailableSpoolItemsInFile.Count Then
                                  CompleteSetOrNot =True
                            End If

                            'If there is not a complete set of spool parts
in the file
                            If Not CompleteSetOrNot Then
                                  Exit Sub
                            End If
```

```
                                'The ID of the available fitting station
                                Dim MyFittingStationID As String =
entity("RequestFittingStation")
                                'To determine the fitting station
                                Dim MyFittingStation As
CFCSim_ModelingElementInstance
                                For each MyFittingStation in
SimEnvironment.Elements.Values
                                      If MyFittingStation.ElementType =
"StationElement" Then
                                            If
MyFittingStation("StationID").Value = MyFittingStationID Then
                                                  Exit For
                                            End If
                                      End If
                                Next


                                Dim NewEntity As CFCSim_Entity
                                With ob.File("LayDownFile")
                                      'Whether or not the 'LayDownFile' is
empty
                                If .Length <> 0 Then
                                            .MoveFirst()
                                            'Find the ID of spool with highest
priority in the file
                                            If .EOF =False Then
                                                  Ent = .Entity
                                                  SpoolID =
Ent("JobControlNumber")
                                            End If


                                            While (.EOF =False And .Length > 0)
                                                  Ent = .Entity
                                                  If Ent("JobControlNumber") =
SpoolID Then
                                                        NewEntity =
ob.cloneEntity(Ent)
                                                        'Send the entity to the
destination
                                                        'There could be two
situations: 1) send to roll fitting station; 2) send to fixed fitting station

                                                        'Check if the spool
item should be assembled at final stage
                                                        Dim PrimaryOrFinal As
String
                                                        For j As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                                                              If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(j,0) =
NewEntity("JobControlNumber") And
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(j,3) =
NewEntity("SpoolPartID") Then
```

```
      PrimaryOrFinal =
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(j,6)
                                                                  End If
                                                            Next

                                                            If PrimaryOrFinal =
"Roll" Then
                                                                  Dim Incp As
CFCSim_ModelingElementInstance
                                                                  For Each Incp In
MyFittingStation.ChildElements.Values
                                                                        If
Incp.ElementType = "InPort"  Then

      Incp.OnSimulationTransferIn(NewEntity,ob.ConnectionPoints("Out"),Incp.C
onnectionPoints("In"),True)
                                                                              If
SpoolID = "B969-00384" then

      'MessageBox.Show("Send Spool Items to Roll Fitting!")

      'MessageBox.Show(Ent("JobControlNumber"))

      'MessageBox.Show(Ent("SpoolPartID"))

      'MessageBox.Show(MyFittingStation("StationID").Value)

      'MessageBox.Show(SimEnvironment.SimTime)
                                                                              End
If
                                                                        End If
                                                                  Next
                                                            Else
                                                                  'to locate the
fixed fitting table
                                                                  Dim
MyFixedFittingTable As CFCSim_ModelingElementInstance
                                                                        For each
MyFixedFittingTable in SimEnvironment.Elements.Values
                                                                              If
MyFixedFittingTable.ElementType = "StationElement" Then

      If MyFixedFittingTable("StationType").Value = "FittingStation" Then

      If MyFixedFittingTable("RollOrFixed").Value = "Fixed" And
MyFixedFittingTable("BayID").Value = ob("BayID").Value Then

            Exit For

      End If

      End If
                                                                              End
If
                                                                  Next
```

```
                                                        'Send the
newentity to the fixed fitting table
                                                        Dim Incp As
CFCSim_ModelingElementInstance
                                                        For Each Incp In
MyFixedFittingTable.ChildElements.Values
                                                              If
Incp.ElementType = "InPort"  Then

      Incp.OnSimulationTransferIn(NewEntity,ob.ConnectionPoints("Out"),Incp.C
onnectionPoints("In"),True)
                                                                    'If
SpoolID = "B969-00384" then

      'MessageBox.Show("Send Spool Items to Fixed Fitting!")

      'MessageBox.Show(Ent("SpoolPartID"))
                                                                    'End
If
                                                              End If
                                                        Next

                                                  End If
                                            End If
                                            .MoveNext()
                                      End While


                              End If
                              End With

                              'Remove the spool items from the file

                              With ob.File("LayDownFile")
                                    If .Length <> 0 Then
                                          .MoveFirst()
                                          While (.EOF =False And .Length > 0)

                                                If  .Entity("JobControlNumber")
= SpoolID Then
                                                      .Remove(.Entity)
                                                Else
                                                      .MoveNext()
                                                End If


                                          End While
                                    End If
                              End With

                              ob.DeleteEntity(entity)

                        End If

                  End If

      End Select
```

```
End Sub

End Module
```

This is the VB.NET code for the 'Dispatching' Element

```
'Simphony.NET Template Code

'Force explicit variable declaration and have automatic conversion of data
types

Option Explicit

Option Strict Off

'Imports for commonly used namespaces

Imports System

Imports System.Collections

Imports System.Diagnostics

Imports System.Math

Imports Simphony.NET

Imports System.Drawing

Imports System.Drawing.Drawing2D

Imports System.Windows.Forms

Imports Northwoods.Go


Namespace SimphonyScript

Public Module Script

'Your functions here

Public Function Dispatching_OnCreate(ob As CFCSim_ModelingElementInstance, x
As Double, y As Double) As Boolean handles Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)

      ob.SetNumCoordinates(2)

      ob.Coordinates(0) = new System.Drawing.PointF(x, y)

      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)
```

```
      ob.AddAttribute("Count", "Number of entities passing the counter",
CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalRepresen
tation.CFC_Singular,CFC_AttributeAccess.CFC_ReadOnly)

      ob("Count").Value=0

      ob.AddConnectionPoint("In" , x - 15, y + 25, TConnectionType.CInput, 5)

     ob.AddConnectionPoint("Out", x + 125, y + 25, TConnectionType.COutput,5)

      Return True

End Function

Public Function Dispatching_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle

      r.Size = new SizeF(120, 50)

      Dim text1 As new GoText

      text1.Text = "Dispatching"

      text1.FontSize = 9

      text1.Position = new PointF(20, 15)

      Dim g As new GoGroup

      g.Add(r)

      g.Add(text1)

      return g

End Function


Public Sub Dispatching_OnSimulationInitialize(ob As
CFCSim_ModelingElementInstance) handles Scripting.OnSimulationInitializeEvent

      ob.AddEvent("EnterIn")

End Sub
```

```
Public Sub Dispatching_OnSimulationInitializeRun(ob As
CFCSim_ModelingElementInstance, runNum As Int32) handles
Scripting.OnSimulationInitializeRunEvent

      ob("Count").Value=0

End Sub



Public Sub Dispatching_OnSimulationTransferIn(ob As
CFCSim_ModelingElementInstance, entity As CFCSim_Entity, srcPt As
CFCSim_ConnectionPoint, dstPt As CFCSim_ConnectionPoint) handles
Scripting.OnSimulationTransferInEvent

      ob.ScheduleEvent(Entity,"EnterIn", 0)

End Sub



Public Sub Dispatching_OnSimulationProcessEvent(ob As
CFCSim_ModelingElementInstance, myEvent As String, entity As CFCSim_Entity)
handles Scripting.OnSimulationProcessEventEvent

      Select Case MyEvent

            Case "EnterIn"

            ob("Count").Value=ob("Count").Value+1



            'For spools that are just issued

            If entity("EntityType") = "Spool" Then

                  Dim BayID As Integer

                  Dim NewEntity As CFCSim_Entity

                  BayID = entity("BayID")

                  NewEntity = ob.cloneEntity(Entity)

                  'Send the entity to the bay

                  TransferTo(BayID, ob, NewEntity)

            End If

            'For Dummy Entity

            If entity("EntityType")= "Dummy" Then
```

```
                  Dim NewEntity As CFCSim_Entity

                  NewEntity = ob.cloneEntity(Entity)


                  'To Locate the Lay-Down Area

                  Dim MyLayDownArea As CFCSim_ModelingElementInstance

                  For Each MyLayDownArea In SimEnvironment.Elements.Values

                        If MyLayDownArea.ElementType = "LayDownAreaElement"
Then

                              'Send the dummy entity to the lay-down area

                              If MyLayDownArea("LayDownAreaID").Value =
entity("RequestLayDownArea") Then


      MyLayDownArea.OnSimulationTransferIn(NewEntity,ob.ConnectionPoints("Out
"),MyLayDownArea.ConnectionPoints("In"),True)

                              End If

                        End If

                  Next

                  'Transfer out the entity

                  ob.DeleteEntity(entity)

            End If

            'For Spool Component

            If  entity("EntityType")= "SpoolComponent" Then

                  Dim NewEntity As CFCSim_Entity

                  Dim ActJustFinished As String

                  NewEntity = ob.cloneEntity(Entity)

                  ActJustFinished = entity("ActivityJustFinished")


                  Select Case ActJustFinished


                        Case "RollFitting"
```

```
                        'Check whether it needs return to the 'Assembly'
Element immediately
                        'If NewEntity("ReturnOrNot") = 1 Then


                              'Dim MyFittingStation As
CFCSim_ModelingElementInstance
                              'Dim Incp As CFCSim_ModelingElementInstance
                              'MyFittingStation =
NewEntity("RollFittingTable")
                              'For Each Incp In
MyFittingStation.ChildElements.Values
                                    'If Incp.ElementType = "InPort"  Then

      'Incp.OnSimulationTransferIn(NewEntity,ob.ConnectionPoints("Out"),Incp.
ConnectionPoints("In"),True)
                                    'End If
                              'Next
                              'If entity("JobControlNumber") = "B696-00875"
Then
                                    'MessageBox.Show("Return")
                                    'MessageBox.Show(entity("SpoolPartID"))


                              'End If
                              'ob.DeleteEntity(entity)
                        'Else
                              'Dim MyRollWeldingStation As
CFCSim_ModelingElementInstance
                              'For Each MyRollWeldingStation In
simenvironment.Elements.Values
                                    'If MyRollWeldingStation.ElementType =
"StationElement" Then
```

```
                                        'If
MyRollWeldingStation("StationType").Value = "WeldingStation" And
MyRollWeldingStation("RollOrFixed").Value = "Roll" Then 'And
MyRollWeldingStation("BayID").Value = ob.Parent("BayID").Value

                                                'Dim Incp As
CFCSim_ModelingElementInstance

                                                'For Each Incp In
MyRollWeldingStation.ChildElements.Values

                                                      'If Incp.ElementType =
"InPort"  Then


      'Incp.OnSimulationTransferIn(NewEntity,ob.ConnectionPoints("Out"),Incp.
ConnectionPoints("In"),True)

                                                            'Exit For

                                                      'End If

                                                'Next


                                        'End If

                                   'End If

                              'Next

                              'If entity("JobControlNumber") = "B696-00875"
Then

                                   'MessageBox.Show("GoforWelding")

                                   'MessageBox.Show(entity("SpoolPartID"))

                              'End If

                              'If NewEntity("JobControlNumber") = "B969-
00741" Then

                                   'MessageBox.Show(NewEntity("SpoolPartID"))

                              'End If

                              ob.TransferOut(NewEntity)

                              ob.DeleteEntity(entity)

                         'End If
```

```
                        Case "RollWelding"
                        'Check whether the spool is finished
                        Dim SpoolID As String
                        Dim SpoolPartID As String
                        Dim ParentSpoolPartID As String
                        Dim RollOrFixed As String
                        Dim MySpoolElement As CFCSim_ModelingElementInstance
                        SpoolID = entity("JobControlNumber")
                        SpoolPartID = entity("SpoolPartID")
                        MySpoolElement = entity("SpoolElement")
                        Dim FinishedOrNot As Boolean = True
                        'See if it is a component of another spoolcomponent
                        For i As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                              If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,0) = SpoolID
Then
                                    If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,3) =
SpoolPartID Then
                                          FinishedOrNot = False
                                          ParentSpoolPartID =
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,1)
                                          RollOrFixed =
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,6)
                                          Exit For
                                    End If
                              End If
                        Next

                        If FinishedOrNot Then 'If it is the final spool
```

```
                              'Send the entity to the 'End' Element
                              'MessageBox.Show(entity("JobControlNumber"))
                              Dim MyBayElement As
CFCSim_ModelingElementInstance
                              MyBayElement = ob.Parent
                              Dim Outcp As CFCSim_ModelingElementInstance
                              For Each Outcp In
MyBayElement.ChildElements.Values
                                    If Outcp.ElementType = "OutPort"  Then

      Outcp.OnTransferOut(NewEntity,Outcp.ConnectionPoints("Out"))
                                          ob.DeleteEntity(entity)
                                          Exit For

                                    End If
                              Next

                        Else 'If it needs further processing
                              'Check if the further processing is roll-
fitting
                              If RollOrFixed = "Roll" Then

                                    Dim MyFittingStation As
CFCSim_ModelingElementInstance
                                    MyFittingStation =
entity("RollFittingTable")
                                    Dim Incp As
CFCSim_ModelingElementInstance
                                    For Each Incp In
MyFittingStation.ChildElements.Values
                                          If Incp.ElementType = "InPort"
Then
```

```
      Incp.OnSimulationTransferIn(NewEntity,ob.ConnectionPoints("Out"),Incp.C
onnectionPoints("In"),True)

                                                ob.DeleteEntity(entity)

                                          Exit For

                                          End If

                                    Next


                               Else 'If the further processing is fixed-
fitting

                                    'Send the entity to the fixed fitting
station in the bay

                                    Dim MyFixedFittingElement As
CFCSim_ModelingElementInstance

                                    For Each MyFixedFittingElement In
SimEnvironment.Elements.Values

                                          If
MyFixedFittingElement.ElementType = "StationElement" Then

                                                If
MyFixedFittingElement("StationType").Value = "FittingStation"

                                                      If
MyFixedFittingElement("RollOrFixed").Value = "Fixed" Then

                                                            If
MyFixedFittingElement("BayID").Value = ob.Parent("BayID").Value Then

                                                                  Dim Incp As
CFCSim_ModelingElementInstance

                                                                  For Each
Incp In MyFixedFittingElement.ChildElements.Values

                                                                        If
Incp.ElementType = "InPort"  Then


      Incp.OnSimulationTransferIn(NewEntity,ob.ConnectionPoints("Out"),Incp.C
onnectionPoints("In"),True)


      ob.DeleteEntity(entity)
```

```
      Exit For
                                                                    End
If
                                                                        Next
                                                                End If
                                                            End If


                                                        End If
                                                    End If
                                                Next


                                            End If


                                End If


                            Case "FixedFitting"
                            'If NewEntity("ReturnOrNot") = 1 Then


                                'Dim MyFixedFittingStation As
CFCSim_ModelingElementInstance
                                'Dim Incp As CFCSim_ModelingElementInstance
                                'For Each MyFixedFittingStation In
ob.Parent.ChildElements.Values
                                    'If MyFixedFittingStation.ElementType =
"StationElement" Then
                                        'If
MyFixedFittingStation("StationType").Value = "FittingStation" Then
                                            'If
MyFixedFittingStation("RollOrFixed").Value = "Fixed" Then
                                                'Exit For
```

```
                                                'End If

                                          'End If

                                    'End If

                               'Next


                               'For Each Incp In
MyFixedFittingStation.ChildElements.Values

                                    'If Incp.ElementType = "InPort"  Then


      'Incp.OnSimulationTransferIn(NewEntity,ob.ConnectionPoints("Out"),Incp.
ConnectionPoints("In"),True)

                                    'End If

                               'Next

                               'ob.DeleteEntity(entity)

                          'Else

                               ob.TransferOut(NewEntity)

                               ob.DeleteEntity(entity)

                          'End If


                          Case "FixedWelding"

                               ob.TransferOut(NewEntity)

                               ob.DeleteEntity(entity)

                    End Select

               End If

      End Select


End Sub


Public Sub TransferTo(ID As Integer, Ob As CFCSim_ModelingElementInstance,
Entity As CFCSim_Entity)
```

```
      'To Locate the next destination
      Dim MyBayElement As CFCSim_ModelingElementInstance
      For Each MyBayElement In SimEnvironment.Elements.Values
            If MyBayElement.ElementType = "BayElement" Then
                  If MyBayElement("BayID").Value = ID Then
                        Exit For
                  End If
            End If

      Next

      'Send the entity to the destination
      Dim Incp As CFCSim_ModelingElementInstance
      For Each Incp In MyBayElement.ChildElements.Values
            If Incp.ElementType = "InPort"  Then

      Incp.OnSimulationTransferIn(Entity,ob.ConnectionPoints("Out"),Incp.Conn
ectionPoints("In"),True)
                  Exit For
            End If
      Next
End Sub
End Module
End Namespace
```

This is the VB.NET code for the 'StationElement' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
```

```
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go
Imports System.Data

Namespace SimphonyScript
Public Module Script
'Your functions here



Public Function StationElement_OnCreate(ob As CFCSim_ModelingElementInstance,
x As Double, y As Double) As Boolean handles Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)
      ob.SetNumCoordinates(2)
      ob.Coordinates(0) = new System.Drawing.PointF(x, y)
      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)

      ob.AddAttribute("StationID", "the ID of the fitting
station",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRe
presentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("StationType", "The Type of
Station",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRe
presentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      ob.AddAttribute("BayID", "The ID of the bay where the station is
located",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob("BayID").Value = 0


      Return True

End Function


Public Function StationElement_OnListBoxInitialize(ob As
CFCSim_ModelingElementInstance, attr As CFCSim_Attribute) As ArrayList
handles Scripting.OnListBoxInitializeEvent

      Dim AttrList As New ArrayList
      If attr.Name = "StationType" Then
            AttrList.Add("CuttingStation")
            AttrList.Add("FittingStation")
            AttrList.Add("WeldingStation")
      End If
```

```
      If attr.Name = "RollOrFixed" Then
            AttrList.Add("Roll")
            AttrList.Add("Fixed")
      End If

      Return AttrList
End Function


Public Function StationElement_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle
      r.Size = new SizeF(135, 50)

      Dim text1 As new GoText
      text1.Text = "Station ID = " & ob("StationID").Value
      text1.FontSize = 7
      text1.Position = new PointF(15, 15)

      Dim text2 As New GoText
      text2.Text = ob("StationType").Value
      text2.FontSize = 7
      text2.Position = new PointF(15, 28)

      Dim g As new GoGroup
      g.Add(r)
      g.Add(text1)
      g.Add(text2)
      return g

End Function


Public Sub StationElement_OnAttributeChanged(ob As
CFCSim_ModelingElementInstance, attr As CFCSim_Attribute) handles
Scripting.OnAttributeChangedEvent

      If Attr is ob("StationType")
            If ob("StationType").Value = "FittingStation"  Then

                  ob.AddAttribute("RollOrFixed", "Roll assembly or fixed
assembly",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
                  ob.AddAttribute("FromLaydownArea", "the lay-down area that
feeds spool items or components to the processing
station",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRe
presentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
                  ob.AddAttribute("ToLaydownArea", "the lay-down area that
feeds spool items or components to the processing
station",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRe
presentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
                  ob.AddAttribute("PhysicalLocation_X", "X coordinate of the
physical location of the
```

```
station",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
                  ob("PhysicalLocation_X").Value = 0
                  ob.AddAttribute("PhysicalLocation_Y", "Y coordinate of the
physical location of the
station",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
                  ob("PhysicalLocation_Y").Value = 0

            Else If ob("StationType").Value = "WeldingStation" Then

                  ob("StationID").Value = "WeldingComposite"

                  ob.AddAttribute("NumOfWeldingStations", "The number of
welding stations with the
element",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
                  ob.AddAttribute("RollOrFixed", "Roll assembly or fixed
assembly",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
                  ob.AddAttribute("StationLocations", "the locations of all
welding
stations",CFC_AttributeInternalRepresentation.CFC_Array,CFC_AttributeExternal
Representation.CFC_Table,CFC_AttributeAccess.CFC_ReadWrite)

                  ob("StationLocations").DataTable.Columns.Add("StationID",
GetType(String))

      ob("StationLocations").DataTable.Columns.Add("PhysicalLocation_X",
GetType(Double))

      ob("StationLocations").DataTable.Columns.Add("PhysicalLocation_Y",
GetType(Double))
                  ob("StationLocations").DataTable.Columns.Add("BaySide",
GetType(String))

                  ob.AddAttribute("StationLayDownArea", "the laydown area
where the welding station gets the spool
components",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)


            Else If ob("StationType").Value = "CuttingStation" Then
                  ob.AddAttribute("ToLaydownArea", "the Lay-Down Area where
the cutting station will place cut
pipes",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepr
esentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
            End If
            ob.Invalidate()
      End If

      If Attr is ob("RollOrFixed")
            If ob("RollOrFixed").Value = "Fixed" then
                  If ob("StationType").Value = "FittingStation"  Then

                        ob.RemoveAttribute("FromLaydownArea")
                        ob.RemoveAttribute("ToLaydownArea")
```

```
                        ob.AddAttribute("LayDownArea", "The LayDown Area of
all final
assemblies",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
                        ob("StationID").Value = "FinalFittingComposite"
                  Else
                        ob.RemoveAttribute("StationLocations")
                        ob.RemoveAttribute("StationLayDownArea")
                        ob.RemoveAttribute("NumOfWeldingStations")
                        ob.AddAttribute("LaydownArea","The LayDown Area of
all final
assemblies",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

                        ob("StationID").Value = "FinalWeldingComposite"
                        ob.AddAttribute("PhysicalLocation_X", "X coordinate
of the physical location of the
station",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
                        ob("PhysicalLocation_X").Value = 0
                        ob.AddAttribute("PhysicalLocation_Y", "Y coordinate
of the physical location of the
station",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
                        ob("PhysicalLocation_Y").Value = 0
                  End If
            End If
      End If

      If Attr is ob("NumOfWeldingStations")

            Dim NumRows As Integer
            Dim MyRow As DataRow
            NumRows = ob("NumOfWeldingStations").Value

            For i As Integer = 0 to NumRows - 1
                  MyRow = ob("StationLocations").DataTable.NewRow()
                  MyRow.Item("StationID") = ""
                  MyRow.Item("PhysicalLocation_X") = 0
                  MyRow.Item("PhysicalLocation_Y") = 0
                  MyRow.Item("BaySide") = ""
                  ob("StationLocations").DataTable.Rows.Add(MyRow)
            Next


      End If

End Sub



End Module
End Namespace
```

This is the VB.NET code for the 'CuttingStationDecider' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go

Namespace SimphonyScript
Public Module Script
'Your functions here



Public Function CuttingStationDecider_OnCreate(ob As
CFCSim_ModelingElementInstance, x As Double, y As Double) As Boolean handles
Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)
      ob.SetNumCoordinates(2)
      ob.Coordinates(0) = new System.Drawing.PointF(x, y)
      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)

      ob.AddAttribute("Count", "Number of entities passing the counter",
CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalRepresen
tation.CFC_Singular,CFC_AttributeAccess.CFC_ReadOnly)
      ob("Count").Value=0

      ob.AddAttribute("CuttingCapacityLimit1", "the capacity limit for the
first cutting
machine",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob("CuttingCapacityLimit1").Value = 0

      ob.AddAttribute("CuttingCapacityLimit2", "the capacity limit for the
first cutting
machine",CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExterna
lRepresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob("CuttingCapacityLimit2").Value = 0

      ob.AddConnectionPoint("In" , x - 15, y + 25, TConnectionType.CInput, 5)
      ob.AddConnectionPoint("Out1", x + 125, y + 10,
TConnectionType.COutput,5)
      ob.AddConnectionPoint("Out2", x + 125, y + 35,TConnectionType.COutput,
5)
      ob.AddConnectionPoint("Out3", x + 65, y + 55,TConnectionType.COutput,5)
```

```
      Return True

End Function



Public Function CuttingStationDecider_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle
      r.Size = new SizeF(120, 50)

      Dim text1 As new GoText
      text1.Text = "CuttingDecider"
      text1.FontSize = 9
      text1.Position = new PointF(20, 15)

      Dim g As new GoGroup
      g.Add(r)
      g.Add(text1)
      return g

End Function


Public Sub CuttingStationDecider_OnSimulationInitialize(ob As
CFCSim_ModelingElementInstance) handles Scripting.OnSimulationInitializeEvent

      ob.AddEvent("EnterIn")

End Sub


Public Sub CuttingStationDecider_OnSimulationInitializeRun(ob As
CFCSim_ModelingElementInstance, runNum As Int32) handles
Scripting.OnSimulationInitializeRunEvent

      ob("Count").Value=0

End Sub


Public Sub CuttingStationDecider_OnSimulationTransferIn(ob As
CFCSim_ModelingElementInstance, entity As CFCSim_Entity, srcPt As
CFCSim_ConnectionPoint, dstPt As CFCSim_ConnectionPoint) handles
Scripting.OnSimulationTransferInEvent

      ob.ScheduleEvent(Entity,"EnterIn", 0)

End Sub
```

```
Public Sub CuttingStationDecider_OnSimulationProcessEvent(ob As
CFCSim_ModelingElementInstance, myEvent As String, entity As CFCSim_Entity)
handles Scripting.OnSimulationProcessEventEvent

      Select Case MyEvent
            Case "EnterIn"

                  ob("Count").Value=ob("Count").Value+1

                  'Find the size (or diameter) of cut pipes belonging to the
spool
                  Dim Size As Integer
                  Size = entity("Size")
                  Dim NewEntity As CFCSim_Entity
                  NewEntity = ob.cloneEntity(entity)

                  If entity("PartTypeID") = 2 Then
                        ob.TransferOut(NewEntity,ob.ConnectionPoints("Out3"))
                  Else If Size <= ob("CuttingCapacityLimit1").Value
                        ob.TransferOut(NewEntity,ob.ConnectionPoints("Out1"))
                        Else

      ob.TransferOut(NewEntity,ob.ConnectionPoints("Out2"))
                  End If


      End Select

End Sub

End Module
End Namespace
```

This is the VB.NET code for the 'SpoolItem_Generation' Element

```
'Simphony.NET Template Code

'Force explicit variable declaration and have automatic conversion of data
types

Option Explicit

Option Strict Off



'Imports for commonly used namespaces

Imports System

Imports System.Collections

Imports System.Diagnostics
```

```
Imports System.Math

Imports Simphony.NET

Imports System.Drawing

Imports System.Drawing.Drawing2D

Imports System.Windows.Forms

Imports Northwoods.Go


Namespace SimphonyScript

Public Module Script

'Your functions here


Public Function SpoolItem_Generation_OnCreate(ob As
CFCSim_ModelingElementInstance, x As Double, y As Double) As Boolean handles
Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)

      ob.SetNumCoordinates(2)

      ob.Coordinates(0) = new System.Drawing.PointF(x, y)

      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)

      ob.AddAttribute("Count", "Number of entities passing the counter",
CFC_AttributeInternalRepresentation.CFC_Numeric,CFC_AttributeExternalRepresen
tation.CFC_Singular,CFC_AttributeAccess.CFC_ReadOnly)

      ob("Count").Value=0

      ob.AddConnectionPoint("In" , x - 15, y + 25, TConnectionType.CInput, 5)

     ob.AddConnectionPoint("Out", x + 85, y + 25, TConnectionType.COutput, 5)

      Return True

End Function

Public Function SpoolItem_Generation_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle

      r.Size = new SizeF(75, 50)
```

```
      Dim text1 As new GoText

      text1.Text = "SpoolItems"

      text1.FontSize = 9

      text1.Position = new PointF(5, 15)

      Dim g As new GoGroup

      g.Add(r)

      g.Add(text1)

      return g

End Function

Public Sub SpoolItem_Generation_OnSimulationInitialize(ob As
CFCSim_ModelingElementInstance) handles Scripting.OnSimulationInitializeEvent

      ob.AddEvent("EnterIn")

End Sub

Public Sub SpoolItem_Generation_OnSimulationInitializeRun(ob As
CFCSim_ModelingElementInstance, runNum As Int32) handles
Scripting.OnSimulationInitializeRunEvent

      ob("Count").Value=0

End Sub

Public Sub SpoolItem_Generation_OnSimulationTransferIn(ob As
CFCSim_ModelingElementInstance, entity As CFCSim_Entity, srcPt As
CFCSim_ConnectionPoint, dstPt As CFCSim_ConnectionPoint) handles
Scripting.OnSimulationTransferInEvent

      ob.ScheduleEvent(Entity,"EnterIn", 0)

End Sub




Public Sub SpoolItem_Generation_OnSimulationProcessEvent(ob As
CFCSim_ModelingElementInstance, myEvent As String, entity As CFCSim_Entity)
handles Scripting.OnSimulationProcessEventEvent

      Select Case MyEvent

            Case "EnterIn"

                  ob("Count").Value=ob("Count").Value+1
```

```
                  'Creating spool items entity for the spool
                  Dim MySpoolElement As CFCSim_ModelingElementInstance
                  Dim NewEntity As CFCSim_Entity
                  Dim NumberOfSpoolItems As Integer
                  Dim NumberOfColumns As Integer
                  MySpoolElement = entity("SpoolElement")
                  NumberOfSpoolItems = MySpoolElement("SpoolItems").RowCount
                  NumberOfColumns = MySpoolElement("SpoolItems").ColumnCount

                  'Create spool items entities for the spool
                  'Customize these entities according the database
information
                  For i As Integer = 0 to NumberOfSpoolItems - 1
                        NewEntity = ob.AddEntity()
                        For j As Integer = 0 to NumberOfColumns - 1

      NewEntity(MySpoolElement("SpoolItems").ColumnLabel(j)) =
MySpoolElement("SpoolItems").GetValueRC(i, j)
                              If MySpoolElement("SpoolItems").ColumnLabel(j)
= "SpoolPartState" Then

      MySpoolElement("SpoolItems").SetValueRC(i,j,"Ready")
            NewEntity(MySpoolElement("SpoolItems").ColumnLabel(j)) = "Ready"
                              End If
                        Next
                        NewEntity("EntityType") = "SpoolItem"
                        NewEntity("SpoolElement") = MySpoolElement

                        'Mark the pipe items as double end or single end
                        If NewEntity("PartTypeID") = 1 Then
```

```
                                For k As Integer = 0 to
MySpoolElement("CuttingActivities").RowCount - 1

                                    If
MySpoolElement("CuttingActivities").GetValueRC(k,0) =
NewEntity("JobControlNumber") And
MySpoolElement("CuttingActivities").GetValueRC(k,1)  =  NewEntity("SpoolPartID")
Then

                                        If
MySpoolElement("CuttingActivities").GetValueRC(k,7) = "Double" Then

                                            NewEntity("SingleOrDouble") =
2

                                        Else

                                            NewEntity("SingleOrDouble") =
1

                                        End If

                                        'Exit For

                                        'If entity("JobControlNumber") =
"B969-00750" or entity("JobControlNumber") = "C342-00093" Then


      'MessageBox.Show(entity("JobControlNumber") & " " &
NewEntity("SpoolPartID") & " " & NewEntity("SingleOrDouble"))

                                        'End If

                                    End If

                                Next

                        End If

                        ob.TransferOut(NewEntity)

                Next

      End Select

End Sub

End Module

End Namespace
```

This is the VB.NET code for the 'Before' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go
Imports System.Data
Imports System.Data.OleDb

Namespace SimphonyScript
Public Module Script
'Your functions here

Public Function Before_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Double, y As Double) As Boolean handles Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)
      ob.SetNumCoordinates(2)
      ob.Coordinates(0) = new System.Drawing.PointF(x, y)
      ob.Coordinates(1) = new System.Drawing.PointF(x + 50, y + 50)


      ob.AddConnectionPoint("In" , x - 15, y + 25, TConnectionType.CInput, 5)
      ob.AddConnectionPoint("Out", x + 65, y + 25, TConnectionType.COutput,5)

      Return True

End Function


Public Function Before_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent
      Dim r As new GoRectangle
      r.Size = new SizeF(50, 50)

      Dim text1 As new GoText
      text1.Text = "Before"
      text1.FontSize = 7
      text1.Position = new PointF(15, 20)


      Dim g As new GoGroup
      g.Add(r)
      g.Add(text1)
```

```
      return g

End Function


Public Sub Before_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance,
entity As CFCSim_Entity, srcPt As CFCSim_ConnectionPoint, dstPt As
CFCSim_ConnectionPoint) handles Scripting.OnSimulationTransferInEvent

      If entity("EntityType") = "Dummy" Then
            ob.TransferOut(entity)
            Exit Sub
      End If

      Dim MySpoolElement As CFCSim_ModelingElementInstance
      Dim SpoolID As String
      Dim SpoolPartID As Integer
      MySpoolElement = entity("SpoolElement")
      SpoolID = entity("JobControlNumber")
      SpoolPartID = entity("SpoolPartID")

      If ob.Parent.ElementType = "StationElement" Then

            Dim StationID As String
            Dim StationType As String

            StationType = ob.Parent("StationType").Value

            Select Case StationType

                  Case "CuttingStation"
                  StationID = ob.Parent("StationID").Value
                  'Record the start time of processing (e.g. cutting, fitting
or welding)
                  'Record the ID of the cutting station
                  For i As Integer = 0 to
MySpoolElement("CuttingActivities").RowCount - 1
                        If  MySpoolElement("CuttingActivities").GetValueRC(i,0)
= SpoolID and MySpoolElement("CuttingActivities").GetValueRC(i,1) =
SpoolPartID Then
                              If
MySpoolElement("CuttingActivities").GetValueRC(i,10) = 0 Then

      MySpoolElement("CuttingActivities").SetValueRC(i,3,StationID)

      MySpoolElement("CuttingActivities").SetValueRC(i,9,SimEnvironment.SimTi
me)
                                    entity("ActID") =
MySpoolElement("CuttingActivities").GetValueRC(i,2)
                                    'MessageBox.Show(entity("ActID"))
                                    Exit For
                              End If
                        End If
                  Next

                  Case "FittingStation"
                  'check if it needs more than one fittings
```

```
                    Dim NumOfWelds As Integer
                    Dim CompStage As Integer
                    For i As Integer = 0 to
MySpoolElement("SpoolComponents").RowCount - 1
                        If MySpoolElement("SpoolComponents").GetValueRC(i, 0)
= SpoolID and MySpoolElement("SpoolComponents").GetValueRC(i,1) = SpoolPartID
Then
                            NumOfWelds =
MySpoolElement("SpoolComponents").GetValueRC(i,14)
                            CompStage =
MySpoolElement("SpoolComponents").GetValueRC(i,6)
                        End If
                    Next

                    If NumOfWelds > 1 Then
                        'First, to find all the constituent components of
current Subassemblies
                        Dim AllChildSpoolComponents As New ArrayList

      DepthFirstTravers(MySpoolElement("SpoolComponentsItemsRelationship").Da
taTable, SpoolPartID, AllChildSpoolComponents)

                        'Second, to find all the ChildSpoolComponents that
are 1)at the same stages; and 2) not fitted yet
                        Dim ChildSpoolComponentsAtSameStg As New ArrayList
                        Dim ChildSpoolID As Integer
                        For k As Integer = 0 to AllChildSpoolComponents.Count
- 1
                            ChildSpoolID = AllChildSpoolComponents.Item(k)
                            For m As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                                If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(m,1) =
ChildSpoolID Then
                                    If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(m,5) =
CompStage And
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(m,7) Is
DBNull.Value Then

      ChildSpoolComponentsAtSameStg.Add(ChildSpoolID)
                                    End If
                                End If
                            Next
                        Next

                        ChildSpoolComponentsAtSameStg.Sort()
                        'If SpoolID = "A424-04960SL" Then
                            'For i As Integer = 0 to
AllChildSpoolComponents.Count -1
                                'MessageBox.Show("SubAssmbly " &
SpoolPartID & " has child component " & AllChildSpoolComponents.Item(i))
                            'Next
                        'End If
                        'Get the samllest PartID from
"AllChildSpoolComponents", which is also the first one in the arraylist
                        Dim UnFittedPartID As Integer
```

```
                        UnFittedPartID  =  ChildSpoolComponentsAtSameStg.Item(0)
                         'If SpoolID = "B696-00875" Then
                               'MessageBox.Show(UnFittedPartID)
                         'End If

                         '"1" stands for 'True'; "0" represents 'False'
                         entity("MoreThanOneFittings") = 1
                         entity("CurrentFittingComponentID") = UnFittedPartID
                  Else
                         entity("MoreThanOneFittings") = 0
                         entity("CurrentFittingComponentID") = SpoolPartID


                  End If

                         If ob.Parent("RollOrFixed").Value = "Roll" Then
                               StationID = ob.Parent("StationID").Value
                         Else
                               Dim MyFixedFittingStation As
CFCSim_ModelingElementInstance
                               MyFixedFittingStation =
entity("CEM_Common_RqstdRes")
                               StationID =
MyFixedFittingStation("ResName").Value
                               entity("FixedFittingTable") =
MyFixedFittingStation
                         End If

                  'Record the ID of the fitting station
                  'Record the Start time of processing (e.g. cutting, fitting
or welding)
                  For i As Integer = 0 to
MySpoolElement("FittingActivities").RowCount - 1
                        If  MySpoolElement("FittingActivities").GetValueRC(i,0)
= SpoolID And MySpoolElement("FittingActivities").GetValueRC(i,1) =
entity("CurrentFittingComponentID") Then

      MySpoolElement("FittingActivities").SetValueRC(i,9,SimEnvironment.SimTi
me)

      MySpoolElement("FittingActivities").SetValueRC(i,12,StationID)
                         End If
                  Next


                  Case "WeldingStation"
                  'check if it needs more than one weldings
                  Dim NumOfWelds As Integer
                  Dim CompStage As Integer
                  For i As Integer = 0 to
MySpoolElement("SpoolComponents").RowCount - 1
                         If MySpoolElement("SpoolComponents").GetValueRC(i, 0)
= SpoolID and MySpoolElement("SpoolComponents").GetValueRC(i,1) = SpoolPartID
Then
                               NumOfWelds =
MySpoolElement("SpoolComponents").GetValueRC(i,14)
```

```
                              CompStage =
MySpoolElement("SpoolComponents").GetValueRC(i,6)
                        End If
                  Next

                  If NumOfWelds > 1 Then

                        'to find all the child parts that have not been
processed yet
                        'these child parts must be in the same stage with
current spool component
                        Dim AllChildSpoolComponents As New ArrayList
                        'The subroutine "DepthFirstTravers" will find all the
constituent spool components of the current one, regardless of whether they
are in different stages or not

      DepthFirstTravers(MySpoolElement("SpoolComponentsItemsRelationship").Da
taTable, SpoolPartID, AllChildSpoolComponents)

                        'Now the AllChildSpoolComponents contains all the
child parts and the current spool component itself
                        AllChildSpoolComponents.Add(SpoolPartID)

                        'MessageBox.Show(SpoolPartID)
                        'For j As Integer = 0 to
AllChildSpoolComponents.Count - 1
                              'MessageBox.Show(AllChildSpoolComponents(j))
                        'Next

                        'To check if any of them is in the same stage with
the current spool component
                        'To find all the spool components that are in the
same stage with the current component
                        Dim CompsAtSameStage As New ArrayList
                        For j As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                              If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(j,5) =
CompStage Then

      CompsAtSameStage.Add(MySpoolElement("SpoolComponentsItemsRelationship")
.GetValueRC(j,1))
                              End If
                        Next

                        'To find all the child components that are in the
same stage with the current component
                        'overlap between CompsAtSameStage and
AllChildSpoolComponents
                        Dim ChildPartsAtSameStage As New ArrayList

                        Dim PartID As Integer
                        For j As Integer = 0 to
AllChildSpoolComponents.Count-1
                              PartID = AllChildSpoolcomponents(j)
                              If CompsAtSameStage.Contains(PartID) Then
                                    'To avoid duplication
```

```
                                    If Not
ChildPartsAtSameStage.Contains(PartID) Then
                                          ChildPartsAtSameStage.Add(PartID)
                                    End If
                              End If
                        Next
                        'MessageBox.Show(SpoolPartID)
                        'MessageBox.Show(ChildPartsAtSameStage.Count)

                        'to Check any of them have not welded yet
                        Dim UnweldedPartID As Integer
                        'to assign a very large number to UnweldedPartID
                        UnweldedPartID = 1000
                        For j As Integer = 0 to ChildPartsAtSameStage.Count -
1
                              PartID = ChildPartsAtSameStage(j)
                              For k As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                                    If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k, 1) = PartID
and Not MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k, 7)
Is DBNull.Value Then
                                          If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k, 7) =
"Fitted" Then
                                                'Starts with the child
component with the smallest number
                                                If PartID < UnweldedPartID
Then
                                                      UnweldedPartID = PartID
                                                End If
                                          End If
                                    End If
                              End If
                        Next
                  Next


                  '"1" stands for 'True'; "0" represents 'False'
                  entity("MoreThanOneWelds") = 1
                  entity("CurrentWeldingComponentID") = UnweldedPartID
                  'MessageBox.Show(SpoolPartID)
                  'MessageBox.Show(UnweldedPartID)
            Else
                  entity("MoreThanOneWelds") = 0
                  entity("CurrentWeldingComponentID") = SpoolPartID

            End If

            'If entity("JobControlNumber") = "B696-00875" Then
                  'MessageBox.Show("StartWelding")
                  'MessageBox.Show(entity("CurrentWeldingComponentID"))
            'End If

            'Find the weldingstation resource that the entity has
captured
            Dim MyWeldingStationResource As
CFCSim_ModelingElementInstance
```

```
                  MyWeldingStationResource = entity("CEM_Common_RqstdRes")

                  StationID = MyWeldingStationResource("ResName").Value

                  'Record the ID of the welding station
                  'Record the Start time of processing (e.g. cutting, fitting
or welding)
                  For i As Integer = 0 to
MySpoolElement("WeldingActivities").RowCount - 1
                        If  MySpoolElement("WeldingActivities").GetValueRC(i,0)
= SpoolID and MySpoolElement("WeldingActivities").GetValueRC(i,1) =
entity("CurrentWeldingComponentID") Then

      MySpoolElement("WeldingActivities").SetValueRC(i,9,SimEnvironment.SimTi
me)

      MySpoolElement("WeldingActivities").SetValueRC(i,12,StationID)
                        End If
                  Next


            End Select

      Else If ob.Parent.ElementType = "Handling" Then

            'Creates a ID for the handling activity
            Dim HandlingID As Integer = 0
            'If there is no record in the "HandlingActivities" attribute
            If MySpoolElement("HandlingActivities").RowCount = 0
                  HandlingID = 1
            Else
                  For i As Integer = 0 to
MySpoolElement("HandlingActivities").RowCount - 1
                        If HandlingID <
MySpoolElement("HandlingActivities").GetValueRC(i,2) Then
                              HandlingID =
MySpoolElement("HandlingActivities").GetValueRC(i,2)
                        End If
                  Next
                  HandlingID = HandlingID + 1
            End If
            entity("HandlingID") = HandlingID


            'To find the starting point and destination of this handling
activity
            Dim FromArea As String
            Dim ToArea As String
            Dim StationType As String
            StationType = ob.Parent.Parent("StationType").Value
            Select Case StationType
                  Case "CuttingStation"
                        FromArea = ob.Parent.Parent("StationID").Value
                        ToArea = ob.Parent.Parent("ToLaydownArea").Value

                  Case "FittingStation"
                  If ob.Parent.Parent("RollOrFixed").Value = "Roll" Then
```

```
                        If ob.Parent("HandlingType").Value =
"FromLayDownArea" Then
                              FromArea =
ob.Parent.Parent("FromLaydownArea").Value
                              ToArea = ob.Parent.Parent("StationID").Value
                        Else If ob.Parent("HandlingType").Value =
"ToLayDownArea" Then
                              FromArea = ob.Parent.Parent("StationID").Value
                              ToArea =
ob.Parent.Parent("ToLaydownArea").Value
                        End If
                  Else
                        Dim MyLayDownArea1 As CFCSim_ModelingElementInstance
                        For Each MyLayDownArea1 In
SimEnvironment.Elements.Values
                              If MyLayDownArea1.ElementType =
"LayDownAreaElement" Then
                                    If
MyLayDownArea1("LayDownAreaType").Value = "Raw Cut Pipes Or Partial Pipe
Components" And MyLayDownArea1("BayID").Value =
ob.Parent.Parent("BayID").Value Then
                                          FromArea =
MyLayDownArea1("LayDownAreaID").Value
                                    End If
                              End If
                        Next
                        ToArea = ob.Parent.Parent("LayDownArea").Value

                  End If

                  Case "WeldingStation"
                  If ob.Parent.Parent("RollOrFixed").Value = "Roll" Then
                  'When it is roll welding
                        If ob.Parent("HandlingType").Value =
"FromLayDownArea" Then
                              FromArea =
ob.Parent.Parent("StationLayDownArea").Value
                              ToArea = entity("RollWeldingStation")

                        Else If ob.Parent("HandlingType").Value =
"ToLayDownArea" Then
                              FromArea = entity("RollWeldingStation")
                              If entity("State") = 1 Then
                                    '"1" means the spool is finished, then
send to the end of the bay
                                    ToArea = "Sp-Cmpl-2-1"
                              Else If entity("State") = 2 Then
                                    '"2" means sending the spool part to
laydown area in the primary assembly
                                    Dim MyLayDownArea As
CFCSim_ModelingElementInstance
                                    For Each MyLayDownArea In
SimEnvironment.Elements.Values
                                          If MyLayDownArea.ElementType =
"LayDownAreaElement" Then
                                                If
MyLayDownArea("LayDownAreaType").Value = "Raw Cut Pipes Or Partial Pipe
```

```
Components" And MyLayDownArea("BayID").Value =
ob.Parent.Parent("BayID").Value Then
                                                ToArea =
MyLayDownArea("LayDownAreaID").Value
                                                Exit For
                                            End If
                                        End If
                                    Next
                                    Else If entity("State") = 3 Then
                                    '"3" means sending the spool part to
laydown area in the final assembly
                                        Dim MyFixedFittingElement As
CFCSim_ModelingElementInstance
                                        For Each MyFixedFittingElement In
SimEnvironment.Elements.Values
                                            If
MyFixedFittingElement.ElementType = "StationElement" Then
                                                If
MyFixedFittingElement("StationType").Value = "FittingStation" And
MyFixedFittingElement("RollOrFixed").Value = "Fixed" And
MyFixedFittingElement("BayID").Value= ob.Parent.Parent("BayID").Value Then
                                                ToArea =
MyFixedFittingElement("LayDownArea").Value
                                                    Exit For
                                                End If
                                            End If
                                        Next
                                End If

                            End If
                        Else 'when it is fixed welding
                            FromArea = ob.Parent.Parent("LaydownArea").Value
                            ToArea = "Sp-Cmpl-2-1"

                        End If
                End Select

                'Check if the handling is "FromLayDownArea" or "ToLayDownArea"
and it is in the fiting station
                If ob.Parent("HandlingType").Value = "FromLayDownArea" And
ob.Parent.Parent("StationType").Value = "FittingStation" Then
                    'to find its constituent parts
                    'to find its assembly stage
                    Dim CurrentStage As Integer
                    Dim ConstituentParts As New ArrayList
                    For i As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                            If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i, 1) =
SpoolPartID Then

      ConstituentParts.Add(MySpoolElement("SpoolComponentsItemsRelationship")
.GetValueRC(i,3))
                                CurrentStage =
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,5)
                            End If
                    Next
```

```
                  Dim HandledParts As New ArrayList
                  'Check any of them is in the same stage of the current
spool component; If yes, on handling was involved.
                  For j as Integer = 0 to ConstituentParts.Count - 1
                        Dim ConstituentPartID As Integer
                        ConstituentPartID = ConstituentParts(j)


                        'check if it is a spool item
                        For h As Integer = 0 to
MySpoolElement("SpoolItems").RowCount - 1
                              If MySpoolElement("SpoolItems").GetValueRC(h,1)
= ConstituentPartID Then
                                    HandledParts.Add(ConstituentPartID)
                              End If
                        Next

                        'only in primary assembly, spool components need
handling from laydown area
                        If ob.Parent.Parent("RollOrFixed").Value = "Roll"
Then
                              'check if it is a spool component
                              For k As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                                    If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k, 1) =
ConstituentPartID And
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k, 5) <>
CurrentStage Then
                                          If Not
HandledParts.Contains(ConstituentPartID) Then

      HandledParts.Add(ConstituentPartID)
                                          End If
                                    End If
                              Next
                        End If

                  Next

                  If HandledParts.Count > 0 then
                        Dim ActualFromArea As String
                        For m As Integer = 0 to HandledParts.Count - 1

                              ActualFromArea = FromArea

                              'check if the part is a raw fitting
                              For i As Integer = 0 to
MySpoolElement("SpoolItems").RowCount - 1
                                    If
MySpoolElement("SpoolItems").GetValueRC(i, 1) = HandledParts(m) And
MySpoolElement("SpoolItems").GetValueRC(i, 3) = 2 Then
                                          'If SpoolID = "B696-00875" Then

      'MessageBox.Show(HandledParts(m) & " is a raw fitting")
                                          'End If
```

```
                                        Dim MyFittingLayDownArea As
CFCSim_ModelingElementInstance
                                        For Each MyFittingLayDownArea In
SimEnvironment.Elements.Values
                                            If
MyFittingLayDownArea.ElementType = "LayDownAreaElement" Then
                                                If
MyFittingLayDownArea("BayID").Value = ob.Parent.Parent("BayID").Value And
MyFittingLayDownArea("LayDownAreaType").Value = "Raw Fittings" Then

                                                    ActualFromArea =
MyFittingLayDownArea("LayDownAreaID").Value
                                                    'If SpoolID =
"B696-00875" Then

      'MessageBox.Show(ActualFromArea & " is the lay down area")
                                                    'End If

                                                End If
                                            End If
                                        Next

                                    End If
                                Next

                                'To create a record in the "HandlingActivities"
attribute
                                Dim MyRow As DataRow
                                MyRow =
MySpoolElement("HandlingActivities").DataTable.NewRow()
                                MyRow.Item("JobControlNumber") = SpoolID
                                MyRow.Item("SpoolPartID") = HandledParts(m)
                                MyRow.Item("HandlingID") = HandlingID + m
                                MyRow.Item("Type") = ""
                                MyRow.Item("StartTime") =
SimEnvironment.SimTime
                                MyRow.Item("Duration") = 0
                                MyRow.Item("FinishTime") = 0
                                MyRow.Item("FromArea") = ActualFromArea
                                MyRow.Item("ToArea") = ToArea
                                MyRow.Item("NumberOfPersonnel") = 1
                                MyRow.Item("NumberOfCrane") = 1


      MySpoolElement("HandlingActivities").DataTable.Rows.Add(MyRow)
                            Next
                    End If

                Else
                    'To create a record in the "HandlingActivities" attribute
                    Dim MyRow As DataRow
                    MyRow =
MySpoolElement("HandlingActivities").DataTable.NewRow()
                    MyRow.Item("JobControlNumber") = SpoolID
                    MyRow.Item("SpoolPartID") = SpoolPartID
                    MyRow.Item("HandlingID") = HandlingID
                    MyRow.Item("Type") = ""
```

```
                  MyRow.Item("StartTime") = SimEnvironment.SimTime
                  MyRow.Item("Duration") = 0
                  MyRow.Item("FinishTime") = 0
                  MyRow.Item("FromArea") = FromArea
                  MyRow.Item("ToArea") = ToArea
                  MyRow.Item("NumberOfPersonnel") = 1
                  MyRow.Item("NumberOfCrane") = 1


      MySpoolElement("HandlingActivities").DataTable.Rows.Add(MyRow)
            End If



      End If


      ob.TransferOut(entity)
End Sub

Public sub DepthFirstTravers(MyDataTable As System.Data.DataTable,
CurrentComponentID As Integer, AllChildSpoolComponents As ArrayList)
      'datatable only refers to "SpoolComponentsItemsRelationship" in
Spool_Element
      'to find all the child parts of the current spool component
      Dim ChildParts As New ArrayList
      For i As Integer =0 to MyDataTable.Rows.Count - 1
            If MyDataTable.Rows.Item(i)("ComponentID") = CurrentComponentID
Then
                  ChildParts.Add(MyDataTable.Rows.Item(i)("PartID"))
            End If
      Next

      Dim MyRow As System.Data.DataRow
      For Each MyRow In MyDataTable.Rows
            Dim MyRowCompID As Integer
            MyRowCompID = MyRow.Item("ComponentID")
            If ChildParts.Contains(MyRowCompID) Then
                  DepthFirstTravers(MyDataTable, MyRowCompID,
AllChildSpoolComponents)
            End If
      Next

      'At the bottom level, the constituent parts should be spool items,
which are not listed in "SpoolComponentsItemsRelationship"
      'At this point of time, the recursion will stop.
      'Also from now on, The subroutine starts to collect child component IDs
all the way from the bottom level to the top level
      If Not AllChildSpoolcomponents.Contains(CurrentComponentID) Then
            AllChildSpoolcomponents.Add(CurrentComponentID)
      End If
End Sub

End Module
End Namespace
```

This is the VB.NET code for the 'After' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go

Namespace SimphonyScript
Public Module Script
'Your functions here

Public Function After_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Double, y As Double) As Boolean handles Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)
      ob.SetNumCoordinates(2)
      ob.Coordinates(0) = new System.Drawing.PointF(x, y)
      ob.Coordinates(1) = new System.Drawing.PointF(x + 50, y + 50)


      ob.AddConnectionPoint("In" , x - 15, y + 25, TConnectionType.CInput, 5)
      ob.AddConnectionPoint("Out", x + 65, y + 25, TConnectionType.COutput,5)

      Return True

End Function


Public Function After_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle
      r.Size = new SizeF(50, 50)

      Dim text1 As new GoText
      text1.Text = "After"
      text1.FontSize = 7
      text1.Position = new PointF(15, 20)


      Dim g As new GoGroup
      g.Add(r)
```

```
      g.Add(text1)

      return g

End Function


Public Sub After_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance,
entity As CFCSim_Entity, srcPt As CFCSim_ConnectionPoint, dstPt As
CFCSim_ConnectionPoint) handles Scripting.OnSimulationTransferInEvent


      If entity("EntityType") = "Dummy" Then
            ob.TransferOut(entity)
            Exit Sub
      End If

      Dim MySpoolElement As CFCSim_ModelingElementInstance
      Dim SpoolID As String
      Dim SpoolPartID As Integer
      MySpoolElement = entity("SpoolElement")
      SpoolID = entity("JobControlNumber")
      SpoolPartID = entity("SpoolPartID")

      Dim StartTime As Double
      Dim Duration As Double


      If ob.Parent.ElementType = "StationElement" Then

            Dim StationID As String
            Dim StationType As String
            Dim Location_X As Double
            Dim Location_Y As Double

            StationType = ob.Parent("StationType").Value

            Select Case StationType

            Case "CuttingStation"
                        StationID = ob.Parent("StationID").Value
                        Location_X = ob.Parent("PhysicalLocation_X").Value
                        Location_Y = ob.Parent("PhysicalLocation_Y").Value
                  'Record the finish time and duration of processing (e.g.
cutting, fitting or welding)
                  For i As Integer = 0 to
MySpoolElement("CuttingActivities").RowCount - 1
                     If  MySpoolElement("CuttingActivities").GetValueRC(i,0)
= SpoolID and MySpoolElement("CuttingActivities").GetValueRC(i,1) =
SpoolPartID Then
                              If
MySpoolElement("CuttingActivities").GetValueRC(i,2) = entity("ActID") Then
                                    StartTime =
MySpoolElement("CuttingActivities").GetValueRC(i,9)
                                    Duration = SimEnvironment.SimTime -
StartTime
```

```
      MySpoolElement("CuttingActivities").SetValueRC(i,10,Duration)

      MySpoolElement("CuttingActivities").SetValueRC(i,11,SimEnvironment.SimT
ime)
                                    Exit For
                              End If
                        End If
                  Next

                  'Update the state of the spool item (to 'Cut')
                  'Update the physical location of the spool item
                  For j As Integer = 0 to
MySpoolElement("SpoolItems").RowCount - 1
                        If MySpoolElement("SpoolItems").GetValueRC(j,0) =
SpoolID and MySpoolElement("SpoolItems").GetValueRC(j,1) = SpoolPartID Then

      MySpoolElement("SpoolItems").SetValueRC(j,8,"Cut")

      MySpoolElement("SpoolItems").SetValueRC(j,9,Location_X)

      MySpoolElement("SpoolItems").SetValueRC(j,10,Location_Y)
                        End If
                  Next

                  entity("ActivityJustFinished") = "Cutting"


            Case "FittingStation"
                        StationID = ob.Parent("StationID").Value
                        Location_X = ob.Parent("PhysicalLocation_X").Value
                        Location_Y = ob.Parent("PhysicalLocation_Y").Value

                  If ob.Parent("RollOrFixed").Value = "Roll" Then

                        'First of all, to determine which component has just
been fitted
                        Dim CurrentFittedCompID As Integer
                        CurrentFittedCompID =
entity("CurrentFittingComponentID")

                        'If SpoolID = "A424-04960SL" Then
                              'MessageBox.Show(CurrentFittedCompID)
                        'End If

                        'Record the finish time and duration of processing
(e.g. cutting, fitting or welding)
                        For i As Integer  = 0 to
MySpoolElement("FittingActivities").RowCount - 1
                              If
MySpoolElement("FittingActivities").GetValueRC(i,0) = SpoolID and
MySpoolElement("FittingActivities").GetValueRC(i,1) = CurrentFittedCompID
Then
                                    StartTime =
MySpoolElement("FittingActivities").GetValueRC(i,9)
                                    Duration = SimEnvironment.SimTime -
StartTime
```

```
      MySpoolElement("FittingActivities").SetValueRC(i,10,Duration)

      MySpoolElement("FittingActivities").SetValueRC(i,11,SimEnvironment.SimT
ime)
                                    Exit For
                              End If
                        Next

                       'Update the state of the spool component (to 'Fitted')
                        'Update the physical location of the spool component
                        For j As Integer = 0 to
MySpoolElement("SpoolComponents").RowCount - 1
                              If
MySpoolElement("SpoolComponents").GetValueRC(j,0) = SpoolID And
MySpoolElement("SpoolComponents").GetValueRC(j,1) = CurrentFittedCompID Then

      MySpoolElement("SpoolComponents").SetValueRC(j,11, "Fitted")

      MySpoolElement("SpoolComponents").SetValueRC(j,12, Location_X)

      MySpoolElement("SpoolComponents").SetValueRC(j,13, Location_Y)
                                    Exit For
                              End If
                        Next

                        For k As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                              If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k,0) = SpoolID
Then
                                    If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k,1) =
CurrentFittedCompID Then

      MySpoolElement("SpoolComponentsItemsRelationship").SetValueRC(k,7,"Fitt
ed")
                                    End If
                              End If
                        Next

                        entity("ActivityJustFinished") = "RollFitting"

                        'When the current spool component is fitted, it means
that all its constituent components, if it has, are all fitted as well
                        If CurrentFittedCompID = SpoolPartID Then

                              entity("MoreThanOneFittings") = 0

                        End If

                  Else 'when it is fixed fitting
                        'First of all, to determine which component has just
been fitted
                        Dim CurrentFittedCompID As Integer
                        CurrentFittedCompID =
entity("CurrentFittingComponentID")
```

```
                        'Record the finish time and duration of processing
(e.g. cutting, fitting or welding)
                        For i As Integer  = 0 to
MySpoolElement("FittingActivities").RowCount - 1
                                If
MySpoolElement("FittingActivities").GetValueRC(i,0) = SpoolID and
MySpoolElement("FittingActivities").GetValueRC(i,1) = CurrentFittedCompID
Then
                                        StartTime =
MySpoolElement("FittingActivities").GetValueRC(i,9)
                                        Duration = SimEnvironment.SimTime -
StartTime

      MySpoolElement("FittingActivities").SetValueRC(i,10,Duration)

      MySpoolElement("FittingActivities").SetValueRC(i,11,SimEnvironment.SimT
ime)
                                        Exit For
                                End If
                        Next

                       'Update the state of the spool component (to 'Fitted')
                        'Update the physical location of the spool component
                        For j As Integer = 0 to
MySpoolElement("SpoolComponents").RowCount - 1
                                If
MySpoolElement("SpoolComponents").GetValueRC(j,0) = SpoolID And
MySpoolElement("SpoolComponents").GetValueRC(j,1) = CurrentFittedCompID Then

      MySpoolElement("SpoolComponents").SetValueRC(j,11, "Fitted")

      MySpoolElement("SpoolComponents").SetValueRC(j,12, Location_X)

      MySpoolElement("SpoolComponents").SetValueRC(j,13, Location_Y)
                                        Exit For
                                End If
                        Next

                        For k As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                                If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k,0) = SpoolID
Then
                                        If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k,1) =
CurrentFittedCompID Then

      MySpoolElement("SpoolComponentsItemsRelationship").SetValueRC(k,7,"Fitt
ed")
                                        End If
                                End If
                        Next
                        entity("ActivityJustFinished") = "FixedFitting"

                        'When the current spool component is fitted, it means
that all its constituent components, if it has, are all fitted as well
```

```
                        If CurrentFittedCompID = SpoolPartID Then

                              entity("MoreThanOneFittings") = 0

                        End If

                  End If


            Case "WeldingStation"

                  If ob.Parent("RollOrFixed").Value = "Roll" Then

                        'First of all, to determine which component has just
been welded
                        Dim CurrentWeldedCompID As Integer
                        CurrentWeldedCompID =
entity("CurrentWeldingComponentID")

                        'Record the finish time and duration of processing
(e.g. cutting, fitting or welding)
                        For i As Integer  = 0 to
MySpoolElement("WeldingActivities").RowCount - 1
                              If
MySpoolElement("WeldingActivities").GetValueRC(i,0) = SpoolID and
MySpoolElement("WeldingActivities").GetValueRC(i,1) = CurrentWeldedCompID
Then
                                    StartTime =
MySpoolElement("WeldingActivities").GetValueRC(i,9)
                                    Duration = SimEnvironment.SimTime -
StartTime

      MySpoolElement("WeldingActivities").SetValueRC(i,11,Duration)

      MySpoolElement("WeldingActivities").SetValueRC(i,10,SimEnvironment.SimT
ime)
                                    Exit For
                              End If
                        Next

                        'Find the physical location of the weld station
                        Dim WeldingStationID As String
                        WeldingStationID = entity("RollWeldingStation")
                        For m As Integer = 0  to
ob.Parent("StationLocations").RowCount - 1
                             If  ob.Parent("StationLocations").GetValueRC(m,0)
= WeldingStationID Then
                                    Location_X =
ob.Parent("StationLocations").GetValueRC(m,1)
                                    Location_Y =
ob.Parent("StationLocations").GetValueRC(m,2)
                              End If
                        Next

                       'Update the state of the spool component (to 'Welded')
                        'Update the physical location of the spool component
```

```
                        For j As Integer = 0 to
MySpoolElement("SpoolComponents").RowCount - 1
                            If
MySpoolElement("SpoolComponents").GetValueRC(j,0) = SpoolID And
MySpoolElement("SpoolComponents").GetValueRC(j,1) = CurrentWeldedCompID Then


      MySpoolElement("SpoolComponents").SetValueRC(j,11, "Welded")

      MySpoolElement("SpoolComponents").SetValueRC(j,12, Location_X)

      MySpoolElement("SpoolComponents").SetValueRC(j,13, Location_Y)
                                Exit For
                            End If
                        Next

                        For k As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                            If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k,0) = SpoolID
Then
                                If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k,1) =
CurrentWeldedCompID Then

      MySpoolElement("SpoolComponentsItemsRelationship").SetValueRC(k,7,"Weld
ed")
                                End If
                            End If
                        Next

                        entity("ActivityJustFinished") = "RollWelding"

                        'When the current spool component is welded, it means
that all its constituent components, if it has, are all welded as well
                        If CurrentWeldedCompID = SpoolPartID Then

                            entity("MoreThanOneWelds") = 0

                        End If

                        'Check whether the spool is finished

                        Dim ParentSpoolPartID As Integer
                        Dim RollOrFixed As String
                        Dim FinishedOrNot As Boolean = True
                        'See if it is a component of another spoolcomponent
                        For i As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                            If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,0) = SpoolID
Then
                                If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,3) =
SpoolPartID Then
                                    FinishedOrNot = False
```

```
                                        ParentSpoolPartID =
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,1)
                                        RollOrFixed =
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,6)
                                        Exit For
                                  End If
                            End If
                      Next

                      If FinishedOrNot Then 'If it is the final spool
                            '"1" represents the spool is finished
                            entity("State") = 1
                      Else 'If it needs further processing
                            'Check if the further processing is roll-
fitting
                            If RollOrFixed = "Roll" Then
                                  '"2" represents needing roll assembly
                                  entity("State") = 2
                                  'MessageBox.Show(entity("State"))
                            Else 'If the further processing is fixed-
fitting
                                  entity("State") = 3
                            End If
                      End If

                Else 'if it is fixed welding
                      'First of all, to determine which component has just
been welded
                      Dim CurrentWeldedCompID As Integer
                      CurrentWeldedCompID =
entity("CurrentWeldingComponentID")

                      'Record the finish time and duration of processing
(e.g. cutting, fitting or welding)
                      For i As Integer  = 0 to
MySpoolElement("WeldingActivities").RowCount - 1
                            If
MySpoolElement("WeldingActivities").GetValueRC(i,0) = SpoolID and
MySpoolElement("WeldingActivities").GetValueRC(i,1) = CurrentWeldedCompID
Then
                                  StartTime =
MySpoolElement("WeldingActivities").GetValueRC(i,9)
                                  Duration = SimEnvironment.SimTime -
StartTime

      MySpoolElement("WeldingActivities").SetValueRC(i,11,Duration)

      MySpoolElement("WeldingActivities").SetValueRC(i,10,SimEnvironment.SimT
ime)
                                  Exit For
                            End If
                      Next

                      'Find the physical location of the weld station
                      Location_X = ob.Parent("PhysicalLocation_X").Value
                      Location_Y = ob.Parent("PhysicalLocation_Y").Value
```

```
                        'Update the state of the spool component (to 'Welded')
                         'Update the physical location of the spool component
                         For j As Integer = 0 to
MySpoolElement("SpoolComponents").RowCount - 1
                                If
MySpoolElement("SpoolComponents").GetValueRC(j,0) = SpoolID And
MySpoolElement("SpoolComponents").GetValueRC(j,1) = CurrentWeldedCompID Then


      MySpoolElement("SpoolComponents").SetValueRC(j,11, "Welded")

      MySpoolElement("SpoolComponents").SetValueRC(j,12, Location_X)

      MySpoolElement("SpoolComponents").SetValueRC(j,13, Location_Y)
                                      Exit For
                                End If
                         Next

                         For k As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                                If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k,0) = SpoolID
Then
                                      If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k,1) =
CurrentWeldedCompID Then

      MySpoolElement("SpoolComponentsItemsRelationship").SetValueRC(k,7,"Weld
ed")
                                      End If
                                End If
                         Next

                         entity("ActivityJustFinished") = "FixedWelding"

                         'When the current spool component is welded, it means
that all its constituent components, if it has, are all welded as well
                         'it also means that the spool is finished


                         If CurrentWeldedCompID = SpoolPartID Then

                               entity("MoreThanOneWelds") = 0
                               entity("State") = 1
                         End If


                  End If

            End Select

      Else If ob.Parent.ElementType = "Handling" Then

            If ob.Parent("HandlingType").Value = "FromLayDownArea" And
ob.Parent.Parent("StationType").Value = "FittingStation" Then
                  'to find its constituent parts
```

```
                  'to find its assembly stage
                  Dim CurrentStage As Integer
                  Dim ConstituentParts As New ArrayList
                  For i As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                        If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i, 1) =
SpoolPartID Then

      ConstituentParts.Add(MySpoolElement("SpoolComponentsItemsRelationship")
.GetValueRC(i,3))
                              CurrentStage =
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,5)
                        End If
                  Next

                  Dim HandledParts As New ArrayList
                  'Check any of them is in the same stage of the current
spool component; If yes, on handling was involved.
                  For j as Integer = 0 to ConstituentParts.Count - 1
                        Dim ConstituentPartID As Integer
                        ConstituentPartID = ConstituentParts(j)


                        'check if it is a spool item
                        For h As Integer = 0 to
MySpoolElement("SpoolItems").RowCount - 1
                              If MySpoolElement("SpoolItems").GetValueRC(h,1)
= ConstituentPartID Then
                                    HandledParts.Add(ConstituentPartID)
                              End If
                        Next

                        'check if it is a spool component
                        For k As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1
                              If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k, 1) =
ConstituentPartID And
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k, 5) <>
CurrentStage Then
                                    If Not
HandledParts.Contains(ConstituentPartID) Then
                                          HandledParts.Add(ConstituentPartID)
                                    End If
                              End If
                        Next
                  Next

                  'record the duration of each handling
                  Dim HandlingID As Integer
                  HandlingID = entity("HandlingID")
                  For m As Integer = 0 to HandledParts.Count - 1

                        HandlingID = HandlingID + m
                        For n As Integer = 0 to
MySpoolElement("HandlingActivities").RowCount - 1
```

```
                                If
MySpoolElement("HandlingActivities").GetValueRC(n,2) = HandlingID Then
                                    StartTime =
MySpoolElement("HandlingActivities").GetValueRC(n,4)
                                    Duration = SimEnvironment.SimTime -
StartTime

      MySpoolElement("HandlingActivities").SetValueRC(n,5,Duration)

      MySpoolElement("HandlingActivities").SetValueRC(n,6,SimEnvironment.SimT
ime)
                                End If
                            Next
                    Next

            Else
                    'record the duration of handling
                    For i As Integer = 0 to
MySpoolElement("HandlingActivities").RowCount - 1
                            If
MySpoolElement("HandlingActivities").GetValueRC(i,0) = SpoolID And
MySpoolElement("HandlingActivities").GetValueRC(i,1) = SpoolPartID Then
                                If
MySpoolElement("HandlingActivities").GetValueRC(i,2) = entity("HandlingID")
Then
                                    StartTime =
MySpoolElement("HandlingActivities").GetValueRC(i,4)
                                    Duration = SimEnvironment.SimTime -
StartTime

      MySpoolElement("HandlingActivities").SetValueRC(i,5,Duration)

      MySpoolElement("HandlingActivities").SetValueRC(i,6,SimEnvironment.SimT
ime)
                                        Exit For
                                End If
                            End If
                    Next

                    'Update the current phisical location


            End If

      End If

      ob.TransferOut(entity)

End Sub

End Module
End Namespace
```

This is the VB.NET code for the 'Assembly' Element

```
'Simphony.NET Template Code

'Force explicit variable declaration and have automatic conversion of data
types

Option Explicit

Option Strict Off


'Imports for commonly used namespaces

Imports System

Imports System.Collections

Imports System.Diagnostics

Imports System.Math

Imports Simphony.NET

Imports System.Drawing

Imports System.Drawing.Drawing2D

Imports System.Windows.Forms

Imports Northwoods.Go


Namespace SimphonyScript

Public Module Script

'Your functions here


Public Structure SpoolComponent

      Dim SpoolComponentID As Integer

      Dim SpoolPartID_1 As Integer

      Dim SpoolPartID_2 As Integer

End Structure
```

```
Public Function Assembly_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Double, y As Double) As Boolean handles Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)

      ob.SetNumCoordinates(2)

      ob.Coordinates(0) = new System.Drawing.PointF(x, y)

      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)

      ob.AddFile("AssemblyFile",CFC_FileType.QUEUE)

      ob.AddConnectionPoint("In" , x - 15, y + 25, TConnectionType.CInput, 5)

      ob.AddConnectionPoint("Out1", x + 110, y + 25,
TConnectionType.COutput,5)

      ob.AddConnectionPoint("Out2", x + 65, y + 55,TConnectionType.COutput,5)

      Return True

End Function


Public Function Assembly_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle

      r.Size = new SizeF(100, 50)

      Dim text1 As new GoText

      text1.Text = "Assembly Element"

      text1.FontSize = 7

      text1.Position = new PointF(15, 20)

      Dim g As new GoGroup

      g.Add(r)

      g.Add(text1)

      return g

End Function
```

```
Public Sub Assembly_OnSimulationInitialize(ob As
CFCSim_ModelingElementInstance) handles Scripting.OnSimulationInitializeEvent

      ob.AddEvent("CheckPossibleAssembly")

End Sub

Public Sub Assembly_OnSimulationTransferIn(ob As
CFCSim_ModelingElementInstance, entity As CFCSim_Entity, srcPt As
CFCSim_ConnectionPoint, dstPt As CFCSim_ConnectionPoint) handles
Scripting.OnSimulationTransferInEvent

      If entity("EntityType") = "SpoolItem" or entity("EntityType") =
"SpoolComponent" Or entity("EntityType") = "Dummy1" Then

            ob.ScheduleEvent(entity,"CheckPossibleAssembly",0)

            'MessageBox.Show("A SpoolItem Enters In!")

      End If

      If entity("EntityType") = "Dummy" Then

            ob.TransferOut(entity, ob.ConnectionPoints("Out2"))

      End If

End Sub

Public Sub Assembly_OnSimulationProcessEvent(ob As
CFCSim_ModelingElementInstance, myEvent As String, entity As CFCSim_Entity)
handles Scripting.OnSimulationProcessEventEvent

      If Not entity("EntityType") = "Dummy1" Then

            'Add the entity into the 'AssemblyFile'

            Dim Priority As Integer

            Priority = entity("Priority")

            ob.File("AssemblyFile").Add(entity, Priority)

      End If

      'Determine the 'stage' of assembly

      Dim CurrentAssemblyStage As Integer

      Dim Stages As New ArrayList

      Dim MySpoolElement As CFCSim_ModelingElementInstance

      Dim SpoolID As String
```

```
      MySpoolElement = entity("SpoolElement")

      SpoolID = entity("JobControlNumber")

      'If SpoolID = "B696-00875" And Not entity("EntityType") = "Dummy1" Then

            'Dim FR As String

            'FR = ob.Parent("RollOrFixed").Value

            'MessageBox.Show(entity("SpoolPartID") & " at " & FR & " Stage")

      'End If

      'To find the minimum value in the 'Stages' arraylist

      For i As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1

            If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i, 7) Is
DBNull.Value Then


      Stages.Add(MySpoolElement("SpoolComponentsItemsRelationship").GetValueR
C(i, 5))

            End If

      Next

      CurrentAssemblyStage = 100 ' a very large number

      For j As Integer = 0 to Stages.Count - 1

            If CurrentAssemblyStage > Stages(j) Then

                  CurrentAssemblyStage = Stages(j)

            End If

      Next

      'MessageBox.Show("SpoolID " & SpoolID & " is at " &
CurrentAssemblyStage)



      'To find possible subassembly at this stage. Subassembly means the
spool components that come out of this stage, not any of the intermediate
spool components
```

```
      'For example, if 18 (1, 13) and 19 (10, 18), or to say if comp 18 is
composed by 1 and 13; while comp 19 is formed by 10 and 18, then only comp 19
is the subassembly and comp 18 is just the intermediate spool components

      'and to find the IDs of possible subassemblies and their constituent
parts

      Dim SpoolComID As Integer = 0

      Dim WhetherOrNotSubassembly As Boolean

      Dim SubassemblyAtCurrStg As New ArrayList

      'Find spool components that are 1) a subassembly; 2) at the current
stage; 3) not finished yet

      For i As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1

            WhetherOrNotSubassembly = True

            'Belongs to the current assembly stage

            If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,0) = SpoolID
And MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i, 5) =
CurrentAssemblyStage And
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i, 7) Is
DBNull.Value Then

                  'find the spool components for which the fitting is not
finished yet

                        SpoolComID =
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(i,1)



                        'If the current spool component is NOT a constituent
part of any other spool components at the same stage

                        For j As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1

                              If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(j, 5) =
CurrentAssemblyStage Then

                                    If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(j, 3) =
SpoolComID Then

                                          WhetherOrNotSubassembly = False
```

```
                                    End If

                                End If

                            Next

                            'If yes, add to SubassemblyAtCurrStg. Otherwise, it
is not a subassembly

                            If WhetherOrNotSubassembly Then

                                If  Not  SubassemblyAtCurrStg.Contains(SpoolComID)
Then

                                    SubassemblyAtCurrStg.Add(SpoolComID)

                                End If

                            End If

                End If

        Next


        'If SpoolID = "B696-00875" Then

                'MessageBox.Show("B696-00875 has " & SubassemblyAtCurrStg.Count &
" Subassembly")

                'For i As Integer = 0 to SubassemblyAtCurrStg.Count - 1

                    'MessageBox.Show("At Stage " & CurrentAssemblyStage & ",
B696-00875 has Subassembly " & SubassemblyAtCurrStg.Item(i))

                'Next

        'End If


        'ReadySubassemblies means subassemblies that can be assembled right now

        'Dim ReadySubassemblies As New ArrayList


        Dim SubasssemblyID As Integer

        For i As Integer = 0 to SubassemblyAtCurrStg.Count - 1

                Dim PreconditionsForSubassembly As New ArrayList

                SubasssemblyID = SubassemblyAtCurrStg.Item(i)
```

```
            'To check if the Subasssembly has more than one welds

            Dim NumOfWelds As Integer

            For j As Integer = 0 to
MySpoolElement("SpoolComponents").RowCount - 1

                  If MySpoolElement("SpoolComponents").GetValueRC(j, 0) =
SpoolID and MySpoolElement("SpoolComponents").GetValueRC(j,1) =
SubasssemblyID Then

                        NumOfWelds =
MySpoolElement("SpoolComponents").GetValueRC(j,14)

                  End If

            Next

            If NumOfWelds > 1 Then

                  'First, to find all the constituent components of current
Subassemblies

                  Dim AllChildSpoolComponents As New ArrayList


      DepthFirstTravers(MySpoolElement("SpoolComponentsItemsRelationship").Da
taTable, SubasssemblyID, AllChildSpoolComponents)

                  'preconditions to start assemble the subassembly include:

                  '1)Constituent Spool component(s) from the immediately
previous stage;

                  '2)Constituent Spool item(s) that are required at the
current stage

                  'These two types are included in the "PardID" of
Constituent Spool component(s) at the same stage



                  'Second, to find all the ChildSpoolComponents that are at
the same stages

                  Dim ChildSpoolComponentsAtSameStg As New ArrayList

                  Dim ChildSpoolID

                  For k As Integer = 0 to AllChildSpoolComponents.Count - 1

                        ChildSpoolID = AllChildSpoolComponents.Item(k)
```

```
                        For m As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1

                            If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(m,1) =
ChildSpoolID Then

                                If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(m,5) =
CurrentAssemblyStage Then


      ChildSpoolComponentsAtSameStg.Add(ChildSpoolID)

                                End If

                            End If

                        Next

                Next



                'Should include the ID of the Subassebly as well

                ChildSpoolComponentsAtSameStg.Add(SubasssemblyID)



                'Third, to find the constituent parts of all these spool
components

                Dim ChildSpoolParts As New ArrayList

                Dim ChildSpoolComponentID As Integer

                For n As Integer = 0 to ChildSpoolComponentsAtSameStg.Count
-1

                        ChildSpoolComponentID =
ChildSpoolComponentsAtSameStg.Item(n)

                        For p As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1

                            If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(p,1) =
ChildSpoolComponentID Then


      ChildSpoolParts.Add(MySpoolElement("SpoolComponentsItemsRelationship").
GetValueRC(p,3))
```

```
                        End If

                    Next

                Next


                'Fourth, to rule out intermediate spool components (which
are also spool components) at the current stage

                Dim ChildPartID As Integer

                For q As Integer = 0 to ChildSpoolParts.Count - 1

                    ChildPartID = ChildSpoolParts.Item(q)

                    If Not
ChildSpoolComponentsAtSameStg.Contains(ChildPartID) Then

                        'To avoid repetition

                        If Not PreconditionsForSubassembly.Contains
(ChildPartID) Then


    PreconditionsForSubassembly.Add(ChildPartID)

                        End If

                    End If

                Next

            Else

                'Just to find its constituent parts at the same stage

                For k As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1

                    If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(k,1) =
SubasssemblyID Then


    PreconditionsForSubassembly.Add(MySpoolElement("SpoolComponentsItemsRel
ationship").GetValueRC(k,3))

                    End If

                Next

            End If
```

```
            'To find existing spool parts in the 'AssemblyFile'
            Dim Ent As CFCSim_Entity
            Dim AvailableSpoolParts As New ArrayList
            With ob.File("AssemblyFile")
                  'Whether or not the 'LayDownFile' is empty
                  If .Length <> 0 Then
                        .MoveFirst()
                        'Find the ID of spool with highest priority in the
file
                        While (.EOF =False And .Length > 0)
                              Ent = .Entity
                              If Ent("JobControlNumber") = SpoolID Then
                        AvailableSpoolParts.Add(Ent("SpoolPartID"))
                              End If
                              .MoveNext()

                        End While
                  End If
            End With
            'Find subassembly or spool components that can be assembled right
now
            Dim WhetherOrNotReady As Boolean
            WhetherOrNotReady = True
            Dim PreconditionPartID As Integer
            For j As Integer = 0 to PreconditionsForSubassembly.Count -1
                  PreconditionPartID = PreconditionsForSubassembly.Item(j)
                  'If SpoolID = "B696-00875" Then
                        'MessageBox.Show(SubasssemblyID & " needs item " &
PreconditionPartID)
```

```
                'End If

                If Not AvailableSpoolParts.Contains(PreconditionPartID)

                      WhetherOrNotReady = False

                      Exit For

                End If

          Next

          If WhetherOrNotReady Then

          'If all the preconditions have been satisfied

          'Then, Generate entity that represents these spool components and
remove the corresponding spool parts from the 'AssemblyFile'



                'If SpoolID = "B969-01361"

                      'Dim M As String

                      'For a As Integer = 0 to
PreconditionsForSubassembly.Count - 1

                            'M = M & PreconditionsForSubassembly.Item(a) &
", "

                      'Next

                      'MessageBox.Show("Subassembly " & SubasssemblyID & "
can be assembled now; it preconditions include " & M)

                'End If



                'Generate the spool component entity

                Dim NewEntity As CFCSim_Entity

                NewEntity = ob.AddEntity()

                For m As integer = 0 to
MySpoolElement("SpoolComponents").RowCount - 1

                      If MySpoolElement("SpoolComponents").GetValueRC(m,0)
= SpoolID And MySpoolElement("SpoolComponents").GetValueRC(m, 1) =
SubasssemblyID Then

                            For n As integer = 0 to
MySpoolElement("SpoolComponents").ColumnCount - 1
```

```
      NewEntity(MySpoolElement("SpoolComponents").ColumnLabel(n)) =
MySpoolElement("SpoolComponents").GetValueRC(m, n)

                              Next

                              NewEntity("EntityType") = "SpoolComponent"

                              NewEntity("SpoolElement") = MySpoolElement

                              'If the current station is a roll fitting
station

                              If ob.Parent("RollOrFixed").Value = "Roll" Then

                                    NewEntity("RollFittingTable") = ob.Parent

                              Else 'If it is a fixed fitting station

                                    NewEntity("FixedFittingTable") =
ob.Parent

                              End If



                              'when it is in the final assembly stage, check
if the spool assembly requires handling from Lay down area

                              NewEntity("NeedHandlingOrNot") = 0

                              If ob.Parent("StationType").Value =
"FittingStation" And ob.Parent("RollOrFixed").Value = "Fixed" Then


                                    For p As Integer = 0 to
MySpoolElement("SpoolItems").RowCount - 1

                                          If
PreconditionsForSubassembly.Contains(MySpoolElement("SpoolItems").GetValueRC(
p,1))Then

                                                NewEntity("NeedHandlingOrNot")
= 1

                                          End If

                                    Next

                              End If
```

```
                            'Before transfering out the NewEntity, we
should check if it is the last assembly at the current stage.

                            'If it is the case, we have to check if there
are possible assemblies in the next stage

                            'First, finding the last spool part ID at
current stage

                            If ob.Parent("RollOrFixed").Value = "Roll" Then


                                  'First of all, check if it is the final
stage of the spool(i.e. meaning the spool is already finished)

                                  Dim FinalStage As Integer

                                  FinalStage = 1

                                  For p As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1

                                        If FinalStage <
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(p, 5) Then

                                              FinalStage =
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(p, 5)

                                        End If

                                  Next

                                  Dim LastSpoolCompAtCurrStg As Integer = 0

                                  'Find the maximum Spool Component ID at
the current stage

                                  For n As Integer = 0 to
MySpoolElement("SpoolComponentsItemsRelationship").RowCount - 1

                                        If
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(n, 5) =
CurrentAssemblyStage Then

                                              If LastSpoolCompAtCurrStg <
MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(n,1) Then

                                                    LastSpoolCompAtCurrStg
= MySpoolElement("SpoolComponentsItemsRelationship").GetValueRC(n,1)

                                              End If

                                        End If
```

```
                                    Next

                                    '"Dummy1" is a dummy entity that is used
to check if there is any possible assembly in the next stage
                                    If NewEntity("SpoolPartID") =
LastSpoolCompAtCurrStg Then
                                        'If it is the final stage and final
assemnbly, or to say it is the spool already, there is no need to send
'Dummy1' entity, since there is no more assembly
                                        If Not CurrentAssemblyStage =
FinalStage Then
                                            Dim DummyEnt As CFCSim_Entity
                                            DummyEnt = ob.AddEntity()
                                            DummyEnt("SpoolElement") =
NewEntity("SpoolElement")
                                            DummyEnt("JobControlNumber")
= NewEntity("JobControlNumber")
                                            DummyEnt("EntityType") =
"Dummy1"

      ob.ScheduleEvent(DummyEnt,"CheckPossibleAssembly",0)
                                        End If
                                    End If
                                End If


                                ob.TransferOut(NewEntity,
ob.ConnectionPoints("Out1"))
                                'MessageBox.Show(NewEntity("SpoolPartID"))
                                Exit For
                            End If
                        Next


                        'Remove the IDs of the spool parts in the 'AssemblyFile'
```

```
                  With ob.File("AssemblyFile")
                        'Whether or not the 'LayDownFile' is empty
                        If .Length <> 0 Then
                              .MoveFirst()
                              'Find the ID of spool with highest priority in
the file
                              While (.EOF =False And .Length > 0)
                                    Ent = .Entity
                                    If Ent("JobControlNumber") = SpoolID Then
                                          If
PreconditionsForSubassembly.Contains(Ent("SpoolPartID"))Then
                                                .Remove(Ent)
                                          Else
                                                .MoveNext()
                                          End If
                                    Else
                                          .MoveNext()
                                    End If
                              End While
                        End If
                  End With
            End If
      Next
      'MessageBox.Show(CanbeAssembledSpoolComponents.Count)
End Sub


Public sub DepthFirstTravers(MyDataTable As System.Data.DataTable,
CurrentComponentID As Integer, AllChildSpoolComponents As ArrayList)
      'datatable only refers to "SpoolComponentsItemsRelationship" in
Spool_Element
```

```
      'to find all the child parts of the current spool component
      Dim ChildParts As New ArrayList
      For i As Integer =0 to MyDataTable.Rows.Count - 1
            If MyDataTable.Rows.Item(i)("ComponentID") = CurrentComponentID
Then
                  ChildParts.Add(MyDataTable.Rows.Item(i)("PartID"))
            End If
      Next
      Dim MyRow As System.Data.DataRow
      For Each MyRow In MyDataTable.Rows
            Dim MyRowCompID As Integer
            MyRowCompID = MyRow.Item("ComponentID")
            If ChildParts.Contains(MyRowCompID) Then
                  DepthFirstTravers(MyDataTable, MyRowCompID,
AllChildSpoolComponents)
            End If
      Next

      'At the bottom level, the constituent parts should be spool items,
which are not listed in "SpoolComponentsItemsRelationship"
      'At this point of time, the recursion will stop.
      'Also from now on, The subroutine starts to collect child component IDs
all the way from the bottom level to the top level
      If Not AllChildSpoolcomponents.Contains(CurrentComponentID) Then
            AllChildSpoolcomponents.Add(CurrentComponentID)
      End If
End Sub

End Module
End Namespace
```

This is the VB.NET code for the 'StationIdleCheck' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go

Namespace SimphonyScript
Public Module Script
'Your functions here


Public Function StationIdleCheck_OnCreate(ob As
CFCSim_ModelingElementInstance, x As Double, y As Double) As Boolean handles
Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)
      ob.SetNumCoordinates(2)
      ob.Coordinates(0) = new System.Drawing.PointF(x, y)
      ob.Coordinates(1) = new System.Drawing.PointF(x + 50, y + 50)


      ob.AddConnectionPoint("In" , x - 15, y + 25, TConnectionType.CInput, 5)
      ob.AddConnectionPoint("Out", x + 65, y + 25, TConnectionType.COutput,5)

      Return True

End Function


Public Function StationIdleCheck_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle
      r.Size = new SizeF(50, 50)

      Dim text1 As new GoText
      text1.Text = "IdleCheck"
      text1.FontSize = 7
      text1.Position = new PointF(10, 20)
```

```
	Dim g As new GoGroup
	g.Add(r)
	g.Add(text1)

	return g

End Function


Public Sub StationIdleCheck_OnSimulationTransferIn(ob As
CFCSim_ModelingElementInstance, entity As CFCSim_Entity, srcPt As
CFCSim_ConnectionPoint, dstPt As CFCSim_ConnectionPoint) handles
Scripting.OnSimulationTransferInEvent

	If entity("EntityType")="Dummy" Then

		Dim MyWaitingFile As CFCSim_ModelingElementInstance
		'check whether the waiting file is empty
		For each MyWaitingFile in ob.Parent.ChildElements.Values
			If MyWaitingFile.ElementType = "WaitingFile" Then
				If MyWaitingFile.File("Waiting_File").Length = 0 Then
					entity("AvailableOrNot") = "Avail"
				Else
					entity("AvailableOrNot") = "NotAvail"
				End If
				ob.TransferOut(Entity)
			End If
		Next

	Else If entity("EntityType")= "SpoolComponent" Then

		Dim MyWaitingFile As CFCSim_ModelingElementInstance
		'check whether the waiting file is empty
		For each MyWaitingFile in ob.Parent.ChildElements.Values
			If MyWaitingFile.ElementType = "WaitingFile" Then
				If MyWaitingFile.File("Waiting_File").Length = 0 Then
					'If it is empty, send a dummy entity to laydown
area so that it would send a new spool to the fitting station
					Dim NewDummyEntity As CFCSim_Entity
					NewDummyEntity = ob.AddEntity()
					NewDummyEntity("EntityType") = "Dummy"
					NewDummyEntity("RequestFittingStation") =
ob.Parent("StationID").Value
					NewDummyEntity("RequestLayDownArea") =
ob.Parent("FromLaydownArea").Value
					NewDummyEntity("AvailableOrNot") = "Avail"
					ob.TransferOut(NewDummyEntity)
				End If
			End If
		Next

		ob.TransferOut(entity)
	End If
```

```
End Sub

End Module
End Namespace
```

This is the VB.NET code for the 'Handling' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go

Namespace SimphonyScript
Public Module Script
'Your functions here



Public Function Handling_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Double, y As Double) As Boolean handles Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)
      ob.SetNumCoordinates(2)
      ob.Coordinates(0) = new System.Drawing.PointF(x, y)
      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)

      ob.AddAttribute("HandlingType", "Number of entities passing the
counter",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRe
presentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)

      Return True

End Function


Public Function Handling_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle
      r.Size = new SizeF(120, 50)
```

```
        Dim text1 As new GoText
        text1.Text = "Handling"
        text1.FontSize = 9
        text1.Position = new PointF(20, 15)


        Dim g As new GoGroup
        g.Add(r)
        g.Add(text1)

        return g

End Function


Public Function Handling_OnListBoxInitialize(ob As
CFCSim_ModelingElementInstance, attr As CFCSim_Attribute) As ArrayList
handles Scripting.OnListBoxInitializeEvent

        Dim AttrList As New ArrayList
        If attr.Name = "HandlingType" Then
              AttrList.Add("FromLayDownArea")
              AttrList.Add("ToLayDownArea")
        End If

        Return AttrList

End Function

End Module
End Namespace
```

This is the VB.NET code for the 'CraneDecider' Element

```
'Simphony.NET Template Code

'Force explicit variable declaration and have automatic conversion of data
types

Option Explicit

Option Strict Off



'Imports for commonly used namespaces

Imports System

Imports System.Collections

Imports System.Diagnostics
```

```
Imports System.Math

Imports Simphony.NET

Imports System.Drawing

Imports System.Drawing.Drawing2D

Imports System.Windows.Forms

Imports Northwoods.Go


Namespace SimphonyScript

Public Module Script

'Your functions here

Public Function CraneDecider_OnCreate(ob As CFCSim_ModelingElementInstance, x
As Double, y As Double) As Boolean handles Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)

      ob.SetNumCoordinates(2)

      ob.Coordinates(0) = new System.Drawing.PointF(x, y)

      ob.Coordinates(1) = new System.Drawing.PointF(x + 100, y + 50)

      ob.AddConnectionPoint("In" , x - 15, y + 25, TConnectionType.CInput, 5)

      ob.AddConnectionPoint("Out1", x + 125, y + 10,
TConnectionType.COutput,5)

      ob.AddConnectionPoint("Out2", x + 125, y + 35,TConnectionType.COutput,
5)

      Return True

End Function


Public Function CraneDecider_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent


      Dim r As new GoRectangle

      r.Size = new SizeF(120, 50)
```

```
	Dim text1 As new GoText

	text1.Text = "CraneDecider"

	text1.FontSize = 9

	text1.Position = new PointF(20, 15)

	Dim g As new GoGroup

	g.Add(r)

	g.Add(text1)

	return g

End Function


Public Sub CraneDecider_OnSimulationTransferIn(ob As
CFCSim_ModelingElementInstance, entity As CFCSim_Entity, srcPt As
CFCSim_ConnectionPoint, dstPt As CFCSim_ConnectionPoint) handles
Scripting.OnSimulationTransferInEvent

	If ob.Parent("HandlingType").Value = "FromLayDownArea" Then

		'To find which welding station it captured

		Dim MyWeldingStationResource As CFCSim_ModelingElementInstance

		MyWeldingStationResource = entity("CEM_Common_RqstdRes")

		entity("RollWeldingStation") =
MyWeldingStationResource("ResName").Value

	End If

	Dim SideOfBay As String

	For i As Integer = 0 to ob.Parent.Parent("StationLocations").RowCount -
1

		If ob.Parent.Parent("StationLocations").GetValueRC(i,0) =
entity("RollWeldingStation") Then

			SideOfBay =
ob.Parent.Parent("StationLocations").GetValueRC(i,3)

			'MessageBox.Show(SideOfBay)

			Exit For
```

```
            End If
      Next
      If SideOfBay = "Left" then
            ob.TransferOut(entity, ob.ConnectionPoints("Out1"))
      Else
            ob.TransferOut(entity, ob.ConnectionPoints("Out2"))
      End If
End Sub
End Module
End Namespace
```

This is the VB.NET code for the 'WeldingResReminder' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off
'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go
```

```
Namespace SimphonyScript

Public Module Script

'Your functions here

Public Function WeldingResReminder_OnCreate(ob As
CFCSim_ModelingElementInstance, x As Double, y As Double) As Boolean handles
Scripting.OnCreateEvent

      ob.OnCreate(x,y,True)

      ob.SetNumCoordinates(2)

      ob.Coordinates(0) = new System.Drawing.PointF(x, y)

      ob.Coordinates(1) = new System.Drawing.PointF(x + 50, y + 50)

      ob.AddConnectionPoint("In" , x - 15, y + 25, TConnectionType.CInput, 5)

      ob.AddConnectionPoint("Out", x + 65, y + 25, TConnectionType.COutput,5)

      Return true

End Function


Public Function WeldingResReminder_OnGraphicsInitialize(ob As
CFCSim_ModelingElementInstance) As GoObject handles
Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle

      r.Size = new SizeF(50, 50)

      Dim text1 As new GoText

      text1.Text = "Reminder"

      text1.FontSize = 7

      text1.Position = new PointF(15, 20)

      Dim g As new GoGroup

      g.Add(r)

      g.Add(text1)

      return g

End Function
```

```
Public Sub WeldingResReminder_OnSimulationTransferIn(ob As
CFCSim_ModelingElementInstance, entity As CFCSim_Entity, srcPt As
CFCSim_ConnectionPoint, dstPt As CFCSim_ConnectionPoint) handles
Scripting.OnSimulationTransferInEvent

      Dim WeldingResourceID As String

      WeldingResourceID = entity("RollWeldingStation")

      'change the entity attributes

      'one of the attributes is "CEM_Common_RqstdRes"

      Dim MyResourceElement As CFCSim_ModelingElementInstance

      For Each MyResourceElement In SimEnvironment.Elements.Values

            If MyResourceElement.ElementType = "Resource" Then

                  If MyResourceElement("ResName").Value = WeldingResourceID
Then

                        entity("CEM_Common_RqstdRes") = MyResourceElement

                        Exit For

                  End If

            End If

      Next

      'Another attribute is "CEM_Common_RqstElmnt"

      Dim MyResourceRequestElement As CFCSim_ModelingElementInstance

      For Each MyResourceRequestElement In
ob.Parent.Parent.ChildElements.Values

            If MyResourceRequestElement.ElementType = "Capture" Then

                  If MyResourceRequestElement.Parent.ElementType =
"StationElement" Then

                        entity("CEM_Common_RqstElmnt") =
MyResourceRequestElement
```

```
                        'MessageBox.Show("OK")

                        Exit For

                    End If

                End If

        Next

        ob.TransferOut(entity)

End Sub


End Module

End Namespace
```

This is the VB.NET code for the 'End' Element

```
'Simphony.NET Template Code
'Force explicit variable declaration and have automatic conversion of data
types
Option Explicit
Option Strict Off

'Imports for commonly used namespaces
Imports System
Imports System.Collections
Imports System.Diagnostics
Imports System.Math
Imports Simphony.NET
Imports System.Drawing
Imports System.Drawing.Drawing2D
Imports System.Windows.Forms
Imports Northwoods.Go
Imports System.Data
Imports System.Data.OleDb

Namespace SimphonyScript
Public Module Script
'Your functions here


Public Function End_OnCreate(ob As CFCSim_ModelingElementInstance, x As
Double, y As Double) As Boolean handles Scripting.OnCreateEvent

        ob.OnCreate(x,y,True)
        ob.SetNumCoordinates(2)
```

```
      ob.Coordinates(0) = new System.Drawing.PointF(x, y)
      ob.Coordinates(1) = new System.Drawing.PointF(x + 50, y + 50)
      ob.AddConnectionPoint("In" , x - 15, y + 25, TConnectionType.CInput, 5)
      ob.AddConnectionPoint("Out", x + 65, y + 25, TConnectionType.COutput,5)

      ob.AddAttribute("CuttingActivityTableName", "The table name of
'cutting' activity in the
database",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("FittingActivityTableName", "The table name of
'fitting' activity in the
database",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("WeldingActivityTableName", "The table name of
'welding' activity in the
database",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("HandlingActivityTableName", "The table name of
'handling' activity in the
database",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalR
epresentation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      ob.AddAttribute("OutputDatabaseFileAddress","The address of the
database to which the simulation model will send the resulting schedule
to",CFC_AttributeInternalRepresentation.CFC_Text,CFC_AttributeExternalRepresе
ntation.CFC_Singular,CFC_AttributeAccess.CFC_ReadWrite)
      return true
End Function


Public Function End_OnGraphicsInitialize(ob As CFCSim_ModelingElementInstance)
As GoObject handles Scripting.OnGraphicsInitializeEvent

      Dim r As new GoRectangle
      r.Size = new SizeF(50, 50)

      Dim text1 As new GoText
      text1.Text = "End"
      text1.FontSize = 9
      text1.Position = new PointF(15, 20)


      Dim g As new GoGroup
      g.Add(r)
      g.Add(text1)

      return g

End Function


Public Sub End_OnSimulationTransferIn(ob As CFCSim_ModelingElementInstance,
entity As CFCSim_Entity, srcPt As CFCSim_ConnectionPoint, dstPt As
CFCSim_ConnectionPoint) handles Scripting.OnSimulationTransferInEvent

      'If entity("JobControlNumber") = "B696-00875" Then
            'MessageBox.Show("Spool " & entity("JobControlNumber") & " Comes
In!")
```

```
      'End If

      Dim MySpoolElement As CFCSim_ModelingElementInstance
      MySpoolElement = entity("SpoolElement")

      Dim StartTime As Double
      Dim FinishTime As Double
      Dim CycleTime As Double

      'To find the starting point of processing the current spool
      StartTime = 1000000
      For i As Integer = 0 to MySpoolElement("CuttingActivities").RowCount -
1
            If StartTime > MySpoolElement("CuttingActivities").GetValueRC(i,
9) Then
                  StartTime =
MySpoolElement("CuttingActivities").GetValueRC(i, 9)
            End If
      Next

      'To find the finishing poing of processing the current spool
      FinishTime = 0
      For i As Integer = 0 to MySpoolElement("WeldingActivities").RowCount -
1
            If FinishTime <
MySpoolElement("WeldingActivities").GetValueRC(i,10) Then
                  FinishTime =
MySpoolElement("WeldingActivities").GetValueRC(i,10)
            End If
      Next

      'Calculate the cycle time
      CycleTime = FinishTime - StartTime
      MySpoolElement("Duration").Value = CycleTime
      'Update the state of the spool
      MySpoolElement("SpoolState").Value = "Completed"

      'To find how many handlings involved during the fabrication
      Dim NumOfHandlings As Integer
      NumOfHandlings = MySpoolElement("HandlingActivities").RowCount
      MySpoolElement("NumOfHandling").Value = NumOfHandlings

      ob.TransferOut(entity)
End Sub


Public Sub End_OnSimulationPostRun(ob As CFCSim_ModelingElementInstance,
runNum As Int32, m_status As CFC_Simulationstatus) handles
Scripting.OnSimulationPostRunEvent

      Dim cn As OleDb.OleDbConnection
      'Dim MydataAdapter3 As OleDb.OleDbDataAdapter
      'Dim MydataAdapter4 As OleDb.OleDbDataAdapter
      Dim DatabaseAddress As String
      Dim CuttingTableName As String
      Dim FittingTableName As String
      Dim WeldingTableName As String
```

```
      Dim HandlingTableName As String

      'To get the table name for each type of actvity
      CuttingTableName = ob("CuttingActivityTableName").Value
      FittingTableName = ob("FittingActivityTableName").Value
      WeldingTableName = ob("WeldingActivityTableName").Value
      HandlingTableName = ob("HandlingActivityTableName").Value
      DatabaseAddress = ob("OutputDatabaseFileAddress").Value


      'to define the connection
      cn = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & DatabaseAddress)


      '***************Cutting
Data*************************************************************************
***************

      'to define the DataAdapter for cutting
      Dim MydataAdapter1 As OleDb.OleDbDataAdapter
      MyDataAdapter1 = New OleDbDataAdapter("SELECT * FROM " &
CuttingTableName & ";",cn)
      Dim cd1 As New OleDbCommandBuilder(MyDataAdapter1)

      'To create a datatable to hold information from database
      Dim MyDestinationDataTable1 As New DataTable()
      Dim SourceRow1 As DataRow
      Dim DestinationRow1 As DataRow

      'to send the 'Cutting' schedule information back to the database
      Try

            cn.open()

            'To get the original data from database
            MyDataAdapter1.Fill(MyDestinationDataTable1)

            'Update it with data in simulation
            Dim MySpoolElement As CFCSim_ModelingElementInstance
            For Each MySpoolElement In SimEnvironment.Elements.Values
                  If MySpoolElement.ElementType = "Spool_Element" Then
                        For Each SourceRow1 In
MySpoolElement("CuttingActivities").DataTable.Rows
                              For Each DestinationRow1 In
MyDestinationDataTable1.Rows
                                    'to find the corresponding row in the
destination datatable
                                    If
DestinationRow1.Item("JobControlNumber") = SourceRow1.Item("JobControlNumber")
And DestinationRow1.Item("SpoolPartID") = SourceRow1.Item("SpoolPartID") And
DestinationRow1.Item("CuttingID") = SourceRow1.Item("CuttingID") Then
                                          DestinationRow1.Item("StartTime")=
SourceRow1.Item("StartTime")
                                          DestinationRow1.Item("Duration")=
SourceRow1.Item("Duration")
```

```
                                          DestinationRow1.Item("FinishTime")=
SourceRow1.Item("FinishTime")
                                    End If
                              Next
                        Next
                  End If
            Next

            MyDataAdapter1.Update(MyDestinationDataTable1)

      Catch ex As Exception
        MessageBox.Show(ex.Message)
      Finally
            cn.Close()
      End Try

      '*****************Fitting
Data*************************************************************************
**********

      'to define the DataAdapter for fitting
      Dim MydataAdapter2 As OleDb.OleDbDataAdapter
      MyDataAdapter2 = New OleDbDataAdapter("SELECT * FROM " &
FittingTableName & ";",cn)
      Dim cd2 As New OleDbCommandBuilder(MyDataAdapter2)

      'To create a datatable to hold information from database
      Dim MyDestinationDataTable2 As New DataTable()
      Dim SourceRow2 As DataRow
      Dim DestinationRow2 As DataRow

      'to send the 'Fitting' schedule information back to the database
      Try

            cn.open()

            MyDataAdapter2.Fill(MyDestinationDataTable2)

            Dim MySpoolElement As CFCSim_ModelingElementInstance
            For Each MySpoolElement In SimEnvironment.Elements.Values
                  If MySpoolElement.ElementType = "Spool_Element" Then
                        For Each SourceRow2 In
MySpoolElement("FittingActivities").DataTable.Rows
                              For Each DestinationRow2 In
MyDestinationDataTable2.Rows
                                    'to find the corresponding row in the
destination datatable
                                    If
DestinationRow2.Item("JobControlNumber")  =  SourceRow2.Item("JobControlNumber")
And DestinationRow2.Item("WeldID") = SourceRow2.Item("WeldID") And
DestinationRow2.Item("FittingID") = SourceRow2.Item("FittingID") Then
                                          DestinationRow2.Item("StartTime")=
SourceRow2.Item("StartTime")
                                          DestinationRow2.Item("Duration")=
SourceRow2.Item("Duration")
                                          DestinationRow2.Item("FinishTime")=
SourceRow2.Item("FinishTime")
```

```
      DestinationRow2.Item("FittingTableID")=
SourceRow2.Item("FittingTableID")
                                    End If
                              Next
                        Next
                  End If
            Next

            MyDataAdapter2.Update(MyDestinationDataTable2)

      Catch ex As Exception
        MessageBox.Show(ex.Message)
      Finally
            cn.Close()
      End Try


      '***************Welding
Data*************************************************************************
*****************
      'to define the DataAdapter for fitting
      Dim MydataAdapter3 As OleDb.OleDbDataAdapter
      MyDataAdapter3 = New OleDbDataAdapter("SELECT * FROM " &
WeldingTableName & ";",cn)
      Dim cd3 As New OleDbCommandBuilder(MyDataAdapter3)

      'To create a datatable to hold information from database
      Dim MyDestinationDataTable3 As New DataTable()
      Dim SourceRow3 As DataRow
      Dim DestinationRow3 As DataRow

      'to send the 'Fitting' schedule information back to the database
      Try

            cn.open()

            MyDataAdapter3.Fill(MyDestinationDataTable3)

            Dim MySpoolElement As CFCSim_ModelingElementInstance
            For Each MySpoolElement In SimEnvironment.Elements.Values
                  If MySpoolElement.ElementType = "Spool_Element" Then
                        For Each SourceRow3 In
MySpoolElement("WeldingActivities").DataTable.Rows
                              For Each DestinationRow3 In
MyDestinationDataTable3.Rows
                                    'to find the corresponding row in the
destination datatable
                                    If
DestinationRow3.Item("JobControlNumber") = SourceRow3.Item("JobControlNumber")
And DestinationRow3.Item("WeldID") = SourceRow3.Item("WeldID") And
DestinationRow3.Item("WeldingID") = SourceRow3.Item("WeldingID") Then
                                          DestinationRow3.Item("StartTime")=
SourceRow3.Item("StartTime")
                                          DestinationRow3.Item("Duration")=
SourceRow3.Item("Duration")
```

```
                                                DestinationRow3.Item("FinishTime")=
SourceRow3.Item("FinishTime")

      DestinationRow3.Item("WeldingMachineID")=
SourceRow3.Item("WeldingMachineID")
                                        End If
                                    Next
                                Next
                            End If
                        Next

                        MyDataAdapter3.Update(MyDestinationDataTable3)

      Catch ex As Exception
        MessageBox.Show(ex.Message)
      Finally
            cn.Close()
      End Try


      '******************Handling
Data*************************************************************************
****************************
      'to define the DataAdapter for fitting
      Dim MydataAdapter4 As OleDb.OleDbDataAdapter
      MyDataAdapter4 = New OleDbDataAdapter("SELECT * FROM " &
HandlingTableName & ";",cn)
      Dim cd4 As New OleDbCommandBuilder(MyDataAdapter4)

      Dim MyDataTable4 As New DataTable()
      Dim MyDataTable5 As New DataTable()

      Try
            cn.open()
            MyDataAdapter4.Fill(MyDataTable4)

            'To delete the existing data
            If Not MyDataTable4.Rows.Count = 0 Then
                  For i As Integer = 0 to MyDataTable4.Rows.Count - 1
                        MyDataTable4.Rows(i).Delete
                  Next
                  MyDataAdapter4.Update(MyDataTable4)
            End If

            'Send the resulting information back
            Dim MySpoolElement As CFCSim_ModelingElementInstance
            For Each MySpoolElement In SimEnvironment.Elements.Values
                  If MySpoolElement.ElementType = "Spool_Element" Then

                        MyDataTable5 =
MySpoolElement("HandlingActivities").DataTable
                        MydataAdapter4.Update(MyDataTable5)
            End If
            Next

      Catch ex As Exception
        MessageBox.Show(ex.Message)
```

```
      Finally
            cn.Close()
      End Try

      '***************Update the spool
information*************************************************************
************************
      Dim MydataAdapter5 As OleDb.OleDbDataAdapter
      MyDataAdapter5 = New OleDbDataAdapter("SELECT * FROM Spool;",cn)
      Dim cd5 As New OleDbCommandBuilder(MyDataAdapter5)

      Dim MyDestinationDataTable5 As New DataTable()
      Dim DestinationRow5 As DataRow
      Dim SpoolID As String
      Dim SpoolFinishTime As Double
      Dim SpoolDuration As Double

      Try

            cn.open()

            MyDataAdapter5.Fill(MyDestinationDataTable5)

            Dim MySpoolElement As CFCSim_ModelingElementInstance
            For Each MySpoolElement In SimEnvironment.Elements.Values
                  If MySpoolElement.ElementType = "Spool_Element" Then
                        SpoolID = MySpoolElement("JobControlNumber").Value
                        SpoolDuration = MySpoolElement("Duration").Value
                        SpoolFinishTime = 0
                        For i As Integer = 0 to
MySpoolElement("WeldingActivities").RowCount - 1
                              If SpoolFinishTime <
MySpoolElement("WeldingActivities").GetValueRC(i,10) then
                                    SpoolFinishTime =
MySpoolElement("WeldingActivities").GetValueRC(i,10)
                              End If
                        Next
                        For Each DestinationRow5 In
MyDestinationDataTable5.Rows
                              'to find the corresponding row in the
destination datatable
                              If DestinationRow5.Item("JobControlNumber") =
SpoolID Then
                                    DestinationRow5.Item("FinishDate")=
SpoolFinishTime
                                    DestinationRow5.Item("Duration")=
SpoolDuration
                                    DestinationRow5.Item("SpoolState")=
"Completed"
                              End If
                        Next
                  End If
            Next

            MyDataAdapter5.Update(MyDestinationDataTable5)

      Catch ex As Exception
```

```
        MessageBox.Show(ex.Message)
      Finally
            cn.Close()
      End Try

End Sub

End Module
End Namespace
```

Appendix B

The part of the algorithm for pipe spool fabrication sequencing

```
import string
from string import*
import itertools
from itertools import*

# define axis
xAxis=0
yAxis=1
zAxis=2
nAxis=3

# weld class
class weld:
    def __init__(self,id,location,axs):
        self.id=id
        self.loc=location
        self.stg=0
        self.done=False
        self.rollAxs=axs

# part class
class part:
    def __init__(self,id,location,dimensions):
        self.id=id
        self.loc=location
        self.dim=dimensions

# assembly class
class assembly:
    def __init__(self):
        # since Location involves three coordinates (x, y, z) so a list is
need
        self.locMax=[]
        self.locMin=[]
        self.P={}
        self.W={}
    # create the P dictionary indexed by parts
    def prtsIndx(self):
        # add parts that appear in Welds dictionary
        for w,P in self.W.iteritems():
            for p in P:
                if p not in self.P:
                    self.P[p]=[]
        # for each part added, update its weld list
        for prt,w in self.P.iteritems():
            for k,P in self.W.iteritems():
                if prt in P:
                    w.append(k)
    # get max dimensions for the assembly
    def maxDim(self):
        if self.locMax <> [] and self.locMin <> []:
```

```
            return self.locMin, self.locMax
        else:
            # get lowest and highest coordinates for bounding box
            Rx=[]
            Ry=[]
            Rz=[]
            Lx=[]
            Ly=[]
            Lz=[]
            for p in self.P:
                Rx.append(p.loc[0])
                Ry.append(p.loc[1])
                Rz.append(p.loc[2])
                Lx.append(p.loc[0]+p.dim[0])
                Ly.append(p.loc[1]+p.dim[1])
                Lz.append(p.loc[2]+p.dim[2])
            #the first element in the list is the minimum coordinates on X
axis
            #the second element is the minimum coordinates on Y axis
            self.locMin.append(min(Rx))
            self.locMin.append(min(Ry))
            self.locMin.append(min(Rz))

            self.locMax.append(max(Lx))
            self.locMax.append(max(Ly))
            self.locMax.append(max(Lz))

            return self.locMin, self.locMax

    # update the assembly data structure
    def update(self):
        self.prtsIndx()
        self.maxDim()

# SplitAt (assembly A, weld W)
# # split A at weld W and return two sub-assemblies A1, A2
def SplitAt (A,wld):
    A1=assembly()
    A2=assembly()
    p1=A.W[wld][0]
    p2=A.W[wld][1]
    A1P=set([p1]) # set of parts in A1 start with one from the given weld
    A1.P[p1]=[]
    A2P=set([p2]) # set of parts in A2 start with another from the given weld
    A2.P[p2]=[]
    change = True
    while change: # loop until no more changes to A2
        change=False
        for w,P in A.W.iteritems(): # loop welds and associated parts in A
            if w <> wld: # skip split weld
                # if weld has parts shared with A2 part set and weld not in
A2 weld list, add it
                if A2P.intersection(P) <> set() and w not in A2.W:
                    A2P=A2P.union(P)
                    A2.W[w]=P
                    change=True
    for w,P in A.W.iteritems(): # find welds for A1
```

```
        # if weld not the split weld and not in A2, add it to A1
        if w <> wld and w not in A2.W:
            A1.W[w]=P
    A1.update()
    A2.update()
return A1, A2

# GetWeldType(A1,A2,Wld,clrnc)
#   make sure Wld not in A1 or A2
#   get roll axis for Wld
#   get MaxClrnc needed for A1 and A2 and roll axis
#   if MaxClrnc less than or equal clrnc then roll else position
def GetWeldType (A1,A2,wld,clrnc):
    if wld in A1.W or wld in A2.W:
        return "error: weld is part of one of the assemblies"
    else:
        rolldistances=[]
        #rollneed= execluding roll axis, max of difference between
        #weld location on other two axis and the min and max locations of A1
and A2
        for i in range(nAxis): # use 3 later after adding z axis
            if i <> wld.rollAxs:
                rolldistances.append(abs(A1.locMax[i]-wld.loc[i]))
                rolldistances.append(abs(wld.loc[i]-A1.locMin[i]))
                rolldistances.append(abs(A2.locMax[i]-wld.loc[i]))
                rolldistances.append(abs(wld.loc[i]-A2.locMin[i]))
        rollneed=max(rolldistances)
        if rollneed<=clrnc:
            return 1
        else:
            return 2

def GetWeldType2 (A,clrnc):
    if len(A.W) > 1:
        return "error: too many welds"
    if len(A.W)==1: # one weld
        wld=A.W.keys()[0]
        rolldistances=[]
        for i in range(nAxis):
            if i<>wld.rollAxs:
                rolldistances.append(abs(A.locMax[i]-wld.loc[i]))
                rolldistances.append(abs(wld.loc[i]-A.locMin[i]))
        rollneed=max(rolldistances)
        if rollneed<=clrnc:
            return 1
        else:
            return 2
    else:
        return 0 # no welds exist

# MinCost (assembly A)
#  # returns minimum total cost in terms of number of position and roll welds
#  # returns also an ordered list of welds representing the sequence to
follow to acheive that cost
def MinCost(A,clrnc,stg): # return total cost and a list of weld sequence
    # initial values for weld with minimum cost
    # start with any weld and minimum cost = maximum integer
```

```
    # initialize weld sequence to empty list
    minCst=2147483647
    minWld=None
    prvSqnce=[]
    wldSqnce=[]
    stg-=1

    if len(A.W)==0:
        minCst=0
    # the condition should be changed to when all welds in A are on the same
axis, then they can be processed at the same stage
    # except when position welding is required
    # The assumption that only two parts and one weld can be processes at one
time is FALSE!!
    elif len(A.W)==1:
        minCst=GetWeldType2(A,clrnc)
        minWld=A.W.keys()[0].id
    else:
        for wld in A.W:
            A1,A2= SplitAt(A,wld)
            cost1,w1,s1=MinCost(A1,clrnc,stg)
            cost2,w2,s2=MinCost(A2,clrnc,stg)
            cost3=GetWeldType(A1,A2,wld,clrnc)
            totalcost=cost1+cost2+cost3
            #print "w:",wld.id,totalcost

            if totalcost < minCst:
                minCst=totalcost
                minWld=wld.id
                prvSqnce=[s1,s2]

    wldSqnce.append(stg)
    wldSqnce.append(minWld)
    wldSqnce.extend(prvSqnce)

    #print minCst, minWld, wldSqnce,stg
return minCst,minWld, wldSqnce

def main():
    pass

if __name__ == '__main__':
    main()
```

The part for inputting the configuration and geometry of a specific pipe spool and outputting the optimal fabrication sequence

```
# Parts
p7 = part(7, [449,0,171], [229,0,0])
p1 = part(1, [249,0,171], [200,0,0])
p5 = part(5, [329,0,171], [0,0,50])
p6 = part(6, [21,0,171], [228,0,0])
p11 = part(11, [0,0,150], [21,0,42])
p13 = part(13, [0,0,100], [0,0,50])
p15 = part(15, [0,0,0], [0,0,100])
p4 = part(4, [0,0,192], [0,0,100])
p14 = part(14, [0,0,292], [0,0,62])
```

```
p3 = part(3, [0,0,354], [0,0,330])
p10 = part(10, [0,0,684], [0,0,40])
p2 = part(2, [0,0,724], [0,0,1692])
p12 = part(12, [0,0,2416], [0,0,100])
p8 = part(8, [0,0,2516], [0,0,50])
p9 = part(9, [0,0,2566], [68,0,68])

# Welds
wldList=[]
w1 = weld(1, [0,0,2566], SS.zAxis)
wldList.append(w1)
w2 = weld(2, [0,0,2516], SS.zAxis)
wldList.append(w2)
w3 = weld(3, [0,0,2416], SS.zAxis)
wldList.append(w3)
w4 = weld(4, [0,0,724], SS.zAxis)
wldList.append(w4)
w5 = weld(5, [0,0,684], SS.zAxis)
wldList.append(w5)
w6 = weld(6, [0,0,354], SS.zAxis)
wldList.append(w6)
w7 = weld(7, [0,0,292], SS.zAxis)
wldList.append(w7)
w8 = weld(8, [0,0,192], SS.zAxis)
wldList.append(w8)
w9 = weld(9, [21,0,171], SS.xAxis)
wldList.append(w9)
w10 = weld(10, [249,0,171], SS.xAxis)
wldList.append(w10)
w11 = weld(11, [329,0,171], SS.zAxis)
wldList.append(w11)
w12 = weld(12, [449,0,171], SS.xAxis)
wldList.append(w12)
w13 = weld(13, [0,0,150], SS.zAxis)
wldList.append(w13)
w14 = weld(14, [0,0,100], SS.zAxis)
wldList.append(w14)

MA19 = assembly()
MA19.loc=[0,0,0]

MA19.W={w1:[p8,p9],w2:[p8,p12],w3:[p2,p12],w4:[p2,p10],w5:[p3,p10],w6:[p3,p14
],w7:[p4,p14],w8:[p4,p11],w9:[p6,p11],w10:[p1,p6],w11:[p1,p5],w12:[p1,p7],w13
:[p11,p13],w14:[p13,p15]}

def main():
    Assembly=MA19
    MinCost,MinWeld,AssemblSeq=SS.MinCost(Assembly,clrnc,0)
    print MinCost
    #print len(AssemblSeq)
    sqnceList=SS.ParseWldSqnce(AssemblSeq)
print sqnceList

pass

if __name__ == '__main__':
    main()
```

Appendix C

VB.NET code for the main windows form of the CDRASS system

```
Imports System.Diagnostics.Process
Imports System.Threading
Imports Simphony
Imports Simphony.Simulation
Imports Simphony.Modeling
Imports System.Data
Imports System.Data.OleDb
Imports System.Windows.Forms.DataVisualization.Charting


Public Class Form1
    Implements IDiscreteEventModel

    Private MyEngine As New DiscreteEventEngine
    Private MyScenarios As New List(Of IDiscreteEventScenario)
    Public MyDataSet As New DataSet

    Public Sub New()

        ' This call is required by the designer.
        InitializeComponent()

        'Read data from Database
        ReadInformationFromDataBase()


        Dim MyScenario As New Scenario(MyEngine, MyDataSet)

        MyScenarios.Add(MyScenario)

        'Initialization.
        MyEngine.InitializeEngine()

        'Start the simulation
        Cursor = Cursors.WaitCursor
        MyEngine.Simulate(Me)
        Cursor = Cursors.Default

        'Initialize Chart
        'InitializeChart()

    End Sub



    Public Sub FinalizeModel() Implements
Simphony.Simulation.IDiscreteEventModel.FinalizeModel
        'DataGridView1.DataSource =
MyDataSet.Tables("tblResourceUsagePerHour")
        DataGridView1.DataSource = MyDataSet.Tables("tblResourceUsagePerDay")
```

```
        DataGridView2.DataSource = MyDataSet.Tables("tblWorkAreaUsagePerDay")
        DataGridView3.DataSource = MyDataSet.Tables("tblWAWPTotalFloat")
        DataGridView4.DataSource = MyDataSet.Tables("tblWAWPPredecessors")
        DataGridView5.DataSource =
MyDataSet.Tables("tblResourceCapturedPerWAWP")
        WriteResultMethod()
        InitializeChart()
        DisplayChart()
    End Sub

    Public Sub InitializeModel() Implements
Simphony.Simulation.IDiscreteEventModel.InitializeModel

    End Sub

    Public ReadOnly Property Scenarios As
System.Collections.Generic.IEnumerable(Of
Simphony.Simulation.IDiscreteEventScenario) Implements
Simphony.Simulation.IDiscreteEventModel.Scenarios
        Get
            Return MyScenarios
        End Get
    End Property

    Public Sub ReadInformationFromDataBase()
        Dim cn As OleDbConnection
        Dim MyDataAdapter As OleDbDataAdapter
        Dim DBAddress As String = Me.DBAddressTxt.Text
        Dim MyPrimaryKeycolumns As DataColumn()

        'Get data from StiteInstallationDBModifiedJuly132011
        cn = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=" & DBAddress)
        Try
            cn.Open()

            'Get Construction Area Info
            MyDataAdapter = New OleDbDataAdapter("Select * from
tblConstructionArea;", cn)
            MyDataAdapter.Fill(MyDataSet, "tblConstructionArea")
            'DataGridView1.DataSource =
MyDataSet.Tables("tblConstructionArea")

            '************************************
            'Get Project Info
            MyDataAdapter = New OleDbDataAdapter("Select * from
tblProjectParameters;", cn)
            MyDataAdapter.Fill(MyDataSet, "tblProjects")
            MyPrimaryKeycolumns = New DataColumn(0) {}
            MyPrimaryKeycolumns(0) =
MyDataSet.Tables("tblProjects").Columns("ProjectNumber")
            'The property 'PrimaryKey' of a datatable is actually an array of
columns
            MyDataSet.Tables("tblProjects").PrimaryKey = MyPrimaryKeycolumns

            '************************************
            'Get Classification Info
```

```
            MyDataAdapter = New OleDbDataAdapter("Select * from
tblClassification;", cn)
            MyDataAdapter.Fill(MyDataSet, "tblClassification")
            MyPrimaryKeycolumns = New DataColumn(0) {}
            MyPrimaryKeycolumns(0) =
MyDataSet.Tables("tblClassification").Columns("ClassificationID")
            MyDataSet.Tables("tblClassification").PrimaryKey =
MyPrimaryKeycolumns

            '************************************
            'Get Work Area Info
            MyDataAdapter = New OleDbDataAdapter("Select * from tblWorkArea;",
cn)
            'tblModules correspond to tblMain in ModuleYard Database
            MyDataAdapter.Fill(MyDataSet, "tblWorkArea")
            MyPrimaryKeycolumns = New DataColumn(0) {}
            MyPrimaryKeycolumns(0) =
MyDataSet.Tables("tblWorkArea").Columns("WADesignator")
            MyDataSet.Tables("tblWorkArea").PrimaryKey = MyPrimaryKeycolumns

            '************************************
            'Get Workpackage Info
            MyDataAdapter = New OleDbDataAdapter("Select * from
Query_Workpackage;", cn)
            MyDataAdapter.Fill(MyDataSet, "tblWorkpackages")
            MyPrimaryKeycolumns = New DataColumn(0) {}
            MyPrimaryKeycolumns(0) =
MyDataSet.Tables("tblWorkpackages").Columns("WPID")
            MyDataSet.Tables("tblWorkpackages").PrimaryKey =
MyPrimaryKeycolumns
            'DataGridView1.DataSource = MyDataSet.Tables("tblWorkpackages")
            '************************************
            'Get WorkPackagePredecessors Info
            MyDataAdapter = New OleDbDataAdapter("Select * from
tblWPPredecessors;", cn)
            MyDataAdapter.Fill(MyDataSet, "tblWPPredecessors")

            '************************************
            'Get Construction Work Area Info
            MyDataAdapter = New OleDbDataAdapter("Select * From
tblConstructionWorkArea;", cn)
            MyDataAdapter.Fill(MyDataSet, "tblConstructionWorkArea")
            MyPrimaryKeycolumns = New DataColumn(0) {}
            MyPrimaryKeycolumns(0) =
MyDataSet.Tables("tblConstructionWorkArea").Columns("CWAName")
            MyDataSet.Tables("tblConstructionWorkArea").PrimaryKey =
MyPrimaryKeycolumns

            '************************************
            'Get WorkAreaWorkpackage Info
            MyDataAdapter = New OleDbDataAdapter("Select * From
Query_WorkAreaWorkpackage;", cn)
            MyDataAdapter.Fill(MyDataSet, "tblWAWP")
            MyPrimaryKeycolumns = New DataColumn(0) {}
            MyPrimaryKeycolumns(0) =
MyDataSet.Tables("tblWAWP").Columns("WAWPID")
            MyDataSet.Tables("tblWAWP").PrimaryKey = MyPrimaryKeycolumns
```

```
            Dim WAWPTotalFloat As DataColumn = New
DataColumn("WAWPTotalFloat")
            WAWPTotalFloat.DataType = System.Type.GetType("System.Double")
            MyDataSet.Tables("tblWAWP").Columns.Add(WAWPTotalFloat)

            Dim WAWPEstDuration As DataColumn = New
DataColumn("WAWPEstDuration")
            WAWPEstDuration.DataType = System.Type.GetType("System.Double")
            MyDataSet.Tables("tblWAWP").Columns.Add(WAWPEstDuration)

            'Find the number of work area that a work package crosses
            Dim WorkpackageAndNumberOfWorkAreas As New Dictionary(Of Integer,
Integer)
            For Each Row As DataRow In
MyDataSet.Tables("tblWorkpackages").Rows
                Dim WPID As Integer = CInt(Row.Item("WPID"))
                Dim NumberOfWorkAreas As Integer = 0
                For Each Row1 As DataRow In MyDataSet.Tables("tblWAWP").Rows
                    If CStr(Row1.Item("WAWPWorkpackageID")) = WPID Then
                        NumberOfWorkAreas = NumberOfWorkAreas + 1
                    End If
                Next
                WorkpackageAndNumberOfWorkAreas.Add(WPID, NumberOfWorkAreas)
            Next

            'The duration of WAWP is derived from dividing the work package
by the number of work areas
            For Each Row As DataRow In MyDataSet.Tables("tblWAWP").Rows
                Dim WPID As Integer = CInt(Row.Item("WAWPWorkpackageID"))
                Dim NumberOfWorkAreas As Integer =
WorkpackageAndNumberOfWorkAreas(WPID)
                Dim WAWPDuration As Double
                If MyDataSet.Tables("tblWorkpackages").Rows.Contains(WPID)
Then
                    If Not
MyDataSet.Tables("tblWorkpackages").Rows.Find(WPID).Item("WPDurationOverride")
Is DBNull.Value Then
                        WAWPDuration =
CDbl(MyDataSet.Tables("tblWorkpackages").Rows.Find(WPID).Item("WPDurationOver
ride"))
                    End If
                End If
                Row.Item("WAWPEstDuration") = WAWPDuration /
NumberOfWorkAreas
            Next

            'DataGridView1.DataSource = MyDataSet.Tables("tblWAWP")

            '************************************
            'Get Calendar Info
            MyDataAdapter = New OleDbDataAdapter("Select * From tblCalendar;",
cn)
            MyDataAdapter.Fill(MyDataSet, "tblCalendar")
            MyPrimaryKeycolumns = New DataColumn(0) {}
            MyPrimaryKeycolumns(0) =
MyDataSet.Tables("tblCalendar").Columns("Cal ID")
```

```
            MyDataSet.Tables("tblCalendar").PrimaryKey = MyPrimaryKeycolumns

            MyDataAdapter = New OleDbDataAdapter("SELECT * FROM
tblCalendar_Detail ORDER BY DateID;", cn)
            MyDataAdapter.Fill(MyDataSet, "tblCalendar_Detail")
            MyPrimaryKeycolumns = New DataColumn(0) {}
            MyPrimaryKeycolumns(0) =
MyDataSet.Tables("tblCalendar_Detail").Columns("Date")
            MyDataSet.Tables("tblCalendar_Detail").PrimaryKey =
MyPrimaryKeycolumns

            '************************************
            'Get Resource Info
            MyDataAdapter = New OleDbDataAdapter("Select * From tblCraft", cn)
            MyDataAdapter.Fill(MyDataSet, "tblResource")
            MyPrimaryKeycolumns = New DataColumn(0) {}
            MyPrimaryKeycolumns(0) =
MyDataSet.Tables("tblResource").Columns("Craft")
            MyDataSet.Tables("tblResource").PrimaryKey = MyPrimaryKeycolumns

            '************************************
            'Get Time Dependent Resource Limit
            MyDataAdapter = New OleDbDataAdapter("Select * From
tblCraftAvailability;", cn)
            MyDataAdapter.Fill(MyDataSet, "tblResourceAvailability")

        Catch ex As Exception
            Throw New ArgumentException(ex.Message)
        Finally
            cn.Close()
        End Try

        'convert WP relationships to WAWP relationships
        ConvertWPRelsToWAWPRels(MyDataSet.Tables("tblWPPredecessors"),
MyDataSet.Tables("tblWAWP"))

        'DataGridView1.DataSource = MyDataSet.Tables("tblWAWPPredecessors")
    End Sub

    Public Sub ConvertWPRelsToWAWPRels(ByVal tblWPPredecessors As DataTable,
ByVal tblWAWP As DataTable)

        'Create a datatable to contain precedence relationships between WAWPs
        Dim MyPrimaryKeycolumns As DataColumn()
        Dim tblWAWPPredecessors As DataTable =
MyDataSet.Tables.Add("tblWAWPPredecessors")
        tblWAWPPredecessors.Columns.Add("WAWPID",
Type.GetType("System.Int32")) 'Successor
        tblWAWPPredecessors.Columns.Add("WAWPPredID",
Type.GetType("System.Int32"))
        tblWAWPPredecessors.Columns.Add("WAWPRel1",
Type.GetType("System.String")) '"FS" "SS"
        tblWAWPPredecessors.Columns.Add("WAWPLag1",
Type.GetType("System.Double"))
        tblWAWPPredecessors.Columns.Add("WAWPRel2",
Type.GetType("System.String")) '"FF"
```

```
        tblWAWPPredecessors.Columns.Add("WAWPLag2",
Type.GetType("System.Double"))
        tblWAWPPredecessors.Columns.Add("SimPredStart",
Type.GetType("System.String"))
        tblWAWPPredecessors.Columns.Add("EstimatedPredFinish",
Type.GetType("System.String"))
        MyPrimaryKeycolumns = New DataColumn(1) {}
        MyPrimaryKeycolumns(0) = tblWAWPPredecessors.Columns("WAWPID")
        MyPrimaryKeycolumns(1) = tblWAWPPredecessors.Columns("WAWPPredID")
        tblWAWPPredecessors.PrimaryKey = MyPrimaryKeycolumns


        For Each WorkAreaWorkPackageRow As DataRow In tblWAWP.Rows
            Dim WAWPID As Integer =
CInt(WorkAreaWorkPackageRow.Item("WAWPID"))
            Dim WAWPWorkpackageID As Integer =
CInt(WorkAreaWorkPackageRow.Item("WAWPWorkpackageID"))
            Dim WAWPWorkAreaID As Integer =
CInt(WorkAreaWorkPackageRow.Item("WAWPWorkAreaNumber"))

            'Find its predecessor WAWP
            For Each PredecessorRow As DataRow In tblWPPredecessors.Rows
                Dim WPPredecessorID As Integer
                If PredecessorRow.Item("WPID") = WAWPWorkpackageID Then
                    'Get the predecessor WPID first
                    WPPredecessorID = CInt(PredecessorRow.Item("WPPredID"))
                    'Get the corresponding predecessor WAWPID which is in the
same area with current WAWP
                    Dim WAWPPredecessorIDs As New List(Of Integer)
                    For Each WAWPRow As DataRow In tblWAWP.Rows
                        If CInt(WAWPRow.Item("WAWPWorkpackageID")) =
WPPredecessorID Then

WAWPPredecessorIDs.Add(CInt(WAWPRow.Item("WAWPID")))
                        End If
                    Next
                    For Each WAWPPredecessorID In WAWPPredecessorIDs
                        Dim NewRow As DataRow = tblWAWPPredecessors.NewRow
                        If Not PredecessorRow.Item("WPRel1") Is DBNull.Value
Then
                            If Not PredecessorRow.Item("WPRel2") Is
DBNull.Value Then
                                NewRow.Item("WAWPID") = WAWPID
                                NewRow.Item("WAWPPredID") = WAWPPredecessorID
                                NewRow.Item("WAWPRel1") =
PredecessorRow.Item("WPRel1")
                                NewRow.Item("WAWPLag1") =
PredecessorRow.Item("WPLag1")
                                NewRow.Item("WAWPRel2") =
PredecessorRow.Item("WPRel2")
                                NewRow.Item("WAWPLag2") =
PredecessorRow.Item("WPLag2")
                                NewRow.Item("SimPredStart") = ""
                                NewRow.Item("EstimatedPredFinish") = ""
                                tblWAWPPredecessors.Rows.Add(NewRow)
                            Else
                                NewRow.Item("WAWPID") = WAWPID
```

```
                                NewRow.Item("WAWPPredID") = WAWPPredecessorID
                                NewRow.Item("WAWPRel1") =
PredecessorRow.Item("WPRel1")
                                NewRow.Item("WAWPLag1") =
PredecessorRow.Item("WPLag1")
                                NewRow.Item("SimPredStart") = ""
                                NewRow.Item("EstimatedPredFinish") = ""
                                tblWAWPPredecessors.Rows.Add(NewRow)
                            End If
                        Else
                            NewRow.Item("WAWPID") = WAWPID
                            NewRow.Item("WAWPPredID") = WAWPPredecessorID
                            NewRow.Item("WAWPRel2") =
PredecessorRow.Item("WPRel2")
                            NewRow.Item("WAWPLag2") =
PredecessorRow.Item("WPLag2")
                            NewRow.Item("SimPredStart") = ""
                            NewRow.Item("EstimatedPredFinish") = ""
                            tblWAWPPredecessors.Rows.Add(NewRow)
                        End If
                    Next
                End If
            Next
        Next

        'Taking out the excessive precedence relationships
        Dim ExcessivePredRelDic As New Dictionary(Of Integer, Integer)
        For Each WAWPPrecedenceRelRow As DataRow In tblWAWPPredecessors.Rows
            Dim WAWPID As Integer = CInt(WAWPPrecedenceRelRow.Item("WAWPID"))
            Dim WAWPWorkAreaID As Integer =
CInt(tblWAWP.Rows.Find(WAWPID).Item("WAWPWorkAreaNumber"))
            Dim WAWPPredID As Integer
            Dim WAWPPredWorkAreaID As Integer
            Select Case WAWPID
                Case 7 To 12 'Module Support Structure
                    WAWPPredID = CInt(WAWPPrecedenceRelRow.Item("WAWPPredID"))
                    'find WAWP Predecessor's work are
                    WAWPPredWorkAreaID =
CInt(tblWAWP.Rows.Find(WAWPPredID).Item("WAWPWorkAreaNumber"))
                    If WAWPWorkAreaID <> WAWPPredWorkAreaID Then
                        ExcessivePredRelDic.Add(WAWPID, WAWPPredID)
                    End If
                Case 46 To 47 'Hydrotesting on top of 007ABC or 014AB
                    WAWPPredID = CInt(WAWPPrecedenceRelRow.Item("WAWPPredID"))
                    'find WAWP Predecessor's work are
                    WAWPPredWorkAreaID =
CInt(tblWAWP.Rows.Find(WAWPPredID).Item("WAWPWorkAreaNumber"))
                    If WAWPWorkAreaID <> WAWPPredWorkAreaID Then
                        ExcessivePredRelDic.Add(WAWPID, WAWPPredID)
                    End If
                Case 53 To 54 'Insulation on top of 007ABc or 014AB
                    WAWPPredID = CInt(WAWPPrecedenceRelRow.Item("WAWPPredID"))
                    'find WAWP Predecessor's work are
                    WAWPPredWorkAreaID =
CInt(tblWAWP.Rows.Find(WAWPPredID).Item("WAWPWorkAreaNumber"))
                    If WAWPWorkAreaID <> WAWPPredWorkAreaID Then
                        ExcessivePredRelDic.Add(WAWPID, WAWPPredID)
```

```
                    End If
                End Select
            Next

            For Each kvp As KeyValuePair(Of Integer, Integer) In
ExcessivePredRelDic
                ' Create an array for the key values to find.
                Dim FindTheseVals(1) As Object
                FindTheseVals(0) = kvp.Key
                FindTheseVals(1) = kvp.Value
                'find the corresponding data row
                Dim ExcessivePredRelRow As DataRow =
tblWAWPPredecessors.Rows.Find(FindTheseVals)
                If Not (ExcessivePredRelRow Is Nothing) Then
                    tblWAWPPredecessors.Rows.Remove(ExcessivePredRelRow)
                End If

            Next
        End Sub

        Public Sub InitializeChart()
            ResourceLoadingChart1.Series.Clear()
            ResourceLoadingChart2.Series.Clear()

            'Resource Loading Chart for Piling
            ResourceLoadingChart1.ChartAreas.Item(0).AxisX.Interval = 1
            ResourceLoadingChart1.ChartAreas.Item(0).AxisX.IntervalType =
DateTimeIntervalType.Days
            ResourceLoadingChart1.ChartAreas.Item(0).AxisX.LabelAutoFitStyle =
LabelAutoFitStyles.LabelsAngleStep90
            ResourceLoadingChart1.Titles.Add("Piling Manpower Loading Curve")
            ' Set chart title font
            ResourceLoadingChart1.Titles(0).Font = New Font("Arial", 12,
FontStyle.Bold)
            ResourceLoadingChart1.Titles(0).ForeColor = Color.White
            ' Set chart title color
            ResourceLoadingChart1.Titles(0).BackColor = Color.Red
            ' Set  axis title
            ResourceLoadingChart1.ChartAreas.Item(0).AxisY.Title = "Manpower"
            ' Set Title font
            ResourceLoadingChart1.ChartAreas.Item(0).AxisY.TitleFont = New
Font("Arial", 12, FontStyle.Bold)
            ' Set Title color
            ResourceLoadingChart1.ChartAreas.Item(0).AxisY.TitleForeColor =
Color.Gray

            ResourceLoadingChart1.Series.Add("MPiling")
            ResourceLoadingChart1.Series("MPiling").ChartType =
SeriesChartType.Column

            'Resource Loading Chart for Electrical
            ResourceLoadingChart2.ChartAreas.Item(0).AxisX.Interval = 1
            ResourceLoadingChart2.ChartAreas.Item(0).AxisX.IntervalType =
DateTimeIntervalType.Days
            ResourceLoadingChart2.ChartAreas.Item(0).AxisX.LabelAutoFitStyle =
LabelAutoFitStyles.LabelsAngleStep90
            ResourceLoadingChart2.Titles.Add("Electrician Manpower Loading Curve")
```

```
        ' Set chart title font
        ResourceLoadingChart2.Titles(0).Font = New Font("Arial", 12,
FontStyle.Bold)
        ResourceLoadingChart2.Titles(0).ForeColor = Color.White
        ' Set chart title color
        ResourceLoadingChart2.Titles(0).BackColor = Color.Red
        ' Set  axis title
        ResourceLoadingChart2.ChartAreas.Item(0).AxisY.Title = "Manpower"
        ' Set Title font
        ResourceLoadingChart2.ChartAreas.Item(0).AxisY.TitleFont = New
Font("Arial", 12, FontStyle.Bold)
        ' Set Title color
        ResourceLoadingChart2.ChartAreas.Item(0).AxisY.TitleForeColor =
Color.Gray

        ResourceLoadingChart2.Series.Add("MELectrical")
        ResourceLoadingChart2.Series("MELectrical").ChartType =
SeriesChartType.Column

        ''Resource Loading Chart for Piping
        ResourceLoadingChart3.ChartAreas.Item(0).AxisX.Interval = 1
        ResourceLoadingChart3.ChartAreas.Item(0).AxisX.IntervalType =
DateTimeIntervalType.Days
        ResourceLoadingChart3.ChartAreas.Item(0).AxisX.LabelAutoFitStyle =
LabelAutoFitStyles.LabelsAngleStep90
        ResourceLoadingChart3.Titles.Add("Pipe fitter Manpower Loading Curve")
        ' Set chart title font
        ResourceLoadingChart3.Titles(0).Font = New Font("Arial", 12,
FontStyle.Bold)
        ResourceLoadingChart3.Titles(0).ForeColor = Color.White
        ' Set chart title color
        ResourceLoadingChart3.Titles(0).BackColor = Color.Red
        ' Set  axis title
        ResourceLoadingChart3.ChartAreas.Item(0).AxisY.Title = "Manpower"
        ' Set Title font
        ResourceLoadingChart3.ChartAreas.Item(0).AxisY.TitleFont = New
Font("Arial", 12, FontStyle.Bold)
        ' Set Title color
        ResourceLoadingChart3.ChartAreas.Item(0).AxisY.TitleForeColor =
Color.Gray

        ResourceLoadingChart3.Series.RemoveAt(0)
        ResourceLoadingChart3.Series.Add("MPiping")
        ResourceLoadingChart3.Series("MPiping").ChartType =
SeriesChartType.Column

        'Resource Loading Chart for Insulation
        ResourceLoadingChart4.ChartAreas.Item(0).AxisX.Interval = 1
        ResourceLoadingChart4.ChartAreas.Item(0).AxisX.IntervalType =
DateTimeIntervalType.Days
        ResourceLoadingChart4.ChartAreas.Item(0).AxisX.LabelAutoFitStyle =
LabelAutoFitStyles.LabelsAngleStep90
        ResourceLoadingChart4.Titles.Add("Insulation Manpower Loading Curve")
        ' Set chart title font
        ResourceLoadingChart4.Titles(0).Font = New Font("Arial", 12,
FontStyle.Bold)
        ResourceLoadingChart4.Titles(0).ForeColor = Color.White
```

```
        ' Set chart title color
        ResourceLoadingChart4.Titles(0).BackColor = Color.Red
        ' Set  axis title
        ResourceLoadingChart4.ChartAreas.Item(0).AxisY.Title = "Manpower"
        ' Set Title font
        ResourceLoadingChart4.ChartAreas.Item(0).AxisY.TitleFont = New
Font("Arial", 12, FontStyle.Bold)
        ' Set Title color
        ResourceLoadingChart4.ChartAreas.Item(0).AxisY.TitleForeColor =
Color.Gray

        ResourceLoadingChart4.Series.RemoveAt(0)
        ResourceLoadingChart4.Series.Add("MInsulation")
        ResourceLoadingChart4.Series("MInsulation").ChartType =
SeriesChartType.Column

        'Resource Loading Chart for Iron worker
        ResourceLoadingChart5.ChartAreas.Item(0).AxisX.Interval = 1
        ResourceLoadingChart5.ChartAreas.Item(0).AxisX.IntervalType =
DateTimeIntervalType.Days
        ResourceLoadingChart5.ChartAreas.Item(0).AxisX.LabelAutoFitStyle =
LabelAutoFitStyles.LabelsAngleStep90
        ResourceLoadingChart5.Titles.Add("Iron Worker Manpower Loading Curve")
        ' Set chart title font
        ResourceLoadingChart5.Titles(0).Font = New Font("Arial", 12,
FontStyle.Bold)
        ResourceLoadingChart5.Titles(0).ForeColor = Color.White
        ' Set chart title color
        ResourceLoadingChart5.Titles(0).BackColor = Color.Red
        ' Set  axis title
        ResourceLoadingChart5.ChartAreas.Item(0).AxisY.Title = "Manpower"
        ' Set Title font
        ResourceLoadingChart5.ChartAreas.Item(0).AxisY.TitleFont = New
Font("Arial", 12, FontStyle.Bold)
        ' Set Title color
        ResourceLoadingChart5.ChartAreas.Item(0).AxisY.TitleForeColor =
Color.Gray

        ResourceLoadingChart5.Series.RemoveAt(0)
        ResourceLoadingChart5.Series.Add("MIronWorker")
        ResourceLoadingChart5.Series("MIronWorker").ChartType =
SeriesChartType.Column

        ''For 011AB
        'ResourceLoadingChart6.ChartAreas.Item(0).AxisX.Interval = 1
        'ResourceLoadingChart6.ChartAreas.Item(0).AxisX.IntervalType =
DateTimeIntervalType.Days
        'ResourceLoadingChart6.ChartAreas.Item(0).AxisX.LabelAutoFitStyle =
LabelAutoFitStyles.LabelsAngleStep90
        'ResourceLoadingChart6.Titles.Add("011AB Work Area")
        '' Set chart title font
        'ResourceLoadingChart6.Titles(0).Font = New Font("Arial", 12,
FontStyle.Bold)
        'ResourceLoadingChart6.Titles(0).ForeColor = Color.White
        '' Set chart title color
        'ResourceLoadingChart6.Titles(0).BackColor = Color.Red
        '' Set  axis title
```

```
        'ResourceLoadingChart6.ChartAreas.Item(0).AxisY.Title = "Number Of
Craft Persons"
        '' Set Title font
        'ResourceLoadingChart6.ChartAreas.Item(0).AxisY.TitleFont = New
Font("Arial", 12, FontStyle.Bold)
        '' Set Title color
        'ResourceLoadingChart6.ChartAreas.Item(0).AxisY.TitleForeColor =
Color.Gray

        'ResourceLoadingChart6.Series.RemoveAt(0)
        'ResourceLoadingChart6.Series.Add("011AB")
        'ResourceLoadingChart6.Series("011AB").ChartType =
SeriesChartType.Column

        ''For 012ABC
        'ResourceLoadingChart7.ChartAreas.Item(0).AxisX.Interval = 1
        'ResourceLoadingChart7.ChartAreas.Item(0).AxisX.IntervalType =
DateTimeIntervalType.Days
        'ResourceLoadingChart7.ChartAreas.Item(0).AxisX.LabelAutoFitStyle =
LabelAutoFitStyles.LabelsAngleStep90
        'ResourceLoadingChart7.Titles.Add("012ABC Work Area")
        '' Set chart title font
        'ResourceLoadingChart7.Titles(0).Font = New Font("Arial", 12,
FontStyle.Bold)
        'ResourceLoadingChart7.Titles(0).ForeColor = Color.White
        '' Set chart title color
        'ResourceLoadingChart7.Titles(0).BackColor = Color.Red
        '' Set  axis title
        'ResourceLoadingChart7.ChartAreas.Item(0).AxisY.Title = "Number Of
Craft Persons"
        '' Set Title font
        'ResourceLoadingChart7.ChartAreas.Item(0).AxisY.TitleFont = New
Font("Arial", 12, FontStyle.Bold)
        '' Set Title color
        'ResourceLoadingChart7.ChartAreas.Item(0).AxisY.TitleForeColor =
Color.Gray

        'ResourceLoadingChart7.Series.RemoveAt(0)
        'ResourceLoadingChart7.Series.Add("012ABC")
        'ResourceLoadingChart7.Series("012ABC").ChartType =
SeriesChartType.Column

        ''For 005AB
        'ResourceLoadingChart8.ChartAreas.Item(0).AxisX.Interval = 1
        'ResourceLoadingChart8.ChartAreas.Item(0).AxisX.IntervalType =
DateTimeIntervalType.Days
        'ResourceLoadingChart8.ChartAreas.Item(0).AxisX.LabelAutoFitStyle =
LabelAutoFitStyles.LabelsAngleStep90
        'ResourceLoadingChart8.Titles.Add("005AB")
        '' Set chart title font
        'ResourceLoadingChart8.Titles(0).Font = New Font("Arial", 12,
FontStyle.Bold)
        'ResourceLoadingChart8.Titles(0).ForeColor = Color.White
        '' Set chart title color
        'ResourceLoadingChart8.Titles(0).BackColor = Color.Red
        '' Set  axis title
```

```
        'ResourceLoadingChart8.ChartAreas.Item(0).AxisY.Title = "Number Of
Craft Persons"
        '' Set Title font
        'ResourceLoadingChart8.ChartAreas.Item(0).AxisY.TitleFont = New
Font("Arial", 12, FontStyle.Bold)
        '' Set Title color
        'ResourceLoadingChart8.ChartAreas.Item(0).AxisY.TitleForeColor =
Color.Gray

        'ResourceLoadingChart8.Series.RemoveAt(0)
        'ResourceLoadingChart8.Series.Add("005AB")
        'ResourceLoadingChart8.Series("005AB").ChartType =
SeriesChartType.Column

        ''For 006AB
        'ResourceLoadingChart9.ChartAreas.Item(0).AxisX.Interval = 1
        'ResourceLoadingChart9.ChartAreas.Item(0).AxisX.IntervalType =
DateTimeIntervalType.Days
        'ResourceLoadingChart9.ChartAreas.Item(0).AxisX.LabelAutoFitStyle =
LabelAutoFitStyles.LabelsAngleStep90
        'ResourceLoadingChart9.Titles.Add("006AB")
        '' Set chart title font
        'ResourceLoadingChart9.Titles(0).Font = New Font("Arial", 12,
FontStyle.Bold)
        'ResourceLoadingChart9.Titles(0).ForeColor = Color.White
        '' Set chart title color
        'ResourceLoadingChart9.Titles(0).BackColor = Color.Red
        '' Set  axis title
        'ResourceLoadingChart9.ChartAreas.Item(0).AxisY.Title = "Number Of
Craft Persons"
        '' Set Title font
        'ResourceLoadingChart9.ChartAreas.Item(0).AxisY.TitleFont = New
Font("Arial", 12, FontStyle.Bold)
        '' Set Title color
        'ResourceLoadingChart9.ChartAreas.Item(0).AxisY.TitleForeColor =
Color.Gray

        'ResourceLoadingChart9.Series.RemoveAt(0)
        'ResourceLoadingChart9.Series.Add("006AB")
        'ResourceLoadingChart9.Series("006AB").ChartType =
SeriesChartType.Column

        ''Resource Loading Chart for Cable Tray
        'ResourceLoadingChart10.ChartAreas.Item(0).AxisX.Interval = 1
        'ResourceLoadingChart10.ChartAreas.Item(0).AxisX.IntervalType =
DateTimeIntervalType.Days
        'ResourceLoadingChart10.ChartAreas.Item(0).AxisX.LabelAutoFitStyle =
LabelAutoFitStyles.LabelsAngleStep90
        'ResourceLoadingChart10.Titles.Add("Building Cladding Manpower
Loading Curve")
        '' Set chart title font
        'ResourceLoadingChart10.Titles(0).Font = New Font("Arial", 12,
FontStyle.Bold)
        'ResourceLoadingChart10.Titles(0).ForeColor = Color.White
        '' Set chart title color
        'ResourceLoadingChart10.Titles(0).BackColor = Color.Red
        '' Set  axis title
```

```
        'ResourceLoadingChart10.ChartAreas.Item(0).AxisY.Title = "Manpower"
        '' Set Title font
        'ResourceLoadingChart10.ChartAreas.Item(0).AxisY.TitleFont = New
Font("Arial", 12, FontStyle.Bold)
        '' Set Title color
        'ResourceLoadingChart10.ChartAreas.Item(0).AxisY.TitleForeColor =
Color.Gray

        'ResourceLoadingChart10.Series.RemoveAt(0)
        'ResourceLoadingChart10.Series.Add("MClad")
        'ResourceLoadingChart10.Series("MClad").ChartType =
SeriesChartType.Column

        'Resource Loading Chart for Module Prep/Fina Inspection

    End Sub

    Private Sub DisplayChart()
        'For Piling
        Dim MPilingXArray As New List(Of Date)
        Dim MPilingYArray As New List(Of Integer)

        'For Electrical
        Dim MELectricalArray As New List(Of Date)
        Dim MELectricalYArray As New List(Of Integer)

        'For Piping
        Dim MPipingXArray As New List(Of Date)
        Dim MPipingYArray As New List(Of Integer)

        'For Insulation
        Dim MInsulationXArray As New List(Of Date)
        Dim MInsulationYArray As New List(Of Integer)

        'For Iron Worker
        Dim MIronWorkerXArray As New List(Of Date)
        Dim MIronWorkerYArray As New List(Of Integer)

        ''For HVAC
        'Dim HVACXArray As New List(Of Date)
        'Dim HVACYArray As New List(Of Integer)

        ''For Instrumentation
        'Dim InstrXArray As New List(Of Date)
        'Dim InstrYArray As New List(Of Integer)

        ''For Insulation
        'Dim InsulXArray As New List(Of Date)
        'Dim InsulYArray As New List(Of Integer)

        ''For Heat Tracing
        'Dim TracXArray As New List(Of Date)
        'Dim TracYArray As New List(Of Integer)

        ''For Cable Tray
        'Dim CladXArray As New List(Of Date)
        'Dim CladYArray As New List(Of Integer)
```

```
        For Each ResUsageRow As DataRow In
MyDataSet.Tables("tblResourceUsagePerDay").Rows
            Dim ResName As String = CStr(ResUsageRow.Item("ResourceName"))
            Select Case ResName
                Case "PIL"
                    'For Steel Structure
                    MPilingXArray.Add(CDate(ResUsageRow.Item("Date")).Date)
                    MPilingYArray.Add(CInt(ResUsageRow.Item("Usage")))
                Case "EL"
                    MELectricalArray.Add(CDate(ResUsageRow.Item("Date")).Date)
                    MELectricalYArray.Add(CInt(ResUsageRow.Item("Usage")))
                Case "PF"
                    MPipingXArray.Add(CDate(ResUsageRow.Item("Date")).Date)
                    MPipingYArray.Add(CInt(ResUsageRow.Item("Usage")))
                Case "INS"

MInsulationXArray.Add(CDate(ResUsageRow.Item("Date")).Date)
                    MInsulationYArray.Add(CInt(ResUsageRow.Item("Usage")))
                Case "IW"

MIronWorkerXArray.Add(CDate(ResUsageRow.Item("Date")).Date)
                    MIronWorkerYArray.Add(CInt(ResUsageRow.Item("Usage")))
                    'Case "MHVAC"
                    '    HVACXArray.Add(CDate(ResUsageRow.Item("Date")).Date)
                    '    HVACYArray.Add(CInt(ResUsageRow.Item("Usage")))
                    'Case "MInstr"
                    '     InstrXArray.Add(CDate(ResUsageRow.Item("Date")).Date)
                    '    InstrYArray.Add(CInt(ResUsageRow.Item("Usage")))
                    'Case "MInsul"
                    '     InsulXArray.Add(CDate(ResUsageRow.Item("Date")).Date)
                    '    InsulYArray.Add(CInt(ResUsageRow.Item("Usage")))
                    'Case "MTrac"
                    '    TracXArray.Add(CDate(ResUsageRow.Item("Date")).Date)
                    '    TracYArray.Add(CInt(ResUsageRow.Item("Usage")))
                    'Case "MClad"
                    '    CladXArray.Add(CDate(ResUsageRow.Item("Date")).Date)
                    '    CladYArray.Add(CInt(ResUsageRow.Item("Usage")))
            End Select

        Next

        'Dim XArray(NumOfDays) As DateTime
        'Dim YArray(NumOfDays) As Integer
        'Dim i As Integer
        'For Each ResUsageRow As DataRow In
MyDataSet.Tables("tblResourceUsagePerDay").Rows
        '    If ResUsageRow.Item("ResourceName") = "MPipFab" Then
        '        XArray(i) = CDate(ResUsageRow.Item("Date")).Date
        '        YArray(i) = CInt(ResUsageRow.Item("Usage"))
        '        'MessageBox.Show(XArray(i))
        '    End If
        'Next

ResourceLoadingChart1.Series("MPiling").Points.DataBindXY(MPilingXArray,
MPilingYArray)
```

```
ResourceLoadingChart2.Series("MELectrical").Points.DataBindXY(MELectricalArra
y, MELectricalYArray)

ResourceLoadingChart3.Series("MPiping").Points.DataBindXY(MPipingXArray,
MPilingYArray)

ResourceLoadingChart4.Series("MInsulation").Points.DataBindXY(MInsulationXArr
ay, MInsulationYArray)

ResourceLoadingChart5.Series("MIronWorker").Points.DataBindXY(MIronWorkerXArr
ay, MIronWorkerYArray)
        'ResourceLoadingChart6.Series("MHVAC").Points.DataBindXY(HVACXArray,
HVACYArray)
        'ResourceLoadingChart7.Series("MInstr").Points.DataBindXY(InstrXArray,
InstrYArray)
        'ResourceLoadingChart8.Series("MInsul").Points.DataBindXY(InsulXArray,
InsulYArray)
        'ResourceLoadingChart9.Series("MTrac").Points.DataBindXY(TracXArray,
TracYArray)
        'ResourceLoadingChart10.Series("MClad").Points.DataBindXY(CladXArray,
CladYArray)

    End Sub


    Public Sub WriteResultMethod()
        Dim cn As OleDbConnection
        Dim MyDataAdapter As OleDbDataAdapter
        Dim MyCommandBuilder As OleDbCommandBuilder
        Dim DBAddress As String = Me.DBAddressTxt.Text

        'Get data from StiteInstallationDBModifiedJuly132011
        cn = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=I:\SiteInstallationDBPopulatedSeptember262012.accdb")

        Try
            cn.Open()
            MyDataAdapter = New OleDbDataAdapter("Select * From
tblWorkAreaWorkpackage;", cn)
            MyCommandBuilder = New OleDbCommandBuilder(MyDataAdapter)
            MyDataSet.Tables("tblWAWP").Columns.Remove("WADesignator")
            MyDataSet.Tables("tblWAWP").Columns.Remove("WAWPTotalFloat")
            MyDataSet.Tables("tblWAWP").Columns.Remove("WAWPEstDuration")
            'DataGridView1.DataSource = MyDataSet.Tables("tblWAWP")
            MyDataAdapter.Update(MyDataSet.Tables("tblWAWP"))

            MyDataAdapter = New OleDbDataAdapter("Select * from
tblWorkpackage;", cn)
            MyCommandBuilder = New OleDbCommandBuilder(MyDataAdapter)

MyDataSet.Tables("tblWorkpackages").Columns.Remove("ClassificationDescription
")
            MyDataAdapter.Update(MyDataSet.Tables("tblWorkpackages"))

        Catch ex As Exception
            Throw New ArgumentException(ex.Message)
```

```
        Finally
            cn.Close()
        End Try
    End Sub

End Class
```

VB.NET code for the simulation scenario

```
Imports Simphony
Imports Simphony.Simulation
Imports Simphony.Modeling
Imports System.Data
Imports System.Data.OleDb
Imports System.Linq
Imports System.Math
'Imports System.Windows.Forms.DataVisualization.Charting
Imports System.Diagnostics

Public Class Scenario
    Inherits DiscreteEventScenario
    Private TimeAdvanceStep = 1 'Each time step is 1 hour
    Private Converter As Integer = 1 '3600 'Converts a time step to a number
of simulation time units; For example, Each simulation time unit represents a
second; Each time step (= 1 hour) will be 3600 time units

    'Define all related events
    Private SimTimeTickEvent As New Action(Of Entity)(AddressOf SimTimeTick)

    'Declare all attributes
    Private ReadOnly MyEngine As DiscreteEventEngine
    Private MyDataSet As New DataSet
    Private tblProjects, tblWorkArea, tblResource, tblCalendar,
tblCalendarDetail, tblWorkpackages, tblWorkAreaWorkpackage,
tblWorkpackagePredecessor, tblWAWP, tblWAWPPredecessors, tblWAWPTotalFloat,
tblResourceAvailability, tblResourceUsagePerDay, tblResourceUsagePerHour,
tblWorkAreaUsagePerDay, tblWorkAreaUsagePerHour, tblClassification,
tblResourceCapturedPerWAWP As DataTable
    'Update Both Arrival WAWP Lists and Completed WAWP Lists
    'Private UnifiedWaitingList As New List(Of WorkAreaWorkpackageEntity)
    Private CompletedWAWPList As New List(Of WorkAreaWorkpackageEntity)
    Private CompletedWAWPIDList As New List(Of Integer)
    Private ArrivedWAWPList As New List(Of WorkAreaWorkpackageEntity)
    Private ArrivedWAWPIDList As New List(Of Integer)
    Private ResourceCapturedWAWPList As New List(Of WorkAreaWorkpackageEntity)
    Private MyWAWPs As New Dictionary(Of Integer, WorkAreaWorkpackageEntity)

    'TotalCrewResource & Waiting file
    Private TotalCrewResource As CrewResource
    Private MyTrucks As Resource
    Private ReadOnly CrewResourceList As New List(Of CrewResource)
    Private ReadOnly WorkAreaCongestionResourceList As New List(Of
CongestionResource)
    'Private ReadOnly CrewResources As New Dictionary(Of String, CrewResource)
    Private ResourceAvailQuantityPair As New Dictionary(Of String, Integer)
    Private ResourceMinPercentPair As New Dictionary(Of String, Double)
    Private ResourceMaxPercentPair As New Dictionary(Of String, Double)
```

```
    'Private ResourceWaitingLists As New Dictionary(Of String, List(Of
WorkAreaWorkpackageEntity))
    Private ProjectStartDate As Date

    'Set up local time for entity arrival event
    Private CurrentSimTime As Integer = 0
    Private NumWAWPArrived As Integer
    Private NumWAWPCompleted As Integer
    Private TotalNumOfWAWPs As Integer

    Private SuccWAWPIDs, PredWAWPIDs As New List(Of Integer)

    Public Sub New(ByVal MyEngine As DiscreteEventEngine, ByVal MyDataSet As
DataSet)
        Me.MyEngine = MyEngine
        Me.MyDataSet = MyDataSet
    End Sub

    Public Overrides Function InitializeScenario() As Integer
        'Initialize Resources
        tblResource = MyDataSet.Tables("tblResource")
        tblResourceAvailability = MyDataSet.Tables("tblResourceAvailability")
        tblClassification = MyDataSet.Tables("tblClassification")
        For Each MyResourceRow As DataRow In tblResourceAvailability.Rows
            If Not
ResourceAvailQuantityPair.ContainsKey(MyResourceRow.Item("Craft")) Then
                Dim MyCrewResourceName As String =
CStr(MyResourceRow.Item("Craft"))
                Dim MaxRes As Integer =
CInt(MyResourceRow.Item("AvailableAmount"))
                'Create CrewResources
                Dim MyCrewResource As New CrewResource(MyCrewResourceName,
MaxRes)
                MyCrewResource.TaskDesc =
CStr(tblResource.Rows.Find(MyCrewResourceName).Item("CraftDescription"))
                MyCrewResource.OriginalQuantity = MaxRes

                ''Manpower is increased at a rate that must not exceed
certain limits
                ''MyCrewResource.RampUp = CDbl(MyResourceRow.Item("Resource
ramp up")) * 50
                ''MyCrewResource.Level = MyResourceRow.Item("Level")
                ''MyCrewResource.NonManPower =
MyResourceRow.Item("NonManPower")
                ''MyCrewResource.NonYardActivity =
MyResourceRow.Item("NonYardActivity")
                ''MyCrewResource.ZeroFreeFloatActivity =
MyResourceRow.Item("ZeroFreeFloatActivity")

                Dim NormalCraftSize As Integer
                Dim MaxCraftSize As Integer
                Dim MinCraftSize As Integer
                ''Find the MinCraftSize and MaxCraftSize of each trade
                For Each ClassificationRow As DataRow In
tblClassification.rows
                    If CStr(ClassificationRow.Item("Craft")) =
MyCrewResourceName Then
```

```
                        If Not ClassificationRow.Item("MaxCraftSize") Is
DBNull.Value Then
                            NormalCraftSize =
CInt(ClassificationRow.Item("NormalCraftSize"))
                            MaxCraftSize =
CInt(ClassificationRow.Item("MaxCraftSize"))
                            MinCraftSize =
CInt(ClassificationRow.Item("MinCraftSize"))
                            Exit For
                        End If
                    End If
                Next
                MyCrewResource.MinPercent = MinCraftSize / NormalCraftSize
                MyCrewResource.MaxPercent = MaxCraftSize / NormalCraftSize

                'Initialize the ResourceAvailQuantityPair list
                ResourceAvailQuantityPair.Add(MyCrewResourceName, MaxRes)
                ResourceMinPercentPair.Add(MyCrewResourceName,
MyCrewResource.MinPercent)
                ResourceMaxPercentPair.Add(MyCrewResourceName,
MyCrewResource.MaxPercent)
                CrewResourceList.Add(MyCrewResource)
            End If

        Next

        ''Define TotalCrewResources

        'Dim MaxManPerShift As Integer =
CInt(tblProjects.Rows(0).Item("MaxManPerShiftOnsite"))
        'Dim MaxTotalCrewResQuan = MaxManPerShift * 100
        'TotalCrewResource = New CrewResource("TotalCrewResource",
MaxTotalCrewResQuan)
        'TotalCrewResource.TaskDesc = "TotalCrewResource"
        'TotalCrewResource.OriginalQuantity = MaxTotalCrewResQuan
        'ResourceAvailQuantityPair.Add("TotalCrewResource",
MaxTotalCrewResQuan)

        'Define Work Spcae Congestion Resources
        tblWorkArea = MyDataSet.Tables("tblWorkArea")
        For Each WorkAreaRow As DataRow In tblWorkArea.Rows
            Dim MyWorkAreaID As Integer =
CInt(WorkAreaRow.Item("WorkAreaNumber"))
            Dim MyWorkAreaName As String =
CStr(WorkAreaRow.Item("WADesignator"))
            'the maximum number of craft people in a work area
            Dim MyWorkAreaCongestionLimit As Integer = 100

            Dim MyWorkAreaCongestionResource As New
CongestionResource(MyWorkAreaName, MyWorkAreaCongestionLimit)
            MyWorkAreaCongestionResource.WorkAreaDesignator =
CStr(WorkAreaRow.Item("WADesignator"))
            'Belong to which construction work area
            MyWorkAreaCongestionResource.WAConstructionWorkArea =
CStr(WorkAreaRow.Item("WAConstructionWorkArea"))
            MyWorkAreaCongestionResource.OriginalCongestionLimit =
MyWorkAreaCongestionLimit
```

```
            WorkAreaCongestionResourceList.Add(MyWorkAreaCongestionResource)

            'Initialize the ResourceAvailQuantityPair list
            ResourceAvailQuantityPair.Add(MyWorkAreaName,
MyWorkAreaCongestionLimit)
        Next

        'For Each WorkAreaCongestionResource As CongestionResource In
WorkAreaCongestionResourceList
        '    MessageBox.Show(WorkAreaCongestionResource.Name & " " &
WorkAreaCongestionResource.Quantity)
        'Next

        'Initialize start time
        tblCalendar = MyDataSet.Tables("tblCalendar")
        tblProjects = MyDataSet.Tables("tblProjects")
        ProjectStartDate = CDate(tblProjects.Rows(0).Item("Start"))
        Dim ProjectCalID As Integer =
CInt(tblProjects.Rows(0).Item("DefCalendarID"))
        Dim StartHour As Double =
CDate(tblCalendar.Rows.Find(ProjectCalID).Item("StartTime").ToString).Hour +
CDate(tblCalendar.Rows.Find(ProjectCalID).Item("StartTime").ToString).Minute
/ 60
        ProjectStartDate = CDate(ProjectStartDate.AddHours(StartHour))

        Return MyBase.InitializeScenario()

    End Function

    Public Overrides Function InitializeRun(ByVal runIndex As Integer) As
Double
        'Initialize all the datatables, except for three datatables that have
already been initialized in InitializeScenario Function
        tblWorkpackages = MyDataSet.Tables("tblWorkpackages")
        tblWorkAreaWorkpackage = MyDataSet.Tables("tblWAWP")
        tblWorkpackagePredecessor = MyDataSet.Tables("tblWAWPPredecessors")
        tblCalendarDetail = MyDataSet.Tables("tblCalendar_Detail")
        tblWAWP = MyDataSet.Tables("tblWAWP")
        tblWAWPPredecessors = MyDataSet.Tables("tblWAWPPredecessors")
        TotalNumOfWAWPs = MyDataSet.Tables("tblWAWP").Rows.Count

        'Trigger Arrival List Updating event
        Dim TriggerEntity As New Entity
        MyEngine.ScheduleEvent(TriggerEntity, SimTimeTickEvent, 0)

        'Find WAWPs that have successors and WAWPs that are succeeding to
some other WAWPs
        For Each Row As DataRow In tblWorkpackagePredecessor.Rows
            If Not SuccWAWPIDs.Contains(Row.Item("WAWPID")) Then
                SuccWAWPIDs.Add(Row.Item("WAWPID"))
            End If
            If Not PredWAWPIDs.Contains(Row.Item("WAWPPredID")) Then
                PredWAWPIDs.Add(Row.Item("WAWPPredID"))
            End If
        Next

        CalculateTotalfloat(tblWorkpackages, tblWAWP, tblWAWPPredecessors)
```

```
        tblWAWPTotalFloat = MyDataSet.Tables("tblWAWPTotalFloat")

        tblResourceUsagePerDay =
MyDataSet.Tables.Add("tblResourceUsagePerDay")
        tblResourceUsagePerDay.Columns.Add("ResourceName",
Type.GetType("System.String"))
        tblResourceUsagePerDay.Columns.Add("Date",
Type.GetType("System.DateTime"))
        tblResourceUsagePerDay.Columns.Add("Usage",
Type.GetType("System.Double"))

        tblResourceUsagePerHour =
MyDataSet.Tables.Add("tblResourceUsagePerHour")
        tblResourceUsagePerHour.Columns.Add("ResourceName",
Type.GetType("System.String"))
        tblResourceUsagePerHour.Columns.Add("Date",
Type.GetType("System.DateTime"))
        tblResourceUsagePerHour.Columns.Add("Usage",
Type.GetType("System.Int32"))

        tblWorkAreaUsagePerDay =
MyDataSet.Tables.Add("tblWorkAreaUsagePerDay")
        tblWorkAreaUsagePerDay.Columns.Add("WorkAreaName",
Type.GetType("System.String"))
        tblWorkAreaUsagePerDay.Columns.Add("Date",
Type.GetType("System.DateTime"))
        tblWorkAreaUsagePerDay.Columns.Add("Usage",
Type.GetType("System.Double"))

        tblWorkAreaUsagePerHour =
MyDataSet.Tables.Add("tblWorkAreaUsagePerHour")
        tblWorkAreaUsagePerHour.Columns.Add("WorkAreaName",
Type.GetType("System.String"))
        tblWorkAreaUsagePerHour.Columns.Add("Date",
Type.GetType("System.DateTime"))
        tblWorkAreaUsagePerHour.Columns.Add("Usage",
Type.GetType("System.Int32"))

        tblResourceCapturedPerWAWP =
MyDataSet.Tables.Add("tblResourceCapturedPerWAWP")
        tblResourceCapturedPerWAWP.Columns.Add("WAWPID",
Type.GetType("System.Int32"))
        tblResourceCapturedPerWAWP.Columns.Add("ResName",
Type.GetType("System.String"))
        tblResourceCapturedPerWAWP.Columns.Add("ResQtyCaptured",
Type.GetType("System.Int32"))
        tblResourceCapturedPerWAWP.Columns.Add("Date",
Type.GetType("System.DateTime"))
        Return 1000000

    End Function

    Public Sub SimTimeTick(ByVal TriggerEntity As Entity)
        'The first step is to update the completion status of every WAWP
entity and Release resource captured in the last time step
        CurrentSimTime = MyEngine.TimeNow
```

```
        Dim CurrentDateTime As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
        'If DateTime.Compare(#12/7/2012 3:00:00 PM#, CurrentDateTime) <= 0
Then
        '    MessageBox.Show("Debugging")
        'End If

        'At the beginning of the simulation, it is not necessary at all to
update the completion status of WAWP entities
        'Resource releasing is included in WAWPStatusUpdate subroutine
        If CurrentSimTime <> 0 Then
            WAWPStatusUpdate()
        End If

        'Record Daily resource usage
        If CurrentDateTime.Hour = 17 Then
            For Each MyResource As CrewResource In CrewResourceList
                Dim ResName As String = MyResource.Name
                'If ResName = "EL" And CurrentDateTime = #10/22/2012 5:00:00
PM# Then
                '    MessageBox.Show("OK!")
                'End If
                Dim TotalUsage As Integer = 0
                For Each ResUsageRow As DataRow In
tblResourceUsagePerHour.Select("ResourceName = '" & ResName & "'")
                    If CDate(ResUsageRow.Item("Date")).Date =
CurrentDateTime.Date Then
                        TotalUsage = TotalUsage +
CInt(ResUsageRow.Item("Usage"))
                    End If
                Next

                'Dim abc = From r In tblResourceUsagePerHour
                '          Where (CDate(r.Item("Date")).Date =
CurrentDateTime.Date) And (r.Item("ResourceName") = ResName)
                '          Select r.Item("Usage")

                'Dim bcd = abc.ToArray().Sum(Function(x) x)

                'Assume 10 working hours per day
                Dim AverageUsagePerDay As Double = TotalUsage / 10
                'If ResName = "MSteel" And AverageUsagePerDay > 5000 Then
                '    MessageBox.Show("OK!")
                'End If
                Dim ResUsagePerdayRow As DataRow =
tblResourceUsagePerDay.NewRow
                ResUsagePerdayRow.Item("ResourceName") = ResName
                ResUsagePerdayRow.Item("Date") = CurrentDateTime
                ResUsagePerdayRow.Item("Usage") = AverageUsagePerDay
                tblResourceUsagePerDay.Rows.Add(ResUsagePerdayRow)
            Next

            For Each MyWorkArea As CongestionResource In
WorkAreaCongestionResourceList
                Dim WorkAreaName As String = MyWorkArea.Name
                Dim TotalUsage As Integer = 0
```

```
                For Each WorkAreaUsageRow As DataRow In
tblWorkAreaUsagePerHour.Select("WorkAreaName = '" & WorkAreaName & "'")
                    If CDate(WorkAreaUsageRow.Item("Date")).Date =
CurrentDateTime.Date Then
                        TotalUsage = TotalUsage +
CInt(WorkAreaUsageRow.Item("Usage"))
                    End If
                Next
                'Assume 10 working hours per day
                Dim AverageUsagePerDay As Double = TotalUsage / 10
                Dim WorkAreaUsagePerDayRow As DataRow =
tblWorkAreaUsagePerDay.NewRow
                WorkAreaUsagePerDayRow.Item("WorkAreaName") = WorkAreaName
                WorkAreaUsagePerDayRow.Item("Date") = CurrentDateTime
                WorkAreaUsagePerDayRow.Item("Usage") = AverageUsagePerDay
                tblWorkAreaUsagePerDay.Rows.Add(WorkAreaUsagePerDayRow)
            Next

        End If

        'Check if Current Simulation Time is within working time range
        'If NOT, step 2--Arrival List update and step 3--Resource Allocate
will be skipped
        Dim WhetherExecuteRoutine As Boolean = False
        If ArrivedWAWPList.Count <> 0 Then
            Dim WhetherAWorkingDay As Boolean = True
            Dim WhetherAWorkingHour As Boolean = True
            For i As Integer = 0 To ArrivedWAWPList.Count - 1
                Dim CalID As Integer = ArrivedWAWPList.Item(i).CalendarID
                CheckWorkingDay_WorkingHour(CalID, CurrentDateTime,
WhetherAWorkingDay, WhetherAWorkingHour)
                'After 17:00 pm, it is not working time any more. therefore,
should stop updating the arrival list or allocating resource
                If CurrentDateTime.Hour = 17 Then
                    WhetherAWorkingHour = False
                End If
                'If it is a working day and within working hours, continue
                If WhetherAWorkingDay And WhetherAWorkingHour Then
                    WhetherExecuteRoutine = True
                    Exit For
                End If
            Next
        Else
            'When Arrival List is empty, check the overall project working
days and working hours
            'If it is true then still carry out 'WAWPArrivalListUpdate' and
'ResourceAllocate'
            Dim WhetherAWorkingDay As Boolean = True
            Dim WhetherAWorkingHour As Boolean = True
            Dim CalID As Integer =
CInt(tblProjects.Rows(0).Item("DefCalendarID"))
            CheckWorkingDay_WorkingHour(CalID, CurrentDateTime,
WhetherAWorkingDay, WhetherAWorkingHour)
            If WhetherAWorkingDay And WhetherAWorkingHour Then
                WhetherExecuteRoutine = True
            End If
        End If
```

```
        If CurrentSimTime = 0 Then
            WhetherExecuteRoutine = True
        End If

        If Not WhetherExecuteRoutine Then
            Dim TriggerEntity1 As New Entity
            MyEngine.ScheduleEvent(TriggerEntity1, SimTimeTickEvent, 1 *
Converter)
            Exit Sub
        End If

        'The second step is to update the arrival list of WAWP entities
        WAWPArrivalListUpdate()

        'The third step is to Allocate resource to WAWP entities
        ResourceAllocate()

        'The fourth step is to forward to the next time tick
        If NumWAWPCompleted < TotalNumOfWAWPs And CurrentSimTime < 1000000
Then
            Dim TriggerEntity1 As New Entity
            MyEngine.ScheduleEvent(TriggerEntity1, SimTimeTickEvent, 1 *
Converter)
        End If
        'Debug.WriteLine("Current Time is " & CurrentDateTime.ToString)
        'Debug.WriteLine(NumWAWPCompleted & " WAWPs have been completed!")
    End Sub

    Public Sub WAWPStatusUpdate()
        Dim CurrentDateTime As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
        '**************************Debug
point**********************************
        'If CurrentDateTime.ToString = "7/26/2011 5:00:00 PM" Then
        '    MessageBox.Show("OK!")
        'End If
        '**************************Debug
point**********************************

        For i As Integer = 0 To ResourceCapturedWAWPList.Count - 1
            Dim CurrentWAWPEntity As WorkAreaWorkpackageEntity =
ResourceCapturedWAWPList.Item(i)

            'check if it is in a working day and if it is withinworking hours
            'We check it for every entity, coz they might use different
calendar
            Dim CalendarID As Integer = CurrentWAWPEntity.CalendarID

            Dim WhetherAWorkingDay As Boolean = True
            Dim WhetherAWorkingHour As Boolean = True
            CheckWorkingDay_WorkingHour(CalendarID, CurrentDateTime,
WhetherAWorkingDay, WhetherAWorkingHour)
            'For progress updating, 7 am is not considered as working hour.
it just marks the start of working time of a working day. 8 am is considered
the first time to get progress
            If CurrentDateTime.Hour = 7 Then
```

```
                WhetherAWorkingHour = False
            End If

            'If it is a working day and it is within working hours, then
update the progress
            If WhetherAWorkingDay And WhetherAWorkingHour Then

                'Update the WAWP completion status
                'Get the Resouce Amount it has captured in the previous time
step
                Dim WAWPID As Integer = CurrentWAWPEntity.WAWPID

                Dim ReqResName As String = CurrentWAWPEntity.ReqRes
                Dim ReqResAmountCaptured As Integer =
CurrentWAWPEntity.CurrentReqResNameQtyPair(ReqResName)

                'Record how much resource it has captured
                Dim NewDataRow As DataRow =
tblResourceCapturedPerWAWP.NewRow()
                NewDataRow.Item("WAWPID") = WAWPID
                NewDataRow.Item("ResName") = ReqResName
                NewDataRow.Item("ResQtyCaptured") = ReqResAmountCaptured
                NewDataRow.Item("Date") = CurrentDateTime
                tblResourceCapturedPerWAWP.Rows.Add(NewDataRow)

                'Calculate the manhours just achieved in the previous time
step
                CurrentWAWPEntity.ManHoursCompleted =
CurrentWAWPEntity.ManHoursCompleted + 1 * ReqResAmountCaptured
                '**************************Debug
point**********************************
                'If WAWPID = 212002 Then
                '    MessageBox.Show(WAWPID & " Manhours have been completed
is " & CurrentWAWPEntity.ManHoursCompleted)
                'End If
                '**************************Debug
point**********************************
                'Check If it is completed
                Dim OriginalDuration As Double =
CalculateDurationFromDaysToHours(CalendarID,
CurrentWAWPEntity.DurationOverride)
                Dim NormReqResAmount As Integer =
CurrentWAWPEntity.ReqResAmount
                Dim TotalManHoursToBeAchieved As Double = NormReqResAmount *
OriginalDuration

                If CurrentWAWPEntity.ManHoursCompleted >=
TotalManHoursToBeAchieved Then

                    'Record the finish date
                    CurrentWAWPEntity.SimFinish =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)

                    'Update the WAWP table
                    If tblWAWP.Rows.Contains(CurrentWAWPEntity.WAWPID) Then
```

```
tblWAWP.Rows.Find(CurrentWAWPEntity.WAWPID).Item("WAWPSimulationStart") =
CurrentWAWPEntity.SimStart

tblWAWP.Rows.Find(CurrentWAWPEntity.WAWPID).Item("WAWPSimulationFinish") =
CurrentWAWPEntity.SimFinish
                    End If
                    'Update the corresponding WP table
                    Dim WPID As Integer = CurrentWAWPEntity.WPID
                    If tblWorkpackages.Rows.Contains(WPID) Then
                        If Not
tblWorkpackages.Rows.Find(WPID).Item("WPSimStart") Is DBNull.Value Then
                            If
DateTime.Compare(CDate(tblWorkpackages.Rows.Find(WPID).Item("WPSimStart")),
CurrentWAWPEntity.SimStart) > 0 Then

tblWorkpackages.Rows.Find(WPID).Item("WPSimStart") =
CurrentWAWPEntity.SimStart
                            End If
                        Else
                            tblWorkpackages.Rows.Find(WPID).Item("WPSimStart")
= CurrentWAWPEntity.SimStart
                        End If
                        If Not
tblWorkpackages.Rows.Find(WPID).Item("WPSimFinish") Is DBNull.Value Then
                            If
DateTime.Compare(CDate(tblWorkpackages.Rows.Find(WPID).Item("WPSimFinish")),
CurrentWAWPEntity.SimFinish) < 0 Then

tblWorkpackages.Rows.Find(WPID).Item("WPSimFinish") =
CurrentWAWPEntity.SimFinish
                            End If
                        Else

tblWorkpackages.Rows.Find(WPID).Item("WPSimFinish") =
CurrentWAWPEntity.SimFinish
                        End If

                    End If

                    'Remove from Arrrival List
                    ArrivedWAWPList.Remove(CurrentWAWPEntity)
                    ArrivedWAWPIDList.Remove(CurrentWAWPEntity.WAWPID)
                    'UnifiedWaitingList.Remove(CurrentWAWPEntity)
                    ''Remove from resource waiting lists too!!!

'ResourceWaitingLists(ReqResName).Remove(CurrentWAWPEntity)

                    'Add to completed list
                    If Not CompletedWAWPList.Contains(CurrentWAWPEntity) Then
                        CompletedWAWPList.Add(CurrentWAWPEntity)
                        NumWAWPCompleted = NumWAWPCompleted + 1
                    End If
                    If Not
CompletedWAWPIDList.Contains(CurrentWAWPEntity.WAWPID) Then
                        CompletedWAWPIDList.Add(CurrentWAWPEntity.WAWPID)
                    End If
```

```
                End If
            End If
        Next

        'Record Resource Usage (including workingdays and nonworkingdays)

        For Each MyResource As CrewResource In CrewResourceList
            Dim ResName As String = MyResource.Name
            Dim AvailQty As Integer = ResourceAvailQuantityPair(ResName)
            If CurrentDateTime.Hour = 7 Then
                AvailQty = FindCurrentResourceLimit(ResName, CurrentDateTime)
            End If
            Dim Usage As Integer = FindCurrentResourceLimit(ResName,
CurrentDateTime) - AvailQty

            Dim ResUsageDataRow As DataRow = tblResourceUsagePerHour.NewRow
            ResUsageDataRow.Item("ResourceName") = ResName
            ResUsageDataRow.Item("Date") = CurrentDateTime
            ResUsageDataRow.Item("Usage") = Usage
            tblResourceUsagePerHour.Rows.Add(ResUsageDataRow)

            'If ResName = "MSteel" And Usage > 5000 Then
            '    MessageBox.Show("OK!")
            'End If
        Next

        For Each MyWorkArea As CongestionResource In
WorkAreaCongestionResourceList
            Dim WorkAreaName As String = MyWorkArea.WorkAreaDesignator
            Dim AvailWorkArea As Integer =
ResourceAvailQuantityPair(WorkAreaName)
            Dim Usage As Integer = MyWorkArea.OriginalCongestionLimit -
AvailWorkArea

            Dim WorkAreaUsageDataRow As DataRow =
tblWorkAreaUsagePerHour.NewRow
            WorkAreaUsageDataRow.Item("WorkAreaName") = WorkAreaName
            WorkAreaUsageDataRow.Item("Date") = CurrentDateTime
            WorkAreaUsageDataRow.Item("Usage") = Usage
            tblWorkAreaUsagePerHour.Rows.Add(WorkAreaUsageDataRow)
        Next

        'Release Resource
        For Each MyResource As CrewResource In CrewResourceList
            Dim ResName As String = MyResource.Name
            Dim AvailQty As Integer = FindCurrentResourceLimit(ResName,
CurrentDateTime)
            ResourceAvailQuantityPair(ResName) = AvailQty
        Next

        'release work space
        For Each MyCongestionResouce As CongestionResource In
WorkAreaCongestionResourceList
            Dim WorkAreaName As String = MyCongestionResouce.Name
            Dim Limit As Integer = MyCongestionResouce.Quantity
            ResourceAvailQuantityPair(WorkAreaName) = Limit
```

```
        Next

    End Sub

    Public Function FindCurrentResourceLimit(ByVal ResName As String, ByVal
CurrentTime As Date) As Integer
        Dim Limit As Integer
        For Each LimitRow As DataRow In tblResourceAvailability.Select("Craft
= '" & ResName & "'")
            Dim StartDate As Date = CDate(LimitRow.Item("StartDate"))
            Dim EndDate As Date = CDate(LimitRow.Item("EndDate"))
            If DateTime.Compare(CurrentTime, StartDate) >= 0 And
DateTime.Compare(CurrentTime, EndDate) <= 0 Then
                Limit = CInt(LimitRow.Item("AvailableAmount"))
            End If
        Next
        Return Limit
    End Function

    Public Sub CheckWorkingDay_WorkingHour(ByVal CalendarID As Integer, ByVal
CurrentDateTime As Date, ByRef WhetherAWorkingDay As Boolean, ByRef
WhetherAWorkingHour As Boolean)
        'Check if it is a working day
        Dim CalendarDesc As String =
CStr(tblCalendar.Rows.Find(CalendarID).Item("Calendar Description"))
        If tblCalendarDetail.Rows.Contains(CurrentDateTime.ToShortDateString)
Then
            If
tblCalendarDetail.Rows.Find(CurrentDateTime.ToShortDateString).Item(CalendarD
esc) = 0 Then
                WhetherAWorkingDay = False
            Else
                WhetherAWorkingDay = True
            End If
        End If

        'Check if it is within working hours
        Dim StartTimeOfWorkingDay As Date
        Dim EndTimeOfWorkingDay As Date
        Dim HoursPerDay, StartHour As Double
        FindStartHour_HoursPerDay(CalendarID, HoursPerDay, StartHour)
        StartTimeOfWorkingDay =
CDate(CurrentDateTime.ToShortDateString).AddHours(StartHour)
        EndTimeOfWorkingDay =
CDate(CurrentDateTime.ToShortDateString).AddHours(StartHour + HoursPerDay)
        If DateTime.Compare(CurrentDateTime, StartTimeOfWorkingDay) < 0 Or
DateTime.Compare(CurrentDateTime, EndTimeOfWorkingDay) > 0 Then
            WhetherAWorkingHour = False
        Else
            WhetherAWorkingHour = True
        End If

    End Sub

    Public Sub WAWPArrivalListUpdate()
        'Update the list of arrived WAWP
```

```
        'Schedule Arrival of WAWPs (Entities) that have NO predecessors or
that all predecessors are finished already
        Dim UnfinishedFirstWAWPs, ReadyWAWPs As New List(Of Integer)

        'Find WAWPs that have NO predecessors
        For i As Integer = 0 To PredWAWPIDs.Count - 1
            'Have NO predecessors
            If Not SuccWAWPIDs.Contains(PredWAWPIDs.Item(i)) Then
                'NOT finished and NOT included in the arrival list yet
                If Not CompletedWAWPIDList.Contains(PredWAWPIDs.Item(i)) And
Not ArrivedWAWPIDList.Contains(PredWAWPIDs.Item(i)) Then
                    If Not UnfinishedFirstWAWPs.Contains(PredWAWPIDs.Item(i))
Then
                        UnfinishedFirstWAWPs.Add(PredWAWPIDs.Item(i))
                    End If
                End If
            End If
        Next

        If UnfinishedFirstWAWPs.Count <> 0 Then
            For i As Integer = 0 To UnfinishedFirstWAWPs.Count - 1
                Dim WAWPID As Integer = UnfinishedFirstWAWPs.Item(i)
                Dim WPID As Integer =
tblWAWP.Rows.Find(WAWPID).Item("WAWPWorkpackageID")
                If Not tblWorkpackages.Rows.Find(WPID).Item("WPEarlyStart")
Is DBNull.Value Or Not tblWorkpackages.Rows.Find(WPID).Item("WPEarlyfinish")
Is DBNull.Value Then
                    'Check both ES condition and EF condition, coz they don't
have any predecessor
                    Dim EarlyStartFromESConstraint As Date =
FindEarlyStartFromESConstraint(WAWPID)
                    Dim EarlyStartFromEFConstraint As Date =
FindEarlyStartFromEFConstraint(WAWPID)
                    Dim CurrentDateTime As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
                    'using '<=' instead of '=' is because ESDate might be one
of holidays (non-working days) which will be skipped by this simulation model
                    If DateTime.Compare(EarlyStartFromESConstraint,
CurrentDateTime) <= 0 And DateTime.Compare(EarlyStartFromEFConstraint,
CurrentDateTime) <= 0 Then
                        CreateWAWPEntity(WAWPID)
                    End If
                Else
                    'If no early start or early finish constraint, then
create WAWPEntity right away
                    CreateWAWPEntity(WAWPID)
                End If
            Next
        End If

        'Find WAWPs that all predecessors have been at least started
        Dim PossibleReadyWAWPList As New List(Of Integer)
        Dim NotReadyWAWPList As New List(Of Integer)
        For Each Row As DataRow In tblWAWPPredecessors.Rows
            Dim WAWPID As Integer = Row.Item("WAWPID")
            Dim WhetherOrNotReady As Boolean = True
```

```
            'Not consider those that have been completed or those that have
been proved not ready to start or those that have already been included in
arrival list
            If Not CompletedWAWPIDList.Contains(WAWPID) And Not
NotReadyWAWPList.Contains(WAWPID) And Not ArrivedWAWPIDList.Contains(WAWPID)
Then
                'all predecessors should have at least been started
                'And for predecessors that have FF or FS relationship, they
should have estimated finish date
                For Each Row1 As DataRow In
tblWAWPPredecessors.Select("WAWPID = '" & WAWPID & "'")
                    'Every predecessor should have a start date
                    If Row1.Item("SimPredStart") = "" Then
                        WhetherOrNotReady = False
                        Exit For
                    End If
                    'For 'FS' predecessors, they should have estimated finish
date
                    If Not Row1.Item("WAWPRel1") Is DBNull.Value Then
                        If CStr(Row1.Item("WAWPRel1")) = "FS" Then
                            If Row1.Item("EstimatedPredFinish") = "" Then
                                WhetherOrNotReady = False
                                Exit For
                            End If
                        End If
                    End If
                    'for 'FF' predecessors, they should have estimated finish
date
                    If Not Row1.Item("WAWPRel2") Is DBNull.Value Then
                        If CStr(Row1.Item("WAWPRel2")) = "FF" Then
                            If Row1.Item("EstimatedPredFinish") = "" Then
                                WhetherOrNotReady = False
                                Exit For
                            End If
                        End If
                    End If
                Next

                If WhetherOrNotReady Then
                    If Not PossibleReadyWAWPList.Contains(WAWPID) Then
                        PossibleReadyWAWPList.Add(WAWPID)
                    End If
                Else
                    NotReadyWAWPList.Add(WAWPID)
                End If
            End If
        Next

        'Find Earliest possible start date from precedence relationships and
from Early start constraint
        'This check is performed in every time step. it is beneficial to the
cases when predecessors' durations are variable.
        For i As Integer = 0 To PossibleReadyWAWPList.Count - 1

            Dim WAWPID As Integer = PossibleReadyWAWPList.Item(i)

            'Find corresponding WP
```

```
            Dim WPID As Integer
            If tblWAWP.Rows.Contains(WAWPID) Then
                WPID = tblWAWP.Rows.Find(WAWPID).Item("WAWPWorkpackageID")
            End If
            'Find its own duration and calendar ID
            Dim WAWPDuration As Double
            Dim WAWPCalendarID As Integer
            If tblWorkpackages.Rows.Contains(WPID) Then
                WAWPCalendarID =
CInt(tblWorkpackages.Rows.Find(WPID).Item("WPCalendarID"))
            End If
            If tblWAWP.Rows.Contains(WAWPID) Then
                WAWPDuration =
CDbl(tblWAWP.Rows.Find(WAWPID).Item("WAWPEstDuration"))
            End If
            WAWPDuration = CalculateDurationFromDaysToHours(WAWPCalendarID,
WAWPDuration)

            'From predecessors
            Dim EarliestPossibleStartDate As Date = #1/1/2000#
            For Each Row As DataRow In tblWAWPPredecessors.Select("WAWPID =
'" & WAWPID & "'")
                'Get Predecessor Info
                Dim WAWPPredID As Integer = CInt(Row.Item("WAWPPredID"))
                Dim WAWPPredSimStart As Date = CDate(Row.Item("SimPredStart"))
                Dim WAWPPredEstimatedFinish As Date =
System.DateTime.MinValue
                If Not Row.Item("EstimatedPredFinish") = "" Then
                    WAWPPredEstimatedFinish =
CDate(Row.Item("EstimatedPredFinish"))
                End If
                Dim Rel1 As String = ""
                Dim Rel2 As String = ""
                If Not Row.Item("WAWPRel1") Is DBNull.Value Then
                    Rel1 = CStr(Row.Item("WAWPRel1"))
                End If
                If Not Row.Item("WAWPRel2") Is DBNull.Value Then
                    Rel2 = CStr(Row.Item("WAWPRel2"))
                End If
                Dim Lag1 As Double = 0
                Dim Lag2 As Double = 0
                If Not Row.Item("WAWPLag1") Is DBNull.Value Then
                    Lag1 = CStr(Row.Item("WAWPLag1"))
                End If
                If Not Row.Item("WAWPLag2") Is DBNull.Value Then
                    Lag2 = CStr(Row.Item("WAWPLag2"))
                End If

                'Get Predecessor duration
                Dim WAWPPredDuration As Double =
MyWAWPs(WAWPPredID).DurationOverride
                Dim WAWPPredCalendarID As Integer =
MyWAWPs(WAWPPredID).CalendarID
                WAWPPredDuration =
CalculateDurationFromDaysToHours(WAWPPredCalendarID, WAWPPredDuration)
```

```
                '*************************Debug
point*********************************
                'Dim SimTimeNow1 As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
                'If WAWPID = 324003 And DateTime.Compare(SimTimeNow1,
#6/14/2011 12:00:00 PM#) >= 0 Then
                '    MessageBox.Show("the start time of 106001 is estimated
as " & EarliestPossibleStartDate & " at time " & SimTimeNow1)
                'End If
                '*************************Debug
point*********************************

                'Calculate the 'Lastest' early start date (from predecessor)
                If Not Rel1 = "" Then
                    If Rel1 = "SS" Then
                        'If the relationship is 'SS', calculate
EarlyStartDeterminedByRel1 From start time
                        Dim EarlyStartDeterminedByRel1 As Date =
CalculateSuccessorTaskStartDate(WAWPPredSimStart, Rel1, Lag1,
WAWPPredCalendarID, WAWPPredDuration, WAWPDuration)
                        If DateTime.Compare(EarliestPossibleStartDate,
EarlyStartDeterminedByRel1) < 0 Then
                            EarliestPossibleStartDate =
EarlyStartDeterminedByRel1
                        End If
                    ElseIf Rel1 = "FS" Then
                        'If the relationship is 'FS', calculate
EarlyStartDeterminedByRel1 From estimated finish time
                        Dim EarlyStartDeterminedByRel1 As Date =
CalculateSecondDate(WAWPPredEstimatedFinish, Lag1, WAWPPredCalendarID)
                        If DateTime.Compare(EarliestPossibleStartDate,
EarlyStartDeterminedByRel1) < 0 Then
                            EarliestPossibleStartDate =
EarlyStartDeterminedByRel1
                        End If
                    End If
                End If

                If Not Rel2 = "" Then
                    If Rel2 = "FF" Then
                        'If the relationship is 'FF', calculate
EarlyStartDeterminedByRel1 From estimated finish time
                        Dim EarlyStartDeterminedByRel2 As Date =
CalculateSecondDate(WAWPPredEstimatedFinish, Lag2 - WAWPDuration,
WAWPPredCalendarID)
                        If DateTime.Compare(EarliestPossibleStartDate,
EarlyStartDeterminedByRel2) < 0 Then
                            EarliestPossibleStartDate =
EarlyStartDeterminedByRel2
                        End If
                    End If
                End If
            Next

            'Convert Current Sim Time to date
            Dim SimTimeNow As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
```

```
            '*************************Debug
point*********************************
            'If WAWPID = 324003 And DateTime.Compare(SimTimeNow, #6/14/2011
12:00:00 PM#) >= 0 Then 'And DateTime.Compare(EarliestPossibleStartDate,
#5/17/2011 1:00:00 PM#) <> 0 Then
            '    MessageBox.Show("the start time of 324003 is estimated as "
& EarliestPossibleStartDate & " at time " & SimTimeNow)
            'End If
            '*************************Debug
point*********************************

            'From ES constraint
            Dim EarlyStartFromESConstraint As Date =
FindEarlyStartFromESConstraint(WAWPID)
            If DateTime.Compare(EarliestPossibleStartDate,
EarlyStartFromESConstraint) < 0 Then
                EarliestPossibleStartDate = EarlyStartFromESConstraint
            End If

            'From EF constraint
            Dim EarlyStartFromEFConstraint As Date =
FindEarlyStartFromEFConstraint(WAWPID)
            If DateTime.Compare(EarliestPossibleStartDate,
EarlyStartFromEFConstraint) < 0 Then
                EarliestPossibleStartDate = EarlyStartFromEFConstraint
            End If

            'if the earliest possible start date is at 5:00:00 pm which is
the end of that day, the system will automatically convert it to 7:00:00 am
of the next working day
            If EarliestPossibleStartDate.Hour = 17 Then
                EarliestPossibleStartDate =
FindTheNextWorkingDay(WAWPCalendarID, EarliestPossibleStartDate)
            End If


            'One of reasons for useing '<=' instead of "=", is beacause the
early start of a workpackage could be holidays, so when simulation jumps over
these holidays and if the condition is defined as ESSimTime =
MyEngine.TimeNow, then some of these workpackages might be overlooked!!!
            'Aonther reason is that because the simulation will jump over
5:00:00pm and directly to 7:00:00 am of the next working day
            If DateTime.Compare(EarliestPossibleStartDate, SimTimeNow) <= 0
Then
                ReadyWAWPs.Add(WAWPID)
            End If
        Next

        If ReadyWAWPs.Count <> 0 Then
            For i As Integer = 0 To ReadyWAWPs.Count - 1
                Dim WAWPID As Integer = ReadyWAWPs.Item(i)
                If Not ArrivedWAWPIDList.Contains(WAWPID) And Not
CompletedWAWPIDList.Contains(WAWPID) Then
                    CreateWAWPEntity(WAWPID)
                    '*************************Debug
point*********************************
                    'If WAWPID = 57 Then
```

```
                    '    Dim SimTimenow As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
                    '    MessageBox.Show("WAWPEntity " & WAWPID & " arrives
at " & SimTimenow)
                    'End If
                    '*************************Debug
point***********************************
                End If
            Next
        End If

    End Sub

    Public Function FindTheNextWorkingDay(ByVal CalendarID As Integer, ByVal
OriginalDate As Date) As Date

        'Find Calendar Description
        Dim CalendarDesc As String = ""
        If tblCalendar.Rows.Contains(CalendarID) Then
            CalendarDesc =
CStr(tblCalendar.Rows.Find(CalendarID).Item("Calendar Description"))
        End If

        'First Find the next working day
        Dim IndexOfOriginalDate As Integer
        If tblCalendarDetail.Rows.Contains(OriginalDate.ToShortDateString)
Then
            IndexOfOriginalDate =
tblCalendarDetail.Rows.IndexOf(tblCalendarDetail.Rows.Find(OriginalDate.ToSho
rtDateString))
        End If

        Dim TheNextWorkingDay As Date
        Dim Workinghours As Integer = 0
        While Workinghours = 0
            IndexOfOriginalDate = IndexOfOriginalDate + 1
            Workinghours =
CInt(tblCalendarDetail.Rows(IndexOfOriginalDate).Item(CalendarDesc))
            TheNextWorkingDay =
tblCalendarDetail.Rows(IndexOfOriginalDate).Item("Date")
        End While

        'Add Start hour to the next working day
        Dim StartHour, HoursPerDay As Double
        FindStartHour_HoursPerDay(CalendarID, HoursPerDay, StartHour)
        TheNextWorkingDay = TheNextWorkingDay.AddHours(StartHour)

        Return TheNextWorkingDay
    End Function

    Public Function CalculateDurationFromDaysToHours(ByVal CalendarID As
Integer, ByVal Duration As Double) As Double
        Dim ShiftsPerDay As Integer
        Dim WorkHoursPerShift As Integer
        If tblCalendar.Rows.Contains(CalendarID) Then
            ShiftsPerDay =
CInt(tblCalendar.Rows.Find(CalendarID).Item("Shifts per Day"))
```

```
            WorkHoursPerShift =
CInt(tblCalendar.Rows.Find(CalendarID).Item("Hours per Shift"))
        End If
        Duration = Duration * ShiftsPerDay * WorkHoursPerShift
        Return Duration
    End Function

    Public Sub CreateWAWPEntity(ByVal WAWPID As Integer)
        'Create WAWP entities
        'Add them to the arrival list
        Dim WAWP As New WorkAreaWorkpackageEntity
        WAWP.WAWPID = WAWPID
        InitializeWAWPEntity(WAWP)
        'Distinguish fully-completed, partially-completed or zero-completed
entities
        Dim OriginalDuration As Double =
CalculateDurationFromDaysToHours(WAWP.CalendarID, WAWP.DurationOverride)
        Dim NormReqResAmount As Integer = WAWP.ReqResAmount
        Dim TotalManHoursToBeAchieved As Double = NormReqResAmount *
OriginalDuration
        If WAWP.ManHoursCompleted = TotalManHoursToBeAchieved Then
            CompletedWAWPList.Add(WAWP)
            CompletedWAWPIDList.Add(WAWP.WAWPID)
            NumWAWPCompleted = NumWAWPCompleted + 1
        Else '0 =< completion percent < 100
            ArrivedWAWPList.Add(WAWP)
            ArrivedWAWPIDList.Add(WAWP.WAWPID)
            'Dim ReqResName As String = WAWP.ReqRes
            'ResourceWaitingLists(ReqResName).Add(WAWP)
            'Dim WorkAreaName As String = "WorkArea" &
WAWP.WorkAreaID.ToString
            'ResourceWaitingLists(WorkAreaName).Add(WAWP)
            'UnifiedWaitingList.Add(WAWP)
            NumWAWPArrived = NumWAWPArrived + 1
        End If
        'ActiveWAWPList.Add(WAWP)
        MyWAWPs.Add(WAWP.WAWPID, WAWP)
    End Sub

    Public Sub InitializeWAWPEntity(ByRef WAWP As WorkAreaWorkpackageEntity)
        Dim WAWPID As Integer = WAWP.WAWPID
        Dim WAWPDataRow As DataRow
        'Get WAWP info from tblWAWP
        If MyDataSet.Tables("tblWAWP").Rows.Contains(WAWPID) Then
            WAWPDataRow = MyDataSet.Tables("tblWAWP").Rows.Find(WAWPID)
        Else
            Throw New ArgumentException("WorkAreaWorkpackage does not exist!")
        End If
        WAWP.WAWPName = CStr(WAWPDataRow.Item("WAWPName"))
        WAWP.WPID = CInt(WAWPDataRow.Item("WAWPWorkpackageID"))
        WAWP.WorkAreaID = CInt(WAWPDataRow.Item("WAWPWorkAreaNumber"))
        WAWP.WorkAreaName = CStr(WAWPDataRow.Item("WADesignator"))
        WAWP.DurationOverride = CDbl(WAWPDataRow.Item("WAWPEstDuration"))
        If WAWPDataRow.Item("WAWPManHours") Is DBNull.Value Then
            WAWP.Manhours = 0
        Else
            WAWP.Manhours = CDbl(WAWPDataRow.Item("WAWPManHours"))
```

```
        End If
        If WAWPDataRow.Item("WAWPQauntity") Is DBNull.Value Then
            WAWP.KeyQty = 0
        Else
            WAWP.KeyQty = CDbl(WAWPDataRow.Item("WAWPQauntity"))
        End If

        'Get info from tblWP
        Dim WPDataRow As DataRow
        If MyDataSet.Tables("tblWorkpackages").Rows.Contains(WAWP.WPID) Then
            WPDataRow =
MyDataSet.Tables("tblWorkpackages").Rows.Find(WAWP.WPID)
        Else
            Throw New ArgumentException("Workpackage does not exist!")
        End If
        WAWP.ConstructionWorkArea =
CStr(WPDataRow.Item("WPConstructionWorkArea"))
        WAWP.ClassificationID = CInt(WPDataRow.Item("WPClassification"))
        WAWP.ClassificationDesc =
CStr(WPDataRow.Item("ClassificationDescription"))
        WAWP.CalendarID = CInt(WPDataRow.Item("WPCalendarID"))
        WAWP.Interruptible = CBool(WPDataRow.Item("WPInteruptable"))
        WAWP.WhetherOrNotCaptedRes = False
        'User input priority
        If Not WPDataRow.Item("WPPriority") Is DBNull.Value Then
            WAWP.Priority = CInt(WPDataRow.Item("WPPriority"))
        End If

        'find required resource
        If tblClassification.Rows.Contains(WAWP.ClassificationID) Then
            WAWP.ReqRes =
CStr(tblClassification.Rows.Find(WAWP.ClassificationID).Item("Craft"))
            'Assume that for each craft the normal request amount is 10
            WAWP.ReqResAmount = 10
        End If

        'Calculate hours have been completed
        If WAWPDataRow.Item("PercentComplete") Is DBNull.Value Then
            WAWP.ManHoursCompleted = 0
        Else
            WAWP.ManHoursCompleted =
CInt(WAWPDataRow.Item("PercentCompleted")) / 100 *
CalculateDurationFromDaysToHours(WAWP.CalendarID, WAWP.DurationOverride) *
WAWP.ReqResAmount
        End If

        'Note that not all workpackages would have a Early Start Constraint.
For those who do not have ES constraint, Only precedence relationship governs
its start!!!
        If Not WPDataRow.Item("WPEarlyStartOverride") Is DBNull.Value Then
            WAWP.ES = CDate(WPDataRow.Item("WPEarlyStartOverride"))
        Else
            If Not WPDataRow.Item("WPEarlyStart") Is DBNull.Value Then
                WAWP.ES = CDate(WPDataRow.Item("WPEarlyStart"))
            End If
        End If
        If Not WPDataRow.Item("WPLateFinishOverride") Is DBNull.Value Then
```

```
            WAWP.LF = CDate(WPDataRow.Item("WPLateFinishOverride"))
        Else
            If Not WPDataRow.Item("WPLateFinish") Is DBNull.Value Then
                WAWP.LF = CDate(WPDataRow.Item("WPLateFinish"))
            End If
        End If
        If Not WPDataRow.Item("WPEarlyFinish") Is DBNull.Value Then
            WAWP.EF = CDate(WPDataRow.Item("WPEarlyFinish"))
        End If

        'Priority related_1
        'Add the calculated Late Finish Date (in TotalFloatCalculation
Function)
        'This Calculated Late Finish is used to prioritize entities
        WAWP.TotalFloat =
CInt(tblWAWPTotalFloat.Rows.Find(WAWP.WAWPID).Item("TF"))
        WAWP.CalculatedLF =
CDate(tblWAWPTotalFloat.Rows.Find(WAWP.WAWPID).Item("LF"))
        WAWP.CalculatedES =
CDate(tblWAWPTotalFloat.Rows.Find(WAWP.WAWPID).Item("ES"))

        'Priority related_2
        'Calculate the number of successors that the current WAWP has
        Dim SuccessorList As New List(Of Integer)
        'FindAllSuccessorsToAWorkAreaWorkpackage subroutine does not consider
'SS' or 'SF' relationship and should be fixed!
        FindAllSuccessorsToAWorkAreaWorkpackage(WAWP.WAWPID, SuccessorList)
        WAWP.NumOfSuccessors = SuccessorList.Count

        'Initialize the amount of resource that the WAWP has captured,
usually zero!!
        WAWP.CurrentReqResNameQtyPair.Add(WAWP.ReqRes, 0)
        WAWP.CurrentReqResNameQtyPair.Add(WAWP.WorkAreaName, 0)


        'Calculate the priority for the WAWP entity
        CalculatePriority(WAWP)

        'Find Precedence info for the WAWP entity
        FindPreds(WAWP)
        FindSuccs(WAWP)

    End Sub

    Public Sub FindPreds(ByRef WAWP As WorkAreaWorkpackageEntity)
        Dim WAWPID, WAWPPredID As Integer
        Dim Rel1, Rel2 As String
        Dim Lag1, Lag2 As Double

        WAWPID = WAWP.WAWPID
        For Each Row As DataRow In
MyDataSet.Tables("tblWAWPPredecessors").Select("WAWPID = '" & WAWPID & "'")
            Rel1 = ""
            Rel2 = ""
            Lag1 = 0
            Lag2 = 0
            'Find a predecessor
```

```
            'Get the WAWPID of the predecessor
            WAWPPredID = CInt(Row.Item("WAWPPredID"))
            'Get the Rel1 info
            If Not Row.Item("WAWPRel1") Is DBNull.Value Then
                Rel1 = CStr(Row.Item("WAWPRel1"))
                Lag1 = CDbl(Row.Item("WAWPLag1"))
            End If
            'Get the Rel2 info
            If Not Row.Item("WAWPRel2") Is DBNull.Value Then
                Rel2 = CStr(Row.Item("WAWPRel2"))
                Lag2 = CDbl(Row.Item("WAWPLag2"))
            End If
            'Add the predecessor into 'WAWPPreds' attribute
            If WAWP.WAWPPreds.ContainsKey(WAWPID) Then
                'Not the first time to add predecessors
                If Not WAWP.WAWPPreds(WAWPID).Contains(WAWPPredID) Then
                    WAWP.WAWPPreds(WAWPID).Add(WAWPPredID)
                    'Add relationship type
                    WAWP.WAWPPredRel1.Add(WAWPPredID, Rel1)
                    WAWP.WAWPPredRel2.Add(WAWPPredID, Rel2)
                    'Add Lags
                    WAWP.WAWPPredLag1.Add(WAWPPredID, Lag1)
                    WAWP.WAWPPredLag2.Add(WAWPPredID, Lag2)
                Else
                    Throw New ArgumentException("Error in WAWPPredecessor!")
                End If
            Else
                'it is the first time to add predecessors
                Dim WAWPPreds As New List(Of Integer)
                WAWPPreds.Add(WAWPPredID)
                WAWP.WAWPPreds.Add(WAWPID, WAWPPreds)
                'Every Predecessor has Only ONE Rel1 or/and Rel2 at most
                WAWP.WAWPPredRel1.Add(WAWPPredID, Rel1)
                WAWP.WAWPPredRel2.Add(WAWPPredID, Rel2)
                'Add Lags
                WAWP.WAWPPredLag1.Add(WAWPPredID, Lag1)
                WAWP.WAWPPredLag2.Add(WAWPPredID, Lag2)
            End If

        Next
    End Sub

    Public Sub FindSuccs(ByRef WAWP As WorkAreaWorkpackageEntity)
        Dim WAWPID, WAWPSuccID As Integer
        Dim Rel1, Rel2 As String
        Dim Lag1, Lag2 As Double

        WAWPID = WAWP.WAWPID
        For Each Row As DataRow In
MyDataSet.Tables("tblWAWPPredecessors").Select("WAWPPredID = '" & WAWPID &
"'")
            Rel1 = ""
            Rel2 = ""
            Lag1 = 0
            Lag2 = 0
            If Row.Item("WAWPPredID") = WAWPID Then
                WAWPSuccID = CInt(Row.Item("WAWPID"))
```

```
                If Not Row.Item("WAWPRel1") Is DBNull.Value Then
                    Rel1 = CStr(Row.Item("WAWPRel1"))
                    Lag1 = CDbl(Row.Item("WAWPLag1"))
                End If

                If Not Row.Item("WAWPRel2") Is DBNull.Value Then
                    Rel2 = CStr(Row.Item("WAWPRel2"))
                    Lag2 = CDbl(Row.Item("WAWPLag2"))
                End If

                If WAWP.WAWPSuccs.ContainsKey(WAWPID) Then
                    If Not WAWP.WAWPSuccs(WAWPID).Contains(WAWPSuccID) Then
                        WAWP.WAWPSuccs(WAWPID).Add(WAWPSuccID)
                        WAWP.WAWPSuccRel1.Add(WAWPSuccID, Rel1)
                        WAWP.WAWPSuccRel2.Add(WAWPSuccID, Rel2)

                        WAWP.WAWPSuccLag1.Add(WAWPSuccID, Lag1)
                        WAWP.WAWPSuccLag2.Add(WAWPSuccID, Lag2)
                    Else
                        Throw New ArgumentException("Error in WAWPSuccessor!")
                    End If

                Else
                    Dim WAWPSuccs As New List(Of Integer)
                    WAWPSuccs.Add(WAWPSuccID)
                    WAWP.WAWPSuccs.Add(WAWPID, WAWPSuccs)

                    WAWP.WAWPSuccRel1.Add(WAWPSuccID, Rel1)
                    WAWP.WAWPSuccRel2.Add(WAWPSuccID, Rel2)

                    WAWP.WAWPSuccLag1.Add(WAWPSuccID, Lag1)
                    WAWP.WAWPSuccLag2.Add(WAWPSuccID, Lag2)
                End If

            End If

        Next
    End Sub

    Public Function FindEarlyStartFromESConstraint(ByVal MyWAWPID As Integer)
As Date
        Dim WPID, CalendarID As Integer
        Dim ESDateFromESConstraint As Date = DateTime.MinValue
        'find WP (at this point of time, WAWP has not been generated. thus,
have to use datatable)
        If tblWAWP.Rows.Contains(MyWAWPID) Then
            WPID = CInt(tblWAWP.Rows.Find(MyWAWPID).Item("WAWPWorkpackageID"))
        End If
        'find ES and Calender ID
        If tblWorkpackages.Rows.Contains(WPID) Then
            CalendarID =
CInt(tblWorkpackages.Rows.Find(WPID).Item("WPCalendarID"))
            If Not
tblWorkpackages.Rows.Find(WPID).Item("WPEarlyStartOverride") Is DBNull.Value
Then
```

```
                ESDateFromESConstraint =
CDate(tblWorkpackages.Rows.Find(WPID).Item("WPEarlyStartOverride"))
            Else
                If Not tblWorkpackages.Rows.Find(WPID).Item("WPEarlyStart")
Is DBNull.Value Then
                    ESDateFromESConstraint =
CDate(tblWorkpackages.Rows.Find(WPID).Item("WPEarlyStart"))
                End If
            End If
        End If
        Dim StartHour, HoursPerDay As Double
        FindStartHour_HoursPerDay(CalendarID, HoursPerDay, StartHour)
        'By default, a date is supposed to start at 12:00 am
        ESDateFromESConstraint = ESDateFromESConstraint.AddHours(StartHour)

        Return ESDateFromESConstraint
    End Function

    Public Function FindEarlyStartFromEFConstraint(ByVal MyWAWPID As Integer)
As Date
        Dim WPID, CalendarID As Integer
        Dim Duration As Double
        Dim ESDateFromEFConstraint As Date = DateTime.MinValue
        Dim EFDate As Date = DateTime.MinValue

        'find WP (at this point of time, WAWP has not been generated. thus,
have to use datatable)
        If tblWAWP.Rows.Contains(MyWAWPID) Then
            WPID = CInt(tblWAWP.Rows.Find(MyWAWPID).Item("WAWPWorkpackageID"))
        End If

        'find EF and Calender ID
        If tblWorkpackages.Rows.Contains(WPID) Then
            CalendarID =
CInt(tblWorkpackages.Rows.Find(WPID).Item("WPCalendarID"))
            Duration =
CDbl(tblWorkpackages.Rows.Find(WPID).Item("WPDurationOverride"))
            'convert from days to hours
            Duration = CalculateDurationFromDaysToHours(CalendarID, Duration)
            If Not tblWorkpackages.Rows.Find(WPID).Item("WPEarlyFinish") Is
DBNull.Value Then
                EFDate =
CDate(tblWorkpackages.Rows.Find(WPID).Item("WPEarlyFinish"))
            End If
        End If

        'find start hour
        Dim StartHour, HoursPerDay As Double
        FindStartHour_HoursPerDay(CalendarID, HoursPerDay, StartHour)

        'If Early Finish Constraint exists
        If Not EFDate = DateTime.MinValue Then
            'For finish time, it should add StartHour + HoursPerDay
            EFDate = EFDate.AddHours(StartHour + HoursPerDay)
            ESDateFromEFConstraint = CalculateSecondDate(EFDate, -1 *
Duration, CalendarID)
        End If
```

```
        If ESDateFromEFConstraint.Hour = 17 Then
            ESDateFromEFConstraint = FindTheNextWorkingDay(CalendarID,
ESDateFromEFConstraint)
        End If

        Return ESDateFromEFConstraint
    End Function

    Public Sub FindStartHour_HoursPerDay(ByVal CalendarID As Integer, ByRef
HoursPerDay As Double, ByRef StartHour As Double)

        If tblCalendar.Rows.Contains(CalendarID) Then
            HoursPerDay = CDbl(tblCalendar.Rows.Find(CalendarID).Item("Hours
per Shift")) * CDbl(tblCalendar.Rows.Find(CalendarID).Item("Shifts per Day"))
            StartHour =
CDate(tblCalendar.Rows.Find(CalendarID).Item("StartTime")).Hour +
CDate(tblCalendar.Rows.Find(CalendarID).Item("StartTime")).Minute / 60
        Else
            Throw New ArgumentException("Calendar ID does not exist in
Calendar Table")
        End If
    End Sub

    Public Sub CalculatePriority(ByRef WAWPEntity As
WorkAreaWorkpackageEntity)
        ''Based on free float
        'Dim TimeNow As Integer = MyEngine.TimeNow
        'Dim WAWPLateFinishDate As Date = WAWPEntity.LF
        ''Use default duration (or it could also use default manhours)
        'Dim WAWPDuration As Double = WAWPEntity.DurationOverride
        'Dim WAWPCalendarID As Integer = WAWPEntity.CalendarID
        'WAWPDuration = CalculateDurationFromDaysToHours(WAWPCalendarID,
WAWPDuration)
        ''Calculate LateStart date
        'Dim WAWPLateStartDate As Date =
CalculateSecondDate(WAWPLateFinishDate, -1 * WAWPDuration, WAWPCalendarID)
        ''Convert it to simulation time
        'Dim WAWPLateStartSimulationTime As Integer =
WAWPLateStartDate.Subtract(ProjectStartDate).TotalHours * Converter
        ''Calculate free float
        'Dim FF As Integer = Max(WAWPLateStartSimulationTime - TimeNow, 0)
        'Dim Priority As Integer = Round(100000 / (1 + FF))
        'WAWPEntity.Priority = Priority
        ''MessageBox.Show(WAWPEntity.WAWPID & " has priority of " & Priority)

        'Use current float model
        'Get the Late finish date
        Dim WAWPLateFinishDate As Date = WAWPEntity.CalculatedLF
        'Get the Time NOW
        Dim SimTimenow As Date = ProjectStartDate.AddHours(MyEngine.TimeNow /
Converter)
        'Calculate the balance duration or the duration required to complete
the WAWP
        'There are two situations at this point: 1) WAWP that has not started
yet; 2) WAWP that has been allocated resources earlier
        Dim WAWPDuration As Double
```

```
        Dim WAWPCalendarID As Integer = WAWPEntity.CalendarID
        If WAWPEntity.WhetherOrNotCaptedRes = False Then
            'For WAWP that has not started yet, use the default duration
            WAWPDuration = WAWPEntity.DurationOverride
            WAWPDuration = CalculateDurationFromDaysToHours(WAWPCalendarID,
WAWPDuration)
        Else
            'For WAWP that has been allocated resources earlier
            Dim WAWPEstimedFinishDate As Date = WAWPEntity.SimEstimatedFinish
            WAWPDuration = CalculateSpanBetweenDates(SimTimenow,
WAWPEstimedFinishDate, WAWPCalendarID)
        End If

        'Calculate Current Float
        Dim WAWPCurrentFloat As Integer
        If DateTime.Compare(WAWPLateFinishDate, SimTimenow) >= 0 Then
            WAWPCurrentFloat = CalculateSpanBetweenDates(SimTimenow,
WAWPLateFinishDate, WAWPCalendarID) - WAWPDuration
        Else
            WAWPCurrentFloat = CalculateSpanBetweenDates(WAWPLateFinishDate,
SimTimenow, WAWPCalendarID) - WAWPDuration
        End If

        'Calculate priority
        Dim WAWPPriority As Integer
        WAWPPriority = 100000 + -1 * WAWPCurrentFloat
        WAWPEntity.Priority = WAWPPriority
    End Sub

    Public Sub ResourceAllocate()
        'Clear the ResourceCapturedWAWPList
        ResourceCapturedWAWPList.Clear()

        'At the beginning of every time step, each WAWP entity has captured
ZERO resource
        For i As Integer = 0 To ArrivedWAWPList.Count - 1
            Dim WAWPEntity As WorkAreaWorkpackageEntity =
ArrivedWAWPList.Item(i)
            Dim ReqResName As String = WAWPEntity.ReqRes
            'Right now Has captured 0 resource
            WAWPEntity.CurrentReqResNameQtyPair(ReqResName) = 0
            Dim WorkAreaName As String = WAWPEntity.WorkAreaName
            WAWPEntity.CurrentReqResNameQtyPair(WorkAreaName) = 0
        Next

        'Allocate both resource and work area through the unified waiting
list
        If ArrivedWAWPList.Count = 0 Then
            Exit Sub
        End If

        ''Calculate priority based on 'Current Float'
        'For Each WAWPEntity As WorkAreaWorkpackageEntity In ArrivedWAWPList
        '    CalculatePriority(WAWPEntity)
        'Next
        ArrivedWAWPList.Sort()
```

```
        'Identify currentWAWPList only includes WAWPs that are currently in
progress (or have been processed in the previous time steps)
        'Since no interruption is allowed, then we can use
"WhetherOrNotCaptedRes" to check if the WAWP entity is current or not
        Dim CurrentWAWPList As New List(Of WorkAreaWorkpackageEntity)
        For Each CurrentWAWP As WorkAreaWorkpackageEntity In ArrivedWAWPList
            If CurrentWAWP.WhetherOrNotCaptedRes = True Then
                CurrentWAWPList.Add(CurrentWAWP)
            End If
        Next

        'Allocation part 1
        'Assign Minimum Amount of Resource and required work area to Current
WAWPs
        If CurrentWAWPList.Count <> 0 Then
            For Each CurrentWAWPEntity As WorkAreaWorkpackageEntity In
CurrentWAWPList
                '*************************Debug
point**********************************
                'Dim SimTimenow As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
                'If DateTime.Compare(SimTimenow, #11/12/2012 7:00:00 AM#) >=
0 Then
                '    MessageBox.Show("WAWPEntity " & CurrentWAWPEntity.WAWPID
& " arrives at " & SimTimenow)
                'End If
                '*************************Debug
point**********************************
                'find MinPercent
                Dim NormReqResAmount As Integer =
CurrentWAWPEntity.ReqResAmount
                Dim ReqResName As String = CurrentWAWPEntity.ReqRes
                Dim MinResPercent As Double =
ResourceMinPercentPair(ReqResName)
                Dim MinResAmount As Integer = CInt(MinResPercent *
NormReqResAmount)
                Dim WorkAreaName As String = CurrentWAWPEntity.WorkAreaName
                Dim NumOfWorkers As Integer = MinResAmount

                'Record how much resource it has captured
                CurrentWAWPEntity.CurrentReqResNameQtyPair(ReqResName) =
MinResAmount
                CurrentWAWPEntity.CurrentReqResNameQtyPair(WorkAreaName) =
NumOfWorkers

                'Update the available resource quantity
                Dim AvailResQty As Integer =
ResourceAvailQuantityPair(ReqResName)
                AvailResQty = AvailResQty - MinResAmount
                ResourceAvailQuantityPair(ReqResName) = AvailResQty

                'Update the work area congestion limit
                Dim AvailWorkAreaLimit As Integer =
ResourceAvailQuantityPair(WorkAreaName)
                AvailWorkAreaLimit = AvailWorkAreaLimit - NumOfWorkers
                ResourceAvailQuantityPair(WorkAreaName) = AvailWorkAreaLimit
```

```
                'Add to ResourceCapturedList
                ResourceCapturedWAWPList.Add(CurrentWAWPEntity)
            Next
        End If

        'Allocation part 2
        'All WAWPs (including Current and Newly Arrived ones) will compete
for the rest of the available resource
        'Here, Greed algorithm is used
        GreedyResourceAllocationMethod(ArrivedWAWPList)
        'Record or Update estimated finish date for all WAWP entities that
have captured the resource
        ArrivedWAWPList.Sort()
        For Each CurrentWAWPEntity As WorkAreaWorkpackageEntity In
ArrivedWAWPList
            'If CurrentWAWPEntity.WAWPID = 59 Then
            '    MessageBox.Show("Debugging")
            'End If
            Dim ReqResName As String = CurrentWAWPEntity.ReqRes
            If CurrentWAWPEntity.CurrentReqResNameQtyPair(ReqResName) > 0
Then
                'Calculate the estimated finish date for each WAWP that
captured the resource
                Dim EstimatedFinishDate As Date =
CalculateEstimatedSimFinish(CurrentWAWPEntity)
                '**************************Debug
point**********************************
                'Dim SimTimenow As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
                'If CurrentWAWPEntity.WAWPID = 312003 And
DateTime.Compare(SimTimenow, #6/16/2011 11:00:00 AM#) >= 0 Then
                '    MessageBox.Show("312003 estimated finish date is " &
EstimatedFinishDate & " at time " & SimTimenow)
                'End If
                '**************************Debug
point**********************************
                CurrentWAWPEntity.SimEstimatedFinish = EstimatedFinishDate
                'Inform this finish date to those successors that have 'FS'
or 'FF' relationship with it
                For Each Row As DataRow In
tblWAWPPredecessors.Select("WAWPPredID = '" & CurrentWAWPEntity.WAWPID & "'")
                    Row.Item("EstimatedPredFinish") =
CStr(EstimatedFinishDate)
                Next
            End If
        Next

        Debug.WriteLine("***Break line***")
        Debug.WriteLine("Arrived WAWPs")
        For i As Integer = 0 To ArrivedWAWPIDList.Count - 1
            Debug.WriteLine(ArrivedWAWPIDList.Item(i))
        Next
        Debug.WriteLine("Resource Captured WAWPs")
        For i As Integer = 0 To ResourceCapturedWAWPList.Count - 1
            'Debug.WriteLine(ResourceCapturedWAWPList.Count)
            Debug.WriteLine(ResourceCapturedWAWPList.Item(i).WAWPID)
        Next
```

```
        Debug.WriteLine("Completed WAWPs")
        For i As Integer = 0 To CompletedWAWPIDList.Count - 1
            Debug.WriteLine(CompletedWAWPIDList.Item(i))
        Next

    End Sub

    Public Sub GreedyResourceAllocationMethod(ByRef ArrivedWAWPList As
List(Of WorkAreaWorkpackageEntity))
        '"greedy" means it tends to satisfy higher priority WAWP entity (to
its normal resource level) before it continues with lower priority WAWP
entities
        'Sort the WAWP List according to Calculated Late Finish and Number of
Successors
        ArrivedWAWPList.Sort()

        'Allocate the resource and workarea that is left from the Allocation
Step 1
        For Each WAWPEntity As WorkAreaWorkpackageEntity In ArrivedWAWPList
            '*************************Debug
point**********************************
            'Dim SimTimeNow As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
            'If WAWPEntity.WAWPID = 41 Then
            '    MessageBox.Show(WAWPEntity.WAWPID & " arrives at " &
SimTimeNow)
            'End If
            'If DateTime.Compare(SimTimeNow, #11/12/2012 7:00:00 AM#) >= 0
Then
            '    MessageBox.Show("WAWPEntity " & WAWPEntity.WAWPID & "
arrives")
            'End If
            '*************************Debug
point**********************************
            'find the amount of resource available and the work area
congestion limit
            Dim ResName As String = WAWPEntity.ReqRes
            Dim AvailResQty As Integer = ResourceAvailQuantityPair(ResName)
            Dim WorkAreaName As String = WAWPEntity.WorkAreaName
            Dim AvailWorkArea As Integer =
ResourceAvailQuantityPair(WorkAreaName)
            Dim CombinedLimit As Integer = Min(AvailResQty, AvailWorkArea)
            '*************************Debug
point**********************************
            'If ResName = "PF" Then
            '    If AvailWorkArea < AvailResQty And AvailWorkArea >= 8 Then
            '        Dim SimTimeNow As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
            '        MessageBox.Show("WAWP ID " & WAWPEntity.WAWPID & " is
limited by congest limit at time " & SimTimeNow)
            '    End If
            'End If
            '*************************Debug
point**********************************

            'Save time to go through the whole list
            'If CombinedLimit = 0 Then
```

```
            '    GoTo NextWAWP
            'End If

            'Decide how much resource it can capture
            'The amount of resouce to be requested should be re-calculated
ONLY WHEN the WAWP happen to have 'FF' predecessor AND there is a possibility
that this 'FF' might be violated (due to delay of the predecessors)
            'Note that these estimated finish dates are determined or updated
in the previous time step, NOT the current one!!!
            Dim RequestedResourceAmount As Integer
            'Check if the WAWP has a 'FF' predecessor
            Dim WhetherOrNotHasFFPredecessor As Boolean = False
            For Each DataRow As DataRow In tblWAWPPredecessors.Select("WAWPID
= '" & WAWPEntity.WAWPID & "'")
                If Not DataRow.Item("WAWPRel2") Is DBNull.Value Then
                    If CStr(DataRow.Item("WAWPRel2")) = "FF" Then
                        WhetherOrNotHasFFPredecessor = True
                        Exit For
                    End If
                End If
            Next

            'Get the normal amount it can request
            Dim NormReqResAmount As Integer = WAWPEntity.ReqResAmount
            If WhetherOrNotHasFFPredecessor Then
                RequestedResourceAmount =
CalculateAmountOfResourceToBeRequested(WAWPEntity) -
WAWPEntity.CurrentReqResNameQtyPair(ResName) 'minus the amount that has been
captured in the allocation part 1
            Else
                RequestedResourceAmount = NormReqResAmount -
WAWPEntity.CurrentReqResNameQtyPair(ResName)
            End If

            'Also find its minimum resource requirement
            Dim MinResPercent As Double = ResourceMinPercentPair(ResName)
            Dim MinResAmount As Integer = CInt(MinResPercent *
NormReqResAmount)

            '**************************Debug
point**********************************
            'If WAWPEntity.WAWPID = 312003 And DateTime.Compare(SimTimeNow,
#6/16/2011 11:00:00 AM#) >= 0 Then
            '    MessageBox.Show("The WAWP 312003 want to capture " &
RequestedResourceAmount & " at time " & SimTimeNow & " the available amount
is " & AvailResQty)
            'End If
            '**************************Debug
point**********************************

            'Check if it is possible to allocate resource to current WAWP
            If CombinedLimit >= RequestedResourceAmount Then
                'Update the amount of resource the WAWP has captured
                WAWPEntity.CurrentReqResNameQtyPair(ResName) =
WAWPEntity.CurrentReqResNameQtyPair(ResName) + RequestedResourceAmount
                'Update the available resource quantity
```

```
                ResourceAvailQuantityPair(ResName) = AvailResQty -
RequestedResourceAmount

                'Update the work space the WAWP has captured
                WAWPEntity.CurrentReqResNameQtyPair(WorkAreaName) =
WAWPEntity.CurrentReqResNameQtyPair(WorkAreaName) + RequestedResourceAmount
                'Update the work area congestion limit
                ResourceAvailQuantityPair(WorkAreaName) = AvailWorkArea -
RequestedResourceAmount


            ElseIf CombinedLimit < RequestedResourceAmount And
CombinedLimit >= MinResAmount - WAWPEntity.CurrentReqResNameQtyPair(ResName)
And CombinedLimit > 0 Then
                '**************************Debug
point**********************************
                'Dim SimTimeNow As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
                'MessageBox.Show("WAWP ID " & WAWPEntity.WAWPID & " is
limited by congestion limit at time " & SimTimeNow)
                '**************************Debug
point**********************************
                'if still satisfy the minimum resource requirement, then
assign all the resource left to the current WAWP entity
                WAWPEntity.CurrentReqResNameQtyPair(ResName) =
WAWPEntity.CurrentReqResNameQtyPair(ResName) + CombinedLimit
                'Update the available resource quantity
                ResourceAvailQuantityPair(ResName) = AvailResQty -
CombinedLimit
                'Update the work area congestion limit
                WAWPEntity.CurrentReqResNameQtyPair(WorkAreaName) =
WAWPEntity.CurrentReqResNameQtyPair(WorkAreaName) + CombinedLimit
                ResourceAvailQuantityPair(WorkAreaName) = AvailWorkArea -
CombinedLimit
            Else
                'Dim SimTimeNow As Date =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
                'MessageBox.Show("WAWP ID " & WAWPEntity.WAWPID & " is
limited by congestion limit at time " & SimTimeNow)
                GoTo NextWAWP
            End If

            'Record the simulation start date if it is a newly arrived WAWP
entity
            If WAWPEntity.WhetherOrNotCaptedRes = False Then
                WAWPEntity.WhetherOrNotCaptedRes = True
                WAWPEntity.SimStart =
ProjectStartDate.AddHours(MyEngine.TimeNow / Converter)
                For Each Row As DataRow In
tblWAWPPredecessors.Select("WAWPPredID = '" & WAWPEntity.WAWPID & "'")
                    If Row.Item("SimPredStart") = "" Then
                        Row.Item("SimPredStart") = CStr(WAWPEntity.SimStart)
                    End If
                Next
            End If
```

```
            'For recording the estimated finish date, it should be done after
the whole Greedy Resource Allocation Method has been performed
            'So it is moved to ResourceAllocate subroutine

            'Add it to ResouceCapturedList
            If Not ResourceCapturedWAWPList.Contains(WAWPEntity) Then
                ResourceCapturedWAWPList.Add(WAWPEntity)
            End If
NextWAWP:
        Next

    End Sub

    Public Function CalculateAmountOfResourceToBeRequested(ByVal WAWPEntity
As WorkAreaWorkpackageEntity) As Integer
        ''True' means FF precedence relationship is maintained; while 'False'
means FF precedence relationship has been violated
        Dim WhetherOrNotMaintained As Boolean = True
        'Calculate manhours left
        Dim WAWPID As Integer = WAWPEntity.WAWPID
        Dim WPID As Integer = WAWPEntity.WPID
        Dim WAWPCalendarID As Integer = WAWPEntity.CalendarID
        Dim ResName As String = WAWPEntity.ReqRes
        Dim NormReqResAmount As Integer = WAWPEntity.ReqResAmount
        'Dim AlreadyCapResAmount As Integer =
WAWPEntity.CurrentReqResNameQtyPair(ResName)
        Dim OriginalDuration As Double =
CalculateDurationFromDaysToHours(WAWPCalendarID, WAWPEntity.DurationOverride)
        Dim TotalManHoursToBeAchieved As Double = NormReqResAmount *
OriginalDuration
        Dim ManHoursHaveAchieved As Double = WAWPEntity.ManHoursCompleted
        Dim ManHoursLeft As Double = TotalManHoursToBeAchieved -
ManHoursHaveAchieved

        '**************************Debug
point**********************************
        'Dim SimTimenow As Date = ProjectStartDate.AddHours(MyEngine.TimeNow
/ Converter)
        'If WAWPID = 312003 And DateTime.Compare(SimTimenow, #6/16/2011
11:00:00 AM#) >= 0 Then
        '    MessageBox.Show("312003 manhours left is " & ManHoursLeft & " at
time " & SimTimenow)
        'End If
        '**************************Debug
point**********************************

        'Suppose we can capture 100% of resource
        Dim PossibleDurationLeft As Double = ManHoursLeft / NormReqResAmount
        Dim PossibleFinishDate As Date =
CalculateSecondDate(ProjectStartDate.AddHours(MyEngine.TimeNow),
PossibleDurationLeft, WAWPCalendarID)
        'find Finish date constraint
        Dim FinishDateFromFFConstraint As Date = System.DateTime.MinValue
        For Each Row As DataRow In tblWAWPPredecessors.Select("WAWPID = '" &
WAWPID & "'")
            Dim WAWPPredID As Integer = Row.Item("WAWPPredID")
            Dim WAWPPredEstimatedFinish As Date
```

```
            If Not Row.Item("EstimatedPredFinish") = "" Then
                WAWPPredEstimatedFinish =
CDate(Row.Item("EstimatedPredFinish"))
            End If
            Dim WAWPPredCalendarID As Integer =
MyWAWPs(WAWPPredID).CalendarID
            Dim Rel2 As String = ""
            If Not Row.Item("WAWPRel2") Is DBNull.Value Then
                Rel2 = CStr(Row.Item("WAWPRel2"))
            End If
            Dim Lag2 As Double = 0
            If Not Row.Item("WAWPLag2") Is DBNull.Value Then
                Lag2 = CStr(Row.Item("WAWPLag2"))
            End If

            Dim EstFinDate As Date = System.DateTime.MinValue
            If Rel2 <> "" Then
                If Rel2 = "FF" Then
                    EstFinDate = CalculateSecondDate(WAWPPredEstimatedFinish,
Lag2, WAWPPredCalendarID)
                End If
            End If

            If DateTime.Compare(FinishDateFromFFConstraint, EstFinDate) < 0
Then
                FinishDateFromFFConstraint = EstFinDate
            End If
        Next
        'Check if FF Precedence Constarint(s) will be maintained or not
        If DateTime.Compare(PossibleFinishDate, FinishDateFromFFConstraint) <
0 Then
            WhetherOrNotMaintained = False
        End If

        If WhetherOrNotMaintained Then
            Return NormReqResAmount
        Else
            'find the Correct duration left from now
            Dim CurrentSimDate = ProjectStartDate.AddHours(MyEngine.TimeNow /
Converter)
            Dim DurationLeft As Double =
CalculateSpanBetweenDates(CurrentSimDate, FinishDateFromFFConstraint,
WAWPCalendarID)
            'calculate the amount of resource that can be captured
            Dim ResourceAmount As Integer = CInt(ManHoursLeft / DurationLeft)
            Return ResourceAmount
        End If
    End Function

    Public Sub CalculateTotalfloat(ByVal tblWorkpackages As DataTable, ByVal
tblWAWP As DataTable, ByVal tblWAWPPredecessors As DataTable)
        'Create a new datatable to contain the total float information
        Dim tblTotalFloatCal As DataTable =
MyDataSet.Tables.Add("tblWAWPTotalFloat")
        Dim MyPrimaryKeycolumns As DataColumn()
        tblTotalFloatCal.Columns.Add("WAWPID", Type.GetType("System.Int32"))
        tblTotalFloatCal.Columns.Add("ES", Type.GetType("System.DateTime"))
```

```
        tblTotalFloatCal.Columns.Add("EF", Type.GetType("System.DateTime"))
        tblTotalFloatCal.Columns.Add("LS", Type.GetType("System.DateTime"))
        tblTotalFloatCal.Columns.Add("LF", Type.GetType("System.DateTime"))
        tblTotalFloatCal.Columns.Add("TF", Type.GetType("System.Double"))
        MyPrimaryKeycolumns = New DataColumn(0) {}
        MyPrimaryKeycolumns(0) = tblTotalFloatCal.Columns("WAWPID")
        tblTotalFloatCal.PrimaryKey = MyPrimaryKeycolumns

        'First WAWPs are those who have no predecessor
        Dim FirstWAWPIDs As New List(Of Integer)
        Dim LastWAWPIDs As New List(Of Integer)
        Dim WAWPHasPerformedForwardCal As New List(Of Integer)
        Dim WAWPHasPerformedBackwardCal As New List(Of Integer)

        For i As Integer = 0 To PredWAWPIDs.Count - 1
            If Not SuccWAWPIDs.Contains(PredWAWPIDs.Item(i)) Then
                If Not FirstWAWPIDs.Contains(PredWAWPIDs.Item(i)) Then
                    FirstWAWPIDs.Add(PredWAWPIDs.Item(i))
                End If
            End If
        Next

        'Forward Calculation for WAWPs that do not have any predecessor
        For i As Integer = 0 To FirstWAWPIDs.Count - 1
            Dim WAWPID As Integer = FirstWAWPIDs.Item(i)
            Dim WPID, WAWPCalendarID As Integer
            Dim WAWPDuration As Double
            If tblWAWP.Rows.Contains(WAWPID) Then
                WPID = tblWAWP.Rows.Find(WAWPID).Item("WAWPWorkpackageID")
                WAWPDuration =
tblWAWP.Rows.Find(WAWPID).Item("WAWPEstDuration")
            End If
            If tblWorkpackages.Rows.Contains(WPID) Then
                WAWPCalendarID =
tblWorkpackages.Rows.Find(WPID).Item("WPCalendarID")
                WAWPDuration =
CalculateDurationFromDaysToHours(WAWPCalendarID, WAWPDuration)
            End If

            Dim ES As Date = ProjectStartDate ' at least the early start date
should be later than project start date
            Dim EarlyStartFromESConstraint As Date
            Dim EarlyStartFromEFConstraint As Date
            'if the WAWP has ES constraint, use its ES as its start date;
            If Not tblWorkpackages.Rows.Find(WPID).Item("WPEarlyStart") Is
DBNull.Value Then
                EarlyStartFromESConstraint =
FindEarlyStartFromESConstraint(WAWPID)
                If DateTime.Compare(ES, EarlyStartFromESConstraint) < 0 Then
                    ES = EarlyStartFromESConstraint
                End If
            End If
            'if the WAWP has EF constraint, use its EF to calculate its
possible start date;
            If Not tblWorkpackages.Rows.Find(WPID).Item("WPEarlyFinish") Is
DBNull.Value Then
```

```
                EarlyStartFromEFConstraint =
FindEarlyStartFromEFConstraint(WAWPID)
                If DateTime.Compare(ES, EarlyStartFromEFConstraint) < 0 Then
                    ES = EarlyStartFromEFConstraint
                End If
            End If

            If ES.Hour = 17 Then
                ES = FindTheNextWorkingDay(WAWPCalendarID, ES)
            End If
            Dim EF As Date = CalculateSecondDate(ES, WAWPDuration,
WAWPCalendarID)

            Dim NewRow As DataRow = tblTotalFloatCal.NewRow
            NewRow.Item("WAWPID") = WAWPID
            NewRow.Item("ES") = ES
            NewRow.Item("EF") = EF
            NewRow.Item("LS") = DBNull.Value
            NewRow.Item("LF") = DBNull.Value
            NewRow.Item("TF") = DBNull.Value
            tblTotalFloatCal.Rows.Add(NewRow)
            If Not WAWPHasPerformedForwardCal.Contains(WAWPID) Then
                WAWPHasPerformedForwardCal.Add(WAWPID)
            End If
        Next

        'Forward Calculation Continue
        Dim TotalNumOfWAWP As Integer = tblWAWP.Rows.Count
        Dim WAWPHasAnalyzedInForwardCal As New List(Of Integer)
        Do
            For i As Integer = 0 To WAWPHasPerformedForwardCal.Count - 1
                Dim WAWPPredID As Integer = WAWPHasPerformedForwardCal.Item(i)
                'If WAWPPredID = 102001 Then
                '    MessageBox.Show("OK!")
                'End If
                Dim WAWPPredES As Date
                Dim WAWPPredEF As Date
                If tblTotalFloatCal.Rows.Contains(WAWPPredID) Then
                    WAWPPredES =
CDate(tblTotalFloatCal.Rows.Find(WAWPPredID).Item("ES"))
                    WAWPPredEF =
CDate(tblTotalFloatCal.Rows.Find(WAWPPredID).Item("EF"))
                End If
                If Not WAWPHasAnalyzedInForwardCal.Contains(WAWPPredID) Then
                    For Each Row In tblWAWPPredecessors.Select("WAWPPredID =
'" & WAWPPredID & "'")
                        'Find a succesor
                        Dim WAWPID As Integer = Row.Item("WAWPID")
                        Dim WAWPES As Date = DateTime.MinValue
                        Dim WAWPEF As Date
                        'Find its duration
                        Dim WPID, WAWPCalendarID As Integer
                        Dim WAWPDuration As Double
                        If tblWAWP.Rows.Contains(WAWPID) Then
                            WPID =
tblWAWP.Rows.Find(WAWPID).Item("WAWPWorkpackageID")
```

```
                            WAWPDuration =
tblWAWP.Rows.Find(WAWPID).Item("WAWPEstDuration")
                        End If
                        If tblWorkpackages.Rows.Contains(WPID) Then
                            WAWPCalendarID =
tblWorkpackages.Rows.Find(WPID).Item("WPCalendarID")
                            WAWPDuration =
CalculateDurationFromDaysToHours(WAWPCalendarID, WAWPDuration)
                        End If
                        'Calculate the Early start and Early finish from
precedence relationship
                        Dim Rel1 As String = ""
                        Dim Rel2 As String = ""
                        If Not Row.Item("WAWPRel1") Is DBNull.Value Then
                            Rel1 = CStr(Row.Item("WAWPRel1"))
                        End If
                        If Not Row.Item("WAWPRel2") Is DBNull.Value Then
                            Rel2 = CStr(Row.Item("WAWPRel2"))
                        End If
                        Dim Lag1 As Double = 0
                        Dim Lag2 As Double = 0
                        If Not Row.Item("WAWPLag1") Is DBNull.Value Then
                            Lag1 = CStr(Row.Item("WAWPLag1"))
                        End If
                        If Not Row.Item("WAWPLag2") Is DBNull.Value Then
                            Lag2 = CStr(Row.Item("WAWPLag2"))
                        End If

                        If Not Rel1 = "" Then
                            If Rel1 = "SS" Then
                                WAWPES = CalculateSecondDate(WAWPPredES, Lag1,
WAWPCalendarID)
                            ElseIf Rel1 = "FS" Then
                                WAWPES = CalculateSecondDate(WAWPPredEF, Lag1,
WAWPCalendarID)
                            End If
                        End If
                        If Not Rel2 = "" Then
                            If Rel2 = "FF" Then
                                WAWPEF = CalculateSecondDate(WAWPPredEF, Lag2,
WAWPCalendarID)
                                If DateTime.Compare(WAWPES,
CalculateSecondDate(WAWPEF, -1 * WAWPDuration, WAWPCalendarID)) < 0 Then
                                    WAWPES = CalculateSecondDate(WAWPEF, -1 *
WAWPDuration, WAWPCalendarID)
                                End If
                            End If
                        End If
                        If WAWPES.Hour = 17 Then
                            WAWPES = FindTheNextWorkingDay(WAWPCalendarID,
WAWPES)
                        End If

                        'Check early start and searly finish constraint
                        Dim EarlyStartFromESConstraint As Date
                        Dim EarlyStartFromEFConstraint As Date
```

```
                        If Not
tblWorkpackages.Rows.Find(WPID).Item("WPEarlyStart") Is DBNull.Value Then
                            EarlyStartFromESConstraint =
FindEarlyStartFromESConstraint(WAWPID)
                            If DateTime.Compare(WAWPES,
EarlyStartFromESConstraint) < 0 Then
                                WAWPES = EarlyStartFromESConstraint
                            End If
                        End If
                        If Not
tblWorkpackages.Rows.Find(WPID).Item("WPEarlyFinish") Is DBNull.Value Then
                            EarlyStartFromEFConstraint =
FindEarlyStartFromEFConstraint(WAWPID)
                            If DateTime.Compare(WAWPES,
EarlyStartFromEFConstraint) < 0 Then
                                WAWPES = EarlyStartFromEFConstraint
                            End If
                        End If

                        WAWPEF = CalculateSecondDate(WAWPES, WAWPDuration,
WAWPCalendarID)

                        'Record the Early start and Early finish
                        If Not tblTotalFloatCal.Rows.Contains(WAWPID) Then
                            Dim NewRow As DataRow = tblTotalFloatCal.NewRow
                            NewRow.Item("WAWPID") = WAWPID
                            NewRow.Item("ES") = WAWPES
                            NewRow.Item("EF") = WAWPEF
                            NewRow.Item("LS") = DBNull.Value
                            NewRow.Item("LF") = DBNull.Value
                            NewRow.Item("TF") = DBNull.Value
                            tblTotalFloatCal.Rows.Add(NewRow)
                        Else
                            'update with the maximum early start and early
finish
                            Dim DataRow As DataRow =
tblTotalFloatCal.Rows.Find(WAWPID)
                            If DateTime.Compare(DataRow.Item("ES"), WAWPES) <
0 Then
                                DataRow.Item("ES") = WAWPES
                            End If
                            If DateTime.Compare(DataRow.Item("EF"), WAWPEF) <
0 Then
                                DataRow.Item("EF") = WAWPEF
                            End If
                        End If
                    Next
                    WAWPHasAnalyzedInForwardCal.Add(WAWPPredID)

                    'Check every successor of the current WAWP to see if it
has fulfilled all the precedence relationships
                    'If yes, it should be included in
WAWPHasPerformedForwardCal, meaning it ES and EF has been determined
                    For Each Row1 In tblWAWPPredecessors.Select("WAWPPredID =
'" & WAWPPredID & "'")
                        Dim WAWPID As Integer = Row1.Item("WAWPID")
                        Dim WhetherOrNotDetermined As Boolean = True
```

```
                        For Each DataRow2 As DataRow In
tblWAWPPredecessors.Select("WAWPID = '" & WAWPID & "'")
                            Dim EachWAWPPredID As Integer =
CInt(DataRow2.Item("WAWPPredID"))
                            If Not
WAWPHasAnalyzedInForwardCal.Contains(EachWAWPPredID) Then
                                WhetherOrNotDetermined = False
                                Exit For
                            End If
                        Next
                        If WhetherOrNotDetermined Then
                            WAWPHasPerformedForwardCal.Add(WAWPID)
                        End If
                    Next
                End If
            Next
        Loop While WAWPHasPerformedForwardCal.Count < TotalNumOfWAWP


        'Backward Calculation
        'First all the WAWPs that do not have any successor
        For i As Integer = 0 To SuccWAWPIDs.Count - 1
            If Not PredWAWPIDs.Contains(SuccWAWPIDs.Item(i)) Then
                If Not LastWAWPIDs.Contains(SuccWAWPIDs.Item(i)) Then
                    LastWAWPIDs.Add(SuccWAWPIDs.Item(i))
                End If
            End If
        Next

        For i As Integer = 0 To LastWAWPIDs.Count - 1
            Dim WAWPID As Integer = LastWAWPIDs.Item(i)
            Dim WPID, WAWPCalendarID As Integer
            Dim WAWPDuration As Double
            Dim DueDate As Date
            If tblWAWP.Rows.Contains(WAWPID) Then
                WPID = tblWAWP.Rows.Find(WAWPID).Item("WAWPWorkpackageID")
                WAWPDuration =
tblWAWP.Rows.Find(WAWPID).Item("WAWPEstDuration")
            End If
            If tblWorkpackages.Rows.Contains(WPID) Then
                WAWPCalendarID =
tblWorkpackages.Rows.Find(WPID).Item("WPCalendarID")
                WAWPDuration =
CalculateDurationFromDaysToHours(WAWPCalendarID, WAWPDuration)
                'If WP has a late-finish constraint, the late finish date
should be at least later than its early finish date
                If Not tblWorkpackages.Rows.Find(WPID).Item("WPLateFinish")
Is DBNull.Value Then
                    DueDate =
tblWorkpackages.Rows.Find(WPID).Item("WPLateFinish")
                Else
                    'in case when WP does not have a Late-finish constraint,
use its early finish date obtained in the forward calculation round
                    DueDate = tblTotalFloatCal.Rows.Find(WPID).Item("EF")
                End If
            End If
            Dim StartHour, HoursPerDay As Double
```

```
            FindStartHour_HoursPerDay(WAWPCalendarID, HoursPerDay, StartHour)
            DueDate = DueDate.AddHours(StartHour + HoursPerDay)

            If tblTotalFloatCal.Rows.Contains(WAWPID) Then
                'Should equal to DueDate
                tblTotalFloatCal.Rows.Find(WAWPID).Item("LF") =
tblTotalFloatCal.Rows.Find(WAWPID).Item("EF")
                If
CalculateSecondDate(tblTotalFloatCal.Rows.Find(WAWPID).Item("LF"), -1 *
WAWPDuration, WAWPCalendarID).Hour = 17 Then
                    tblTotalFloatCal.Rows.Find(WAWPID).Item("LS") =
FindTheNextWorkingDay(WAWPCalendarID,
CalculateSecondDate(tblTotalFloatCal.Rows.Find(WAWPID).Item("LF"), -1 *
WAWPDuration, WAWPCalendarID))
                Else
                    tblTotalFloatCal.Rows.Find(WAWPID).Item("LS") =
CalculateSecondDate(tblTotalFloatCal.Rows.Find(WAWPID).Item("LF"), -1 *
WAWPDuration, WAWPCalendarID)
                End If
                tblTotalFloatCal.Rows.Find(WAWPID).Item("TF") =
CalculateSpanBetweenDates(tblTotalFloatCal.Rows.Find(WAWPID).Item("LS"),
tblTotalFloatCal.Rows.Find(WAWPID).Item("ES"), WAWPCalendarID)
            End If
            If Not WAWPHasPerformedBackwardCal.Contains(WAWPID) Then
                WAWPHasPerformedBackwardCal.Add(WAWPID)
            End If
        Next

        'Backward Calculation continues
        Dim WAWPHasAnalyzedInBackwardCal As New List(Of Integer)
        Do
            For i As Integer = 0 To WAWPHasPerformedBackwardCal.Count - 1
                Dim WAWPSuccID As Integer =
WAWPHasPerformedBackwardCal.Item(i)
                Dim WAWPSuccLS As Date
                Dim WAWPSuccLF As Date
                If tblTotalFloatCal.Rows.Contains(WAWPSuccID) Then
                    WAWPSuccLS =
CDate(tblTotalFloatCal.Rows.Find(WAWPSuccID).Item("LS"))
                    WAWPSuccLF =
CDate(tblTotalFloatCal.Rows.Find(WAWPSuccID).Item("LF"))
                End If
                If Not WAWPHasAnalyzedInBackwardCal.Contains(WAWPSuccID) Then
                    For Each Row In tblWAWPPredecessors.Select("WAWPID = '" &
WAWPSuccID & "'")
                        Dim WAWPPredID As Integer = Row.Item("WAWPPredID")
                        'If WAWPPredID = 110001 Then
                        '    MessageBox.Show("OK!")
                        'End If
                        Dim WAWPPredLS As Date = DateTime.MaxValue
                        Dim WAWPPredLF As Date
                        'Find its duration
                        Dim WPPredID, WAWPPredCalendarID As Integer
                        Dim WAWPPredDuration As Double
                        If tblWAWP.Rows.Contains(WAWPPredID) Then
                            WPPredID =
tblWAWP.Rows.Find(WAWPPredID).Item("WAWPWorkpackageID")
```

```
                            WAWPPredDuration =
tblWAWP.Rows.Find(WAWPPredID).Item("WAWPEstDuration")
                        End If
                        If tblWorkpackages.Rows.Contains(WPPredID) Then
                            WAWPPredCalendarID =
tblWorkpackages.Rows.Find(WPPredID).Item("WPCalendarID")
                            WAWPPredDuration =
CalculateDurationFromDaysToHours(WAWPPredCalendarID, WAWPPredDuration)
                            'MessageBox.Show(WAWPPredDuration)
                        End If
                        'Calculate the Late start and Late finish
                        Dim Rel1 As String = ""
                        Dim Rel2 As String = ""
                        If Not Row.Item("WAWPRel1") Is DBNull.Value Then
                            Rel1 = CStr(Row.Item("WAWPRel1"))
                        End If
                        If Not Row.Item("WAWPRel2") Is DBNull.Value Then
                            Rel2 = CStr(Row.Item("WAWPRel2"))
                        End If
                        Dim Lag1 As Double = 0
                        Dim Lag2 As Double = 0
                        If Not Row.Item("WAWPLag1") Is DBNull.Value Then
                            Lag1 = CStr(Row.Item("WAWPLag1"))
                        End If
                        If Not Row.Item("WAWPLag2") Is DBNull.Value Then
                            Lag2 = CStr(Row.Item("WAWPLag2"))
                        End If

                        If Not Rel1 = "" Then
                            If Rel1 = "SS" Then
                                WAWPPredLS = CalculateSecondDate(WAWPSuccLS,
-1 * Lag1, WAWPPredCalendarID)
                            ElseIf Rel1 = "FS" Then
                                WAWPPredLS = CalculateSecondDate(WAWPSuccLS,
-1 * Lag1 - WAWPPredDuration, WAWPPredCalendarID)
                            End If
                        End If

                        If Not Rel2 = "" Then
                            If Rel2 = "FF" Then
                                If DateTime.Compare(WAWPPredLS,
CalculateSecondDate(WAWPSuccLF, -1 * Lag2 - WAWPPredDuration,
WAWPPredCalendarID)) > 0 Then
                                    WAWPPredLS =
CalculateSecondDate(WAWPSuccLF, -1 * Lag2 - WAWPPredDuration,
WAWPPredCalendarID)
                                End If
                            End If
                        End If

                        WAWPPredLF = CalculateSecondDate(WAWPPredLS,
WAWPPredDuration, WAWPPredCalendarID)
                        If WAWPPredLS.Hour = 17 Then
                            WAWPPredLS =
FindTheNextWorkingDay(WAWPPredCalendarID, WAWPPredLS)
                        End If
```

```
                        'Record the Late start and Late finish
                        If tblTotalFloatCal.Rows.Contains(WAWPPredID) Then
                            'update with the minimum late start and late
finish
                            Dim DataRow As DataRow =
tblTotalFloatCal.Rows.Find(WAWPPredID)
                            If DataRow.Item("LF") Is DBNull.Value Then
                                DataRow.Item("LF") = WAWPPredLF
                            ElseIf DateTime.Compare(DataRow.Item("LF"),
WAWPPredLF) > 0 Then
                                DataRow.Item("LF") = WAWPPredLF
                            End If
                            If DataRow.Item("LS") Is DBNull.Value Then
                                DataRow.Item("LS") = WAWPPredLS
                            ElseIf DateTime.Compare(DataRow.Item("LS"),
WAWPPredLS) > 0 Then
                                DataRow.Item("LS") = WAWPPredLS
                            End If
                            DataRow.Item("TF") =
CalculateSpanBetweenDates(DataRow.Item("ES"), DataRow.Item("LS"),
WAWPPredCalendarID)
                        End If
                    Next
                    WAWPHasAnalyzedInBackwardCal.Add(WAWPSuccID)
                    'Check every predecessor of the current WAWP to see if it
has checked all successors
                    For Each Row1 As DataRow In
tblWAWPPredecessors.Select("WAWPID = '" & WAWPSuccID & "'")
                        Dim WAWPPredID As Integer = Row1.Item("WAWPPredID")
                        Dim WhetherOrNotDetermined As Boolean = True
                        For Each Row2 As DataRow In
tblWAWPPredecessors.Select("WAWPPredID = '" & WAWPPredID & "'")
                            Dim EachWAWPSuccID As Integer =
Row2.Item("WAWPID")
                            If Not
WAWPHasAnalyzedInBackwardCal.Contains(EachWAWPSuccID) Then
                                WhetherOrNotDetermined = False
                                Exit For
                            End If
                        Next
                        If WhetherOrNotDetermined Then
                            WAWPHasPerformedBackwardCal.Add(WAWPPredID)
                        End If
                    Next
                End If
            Next

        Loop While WAWPHasPerformedBackwardCal.Count < TotalNumOfWAWP

        tblTotalFloatCal.DefaultView.Sort = "WAWPID"
    End Sub

    Public Function CalculateEstimatedSimFinish(ByVal MyWAWPEntity As
WorkAreaWorkpackageEntity) As Date
        'Calculate the part of manhours that haven't achieved yet
        Dim WAWPID As Integer = MyWAWPEntity.WAWPID
        Dim WAWPCalendarID As Integer = MyWAWPEntity.CalendarID
```

```
        Dim OriginalDuration As Double =
CalculateDurationFromDaysToHours(WAWPCalendarID,
MyWAWPEntity.DurationOverride)
        Dim ResName As String = MyWAWPEntity.ReqRes
        Dim NormReqResAmount As Integer = MyWAWPEntity.ReqResAmount
        Dim TotalManHoursToBeAchieved As Double = NormReqResAmount *
OriginalDuration
        Dim ManHoursHaveAchieved As Double = MyWAWPEntity.ManHoursCompleted
        Dim ManHoursLeft As Double = TotalManHoursToBeAchieved -
ManHoursHaveAchieved

        'Get the Resouce Amount it has captured right now
        Dim ReqResAmount As Integer =
MyWAWPEntity.CurrentReqResNameQtyPair(ResName)

        'Calculate the duration to finish the WAWP
        Dim DurationLeft As Double = CDbl(ManHoursLeft / ReqResAmount)

        '*************************Debug
point**********************************
        'Dim SimTimenow As Date = ProjectStartDate.AddHours(MyEngine.TimeNow
/ Converter)
        'If WAWPID = 106001 And DateTime.Compare(SimTimenow, #6/3/2011
10:00:00 AM#) >= 0 Then
        '    MessageBox.Show("106001 manhours left is " & ManHoursLeft & " at
time " & SimTimenow & " and the balance duation is " & DurationLeft)
        'End If
        '*************************Debug
point**********************************

        'Calculate the estimated finish date
        Dim CurrSimDate As Date = ProjectStartDate.AddHours(MyEngine.TimeNow
/ Converter)
        Dim EstimatedSimFinish As Date = CalculateSecondDate(CurrSimDate,
DurationLeft, WAWPCalendarID)
        Return EstimatedSimFinish
    End Function

    Public Sub FindAllSuccessorsToAWorkAreaWorkpackage(ByVal CurrentWAWPID As
Integer, ByRef SuccessorsOfCurrentWAWPIDList As List(Of Integer))
        'It is a depth first traverse algorithm
        'Only successors that have 'FS' or 'FF' relationship will be impacted;
Successors that have 'SS' relationship will not be impacted

        'remove its finish date information from tblWAWPPredecessors
datatable
        'And find all its impacted successors
        Dim ImpactedSuccessorIDList As New List(Of Integer)
        For Each Row As DataRow In tblWAWPPredecessors.Select("WAWPPredID =
'" & CurrentWAWPID & "'")

            'Remove the estimated finish date
            If Not Row.Item("EstimatedPredFinish") = "" Then
                Row.Item("EstimatedPredFinish") = ""
            End If

            'If it is 'FS', the successor will be interrupted as a result
```

```
            If Not Row.Item("WAWPRel1") Is DBNull.Value Then
                If CStr(Row.Item("WAWPRel1")) = "FS" Then
                    ImpactedSuccessorIDList.Add(CInt(Row.Item("WAWPID")))
                End If
            End If
            'If it is 'FF', the successor will also be interrupted as a
result
            If Not Row.Item("WAWPRel2") Is DBNull.Value Then
                If CStr(Row.Item("WAWPRel2")) = "FF" Then
                    ImpactedSuccessorIDList.Add(CInt(Row.Item("WAWPID")))
                End If
            End If
        Next

        'Find if there is any succeeding WAWP to the successors (identified
in the last step)
        For i As Integer = 0 To ImpactedSuccessorIDList.Count - 1
            Dim CurrentSuccessorID As Integer =
ImpactedSuccessorIDList.Item(i)
            If SuccWAWPIDs.Contains(CurrentSuccessorID) Then
                'If the successor has its own succeeding WAWP, then it
repeats the same procedure
                FindAllSuccessorsToAWorkAreaWorkpackage(CurrentSuccessorID,
SuccessorsOfCurrentWAWPIDList)
            End If
        Next

        'Regardless, it is one of current WAWP's successors
        'In this way, WAWP itself is considered as one of successors
        If Not SuccessorsOfCurrentWAWPIDList.Contains(CurrentWAWPID) Then
            SuccessorsOfCurrentWAWPIDList.Add(CurrentWAWPID)
        End If

    End Sub

    Private Function CalculateSecondDate(ByVal FirstDate As Date, ByVal
Interval As Double, ByVal CalendarID As Integer) As Date

        'FirstDate =<    < secondDate
        Dim i, DateID1, TotalHoursInFirstDay As Integer
        Dim sum, dt, HoursLeftInFirstDay As Double
        Dim MySecondDate As Date

        Dim StartHour, HoursPerDay As Double
        FindStartHour_HoursPerDay(CalendarID, HoursPerDay, StartHour)

        'Set hours between StartHour and StartHour+HoursPerDay
        If FirstDate.Hour < StartHour Then
            ' if it happens before the start hour, then the first date should
be a day before it shows
            ' the time should be backward one day, and dt (amount of time)
before the end of the working time
            dt = FirstDate.Hour + FirstDate.Minute / 60 - StartHour
            FirstDate = FirstDate.AddHours(-FirstDate.Hour - 24 + (StartHour
+ HoursPerDay))
            'FirstDate = FirstDate.AddHours(-dt - 24 + (StartHour +
HoursPerDay))
```

```
        ElseIf FirstDate.Hour > StartHour + HoursPerDay Then
            'if it happens after the knock-off time, then the dt is useless
and set the datetime to the end of the working time
            dt = FirstDate.Hour + FirstDate.Minute / 60 - StartHour -
HoursPerDay
            FirstDate = FirstDate.AddHours(-dt)
        End If

        Dim CalendarDescription As String =
tblCalendar.Rows.Find(CalendarID).Item("Calendar Description").ToString

        If tblCalendarDetail.Rows.Contains(FirstDate.ToShortDateString) Then
            DateID1 =
CInt(tblCalendarDetail.Rows.Find(FirstDate.ToShortDateString).Item("DateID").
ToString)
        Else
            Throw New ArgumentException("FirstDate does not exist in
Calendar_Detail")
            Return FirstDate
        End If

        sum = 0

        TotalHoursInFirstDay = CInt(tblCalendarDetail.Rows(DateID1 -
1).Item(CalendarDescription).ToString)

        If Interval >= 0 Then
            'there are two situations: (1) the firstdate is a working day; (2)
the firstdate is a non-working day
            If TotalHoursInFirstDay > 0 Then
                'If start time is later than the starthour in the calendar
                If FirstDate.Hour + FirstDate.Minute / 60 + FirstDate.Second
/ 360 > StartHour Then
                    'the actual working time will be less than
HoursLeftInFirstDay
                    HoursLeftInFirstDay = TotalHoursInFirstDay -
(FirstDate.Hour + FirstDate.Minute / 60 + FirstDate.Second / 360 - StartHour)
                Else
                    'otherwise it equals to HoursLeftInFirstDay
                    HoursLeftInFirstDay = TotalHoursInFirstDay
                End If
            Else
                HoursLeftInFirstDay = 0
            End If

            sum = HoursLeftInFirstDay
            i = DateID1
            'sum is total working hours with one more day after the end date
            Do Until sum >= Interval
                i = i + 1
                sum = sum + CDbl(tblCalendarDetail.Rows(i -
1).Item(CalendarDescription))
            Loop

            MySecondDate = CDate(tblCalendarDetail.Rows(i - 1).Item("Date"))
```

```
            Dim DurLastDay As Integer = CInt(tblCalendarDetail.Rows(i -
1).Item(CalendarDescription).ToString)
            Dim HourLastDay As Double = Interval - (sum - DurLastDay)

            'Calculate the closing time of the last day
            MySecondDate = MySecondDate.AddHours(StartHour + HourLastDay)
        Else

            If TotalHoursInFirstDay > 0 Then
                If FirstDate.Hour + FirstDate.Minute / 60 + FirstDate.Second
/ 360 > StartHour Then
                    HoursLeftInFirstDay = (FirstDate.Hour + FirstDate.Minute
/ 60 + FirstDate.Second / 360 - StartHour)
                Else
                    HoursLeftInFirstDay = 0
                End If
            Else
                HoursLeftInFirstDay = 0
            End If

            sum = -HoursLeftInFirstDay
            i = DateID1

            Do Until sum < Interval
                i = i - 1
                If i > 0 Then
                    sum = sum - CDbl(tblCalendarDetail.Rows(i -
1).Item(CalendarDescription))
                Else
                    Throw New ArgumentException("error in negative interval")
                End If
            Loop

            MySecondDate = CDate(tblCalendarDetail.Rows(i - 1).Item("Date"))
            Dim DurLastDay As Integer = CInt(tblCalendarDetail.Rows(i -
1).Item(CalendarDescription).ToString)
            Dim HourLastDay As Double = -Interval - (-sum - DurLastDay)
            MySecondDate = MySecondDate.AddHours(StartHour + DurLastDay -
HourLastDay)

        End If

        Return MySecondDate
    End Function

    Private Function CalculateSpanBetweenDates(ByVal FirstDate As Date, ByVal
SecondDate As Date, ByVal CalendarID As Integer) As Double
        'Calculate the Actual working hours between two dates, which
        ' FirstDate =<    < secondDate

        Dim DateID1, DateID2 As Integer
        Dim sum As Double

        Dim CalendarDescription As String =
tblCalendar.Rows.Find(CalendarID).Item("Calendar Description").ToString
        Dim StartHour, HoursPerDay As Double
```

```
        FindStartHour_HoursPerDay(CalendarID, HoursPerDay, StartHour)

        If tblCalendarDetail.Rows.Contains(FirstDate.ToShortDateString) Then
            DateID1 =
CInt(tblCalendarDetail.Rows.Find(FirstDate.ToShortDateString).Item("DateID").
ToString)
        Else
            Throw New ArgumentException("FirstDate does not exist in
Calendar_Detail")
            Return 0
        End If

        If tblCalendarDetail.Rows.Contains(SecondDate.ToShortDateString) Then
            DateID2 =
CInt(tblCalendarDetail.Rows.Find(SecondDate.ToShortDateString).Item("DateID")
.ToString)
        Else
            Throw New ArgumentException("SecondDate does not exist in
Calendar_Detail")
            Return 0
        End If

        sum = CInt(tblCalendarDetail.Compute("Sum([" & CalendarDescription &
"])", ("DateID >=" & Min(DateID1, DateID2) & " And DateID <=" & Max(DateID1,
DateID2))).ToString)

        'If the start time is later than the start hour of the first day,
then actual working time should be less
        Dim DurFirstDay As Integer = CInt(tblCalendarDetail.Rows(DateID1 -
1).Item(CalendarDescription).ToString)

        If FirstDate.Hour + FirstDate.Minute / 60 + FirstDate.Second / 360 >
StartHour Then
            sum = sum - Min(FirstDate.Hour - StartHour, DurFirstDay)
        End If

        'If the end time is before the end of working time, then actual
working time should be less
        Dim DurLastDay As Integer = CInt(tblCalendarDetail.Rows(DateID2 -
1).Item(CalendarDescription).ToString)

        If SecondDate.Hour + SecondDate.Minute / 60 + SecondDate.Second / 360
< StartHour + DurLastDay Then
            sum = sum - Min(StartHour + DurLastDay - SecondDate.Hour,
DurLastDay)
        End If

        If sum = 0 Then
            sum = SecondDate.Subtract(FirstDate).TotalHours
        End If
        Return sum
    End Function

    Private Function CalculateSuccessorTaskStartDate(ByVal
PredecessorStartDate As Date, ByVal Rel As String, ByVal Lag As Double, ByVal
CalendarID As Integer, ByVal PredecessorDuration As Double, ByVal
TaskDuration As Double) As Date
```

```
        Dim TaskESDate As Date
        Select Case Rel
            Case "SS"
                TaskESDate = CalculateSecondDate(PredecessorStartDate, Lag,
CalendarID)
            Case "SF"
                TaskESDate = CalculateSecondDate(PredecessorStartDate, Lag -
TaskDuration, CalendarID)
            Case "FS"
                TaskESDate = CalculateSecondDate(PredecessorStartDate,
PredecessorDuration + Lag, CalendarID)
            Case "FF"
                TaskESDate = CalculateSecondDate(PredecessorStartDate,
PredecessorDuration + Lag - TaskDuration, CalendarID)
            Case Else
                Throw New ArgumentException("error in logic")
        End Select

        Return TaskESDate
    End Function

End Class
```

VB.NET code for the simulation entity

```
Imports Simphony.Simulation

Public Class WorkAreaWorkpackageEntity
    Inherits Entity
    Implements IComparable(Of WorkAreaWorkpackageEntity)

    Public WAWPID, WPID, WorkAreaID, ClassificationID, Priority, CalendarID,
ReqResAmount, NumOfSuccessors, TotalFloat As Integer
    Public KeyQty, Manhours, DurationOverride, SimDuration, ManHoursCompleted
As Double
    Public WAWPName, WorkAreaName, ClassificationDesc, ReqRes,
ConstructionWorkArea As String
    Public ES, EF, LF, CalculatedLF, CalculatedES, SimStart, SimFinish,
SimEstimatedFinish As Date
    Public Interruptible, WhetherOrNotCaptedRes As Boolean
    'Precedence relationships
    Public CurrentReqResNameQtyPair As New Dictionary(Of String, Integer)
    Public WAWPPreds, WAWPSuccs As New Dictionary(Of Integer, List(Of
Integer))
    'Public WAWPPredEstimatedFinish As New Dictionary(Of Integer, Date)
    'Every Predecessor (or successor) will just one Rel1 or/and Rel2
    Public WAWPPredRel1, WAWPPredRel2, WAWPSuccRel1, WAWPSuccRel2 As New
Dictionary(Of Integer, String)
    Public WAWPPredLag1, WAWPPredLag2, WAWPSuccLag1, WAWPSuccLag2 As New
Dictionary(Of Integer, Double)
    Private Enum DateComparisonResult As Integer
        Earlier = -1
        Later = 1
        TheSame = 0
    End Enum
```

```
    Private Enum NumComparisonResult As Integer
        'the more successors it has, the more front it should be placed in
the waiting list
        Less = 1
        More = -1
        TheSame = 0
    End Enum

    Public Function CompareTo(ByVal other As WorkAreaWorkpackageEntity) As
Integer Implements System.IComparable(Of WorkAreaWorkpackageEntity).CompareTo
        ''first by minimum total float
        'Dim result As Integer
        'If Me.TotalFloat < other.TotalFloat Then
        '    result = -1
        'ElseIf Me.TotalFloat > other.TotalFloat Then
        '    result = 1
        'Else
        '    result = 0
        'End If

        'If result = 0 Then
        '    'comparison =
CType(Me.CalculatedES.CompareTo(other.CalculatedES), DateComparisonResult)
        '    'If CInt(comparison) = 0 Then
        '    If Me.WAWPID < other.WAWPID Then
        '        Return -1
        '    Else
        '        Return 1
        '    End If
        '    'Else
        '    'Return CInt(comparison)
        '    'End If
        'Else
        '    Return result
        'End If

        'First, compare Calculate Total float
        'Second, compare The number of successors
        'Third, compare whether the entity has started or not; already
started entities have higher priority than those haven't started
        'Fourth, compare the WAWPID. the larger, the higher priority

        Dim Comparison As Integer
        If Me.TotalFloat < other.TotalFloat Then
            Comparison = -1
        ElseIf Me.TotalFloat > other.TotalFloat Then
            Comparison = 1
        Else
            Comparison = 0
        End If
        Return Comparison


        'Dim Comparison As DateComparisonResult
        'Comparison = CType(Me.CalculatedLF.CompareTo(other.CalculatedLF),
DateComparisonResult)
        'If CInt(Comparison) = 0 Then
```

```
        '    Dim Comparison1 As NumComparisonResult
        '    Comparison1 =
CType(Me.NumOfSuccessors.CompareTo(other.NumOfSuccessors),
NumComparisonResult)
        '    If Comparison1 = 0 Then
        '        If Me.WhetherOrNotCaptedRes <> other.WhetherOrNotCaptedRes
Then
        '            If Me.WhetherOrNotCaptedRes = True Then
        '                Return -1
        '            Else
        '                Return 1
        '            End If
        '        Else
        '            If Me.WAWPID > other.WAWPID Then
        '                Return -1
        '            Else
        '                Return 1
        '            End If
        '        End If
        '    Else
        '        Return CInt(Comparison1)
        '    End If

        'Else
        '    Return CInt(Comparison)
        'End If

        ''Use pre-calculated (or pre-defined) priority values
        'Dim result As Integer
        'If Me.Priority > other.Priority Then
        '    result = -1
        'ElseIf Me.Priority < other.Priority Then
        '    result = 1
        'Else
        '    result = 0
        'End If

        'If result = 0 Then
        '    'If there is a tie
        '    'The less Id is, the higher priority it has
        '    If Me.WAWPID < other.WAWPID Then
        '        Return -1
        '    Else
        '        Return 1
        '    End If
        'End If
        'Return result


    End Function
End Class
```

VB.NET code for the crew resource

```
Imports Simphony
Imports Simphony.Simulation

Public Class CrewResource
    Inherits Resource

    Public TaskDesc As String
    Public OriginalQuantity, RampUp As Integer
    Public MinPercent, MaxPercent As Double
    Public Level, NonManPower, NonYardActivity, ZeroFreeFloatActivity As
Boolean
    Public Sub New(ByVal name As String, ByVal Quantity As Integer)
        MyBase.New(name, Quantity)
    End Sub
End Class
```

VB.NET code for the congestion resource

```
Imports Simphony
Imports Simphony.Simulation

Public Class CongestionResource
    Inherits Resource
    Public WorkAreaID, WAPriority, OriginalCongestionLimit As Integer
    Public WorkAreaDesignator, WAConstructionWorkArea As String
    Public WALength, WAWidth, WAHeight, WASouthWestPointX, WASouthWestPointY,
WASouthWestPointZ, WAAngle As Double

    Public Sub New(ByVal name As String, ByVal Quantity As Integer)
        MyBase.New(name, Quantity)
    End Sub
End Class
```

Appendix D

Complete List of Work Packages for Case Study

Work Package No. (1)	Description (2)	Quantity (man hours) (3)	Work Areas (4)	Predecessors (FS) (5)	Craft Personnel Requirements		
					Trade (6)	Normal Crew Size (7)	Min Crew Size (8)
1	Piling@011AB012ABC	40	011AB, 012ABC	—	PIL	10	8
2	Piling@006AB005AB	40	005AB, 006AB	1	PIL	10	8
3	Piling@014AB007ABC	40	007ABC, 014AB	2	PIL	10	8
4	ModuleSupportStructure@011AB	40	011AB	1	IW	10	8
5	ModuleSupportStructure@012ABC	40	012ABC	1	IW	10	8
6	ModuleSupportStructure@005AB	40	005AB	2	IW	10	8
7	ModuleSupportStructure@006AB	40	006AB	2	IW	10	8
8	ModuleSupportStructure@007ABC	40	007ABC	3	IW	10	8
9	ModuleSupportStructure@014AB	40	014AB	3	IW	10	8
10	1600-PR-011A	20	011AB	4	IW	10	8
11	1600-PR-011B	20	011AB	10	IW	10	8
12	1610-PR-005	20	011AB	11	IW	10	8
13	1600-PR-012A	20	012ABC	5,12	IW	10	8
14	1600-PR-012B	20	012ABC	13	IW	10	8

15	1600-PR-012C	20	012ABC	14	IW	10	8
16	1610-PR-004	20	012ABC	15	IW	10	8
17	1600-PR-005A	20	005AB	6,16	IW	10	8
18	1600-PR-005B	20	005AB	17	IW	10	8
19	1610-PR-002	20	005AB	18	IW	10	8
20	1610-PR-003	20	005AB	19	IW	10	8
21	1600-PR-006A	20	006AB	7,20	IW	10	8
22	1600-PR-006B	20	006AB	21	IW	10	8
23	1610-PR-001	20	006AB	22	IW	10	8
24	1600-PR-007A	20	007ABC	8,23	IW	10	8
25	1600-PR-007B	20	007ABC	24	IW	10	8
26	1600-PR-007C	20	007ABC	25	IW	10	8
27	1600-PR-014A	20	014AB	26,9	IW	10	8
28	1600-PR-014B	20	014AB	27	IW	10	8
29	PipingBetw011AB012ABC	60	011AB	11,15	PF	10	8
30	PipingBetw012ABC005AB	60	005AB	15,18	PF	10	8
31	PipingBetw005AB006AB	60	006AB	18,22	PF	10	8
32	PipingBetw006AB007ABC	60	007ABC	22,26	PF	10	8
33	PipingBetw007ABC014AB	60	014AB	26,28	PF	10	8
34	PipingSilencerFrameTopOf007C	20	007ABC	26	PF	10	8
35	PipingSilencerFrameTopOf014B	20	014AB	28	PF	10	8
36	Silencer@007ABC	60	007ABc	34	PF	10	8
37	Silencer@014AB	30	014AB	35	PF	10	8
38	HydrotesingBetw011AB012ABC	30	011AB	29(-2 days)	PF	10	8

39	HydrotesingBetw012ABC005AB	30	012ABC	30(-2 days)	PF	10	8
40	HydrotesingBetw005AB006AB	30	005AB	31(-2 days)	PF	10	8
41	HydrotesingBetw006AB007ABC	30	006AB	32(-2 days)	PF	10	8
42	HydrotesingBetw007ABC014AB	30	007ABC	33(-2 days)	PF	10	8
43	HydrotestingSilencerOnTopOf007ABC014AB	80	007ABC, 014AB	36,37	PF	10	8
44	InsulationBetw011AB012ABC	20	011AB	38(-1 day)	INS	10	8
45	InsulationBetw012ABC005AB	20	012ABC	39(-1 day)	INS	10	8
46	InsulationBetw005AB006AB	20	005AB	40(-1 day)	INS	10	8
47	InsulationBetw006AB007ABC	20	006AB	41(-1 day)	INS	10	8
48	InsulationBetw007ABC014AB	20	007ABC	42(-1 day)	INS	10	8
49	InsulationSilencerOntopOf007ABC014AB	40	014AB	43(-1 day)	INS	10	8
50	ElectricalCableTray@011AB012ABC	20	011AB, 012ABC	44	EL	10	8
51	ElectricalCableTray@005AB006AB007ABC014AB	40	005AB,006AB,007ABC,014AB	45,46,47,48,50,49	EL	10	8

Appendix E

PDDL Domain File for Pipe Spool Fabrication Sequencing Experiment 1

```
(define (domain pipespool)
  (:requirements :strips :typing)
  (:types assembly weldpt)
  (:predicates
   (belong ?w - weldpt ?p - assembly)
   (fitted ?w - weldpt)
   (representative ?p - assembly))
  (:action roll-fitting
         :parameters (?p1 ?p2 - assembly ?w1 - weldpt)
         :precondition (and (belong ?w1 ?p1)
                        (belong ?w1 ?p2)
                        (representative ?p1)
                        (representative ?p2)
                        (not(fitted ?w1))
                          (not(= ?p1 ?p2)))
         :effect                                                    (and
(fitted ?w1)(representative ?p1)(not(representative ?p2))
                   (forall (?w2 - weldpt)
                        (when (belong ?w2 ?p2)
                        (belong ?w2 ?p1))))
         )
  )
```

PDDL Problem File for Pipe Spool Fabrication Sequencing Experiment 1

```
(define (problem pipespool2)
  (:domain pipespool)
```

```
  (:objects p1 p2 p3 - assembly w1 w2 - weldpt)
  (:init (not(fitted w1))
       (not(fitted w2))
       (belong w1 p1)
       (belong w1 p2)
       (belong w2 p2)
       (belong w2 p3)
       (representative p1)
       (representative p2)
       (representative p3)
       )
  (:goal (and (representative p1)(fitted w1)(fitted w2)))
  )
```

PDDL domain File for Pipe Spool Fabrication Sequencing Experiment 2

```
(define (domain pipespool)
  (:requirements :strips :typing)
  (:types assembly weldpt)
  (:constants x y z - axis)
  (:predicates
   (belong ?w - weldpt ?p - assembly)
   (fitted ?w - weldpt)
   (representative ?p - assembly)
   (on ?w - weldpt ?a - axis))
  (:action roll-fitting-x
         :parameters (?p1 ?p2 - assembly ?w1 - weldpt)
         :precondition (and (belong ?w1 ?p1)
                        (belong ?w1 ?p2)
                        (representative ?p1)
```

```
                        (representative ?p2)
                        (not(fitted ?w1))
                    (on ?w1 x)
                            (not(= ?p1 ?p2))
                        )
        :effect                 (and                (fitted                 ?w1)
(representative ?p1)(not(representative ?p2))
                    (forall (?w2 - weldpt)
                        (when (belong ?w2 ?p2)
                        (belong ?w2 ?p1)))))
  (:action roll-fitting-y
        :parameters (?p1 ?p2 - assembly ?w1 - weldpt)
        :precondition (and (belong ?w1 ?p1)
                        (belong ?w1 ?p2)
                        (representative ?p1)
                        (representative ?p2)
                        (not(fitted ?w1))
                    (on ?w1 y)
                            (not(= ?p1 ?p2))
                        )
        :effect                 (and                (fitted                 ?w1)
(representative ?p1)(not(representative ?p2))
                    (forall (?w2 - weldpt)
                        (when (belong ?w2 ?p2)
                        (belong ?w2 ?p1)))))
   (:action roll-fitting-z
        :parameters (?p1 ?p2 - assembly ?w1 - weldpt)
        :precondition (and (belong ?w1 ?p1)
                        (belong ?w1 ?p2)
```

```
                        (representative ?p1)
                        (representative ?p2)
                        (not(fitted ?w1))
                  (on ?w1 z)
                          (not(= ?p1 ?p2))
                        )
        :effect               (and               (fitted               ?w1)
(representative ?p1)(not(representative ?p2))
                  (forall (?w2 - weldpt)
                        (when (belong ?w2 ?p2)
                        (belong ?w2 ?p1))))))
```

PDDL problem File for Pipe Spool Fabrication Sequencing Experiment 2

```
(define (problem pipespoolB969-SS00520)
  (:domain pipespool)
  (:objects p1 p2 p3 p4 p5 p6 p7 p8 - assembly w1 w2 w3 w4 w5 w6 w7 - weldpt)
  (:init (not(fitted w1))
         (not(fitted w2))
         (not(fitted w3))
       (not(fitted w4))
       (not(fitted w5))
       (not(fitted w6))
       (not(fitted w7))
       (belong w1 p1)
       (belong w1 p7)
       (belong w2 p2)
       (belong w2 p7)
       (belong w3 p2)
       (belong w3 p8)
```

```
        (belong w4 p3)
        (belong w4 p8)
        (belong w5 p3)
        (belong w5 p6)
        (belong w6 p6)
        (belong w6 p4)
        (belong w7 p4)
        (belong w7 p5)
        (representative p1)
        (representative p2)
        (representative p3)
        (representative p4)
        (representative p5)
        (representative p6)
        (representative p7)
        (representative p8)
        (on w1 x)
        (on w2 y)
        (on w3 y)
        (on w4 z)
        (on w5 z)
        (on w6 z)
        (on w7 z))
  (:goal  (and  (representative  p1)(fitted  w1)(fitted  w2)(fitted  w3)(fitted
w4)(fitted w5)(fitted w6)(fitted w7))))
```

PDDL domain File for Pipe Spool Fabrication Sequencing Experiment 3

```
(define (domain pipespool)
  (:requirements :adl :equality :strips :typing :negative-preconditions )
```

```
  (:types assmbly wldpt axs clrnc);; assmbly, weldpoint, axis, axis type,
availabale clearance for the spool
  (:constants x y z - axs)
  (:predicates
   (belong ?w - wldpt ?p - assmbly)     ;; weld point belongs to assembly
   (welded ?w - wldpt)                  ;; weld point is welded
   (active ?p - assmbly)                ;; assembly is still active. when two
assemblies are welded only one of them remain active
   (on ?w - wldpt ?ax - axs)              ;; a weld point exists on an axis
type and axis number. e.g. x1, x2, .. or y1, y2 ...
   (equal ?p1 - assmbly ?p2 - assmbly)
  )
  (:functions
          (rfx ?p - assmbly)         ;; x,y coordinates for lowest corner of
an assembly
            (rfy ?p - assmbly)
            (lnx ?p - assmbly)         ;; x,y dimensions of assembly bounding
rectangle
            (lny ?p - assmbly)
            (avlblclrnc ?c - clrnc) ;; value of available clearance
            (dx ?w - wldpt)           ;; x,y coordinates for welding points
            (dy ?w - wldpt)
   )
  (:action roll-x-P1-P2-1
        :parameters (?p1 ?p2 - assmbly ?w1 ?w2 ?w3 - wldpt ?c - clrnc)
        :precondition (and (belong ?w1 ?p1)   ;; w1 belongs to p1 and p2
                           (belong ?w1 ?p2)
                               (not(equal ?p1 ?p2))
                               (not (belong ?w2 ?p1))
```

```
                                (not (belong ?w3 ?p1))
                            (active ?p1)       ;; p1 and p2 are active
                            (active ?p2)
                            (not(welded ?w1))  ;; w1 not welded yet
                        (on ?w1 x)       ;; w1 on given axis
                                (<= (rfx ?p1)(rfx ?p2))  ;; consider p1 as the
one closest to the origin (lowest x)
                            (<=  (-  (+  (lny  ?p1)  (rfy  ?p1))  (dy  ?w1))
(avlblclrnc ?c))  ;; difference between y coordinate of weld and
                            (<=  (-  (+  (lny  ?p2)  (rfy  ?p2))  (dy  ?w1))
(avlblclrnc ?c))  ;; max and min y limits of the bounding rectangle
                            (<=  (+  (rfx  ?p1)  (lnx  ?p1))  (+  (rfx  ?p2)
(lnx ?p2)))     ;; if max x limit of bounding box of p1 is less than p2
                             )
          :effect (and (welded ?w1)
                       (active ?p1)
                       (not(active ?p2))
                       (belong ?w2 ?p1)
                       (belong ?w3 ?p1)
                    (assign (lnx ?p1) (- (+ (rfx ?p2) (lnx ?p2)) (rfx ?p1)))) ;;
lenx p1 = refx p2 + lenx p2 - refx p1
  )
  (:action roll-x-P1-P2-2
          :parameters (?p1 ?p2 - assmbly ?w1 ?w2 ?w3 - wldpt ?c - clrnc)
          :precondition (and (belong ?w1 ?p1)   ;; w1 belongs to p1 and p2
                            (belong ?w1 ?p2)
                                (not(equal ?p1 ?p2))
                                (not (belong ?w2 ?p1))
                                (belong ?w3 ?p1)
```

```
                        (active ?p1)       ;; p1 and p2 are active
                        (active ?p2)
                        (not(welded ?w1))  ;; w1 not welded yet
                    (on ?w1 x)       ;; w1 on given axis
                            (<= (rfx ?p1)(rfx ?p2))  ;; consider p1 as the
one closest to the origin (lowest x)
                        (<=  (-  (+  (lny  ?p1)  (rfy  ?p1))  (dy  ?w1))
(avlblclrnc ?c))  ;; difference between y coordinate of weld and
                        (<=  (-  (+  (lny  ?p2)  (rfy  ?p2))  (dy  ?w1))
(avlblclrnc ?c))  ;; max and min y limits of the bounding rectangle
                        (<=  (+  (rfx  ?p1)  (lnx  ?p1))  (+  (rfx  ?p2)
(lnx ?p2)))     ;; if max x limit of bounding box of p1 is less than p2
                       )
        :effect (and (welded ?w1)
                    (active ?p1)
                    (not(active ?p2))
                    (belong ?w2 ?p1)
                (assign (lnx ?p1) (- (+ (rfx ?p2) (lnx ?p2)) (rfx ?p1)))) ;;
lenx p1 = refx p2 + lenx p2 - refx p1
  )
  (:action roll-x-P1-P2-3
        :parameters (?p1 ?p2 - assmbly ?w1 ?w2 ?w3 - wldpt ?c - clrnc)
        :precondition (and (belong ?w1 ?p1)   ;; w1 belongs to p1 and p2
                        (belong ?w1 ?p2)
                            (not(equal ?p1 ?p2))
                            (belong ?w2 ?p1)
                            (not (belong ?w3 ?p1))
                        (active ?p1)       ;; p1 and p2 are active
                        (active ?p2)
```

```
                        (not(welded ?w1))  ;; w1 not welded yet
                    (on ?w1 x)       ;; w1 on given axis
                            (<= (rfx ?p1)(rfx ?p2))  ;; consider p1 as the
one closest to the origin (lowest x)
                        (<=  (-  (+  (lny  ?p1)  (rfy  ?p1))  (dy  ?w1))
(avlblclrnc ?c))  ;; difference between y coordinate of weld and
                        (<=  (-  (+  (lny  ?p2)  (rfy  ?p2))  (dy  ?w1))
(avlblclrnc ?c))  ;; max and min y limits of the bounding rectangle
                        (<=  (+  (rfx  ?p1)  (lnx  ?p1))  (+  (rfx  ?p2)
(lnx ?p2)))     ;; if max x limit of bounding box of p1 is less than p2
                       )
        :effect (and (welded ?w1)
                    (active ?p1)
                    (not(active ?p2))
                    (belong ?w3 ?p1)
                (assign (lnx ?p1) (- (+ (rfx ?p2) (lnx ?p2)) (rfx ?p1)))) ;;
lenx p1 = refx p2 + lenx p2 - refx p1
  )
  (:action roll-x-P1-P2-4
        :parameters (?p1 ?p2 - assmbly ?w1 ?w2 ?w3 - wldpt ?c - clrnc)
        :precondition (and (belong ?w1 ?p1)   ;; w1 belongs to p1 and p2
                        (belong ?w1 ?p2)
                            (not(equal ?p1 ?p2))
                            (belong ?w2 ?p1)
                            (belong ?w3 ?p1)
                        (active ?p1)       ;; p1 and p2 are active
                        (active ?p2)
                        (not(welded ?w1))  ;; w1 not welded yet
                    (on ?w1 x)       ;; w1 on given axis
```

```
                              (<= (rfx ?p1)(rfx ?p2))  ;; consider p1 as the
one closest to the origin (lowest x)
                           (<=  (-  (+  (lny  ?p1)  (rfy  ?p1))  (dy  ?w1))
(avlblclrnc ?c))  ;; difference between y coordinate of weld and
                           (<=  (-  (+  (lny  ?p2)  (rfy  ?p2))  (dy  ?w1))
(avlblclrnc ?c))  ;; max and min y limits of the bounding rectangle
                           (<=  (+  (rfx  ?p1)  (lnx  ?p1))  (+  (rfx  ?p2)
(lnx ?p2)))     ;; if max x limit of bounding box of p1 is less than p2
                         )
        :effect (and (welded ?w1)
                     (active ?p1)
                     (not(active ?p2))
                     (assign  (lnx  ?p1)  (-  (+  (rfx  ?p2)  (lnx  ?p2))
(rfx ?p1)))) ;; lenx p1 = refx p2 + lenx p2 - refx p1
  )
  (:action roll-x-P2-P1-1
        :parameters (?p1 ?p2 - assmbly ?w1 ?w2 ?w3 - wldpt ?c - clrnc)
        :precondition (and (belong ?w1 ?p1)   ;; w1 belongs to p1 and p2
                           (belong ?w1 ?p2)
                           (not(equal ?p1 ?p2))
                           (not (belong ?w2 ?p2))
                           (not (belong ?w3 ?p2))
                           (active ?p1)       ;; p1 and p2 are active
                           (active ?p2)
                           (not(welded ?w1))  ;; w1 not welded yet
                     (on ?w1 x)       ;; w1 on given axis
                           (<= (rfx ?p1)(rfx ?p2))  ;; consider p1 as the one
closest to the origin (lowest x)
```

```
                          (<=   (-   (+   (lny   ?p1)   (rfy   ?p1))   (dy   ?w1))
(avlblclrnc ?c))  ;; difference between y coordinate of weld and
                          (<=   (-   (+   (lny   ?p2)   (rfy   ?p2))   (dy   ?w1))
(avlblclrnc ?c))  ;; max and min y limits of the bounding rectangle
                          (<=   (+   (rfx   ?p2)   (lnx   ?p2))   (+   (rfx   ?p1)
(lnx ?p1)))     ;; if max x limit of bounding box of p1 is less than p2
                         )
        :effect (and (welded ?w1)
                  (active ?p1)
                  (not(active ?p2))
                  (belong ?w2 ?p1)
                  (belong ?w3 ?p1)
  ))
  (:action roll-x-P2-P1-2
        :parameters (?p1 ?p2 - assmbly ?w1 ?w2 ?w3 - wldpt ?c - clrnc)
        :precondition (and (belong ?w1 ?p1)   ;; w1 belongs to p1 and p2
                           (belong ?w1 ?p2)
                         (not(equal ?p1 ?p2))
                           (not (belong ?w2 ?p2))
                         (belong ?w3 ?p2)
                         (active ?p1)       ;; p1 and p2 are active
                         (active ?p2)
                         (not(welded ?w1))  ;; w1 not welded yet
                  (on ?w1 x)       ;; w1 on given axis
                         (<= (rfx ?p1)(rfx ?p2))   ;; consider p1 as the one
closest to the origin (lowest x)
                         (<=   (-   (+   (lny   ?p1)   (rfy   ?p1))   (dy   ?w1))
(avlblclrnc ?c))  ;; difference between y coordinate of weld and
```

```
                        (<=   (-   (+   (lny   ?p2)   (rfy   ?p2))   (dy   ?w1))
(avlblclrnc ?c))  ;; max and min y limits of the bounding rectangle
                        (<=   (+   (rfx   ?p2)   (lnx   ?p2))   (+   (rfx   ?p1)
(lnx ?p1)))     ;; if max x limit of bounding box of p1 is less than p2
                        )
        :effect (and (welded ?w1)
                  (active ?p1)
                  (not(active ?p2))
                  (belong ?w2 ?p1)
  ))
  (:action roll-x-P2-P1-3
        :parameters (?p1 ?p2 - assmbly ?w1 ?w2 ?w3 - wldpt ?c - clrnc)
        :precondition (and (belong ?w1 ?p1)   ;; w1 belongs to p1 and p2
                        (belong ?w1 ?p2)
                        (not(equal ?p1 ?p2))
                        (belong ?w2 ?p2)
                        (not (belong ?w3 ?p2))
                        (active ?p1)       ;; p1 and p2 are active
                        (active ?p2)
                        (not(welded ?w1))  ;; w1 not welded yet
                  (on ?w1 x)       ;; w1 on given axis
                        (<= (rfx ?p1)(rfx ?p2))   ;; consider p1 as the one
closest to the origin (lowest x)
                        (<=   (-   (+   (lny   ?p1)   (rfy   ?p1))   (dy   ?w1))
(avlblclrnc ?c))  ;; difference between y coordinate of weld and
                        (<=   (-   (+   (lny   ?p2)   (rfy   ?p2))   (dy   ?w1))
(avlblclrnc ?c))  ;; max and min y limits of the bounding rectangle
                        (<=   (+   (rfx   ?p2)   (lnx   ?p2))   (+   (rfx   ?p1)
(lnx ?p1)))     ;; if max x limit of bounding box of p1 is less than p2
```

```
                        )
        :effect (and (welded ?w1)
                     (active ?p1)
                     (not(active ?p2))
                     (belong ?w3 ?p1)
  ))
  (:action roll-x-P2-P1-4
         :parameters (?p1 ?p2 - assmbly ?w1 ?w2 ?w3 - wldpt ?c - clrnc)
         :precondition (and (belong ?w1 ?p1)   ;; w1 belongs to p1 and p2
                            (belong ?w1 ?p2)
                           (not(equal ?p1 ?p2))
                            (belong ?w2 ?p2)
                           (belong ?w3 ?p2)
                           (active ?p1)       ;; p1 and p2 are active
                           (active ?p2)
                           (not(welded ?w1))  ;; w1 not welded yet
                     (on ?w1 x)       ;; w1 on given axis
                           (<= (rfx ?p1)(rfx ?p2))   ;; consider p1 as the one
closest to the origin (lowest x)
                           (<=   (-   (+   (lny   ?p1)   (rfy   ?p1))   (dy   ?w1))
(avlblclrnc ?c))  ;; difference between y coordinate of weld and
                           (<=   (-   (+   (lny   ?p2)   (rfy   ?p2))   (dy   ?w1))
(avlblclrnc ?c))  ;; max and min y limits of the bounding rectangle
                           (<=   (+   (rfx   ?p2)   (lnx   ?p2))   (+   (rfx   ?p1)
(lnx ?p1)))     ;; if max x limit of bounding box of p1 is less than p2
                           )
         :effect (and (welded ?w1)
                     (active ?p1)
                     (not(active ?p2))
```

```
  ))
)
```

PDDL problem File for Pipe Spool Fabrication Sequencing Experiment 3

```
(define (problem pipespool1)
  (:domain pipespool)
  (:requirements :typing :fluents :adl :equality)
  (:objects p1 p2 - assmbly w1 w2 w3 - wldpt c - clrnc)
  (:init (not(welded w1))
            (not(welded w2))
            (not(welded w3))
          (belong w1 p1)
          (belong w1 p2)
            (belong w2 p2)
            (belong w3 p2)
            (not (belong w2 p1))
            (not (belong w3 p1))
          (active p1)
          (active p2)
          (on w1 x)
            (on w2 x)
            (on w3 y)
        (= (rfx p1) 0)
          (= (rfy p1) 0)
          (= (lnx p1) 400)
          (= (lny p1) 0)
          (= (rfx p2) 400)
          (= (rfy p2) 0)
          (= (lnx p2) 220)
```

```
            (= (lny p2) 920)
            (= (avlblclrnc c) 1200)
            (= (dx w1) 114)
            (= (dy w1) 0)
              (= (dx w2) 506)
            (= (dy w2) 0)
              (= (dx w3) 620)
            (= (dy w3) 114)
            (equal p1 p1)
            (equal p2 p2)
  )
  (:goal (and (welded w1)(welded w2)(welded w3)))
)
```

PDDL domain File for module installation Experiment 1 and Experiment 2

```
(define (domain modulesequencing)
  (:requirements :conditional-effects :equality :strips :typing :negative-
preconditions :fluents :disjunctive-preconditions)
  (:types module) ;; module
  (:constants x y z - axs)
  (:predicates
   (placed ?m - module)     ;; module is placed in its final location
   (base ?m - module)       ;; base module (bottom one)
   (AdjZ ?m ?m0 - module)  ;; module m is on top of m0
   (AdjX ?m ?m0 - module) ;; module m is to the Left of m0
   (AdjY ?m ?m0 - module);; module m is in front of m0
   )
  (:action place_1
         :parameters (?m ?m2 - module)
```

```
        :precondition (and (base ?m)  ;; or m is a base module
                           (and(not (=?m ?m2))(AdjX ?m ?m2))
                           (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
                    )
        :effect (and (placed ?m))
   )
   (:action place_2
        :parameters (?m ?m1 ?m2 - module)
       :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m2))(AdjX ?m ?m2))
                           (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
                    )
        :effect (and (placed ?m))
   )
   (:action place_3
        :parameters (?m ?m3 - module)
        :precondition (and (base ?m)  ;; or m is a base module
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
```

```
                    )

        :effect (and (placed ?m))
    )
    (:action place_4
        :parameters (?m ?m1 ?m3 - module)
       :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
                    )
        :effect (and (placed ?m))
    )
    (:action place_5
        :parameters (?m ?m2 ?m3 - module)
        :precondition (and (base ?m)  ;; or m is a base module
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (and(not (=?m ?m2))(AdjX ?m ?m2))
                           (or(not(placed ?m2))(not(placed ?m3)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                         (not(placed ?m))  ;; m is not placed
                    )
        :effect (and (placed ?m))
    )
    (:action place_6
```

```
         :parameters (?m ?m1 ?m2 ?m3 - module)
        :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (and(not (=?m ?m2))(AdjX ?m ?m2))
                           (or(not(placed ?m2))(not(placed ?m3)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                       (not(placed ?m))  ;; m is not placed
                    )
         :effect (and (placed ?m))
   )
  )
```

PDDL problem File for module installation Experiment 2 (Experiment 1 is slightly different)

```
(define (problem cubeOf9modules)
  (:domain modulesequencing)
  (:requirements :typing :fluents)
  ;; m z,x,y z(AdjZ), x(AdjX), y(AdjY)
  (:objects m014A m014B m007A m007B m007C m006A m006B PR001 m005A m005B PR002
- module)
  (:init (base m014A)
       (base m007A)
       (base m006A)
       (base m005A)
       (AdjZ m014B m014A)
       (AdjZ m007B m007A)
       (AdjZ m007C m007B)
       (AdjZ m006B m006A)
```

```
        (AdjZ PR001 m006B)
        (AdjZ m005B m005A)
        (AdjZ PR002 m005B)
        (AdjX m014A m007A)
        (AdjX m014B m007B)
        (AdjX m007A m006A)
        (AdjX m007B m006B)
        (AdjX m007C PR001)
        (AdjX m006A m005A)
        (AdjX m006B m005B)
        (AdjX PR001 PR002)
        (placed m006A)
        (placed m006B)
    )

  (:goal (and (placed m014A)(placed m014B)(placed m007A)(placed m007B)(placed
m007C)(placed   m006A)(placed   m006B)(placed   PR001)(placed   m005A)(placed
m005B)(placed PR002)))
 )
```

PDDL domain File for module installation Experiment 3, Experiment 4 and Experiment 5

```
(define (domain modulesequencing)
  (:requirements :conditional-effects :equality :strips :typing :negative-
preconditions :fluents :disjunctive-preconditions)
  (:types module) ;; module
  (:constants x y z - axs)
  (:predicates
   (placed ?m - module)     ;; module is placed in its final location
   (base ?m - module)       ;; base module (bottom one)
```

```
    (AdjZ ?m ?m0 - module)  ;; module m is on top of m0
    (AdjX ?m ?m0 - module) ;; module m is to the Left of m0
    (AdjY ?m ?m0 - module);; module m is in front of m0
    )
   (:action place_1
          :parameters (?m ?m2 - module)
          :precondition (and (base ?m)  ;; m is a base module
                             (and(not (=?m ?m2))(AdjX ?m ?m2))
                             (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                             (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                             (not (exists (?m4 - module)(AdjY ?m ?m4)))
                            (not(placed ?m))  ;; m is not placed
                      )
          :effect (and (placed ?m))
    )
    (:action place_2
          :parameters (?m ?m1 ?m2 - module)
         :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                             (and(not (=?m ?m2))(AdjX ?m ?m2))
                             (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                             (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                             (not (exists (?m4 - module)(AdjY ?m ?m4)))
                            (not(placed ?m))  ;; m is not placed
                      )
          :effect (and (placed ?m))
    )
    (:action place_3
          :parameters (?m ?m3 - module)
```

```
        :precondition (and (base ?m)  ;; m is a base module
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
                    )
        :effect (and (placed ?m))
   )
   (:action place_4
        :parameters (?m ?m1 ?m3 - module)
       :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
                    )
        :effect (and (placed ?m))
   )
   (:action place_5
        :parameters (?m ?m2 ?m3 - module)
        :precondition (and (base ?m)  ;; m is a base module
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (and(not (=?m ?m2))(AdjX ?m ?m2))
                           (or(not(placed ?m2))(not(placed ?m3)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
```

```
                        (not(placed ?m))  ;; m is not placed
                    )

        :effect (and (placed ?m))
    )
    (:action place_6
        :parameters (?m ?m1 ?m2 ?m3 - module)
       :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                          (and(not (=?m ?m3))(AdjX ?m3 ?m))
                          (and(not (=?m ?m2))(AdjX ?m ?m2))
                          (or(not(placed ?m2))(not(placed ?m3)))
                          (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                          (not (exists (?m4 - module)(AdjY ?m ?m4)))
                        (not(placed ?m))  ;; m is not placed
                    )
        :effect (and (placed ?m))
    )
    (:action place_7
        :parameters (?m ?m4 - module)
        :precondition (and (base ?m)  ;; or m is a base module
                          (and(not (=?m ?m4))(AdjY ?m ?m4))
                          (not (exists (?m2 - module)(AdjX ?m ?m2)))
                          (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                          (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                        (not(placed ?m))  ;; m is not placed
                    )
        :effect (and (placed ?m))
    )
```

```
   (:action place_8
         :parameters (?m ?m1 ?m4 - module)
        :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m4))(AdjY ?m ?m4))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                          (not(placed ?m))  ;; m is not placed
                    )
         :effect (and (placed ?m))
   )
   (:action place_9
         :parameters (?m ?m5 - module)
         :precondition (and (base ?m)  ;; m is a base module
                           (and(not (=?m ?m5))(AdjY ?m5 ?m))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
                    )
         :effect (and (placed ?m))
   )
   (:action place_10
         :parameters (?m ?m1 ?m5 - module)
        :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m5))(AdjY ?m5 ?m))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
```

```
                        (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                        (not (exists (?m4 - module)(AdjY ?m ?m4)))
                      (not(placed ?m))  ;; m is not placed
                )
        :effect (and (placed ?m))
    )
    (:action place_11
        :parameters (?m ?m4 ?m5 - module)
        :precondition (and (base ?m)  ;; m is a base module
                        (and(not (=?m ?m5))(AdjY ?m5 ?m))
                        (and(not (=?m ?m4))(AdjY ?m ?m4))
                        (or(not(placed ?m4))(not(placed ?m5)))
                        (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                        (not (exists (?m2 - module)(AdjX ?m ?m2)))
                      (not(placed ?m))  ;; m is not placed
                )
        :effect (and (placed ?m))
    )
    (:action place_12
        :parameters (?m ?m1 ?m4 ?m5 - module)
       :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                        (and(not (=?m ?m5))(AdjY ?m5 ?m))
                        (and(not (=?m ?m4))(AdjY ?m ?m4))
                        (or(not(placed ?m4))(not(placed ?m5)))
                        (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                        (not (exists (?m2 - module)(AdjX ?m ?m2)))
                      (not(placed ?m))  ;; m is not placed
                )
```

```
        :effect (and (placed ?m))
    )
    (:action place_13
        :parameters (?m ?m3 ?m4 ?m5 - module)
        :precondition (and (base ?m)  ;; m is a base module
                           (and(not (=?m ?m5))(AdjY ?m5 ?m))
                           (and(not (=?m ?m4))(AdjY ?m ?m4))
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (or(not(placed ?m4))(not(placed ?m5)))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                      (not(placed ?m))  ;; m is not placed
                  )

        :effect (and (placed ?m))
    )
    (:action place_14
        :parameters (?m ?m1 ?m3 ?m4 ?m5 - module)
       :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m5))(AdjY ?m5 ?m))
                           (and(not (=?m ?m4))(AdjY ?m ?m4))
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (or(not(placed ?m4))(not(placed ?m5)))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                      (not(placed ?m))  ;; m is not placed
                  )
        :effect (and (placed ?m))
    )
```

```
   (:action place_15
         :parameters (?m ?m1 ?m3 ?m4 - module)
        :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m4))(AdjY ?m ?m4))
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                         (not(placed ?m))  ;; m is not placed
                    )

         :effect (and (placed ?m))
   )
  )
```

PDDL problem File for module installation Experiment 5 (Experiment 3 and Experiment 4 are slightly different)

```
(define (problem cubeOf9modules)
  (:domain modulesequencing)
  (:requirements :typing :fluents)
  ;; m z,x,y z(AdjZ), x(AdjX), y(AdjY)
  (:objects m014A m014B m007A m007B m007C m006A m006B PR001 m005A m005B PR002
m011A m011B m012A m012B m012C m013A m013B m013C m004A m004B m143A m143B -
module)
  (:init (base m014A)
       (base m007A)
       (base m006A)
       (base m005A)
       (base m011A)
```

(base m012A)

(base m013A)

(base m004A)

(base m143A)

(AdjZ m014B m014A)

(AdjZ m007B m007A)

(AdjZ m007C m007B)

(AdjZ m006B m006A)

(AdjZ PR001 m006B)

(AdjZ m005B m005A)

(AdjZ PR002 m005B)

(AdjZ m011B m011A)

(AdjZ m012B m012A)

(AdjZ m012C m012B)

(AdjZ m013B m013A)

(AdjZ m013C m013B)

(AdjZ m004B m004A)

(AdjZ m143B m143A)

(AdjX m014A m007A)

(AdjX m014B m007B)

(AdjX m007A m006A)

(AdjX m007B m006B)

(AdjX m007C PR001)

(AdjX m006A m005A)

(AdjX m006B m005B)

(AdjX PR001 PR002)

(AdjX m005A m012A)

(AdjX m005B m012B)

(AdjX PR002 m012C)

```
        (AdjY m011A m012A)
        (AdjY m011B m012B)
        (AdjY m012A m013A)
        (AdjY m012B m013B)
        (AdjY m012C m013C)
        (AdjY m013A m004A)
        (AdjY m013B m004B)
        (AdjY m004A m143A)
        (AdjY m004B m143B)
        (placed m006A)
        (placed m006B)
        (placed m013A)
        (placed m013B)
    )

  (:goal (and (placed m014A)(placed m014B)(placed m007A)(placed m007B)(placed
m007C)(placed   m006A)(placed   m006B)(placed   PR001)(placed   m005A)(placed
m005B)(placed   PR002)(placed   m011A)(placed   m011B)(placed   m012A)(placed
m012B)(placed   m012C)(placed   m013A)(placed   m013B)(placed   m013C)(placed
m004A)(placed m004B)(placed m143A)(placed m143B)))
 )
```

PDDL domain File for module installation Experiment 6, Experiment 7 and Experiment 8

```
(define (domain modulesequencing)
  (:requirements :conditional-effects :equality :strips :typing :negative-
preconditions :fluents :disjunctive-preconditions)
  (:types module) ;; module
  (:constants x y z - axs)
  (:predicates
```

```
    (placed ?m - module)     ;; module is placed in its final location
    (base ?m - module)       ;; base module (bottom one)
    (AdjZ ?m ?m0 - module)  ;; module m is on top of m0
    (AdjX ?m ?m0 - module) ;; module m is to the Left of m0
    (AdjY ?m ?m0 - module);; module m is in front of m0
   )
    (:action place_0
          :parameters (?m ?m1 - module)
         :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                            (not (exists (?m2 - module)(AdjX ?m ?m2)))
                            (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                            (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                            (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
                     )
          :effect (and (placed ?m))
    )
   (:action place_1
          :parameters (?m ?m2 - module)
          :precondition (and (base ?m)  ;; m is a base module
                            (and(not (=?m ?m2))(AdjX ?m ?m2))
                            (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                            (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                            (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
                     )
          :effect (and (placed ?m))
    )
```

```
   (:action place_2
         :parameters (?m ?m1 ?m2 - module)
        :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m2))(AdjX ?m ?m2))
                           (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
                    )
         :effect (and (placed ?m))
   )
   (:action place_3
         :parameters (?m ?m3 - module)
         :precondition (and (base ?m)  ;; m is a base module
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                           (not (exists (?m4 - module)(AdjY ?m ?m4)))
                          (not(placed ?m))  ;; m is not placed
                    )
         :effect (and (placed ?m))
   )
   (:action place_4
         :parameters (?m ?m1 ?m3 - module)
        :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
```

```
                        (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                        (not (exists (?m4 - module)(AdjY ?m ?m4)))
                      (not(placed ?m))  ;; m is not placed
                )
        :effect (and (placed ?m))
    )
    (:action place_5
        :parameters (?m ?m2 ?m3 - module)
        :precondition (and (base ?m)  ;; m is a base module
                        (and(not (=?m ?m3))(AdjX ?m3 ?m))
                        (and(not (=?m ?m2))(AdjX ?m ?m2))
                        (or(not(placed ?m2))(not(placed ?m3)))
                        (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                        (not (exists (?m4 - module)(AdjY ?m ?m4)))
                      (not(placed ?m))  ;; m is not placed
                )
        :effect (and (placed ?m))
    )
    (:action place_6
        :parameters (?m ?m1 ?m2 ?m3 - module)
       :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                        (and(not (=?m ?m3))(AdjX ?m3 ?m))
                        (and(not (=?m ?m2))(AdjX ?m ?m2))
                        (or(not(placed ?m2))(not(placed ?m3)))
                        (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                        (not (exists (?m4 - module)(AdjY ?m ?m4)))
                      (not(placed ?m))  ;; m is not placed
                )
```

```
        :effect (and (placed ?m))
    )
    (:action place_7
        :parameters (?m ?m4 - module)
        :precondition (and (base ?m)  ;; or m is a base module
                           (and(not (=?m ?m4))(AdjY ?m ?m4))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                          (not(placed ?m))  ;; m is not placed
                    )
        :effect (and (placed ?m)))
    (:action place_8
        :parameters (?m ?m1 ?m4 - module)
       :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m4))(AdjY ?m ?m4))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                          (not(placed ?m))  ;; m is not placed
                    )

        :effect (and (placed ?m))
    )
    (:action place_9
        :parameters (?m ?m5 - module)
        :precondition (and (base ?m)  ;; m is a base module
                           (and(not (=?m ?m5))(AdjY ?m5 ?m))
```

```
                        (not (exists (?m2 - module)(AdjX ?m ?m2)))
                        (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                        (not (exists (?m4 - module)(AdjY ?m ?m4)))
                      (not(placed ?m))  ;; m is not placed
                )
        :effect (and (placed ?m))
   )
   (:action place_10
        :parameters (?m ?m1 ?m5 - module)
       :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                        (and(not (=?m ?m5))(AdjY ?m5 ?m))
                        (not (exists (?m2 - module)(AdjX ?m ?m2)))
                        (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                        (not (exists (?m4 - module)(AdjY ?m ?m4)))
                      (not(placed ?m))  ;; m is not placed
                )
        :effect (and (placed ?m))
   )
   (:action place_11
        :parameters (?m ?m4 ?m5 - module)
        :precondition (and (base ?m)  ;; m is a base module
                        (and(not (=?m ?m5))(AdjY ?m5 ?m))
                        (and(not (=?m ?m4))(AdjY ?m ?m4))
                        (or(not(placed ?m4))(not(placed ?m5)))
                        (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                        (not (exists (?m2 - module)(AdjX ?m ?m2)))
                      (not(placed ?m))  ;; m is not placed
                )
```

```
        :effect (and (placed ?m))
    )
    (:action place_12
        :parameters (?m ?m1 ?m4 ?m5 - module)
       :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m5))(AdjY ?m5 ?m))
                           (and(not (=?m ?m4))(AdjY ?m ?m4))
                           (or(not(placed ?m4))(not(placed ?m5)))
                           (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                      (not(placed ?m))  ;; m is not placed
                  )
        :effect (and (placed ?m))
    )
    (:action place_13
        :parameters (?m ?m3 ?m4 ?m5 - module)
        :precondition (and (base ?m)  ;; m is a base module
                           (and(not (=?m ?m5))(AdjY ?m5 ?m))
                           (and(not (=?m ?m4))(AdjY ?m ?m4))
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (or(not(placed ?m4))(not(placed ?m5)))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                      (not(placed ?m))  ;; m is not placed
                  )
        :effect (and (placed ?m))
    )
    (:action place_14
        :parameters (?m ?m1 ?m3 ?m4 ?m5 - module)
```

```
        :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m5))(AdjY ?m5 ?m))
                           (and(not (=?m ?m4))(AdjY ?m ?m4))
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (or(not(placed ?m4))(not(placed ?m5)))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                         (not(placed ?m))  ;; m is not placed
                    )
         :effect (and (placed ?m))
   )
   (:action place_15
         :parameters (?m ?m1 ?m3 ?m4 - module)
        :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                           (and(not (=?m ?m4))(AdjY ?m ?m4))
                           (and(not (=?m ?m3))(AdjX ?m3 ?m))
                           (not (exists (?m2 - module)(AdjX ?m ?m2)))
                           (not (exists (?m5 - module)(AdjY ?m5 ?m)))
                         (not(placed ?m))  ;; m is not placed
                    )

         :effect (and (placed ?m))
   )
   (:action place_16
         :parameters (?m ?m2 ?m5 - module)
         :precondition (and (base ?m)  ;; m is a base module
                           (and(not (=?m ?m5))(AdjY ?m5 ?m))
                           (and(not (=?m ?m2))(AdjX ?m ?m2))
```

```
                            (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                            (not (exists (?m4 - module)(AdjY ?m ?m4)))
                           (not(placed ?m))  ;; m is not placed
                     )
         :effect (and (placed ?m)
               )
   )
   (:action place_17
         :parameters (?m ?m1 ?m2 ?m5 - module)
        :precondition (and (and (AdjZ ?m ?m1)(not (=?m ?m1))(placed ?m1))  ;;
m is on top of ?m1
                            (and(not (=?m ?m5))(AdjY ?m5 ?m))
                            (and(not (=?m ?m2))(AdjX ?m ?m2))
                            (not (exists (?m3 - module)(AdjX ?m3 ?m)))
                            (not (exists (?m4 - module)(AdjY ?m ?m4)))
                           (not(placed ?m))  ;; m is not placed
                     )
         :effect (and (placed ?m)))
  )
```

PDDL domain File for module installation Experiment 8 (Experiment 6 and Experiment 7 are slightly different)

```
(define (problem cubeOf9modules)
  (:domain modulesequencing)
  (:requirements :typing :fluents)
  ;; m z,x,y z(AdjZ), x(AdjX), y(AdjY)
  (:objects m014A m014B m007A m007B m007C m006A m006B PR001 m005A m005B PR002
m011A m011B m012A m012B m012C m013A m013B m013C m004A m004B m143A m143B PR112
PR113 PR114 PR115 - module)
```

```
(:init (base m014A)
     (base m007A)
     (base m006A)
     (base m005A)
     (base m011A)
     (base m012A)
     (base m013A)
     (base m004A)
     (base m143A)
     (base PR112)
     (base PR113)
     (base PR114)
     (AdjZ m014B m014A)
     (AdjZ m007B m007A)
     (AdjZ m007C m007B)
     (AdjZ m006B m006A)
     (AdjZ PR001 m006B)
     (AdjZ m005B m005A)
     (AdjZ PR002 m005B)
     (AdjZ m011B m011A)
     (AdjZ m012B m012A)
     (AdjZ m012C m012B)
     (AdjZ m013B m013A)
     (AdjZ m013C m013B)
     (AdjZ m004B m004A)
     (AdjZ m143B m143A)
     (AdjZ PR115 PR114)
     (AdjX m014A m007A)
     (AdjX m014B m007B)
```

```
      (AdjX m007A m006A)
      (AdjX m007B m006B)
      (AdjX m007C PR001)
      (AdjX m006A m005A)
      (AdjX m006B m005B)
      (AdjX PR001 PR002)
      (AdjX m005A m012A)
      (AdjX m005B m012B)
      (AdjX PR002 m012C)
      (AdjX m143A PR112)
      (AdjX PR112 PR113)
      (AdjX PR113 PR114)
      (AdjY m011A m012A)
      (AdjY m011B m012B)
      (AdjY m012A m013A)
      (AdjY m012B m013B)
      (AdjY m012C m013C)
      (AdjY m013A m004A)
      (AdjY m013B m004B)
      (AdjY m004A m143A)
      (AdjY m004B m143B)
      (placed m005A)
      (placed m005B)
      (placed m004A)
      (placed m004B)
      (placed PR113)
   )
  (:goal (and (placed m014A)(placed m014B)(placed m007A)(placed m007B)(placed
m007C)(placed   m006A)(placed   m006B)(placed   PR001)(placed   m005A)(placed
```

```
m005B)(placed    PR002)(placed    m011A)(placed    m011B)(placed    m012A)(placed
m012B)(placed    m012C)(placed    m013A)(placed    m013B)(placed    m013C)(placed
m004A)(placed    m004B)(placed    m143A)(placed    m143B)(placed    PR112)(placed
PR113)(placed PR114)(placed PR115)))
 )
```