A Methodology for the Automated Creation of Construction Simulation Models

by

Ramzi Roy Labban

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Construction Engineering and Management

Department of Civil and Environmental Engineering
University of Alberta

Abstract

Although much academic research has been performed in the study of construction process simulation, this research has not seen mainstream application in the construction industry. Many factors play a deterrent role in the adoption of simulation by construction end users, mainly the time, cost and skills required to utilize simulation as a viable tool to analyze construction operations. In addition to construction domain expertise, simulation modeling and development skills are required to build a simulation model. The modeling process is the most difficult and time consuming part of the process of building and utilizing a simulation model. The time, effort and technical expertise required to build and experiment with a simulation model balanced against the uniqueness and relatively short life cycle of a construction project is what leads to the slow adoption of simulation by the construction industry.

The objective of this research is to make construction simulation more accessible to construction domain expert end users by reducing the modeling effort required for building construction simulation models. It aims at doing so by developing a methodology which will allow construction end users to rapidly build simulation models using information they are familiar with and use as part of their work. The proposed methodology describes the product, process and environment definitions that describe a construction operation with the purpose of constructing a simulation model to mimic it. It also describes the algorithms and programming required to build a discrete event simulation model compiler that would use the provided product, process and environment definitions to compile a DES model of the described operation, run the model, and produce simulation run results.

The methodology will help simulation practitioners develop systems composed of data structures which hold model descriptive information and simulation run result sets, and a DES model compiler program which will compile the model and execute it for the user. Three case studies of actual construction project simulators are examined to establish the commonalities in building construction simulation models. The commonalities found are then used to describe the different components of the methodology. The methodology is prototyped in a proof of concept setting and then applied to rebuild one of the case study simulation models using the prototype system. The methodology is then applied to build an enterprise level production version system using the methodology. The production version is then utilized to redevelop the same case study which was rebuilt using the prototype, and to build a simulation model for a new construction operation.

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to the many people who have helped and inspired me during my doctoral studies at the University of Alberta. Foremost, I would like to express my respect and appreciation to my advisor, Dr. Simaan M. AbouRizk, for his endless patience, encouragement, support and guidance. It was a great privilege to work under his the supervision and experience his immense wisdom and knowledge, invaluable advice and deep insightful guidance throughout my research. I am also very grateful to Dr. H. James Hoover whose wisdom, knowledge and direction were crucial to my research. I would also like to thank Dr. Yasser Mohammed for his constructive comments, guidance and valuable suggestions. I also wish to express my gratitude to members of my doctoral committee, namely, Dr. Mohammed Al-Hussein, Dr. Chrysostomos D. Stylios, and Dr. Zhijiun Qiu for their valuable suggestions. I extend my heartfelt thanks to all my colleagues and the faculty and staff in the Construction Engineering and Management group for their advice and support and would like to thank, in particular, Ms. Brenda Penner, Mr. Stephen Hague, and Ms. Amy Carter for all their insights and assistance throughout. I also would like to express my thanks and gratitude to Mr. Zuhair Haddad and Dr. Amr El-Sersy from CCC for their continued encouragement and support throughout my doctoral studies.

Last, but certainly not least, I would like to thank my lovely wife, Solange, and my twin girls, Alexia and Jaimie, for their love, patience, understanding and unrelenting support for me during this journey.

Table of Contents

1 Introduction 1

1.1 Background and Problem Statement 1

1.2 Research Objectives 2

1.3 Anticipated Contributions 3

1.4 Research Methodology 3

1.5 Research Scope 5

1.6 Thesis Organization 6

2 Literature Review 7

2.1 Introduction 7

2.2 Application of Simulation in the Construction Industry 7

2.3 Deterrents to Adoption of Simulation in Construction 9

2.4 Construction Simulation Modeling in Simphony 12

3 Three Construction Simulation Models 14

3.1 Introduction 14

3.2 Pipe Spool Fabrication Model 14

3.2.1 Background 14

3.2.2 Simulator Design and Development 15

3.2.3 Product Definitions 16

3.2.4 Process Definitions 16

3.2.5 Resource Definitions 18

3.2.6 Model Structure 19

3.2.7 Simulation Outputs 21

3.3 Asphalt Paving Simulation Model 22

3.3.1 Background 22

3.3.2 Simulator Design and Development 23

3.3.3 Product Definitions 23

3.3.4 Process Definitions 24

3.3.5 Model Structure 27

3.3.6 Simulation Outputs 29

3.4 Pipeline Construction Simulation Model 31

3.4.1 Background 31

3.4.2 Simulator Design and Development 31

3.4.3 Product Definitions 32
3.4.4 Process Definitions 32
3.4.5 Resource Definitions 34
3.4.6 Model Structure 35
3.4.7 Simulation Outputs 36
3.5 Verification and Validation 37
3.6 Conclusion 38
4 Construction Simulation Model Commonalities 39
4.1 Introduction 39
4.2 Product 39
4.2.1 Product Hierarchy 40
4.3 Process Definition 42
4.3.1 Template of Tasks 42
4.3.2 Resources 45
4.3.3 Product-Process Mappings 47
4.3.4 Task Duration Calculation 49
4.4 Environment 51
4.5 DES Model 52
4.5.1 DES Model Structure 52
4.5.2 DES Model Development Requirements 55
4.6 Artificial History Produced by Simulation Run 55
4.7 Conclusion 58
5 Automated DES Model Creation Methodology 60
5.1 Introduction 60
5.2 Automated DES Model Creation Methodology 63
5.2.1 Data Structure Classes 65
5.2.2 Product 65
5.2.3 Task Template 66
5.2.4 Resource 68
5.2.5 Product-Task-Level Mapping 69
5.2.6 Product-Task-Quantity Mapping 69
5.2.7 Result Set 70
5.2.8 Basic DES Model Class 70

5.2.9 Algorithms 71
5.3 Methodology Prototype 76
5.3.1 Simple Construction Example 76
5.3.2 Prototype Development 79
5.3.3 Redevelopment of Pipe Spool Fabrication Model 85
5.4 Discussion 90
5.5 Conclusion 91
6 Implementing the Methodology 93
6.1 Introduction 93
6.2 DES Model Compiler System Development 93
6.2.1 DES Compiler 94
6.2.2 Data Structures 97
6.3 Redevelopment of the Pipe Spool Fabrication Model 99
6.3.1 Product Definitions 99
6.3.2 Task Definitions 100
6.3.3 Sequence 100
6.3.4 Resources 101
6.3.5 Product-Task-Level Mapping 101
6.3.6 Product-Task-Quantity Mapping 102
6.3.7 Result Set 102
6.4 Development of the Building Finishes Model 103
6.4.1 Product Definitions 104
6.4.2 Task Definitions 104
6.4.3 Sequence 104
6.4.4 Resources 105
6.4.5 Product-Task-Level Mapping 105
6.4.6 Product-Task-Qty Mapping 106
6.4.7 Result Set 106
6.5 Verification and Validation of the Models 107
6.6 Conclusion 108
7 Conclusion 111
7.1 Research Contributions 111
7.2 Limitations 113

7.3 Recommendations for Future Research and Development 114
References .....**Error! Bookmark not defined.**

List of Tables

Table 3.1: Sample Spool-Weld Hierarchy 16
Table 3.2: Typical Crew Compositions 19
Table 3.3: Typical Worker Availability over Time 19
Table 3.4: Sample Simulation Outputs 22
Table 3.5: Sample Crew Compositions 34
Table 3.6: Typical Worker Availability over Time 34
Table 3.7: Sample Simulation Outputs 37
Table 4.1: Pipe Spool Fabrication Simulator – Tasks 44
Table 4.2: Pipeline Construction Simulator – Tasks 44
Table 4.3: Asphalt Paving Simulator – Tasks 44
Table 4.4: Asphalt Paving Simulator - Resource List 46
Table 4.5: Pipe Spool Fabrication Simulator - Resource List 46
Table 4.6: Pipeline Construction Simulator - Resource List 47
Table 4.7: Pipe Spool Fabrication Simulator – Spool-Task Level Mapping 48
Table 4.8: Pipeline Construction Simulator – Section-Task Level Mapping 48
Table 4.9: Asphalt Paving Simulator – Section-Task Level Mapping 48
Table 4.10: Pipe Spool Fabrication Simulator – Spool-Task Quantity Mapping 49
Table 4.11: Pipeline Construction Simulator – Section-Task Quantity Mapping 49
Table 4.12: Asphalt Paving Simulator – Section-Task Quantity Mapping 49
Table 4.13: Pipe Spool Fabrication – Artificial History Output Sample 57
Table 4.14: Summary of Findings 59
Table 5.1: Product Definition 77
Table 5.2: Task Definition 77
Table 5.3: Sequence Definitions 77
Table 5.4: Resource Definitions 78
Table 5.5: Resource Availability Definitions 78
Table 5.6: Product-Task-Level Mapping 78
Table 5.7: Product-Task-Qty Mapping 78
Table 5.8: Product List showing Product Name and Nominal Quantity 79
Table 5.9: Task List showing Task name, expected crew production rate, resource crew required, and number of crews 80
Table 5.10: Resource Availability List showing Resource Name and Resource Quantity Available 80
Table 5.11: Product-Task-Level Mapping 80
Table 5.12: Product-Task-Qty Mapping 81
Table 5.13: Simulation Result Set 84
Table 5.14: Product List showing Product Name and Nominal Quantity 86
Table 5.15: Task List showing Task name, expected crew production rate, resource crew required, and number of crews required 87
Table 5.16: Resource Availability List showing Resource Name and Resource Quantity Available 87
Table 5.17: Product-Task-Level Mapping 88

Table 5.18: Product-Task-Qty Mapping 89
Table 5.19: Simulation Result Set 90
Table 6.1: Pipe Spool Fabrication Simulator – Product List 99
Table 6.2: Pipe Spool Fabrication Simulator – Task List 100
Table 6.3: Pipe Spool Fabrication Simulator – Task Sequence 101
Table 6.4: Pipe Spool Fabrication Simulator – Resource List 101
Table 6.5: Pipe Spool Fabrication Simulator – Product-Task-Level Mapping 102
Table 6.6: Pipe Spool Fabrication Simulator – Product-Task-Qty Mapping 102
Table 6.7: Pipe Spool Fabrication Simulator – Simulation Result Set 103
Table 6.8: Building Finishes Simulator – Product List 104
Table 6.9: Building Finishes Simulator – Task List 104
Table 6.10: Building Finishes Simulator – Task Sequence 105
Table 6.11: Building Finishes Simulator – Resource List 105
Table 6.12: Building Finishes Simulator – Product-Task-Level Mapping 106
Table 6.13: Building Finishes Simulator – Product-Task-Qty Mapping 106
Table 6.14: Sample Result Set from Building Finishes Implementation using the Proposed Methodology 107
Table 6.15: Prototype and Production Version Construction Simulation Models 110

List of Figures

Figure 1.1: Research Methodology 5
Figure 2.1: History of Construction Simulation Advancements 11
Figure 3.1: Cut 17
Figure 3.2: Bevel 17
Figure 3.3: Fit-up 17
Figure 3.4: Welding 17
Figure 3.5: PWHT 18
Figure 3.6: NDT 18
Figure 3.7: Blasting and Painting 18
Figure 3.8: The Overall Model 19
Figure 3.9: Parameters Input Interface 20
Figure 3.10: DES Model Components 21
Figure 3.11: DES Model Flow 21
Figure 3.12: The Three Layers 24
Figure 3.13: Properties of the three layers 24
Figure 3.14: DES flow #1 - aggregate material delivery to site 25
Figure 3.15: DES flow #2 - aggregate laying operation 25
Figure 3.16: DES flow #3 - asphalt material delivery to site for base course operation 25
Figure 3.17: DES flow #4 - asphalt base course laying 26
Figure 3.18: DES flow #5 - asphalt material delivery to site for wearing course operation 26
Figure 3.19: DES flow #6 - asphalt wearing course laying 26
Figure 3.20: Selection of equipment 27
Figure 3.21: Crew building interface 27
Figure 3.22: The overall model 28
Figure 3.23: Productivity output measures 29
Figure 3.24: Process animation and layer progress 29
Figure 3.25: Asphalt Paving Simulator Outputs Interface 30
Figure 3.26: Pipeline Subarea/Section/Kilometer/Pipe Sample Hierarchy 32
Figure 3.27: Material Handling DES Flow 33
Figure 3.28: Excavation DES Model Flow 33
Figure 3.29: DES Model Flow of Balance of Tasks 34
Figure 3.30: The Overall Model 35
Figure 3.31: Pipeline Construction Simulator – DES Model 36
Figure 4.1: Conceptual Model of Common Model Structures 39
Figure 4.2: Pipe Spool Fabrication Spool/Weld Sample Hierarchy 40
Figure 4.3: Pipeline Subarea/Section/Kilometer/Pipe Sample Hierarchy 41
Figure 4.4: Asphalt Paving Subarea/Section/Kilometer/Pipe Sample Hierarchy 42
Figure 4.5: Pipe Spool Fabrication Tasks 43
Figure 4.6: Pipeline Construction Tasks 43
Figure 4.7: Asphalt Paving Tasks 43
Figure 4.8: Pipe Spool Fabrication Simulator – Tasks Sequence Showing Welded Spool Proceeding on Three Different Paths to Completion 45

Figure 4.9: Pipe Spool Fabrication Simulator – DES Model 53
Figure 4.10: Asphalt Paving Simulator – DES Model 54
Figure 4.11: Pipeline Construction Simulator – DES Model 55
Figure 5.1: Conceptual Model of Traditional Modeling Methodology 62
Figure 5.2: Conceptual Model of New Modeling Methodology 62
Figure 5.3: Basic DES Model Structure 63
Figure 5.4: Virtual Basic DES Model Created for every Task for every Product to be Built 64
Figure 5.5: Product Class 66
Figure 5.6: TaskTemplate Class 66
Figure 5.7: Task Class 67
Figure 5.8: Sequence Class 67
Figure 5.9: Resource Class 68
Figure 5.10: ResourceAvailability Class 69
Figure 5.11: ProductTaskLevel Class 69
Figure 5.12: ProductTaskQty Class 70
Figure 5.13: ResultSet Class 70
Figure 5.14: DESObject Class 71
Figure 5.15: Pseudo-Code for Creation of DES Models at Product task Intersections 72
Figure 5.16: Workflow for Creation of DES Models at Product task Intersections 73
Figure 5.17: Pseudo-Code for Creation of Initial Arrivals for the First-Task-Model for Each Product 74
Figure 5.18: Workflow for Creation of Initial Arrivals for the First-Task-Model for Each Product 74
Figure 5.19: Pseudo-code for Record End of Task and Create Arrival for Next Task 75
Figure 5.20: Pseudo-code for Record End of Task and Create Arrival for Next Task 75
Figure 5.21: Workflow for Record End of Task and Create Arrival for Next Task 76
Figure 5.22: Prototype User Interface 81
Figure 5.23: Main Simulation Algorithm 82
Figure 5.24: Code Executed on BeginTask 83
Figure 5.25: Code Executed on CapturedResource 83
Figure 5.26: Code Executed on FinishTask 84
Figure 6.1: Anylogic DES Model 94
Figure 6.2: Populate Product Data 95
Figure 6.3: Populate Task and Sequence Data for Each Product 95
Figure 6.4: Populate Resource Data for Activities 96
Figure 6.5: Store Product-Task Start Date in Entity Traversing the Product-Task DES Model 96
Figure 6.6: Output Result of Product-Task DES Model Run and Call Successor 96
Figure 6.7: Inject Entity into Successor Product-Task DES Model 96
Figure 6.8: Spreadsheet and Corresponding SQL Server Data Structure for Product Definitions 97
Figure 6.9: Spreadsheet and Corresponding SQL Server Data Structure for Task Definitions 97
Figure 6.10: Spreadsheet and Corresponding SQL Server Data Structure for Predecessor-Successor Relationships 97

Figure 6.11: Spreadsheet and Corresponding SQL Server Data Structure for Resource Definition 97
Figure 6.12: Spreadsheet and Corresponding SQL Server Data Structure for Resource Availability Definitions 98
Figure 6.13: Spreadsheet and Corresponding SQL Server Data Structure for Product-Task-Level Mapping 98
Figure 6.14: Spreadsheet and Corresponding SQL Server Data Structure for Product-Task-Qty Mapping 98
Figure 6.15: SQL Server Data and Corresponding Spreadsheet Structure for Simulation Result Set 98
Figure 6.16: Pipe Spool Fabrication Spool-Weld Basic Two-Level Hierarchy 99
Figure 6.17: Pipe Spool Fabrication Simulator – Tasks Sequence Showing Welded Spool Proceeding on Three Different Paths to Completion 100

1 Introduction

1.1 Background and Problem Statement

As construction projects become large, complex and tough to manage using traditional techniques, computer simulation can be used effectively to design and analyze construction processes regardless of the complexity or size (Abourizk, 2010). Computer simulation models can be built to characterize the construction activities of a scope of work ranging from a very large and complex project to a sub-area of an industrial facility or to a floor or room of a building. Using simulation, engineers can test out different construction scenarios, estimate resource utilization and find bottlenecks, and forecast time and cost requirements without going to site.

The process of building a simulation model includes four distinct phases: product abstraction phase, process abstraction and modeling phase, experimentation phase, and decision making phase (Abourizk, 2010). While building a new model, simulation practitioners find themselves going through the full four-phase process in its entirety. This rigorous and time consuming cycle is typically repeated for every new construction simulation model to be built, and requires extensive know-how in both simulation and in the subject construction discipline. Accordingly, when faced with a new situation to analyze or a question to be answered, engineers on large and complex projects, even those who are familiar with the usefulness of simulation and value its role, would have to make the choice between using simulation techniques to approach the problem, or resorting to traditional tools. Time permitting and with the right resources and expertise, the choice would go to simulation. But, time and know-how lacking, the decision is to fall back to the traditional techniques which would yield a result quickly and with much less effort, even though the engineers understand the relative inadequacy of the traditional tools.

Modeling is the most difficult and the most time-consuming part of simulation (Abourizk, 1995). The effort and technical expertise needed to build a simulation model and then run experiments compared with the uniqueness and relatively short life cycle of a construction project contribute to the slow adoption of simulation by the industry (Mohamed, 2005). An approach to remedy such a situation would be to diminish the time and skills required to build a simulation model. Shortening this process and taking away from the simulation modeling expertise and effort requirements would enable engineers to concentrate on solving the problem they are facing rather than spending their time within a simulation development environment building a simulation model.

1.2 Research Objectives

The main objective behind this research is to make the use of computer based modeling and simulation more accessible for end users in the construction industry. The strategy to achieve this goal is to develop a methodology for the automated creation and running of construction simulation models. This methodology would allow construction end users to easily create construction simulation models by describing the required models using data in a predetermined format entered into a pre-developed system instead of resorting to the traditional methods usually requiring a substantial amount of time and effort and also requiring the participation of a simulation practitioner to develop the simulation model. Two sub-objectives necessary for the development of the said methodology were identified as follows.

1. Identify the commonalities that exist in the processes of building construction simulation models. This will serve to define all pertinent components required to describe a construction simulation model including product definitions, process definitions, environment definitions and model development.

2. Use the commonalities identified to describe the various required components for the proposed methodology. This will include describing the structures that will hold the common model descriptive information and simulation run outputs, and the algorithms required to automatically generate and run the required discrete event simulation models using the model descriptive information.

1.3 Anticipated Contributions

Execution of this study is expected to produce two main contributions. It will act as a learning experience where the commonalities in building simulation models for diverse construction disciplines will be detailed and documented within the confines of the research scope of this work. This will include detailing the product, process and environment components for three case study construction simulation models implemented on various construction projects.

The execution of this study will also result in the development of an automated modeling methodology aimed at lowering the threshold of adoption of simulation as a viable problem solving technique in the construction industry by reducing the time, effort and expertise required for building simulation models.

1.4 Research Methodology

In order to develop the methodology that will aid in making the use of simulation more accessible by construction end users, a set of research steps is required and outlined as follows. These include a literature review, exploring case studies, identifying commonalities, developing the methodology, and implementing the methodology. The details of those research activities are depicted as follows.

1. Reviewing literature to recognize the issues and role of simulation in the construction industry.
2. Exploring three construction simulation models for commonalities.
3. Identifying and formalizing the commonalities between the three case studies by identifying the common construction information required to describe the products to be built, the construction process utilized, the environment factors affecting the process, the model developed, and simulation practitioner requirements.
4. Developing the proposed methodology and describing all its required components. These components include the data structures to host model descriptive data such as product definition structures, process definition structures, environment definition structures and the simulation output structure, and the model generation algorithms.
5. Building a proof of concept prototype of the methodology. This step encompasses the development of a proof of concept implementation of the proposed methodology by a third party who has no prior knowledge of the research in this work by simply explaining the requirements to this third party.
6. Developing an enterprise level production version of the methodology in a simulation development environment and subsequently:
    a. use that system to redevelop one of the case study simulation models
    b. re-use that same developed system to implement a new construction simulation model
7. Provide recommendations for applicability, and non-applicability, of the model and final comments, discussion and recommendations for future work.

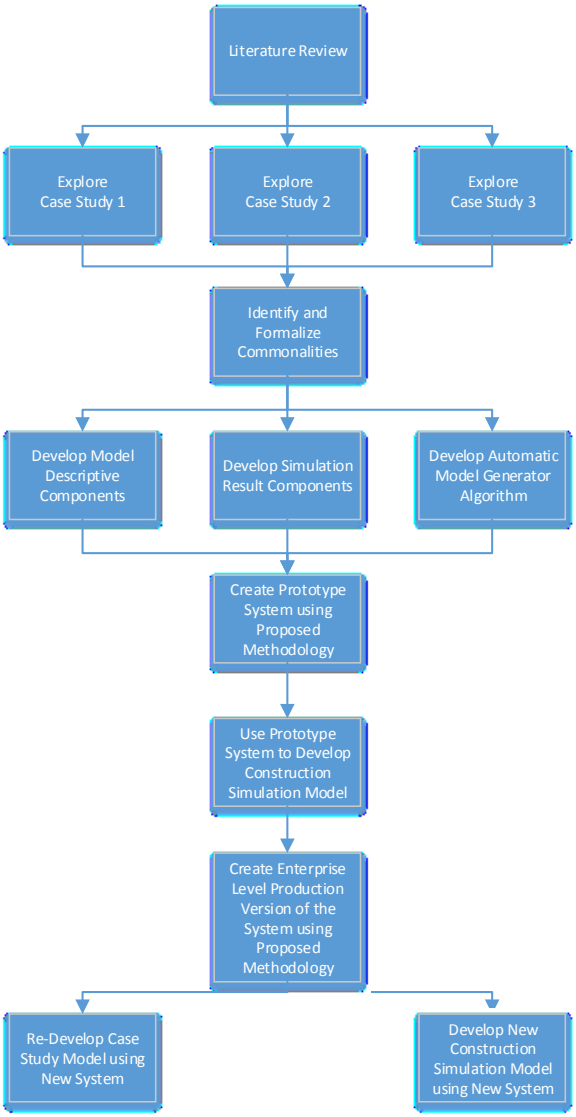

Figure 1.1: Research Methodology

1.5 Research Scope

The proposed automated model creation methodology subject of this work is aimed at enabling construction end users to quickly and simply create simulation models, thus making simulation a more accessible tool for the construction industry. The matter of fact is that this methodology will not be able to encompass all construction related simulation development needs in the construction industry. The proposed methodology will only be able to cater to specific construction situations similar in nature to the case study simulation models explored in this work, and to the prototype and enterprise level instances of the methodology developed as part of this work. Accordingly, this research does not propose

the replacement of all simulation modeling activities for the construction industry with the proposed methodology. Instead, it proposes a way for end users on construction projects to quickly apply simulation techniques to answer questions and solve problems related to their work. The applicability of this methodology to construction situations is recommended for those construction situations which can be described using data that will have a functional fit within the model descriptive structures defined, and where relevant output analysis can be usefully made from simulation result data structured as per the output structures of the methodology being proposed.

1.6 Thesis Organization

This thesis consists of seven chapters. Chapter 1 introduces the research motivation, identifies the research objectives, expected contributions and scope, and summarizes the research methodology. Chapter 2 is a brief overview of the literature on the role of simulation in construction and the deterrents to the adoption of simulation in the construction industry relevant to this research. Chapter 3 explores three case studies of construction simulation models in terms of modeling information and requirements, and expected outputs. Chapter 4 identifies and categorizes the commonalities found in Chapter 3. Chapter 5 proposes a methodology for automated simulation model creation for construction and describes the required components of the methodology; chapter 5 also describes a prototype development of the methodology. Chapter 6 describes the validation of the methodology through the development of an enterprise level production version of the methodology and using it to re-develop the pipe spool fabrication case study from Chapter 3, and to develop a new building finishes model. Chapter 7 summarizes the research conclusion, the limitations of the applicability of the methodology, and suggestions for future work.

2 Literature Review

2.1 Introduction

Chapter 1 summarized the main research objective as the development of a methodology for the automated creation of construction DES models to make simulation more accessible for construction end users. The first step to develop the methodology was to study existing research related to the current role of simulation in construction and the factors that deter the adoption of simulation in the construction industry. The following sections explore these topics.

2.2 Application of Simulation in the Construction Industry

Abourizk (2011) describes construction simulation as a fast growing scientific field engaged in developing and experimenting with computer-based models of construction systems with the goal of understanding their behavior. When dealing with large and complex construction operations, which are more difficult to manage using traditional project management tools, computer simulation methods have shown to be effective in designing and analyzing construction processes, regardless of the complexity or size (Abourizk 2010, Lucko 2009). Construction processes are essentially complex and linked with a great deal of randomness and uncertainty (Halpin, 2003); a simulation model can be built to describe the construction activities of a scope of work ranging from large, complex industrial projects to a simple room of a small building, including the resources required for performing the work and the environment within which the scope of work is being performed (Abourizk, 2010). Using simulation, engineers can test out different construction scenarios in a low-cost, low-pressure (Abourizk 2011) controlled environment much faster than the real system (Lucko 2009), estimate resource utilization and find bottlenecks, and forecast time and cost requirements without having to go to site.

The first to propose the idea of using simulation to study the complexity of construction operations was Teicholz in 1963. He was followed in 1969 by Gaarslev who used some of Teicholz's work to compare queuing theory to simulation. Halpin, in 1973, introduced his CYCLONE system. CYCLONE enabled modeling construction processes using the discrete event simulation technique. This was succeeded by many who built enhancements on the original system including Halpin himself with Mainframe CYCLONE in 1976, INSIGHT (Paulson et al, 1987), RESQUE (Chang & Carr, 1987), UM-CYCLONE (Ioannou, 1989), Micro-CYCLONE (Halpin 1990), COOPS (Liu, 1991), CIPROS (Odeh, 1992), STEPS (McCahill & Bernold, 1993), DISCO (Huang & Halpin, 1994), and STROBOSCOPE (Martinez & Ioannou, 1994). Object oriented concepts were introduced into construction simulation modeling by Chang (1991). The advantages of object oriented concepts in construction were discussed by Oloufa (1993) who concluded that using the object oriented method results in diminished coding and better readability of simulation models. Researching model reusability Tommelein et al. (1994) and Shi and Abourizk (1997) employed a library-based approach allowing the building of models from a set of predefined model components. An object-oriented library-based system for building parameterized simulation models was developed by Oloufa (1994). Abourizk and Hajjar (2002) developed a simulation development environment called Simphony based on their unified modeling methodology. Simphony allowed for the creation of modeling templates which simplified and shortened the development process for new special purpose simulation tools. WorkSim, a framework to automate production of simulation models using templates as building blocks was introduced in 2009 by Lucko. COSYE – Construction Synthetic Environment – a high level architecture distributed simulation development environment was introduced in 2009 by Abourizk & Hague.

2.3 Deterrents to Adoption of Simulation in Construction

Although much academic research has been performed in the study of construction process simulation (Hassan and Gruber 2008, Zayed and Halpin 2004, Zayed and Halpin 2001, Ioannou and Martínez 1996,), this research has not seen mainstream application in the construction industry (Abourizk 2010, Lucko et al. 2009). Lucko (2008) attributes this discrepancy to three important factors, specifically the time, cost, and skills required to construct and analyze simulation models. Abourizk (2011) lists the skills required to include (1) the logic and sequence of the operation being modeled; (2) simulation techniques and algorithms to be utilized; and (3) the necessary software tools and applications for developing the solution. Construction end users possess the first skill as they are the domain expert of the operation they want to model and experiment with. If they are to develop a simulation model, though, they would need to learn the other two skills in a relatively short time frame, as knowledge acquisition in the project-driven construction industry is usually constrained by short time windows (Abourizk 2011). An alternative for construction domain experts learning simulation is for them to partner with a simulation practitioner who already possesses those skills but similarly usually lacks the construction domain knowledge required. Accordingly, construction domain experts are always required to spend a substantial fraction of their time with the simulation practitioner (Lingineni et al. 1995) to transfer the relevant business know-how and aid in defining the product, process, and environment definitions of the model being developed. Once the milestone of either the construction domain expert learning how to model, or partnering thereof with a simulation practitioner is achieved, the process of building a simulation model starts. The model building process includes four distinct phases: product abstraction phase, process abstraction and modeling phase, experimentation phase, and decision making phase

(Abourizk, 2010). While building a new model, simulation practitioners find themselves going through the full four-phase process in its entirety. This expertise- and effort-intensive and time consuming cycle is typically repeated for every new construction simulation model to be built. The modeling part of the cycle is the most difficult and the most time-consuming section (Abourizk, 1995). The effort and technical expertise needed to build a simulation model and then run experiments compared with the uniqueness and relatively short life cycle of a construction project contribute to the slow adoption of simulation by the industry (Mohamed, 2005). Construction end users often find themselves facing a decision of whether to use simulation to analyze their construction operations or resort to traditional techniques. Selecting simulation would necessitate either learning how to develop simulation models in a rather short period of time, or finding, and partnering with a simulation expert to aid in the simulation modeling process. Moreover, they will need to spend a significant amount of time developing the required model. An approach to remedy such a situation would be to reduce the time and skills required to build a simulation model. Shortening this process and taking away from it some of the time, effort and expertise required would enable engineers to concentrate on analyzing and managing their construction operations rather than spending their time within a simulation development environment attempting to build a simulation model.

A solution to this problem which can help further the adoption of the use of simulation in the construction industry is to make simulation a more accessible tool by reducing the simulation skills requirements, and reducing the effort and time requirements to build construction simulation models. To date, and as surveyed above, tools for construction simulation have progressed well over time by offering more advanced simulation development environments. These environments offer a graphical user interface enabling a

more user friendly experience; some of them offer templates and template building capabilities which help users develop special purpose simulation models. Still, these advanced environments all require a simulation practitioner or simulation expertise on top of the domain expertise to develop a simulation model. This research work aims at developing a methodology for the automated creation of construction simulation models by construction end users without requiring simulation practitioners or simulation expertise. It proposes to achieve this by describing the methodology to create a discrete event simulation compiler and its supporting model descriptive data structures which will enable construction end users who possess the domain expertise to describe their construction operations through data and the developed system would compile a discrete event simulation model using that data that mimics their real life operations. Figure 2.1 below is a conceptual depiction of the surveyed history of advancements in construction simulation systems to date and the proposed methodology as a layer on top of the simulation development environments.

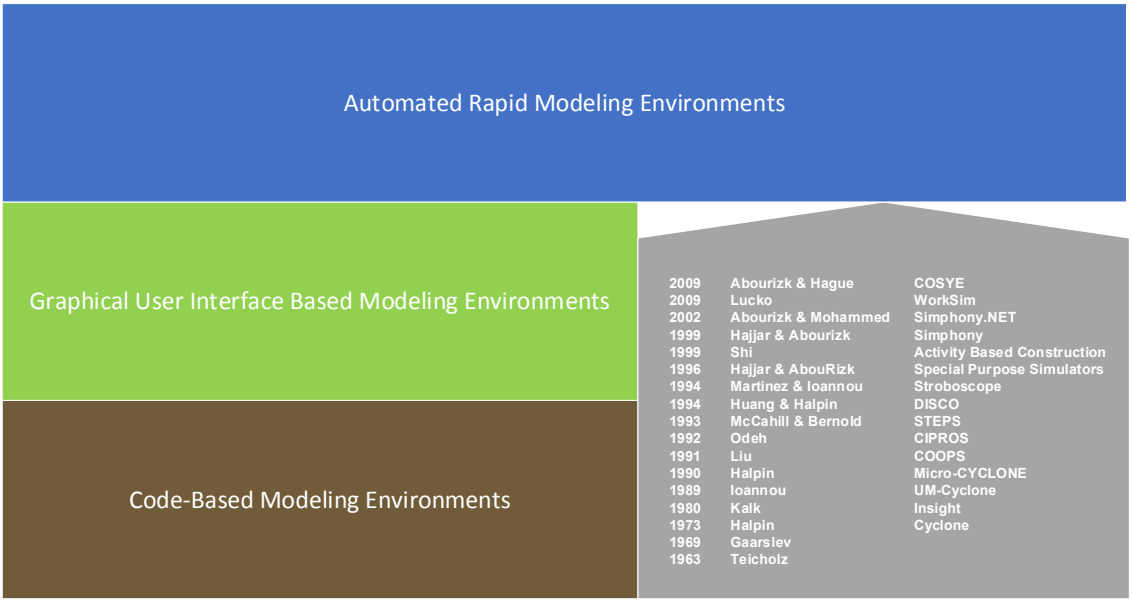

Figure 2.1: History of Construction Simulation Advancements

2.4 Construction Simulation Modeling in Simphony

Simphony was first introduced by Hajjar and AbouRizk (1999). This is a Microsoft Windows-based special purpose simulation tool used to develop flexible and easy to use applications for the construction industry. Simphony provides a graphical, hierarchical, modular, and integrated modeling environment enabling an easier approach to modeling the construction process. In Simphony, a developer would program a domain template encompassing a collection of elements connected together and which represent a set of construction activities in a specific domain. Elements may encapsulate more than one process activity and would have its own graphical representation mapping to the actual process on site. The template building approach enables users to drag and drop the different required items into the modeling environment and link them together to produce a model. The development of a template is a process which requires both a profound understanding of the construction domain being modeled, and advanced simulation modeling development skills to operate within the simulation development environment either through code or through the graphical user interface. Multiple Simphony templates have been created to model real projects and help construction managers in decision making and planning processes including productivity measurement, risk analysis, resource allocation, site planning, claim and dispute resolution, scenario planning, and cost estimation (Sawhney 1994). Using Simphony, a large number of useful construction simulation models were created including earth moving, pavement construction, concrete placement on high-rise buildings, tunneling, underground pipe-jacking, tower crane utilization, equipment management, and site layout optimization. A more recent version of Simphony, “Simphony.NET 4.0”, was developed using the Microsoft.NET 4.0 framework to provide an extensible and flexible environment for modeling and integration purposes. The new version offers simulation practitioners with

advanced simulation development capabilities through its Simphony Core Services, Simphony Modeling Services and its Graphical User Interface. Unlike in the older version of Simphony, the development of a template is not tied to the user interface, thus allowing both development of standalone models or integration of Simphony simulation capabilities into other systems. The new version of Simphony brings a lot of power to the simulation practitioner, but does not remove the requirement of simulation know-how on top of domain specific knowledge for the development of a new simulation template or model, or the amendment of an existing template or model.

3 Three Construction Simulation Models

3.1 Introduction

The development of the automated construction simulation modeling methodology included a study to understand the commonalities found between three traditional discrete event simulation models developed and implemented to help solve problems and answer questions on construction projects.

Multiple factors influenced the decision on which specific simulators to use as case studies for this work. These included the modeling effort required for the simulator, the time required to design and build the simulator, and whether the model was implemented at real life projects or not. Each of the three selected simulators required a substantial amount of time and effort to model its construction problem, and design, build and test the required simulation model. All three simulators were successfully implemented at multiple mega construction projects. This chapter explores the development of the three models and looks at their data requirements, their DES models and their outputs.

3.2 Pipe Spool Fabrication Model

This sections explores the first of the three simulation models studied for commonalities. It describes the development of the pipe spool fabrication simulation model and looks at the data requirements, the DES model which forms the basis of this simulator, and the outputs of this model.

3.2.1 Background

On large industrial projects, pipe spool fabrication is a major component of the construction operation. It is also a relatively short term, complex construction process often riddled with uncertainty due to the intrinsic unique nature of its outputs and the numerous factors affecting its activities. As such, it is important for all stakeholders to have a good grasp of the performance of pipe fabrication shops and their ability to meet the site pipe installation

schedules. The ability of computer based modeling and simulation to model resource and activity interactions, queuing, and uncertainties renders it a good fit for modeling the pipe spool fabrication process. Construction contractors on such large industrial projects often build one or more project specific pipe fabrication shops to handle the pipe spool fabrication scope. These shops are built to handle a specific set of pipe fabrication activities including cut and bevel, fit-up, welding, QC inspection, post weld heat treatment, non-destructive testing, blasting and painting. Each of these activities is repeatedly performed by a specific type of crew on pipe spools. Each time it is performed, a crew is utilized for a certain duration and the result is specific progress of a pipe spool along its path to completion. With the large number of spools and their diverse characteristics and resource requirements, forecasting pipe spool fabrication activity completion and optimizing resource allocation and utilization becomes a complex task well suited to computer modeling and simulation.

3.2.2 Simulator Design and Development

The simulator was developed to aid stakeholders in arriving at answers to the issues stated above. The first step was the abstraction of the real world situation into a simulation model representing the operations of a pipe spool fabrication shop, including detailing the product and process definitions for all the main activities. In order to understand the nature of how pipe spool fabrication activities were performed on construction sites, extended visits to multiple mega industrial projects were conducted to observe and document the above mentioned set of activities. Benchmarking for every activity was conducted via numerous observations of the activity being performed on different spools of varying characteristics. Both crew composition information and productivity figures were collected. The simulator was developed as a discrete event simulation model with spools as the main entity. For the

welding tasks, welds are the entities - where spools are split into their constituent welds - in order to process welds individually and collect their artificial history.

3.2.3 Product Definitions

Product definition for spools to be processed by the simulator is a straightforward process where only those spool characteristics required for simulating the fabrication activities were specified for each spool. It is organized into a two-level hierarchy for spools and their relevant joints. Data for spools include spool ID, current spool status, line class, material type, paint code, surface area, and spool specific priority information. Data for joints include weld type, inch-dia, post weld heat treatment (PWHT) requirement and non-destructive testing (NDT) requirements. All these are used to determine the quantity of work required for each of the tasks involved in building a specific spool.

Spool ID	Stauts	Material	Paint Code	Surface Area	Priority
A140-A141-B92SL-15139D-S101	3	CS	6D	0.03	180
	Weld #	**Weld Type**	**Weld Inch-Dia**	**PWHT Req'd**	**NDT Req'd**
	2	SB	0.75	0	0
	4	SB	0.75	0	0
	5	SB	0.75	0	0
A140-A141-B92SL-15139D-S102	3	CS	6D	0.01	180
A140-A141-B92SL-15139D-S103	3	CS	6D	0.15	180

Table 3.1: Sample Spool-Weld Hierarchy

3.2.4 Process Definitions

For the process definitions we needed to define the activities and flow required to fabricate the different spools. For each activity, the type of resource (crew) required and its relevant productivity had to be identified. Figures 3.1, 3.2 and 3.3 are snapshots of the different activities represented in the DES model. For each activity, the required crews and time to perform the activity is decided based on spool characteristics. Figures 2a, 2b, and 2c, below, depict the DES flow of spools through the “Cut,” “Bevel,” and “Fit-up” activities.

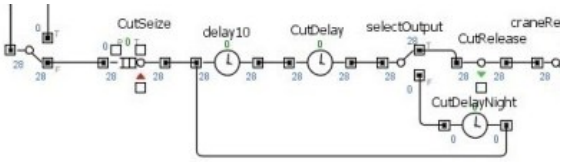

Figure 3.1: Cut

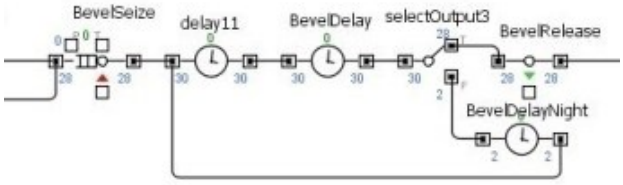

Figure 3.2: Bevel

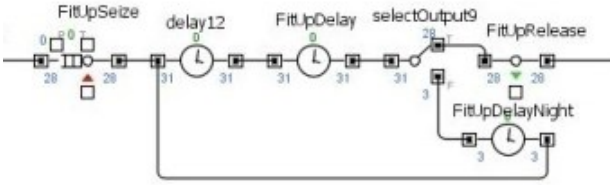

Figure 3.3: Fit-up

Figure 3.4 below depict the DES flow of spools through the welding process. For welding, each spool entity is split into its welding entities, based on the number of shop welds required. Based on spool and weld characteristics, (1) the appropriate number of welders is assigned to each weld, and, accordingly, (2) the weld duration is derived. Splitting the spool entity into weld entities allows us to process welds independently and collect their respective artificial history individually. Once all welds are processed, the weld entities are batched into a spool entity again.

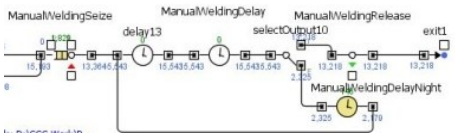

Figure 3.4: Welding

Figures 3.5, 3.6 and 3.7 show the DES flow of spools through the PWHT, NDT and painting activities. Not all spools require PWHT, and not all spools require NDT. The flow and logic control of the model automatically detect this from the spool information and associated tasks are initiated accordingly.

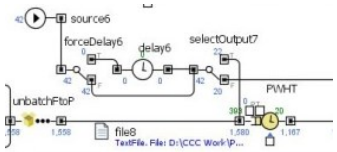

Figure 3.5: PWHT

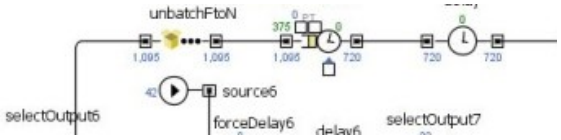

Figure 3.6: NDT

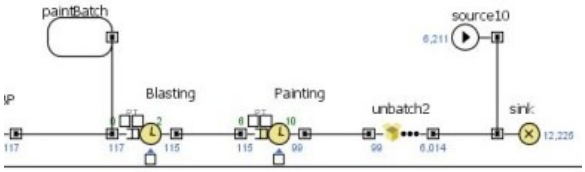

Figure 3.7: Blasting and Painting

3.2.5 Resource Definitions

Each of the pipe fabrication activities is associated with a resource type. Each resource type is typically a crew composed of a group of workers required to perform a specific task. Table 3.2 shows typical crew compositions on a large industrial construction project. Notice that certain worker types are shared amongst the various crew types.

Crew Type	Worker Type 1	Worker Type 2	Worker Type 3	Worker Type 4	Worker Type 5	Worker Type 6	Worker Type 7	Worker Type 8	Worker Type 9
Cut	1	2	1	2					
Bevel		1							
Fit-up		2	2		1	4			
Welding			1				1		
PWHT			1					2	
Blasting	1		1						
Painting			3						4

Table 3.2: Typical Crew Compositions

Table 3.3 shows typical worker availability over time on an industrial project is shown below. Workers available make up the required crews (resources) for the activities which are then captured to simulate the performance of a task on a spool or weld.

Worker Type	Month 1	2	3	4	5	6	7	8
1	21	23	25	25	25	25	25	25
2	85	90	95	95	100	100	100	100
3	80	82	85	86	104	104	104	104
5	37	38	40	41	42	42	42	42
6	90	95	100	100	110	110	110	110
7	92	99	112	129	135	135	135	135

Table 3.3: Typical Worker Availability over Time

3.2.6 Model Structure

All the above pieces come together as in the structure shown in Figure 3.8 below.

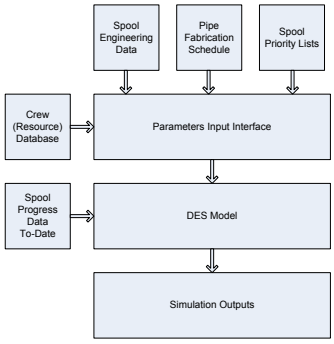

Figure 3.8: The Overall Model

3.2.6.1 Parameters Input Interface

This module allows the user to define the different parameters of the simulator and run the model. “Spool Engineering Data,” “Pipe Fabrication Schedule,” “Spool Priority Lists” and “Crew Database” feed into the “Parameters Input Interface” module. “Spool Engineering

Data" provides engineering characteristics of the spools; "Pipe Fabrication Schedule" and "Spool Priority Lists" provide, respectively, the activity schedule information related to pipe fabrication and the priority lists for spool fabrication requirements produced by the engineers; "Crew Database" provides resource information, namely the number of crews available over time of each crew type. Through the "Parameters Input Interface" the user can change the location of the feed data, assumed productivities for the different tasks, working hours, and fix the number of crews at a constant level throughout the simulation duration instead of reading them from the relevant feed.

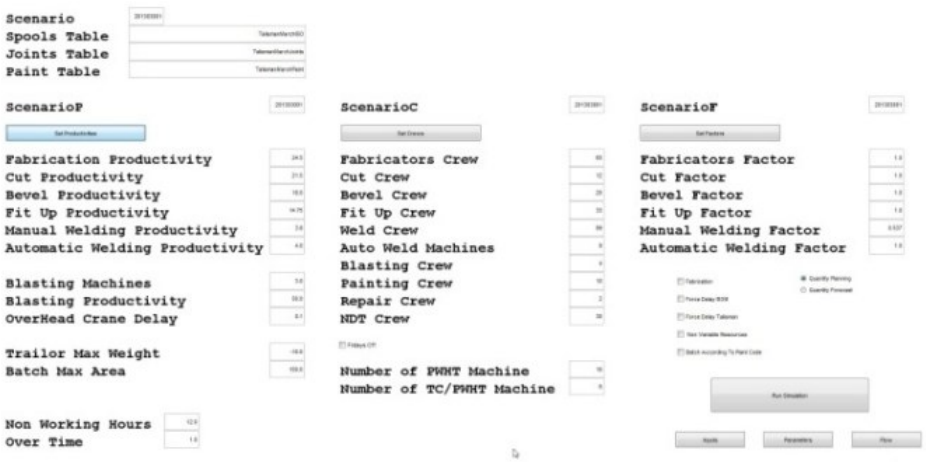

Figure 3.9: Parameters Input Interface

3.2.6.2 Discrete Event Simulation Model

The discrete event simulation model (DES) carries all the DES flow and logic required for running the simulation model. It includes all the tasks along with their corresponding parameters, resource pool requirements, and duration formulas.

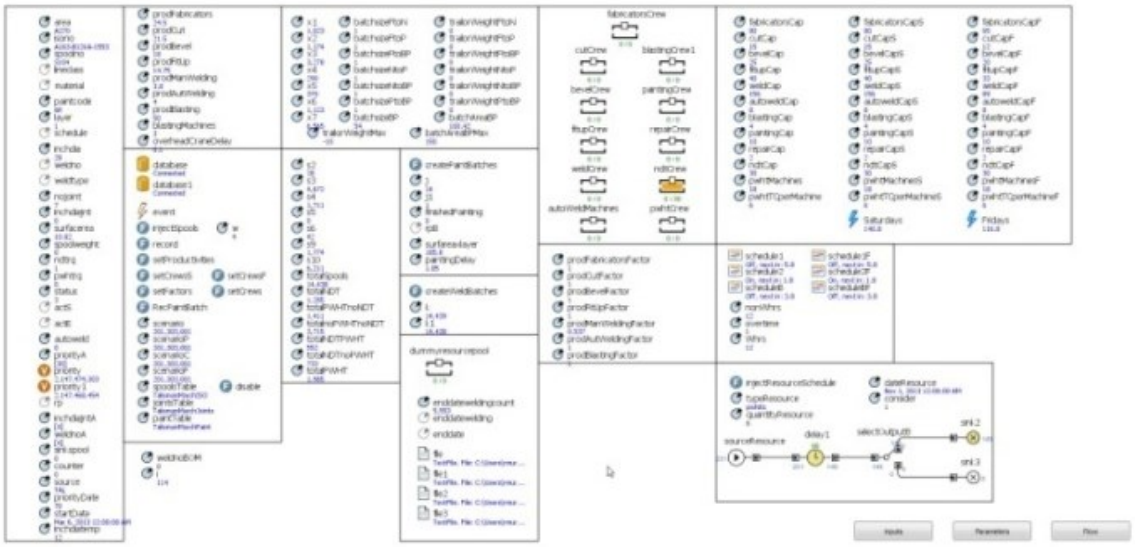

Figure 3.10: DES Model Components

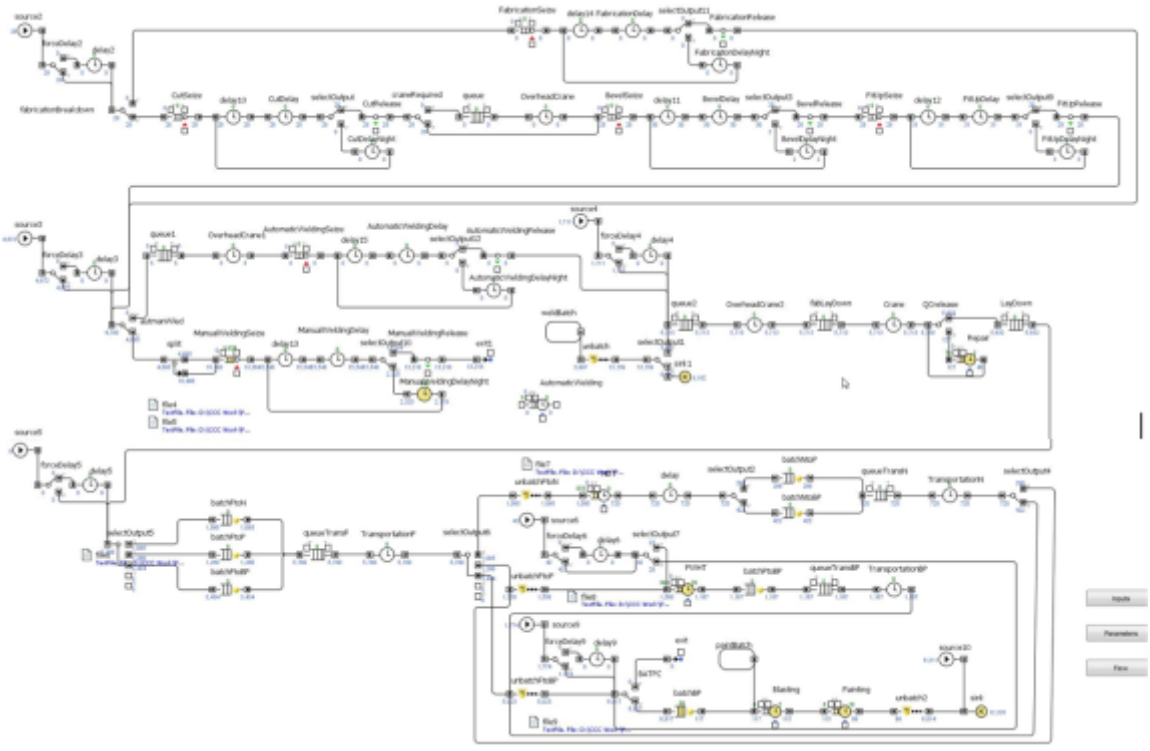

Figure 3.11: DES Model Flow

3.2.7 Simulation Outputs

The simulator produces, as its main output, a comprehensive set of data comprised of the artificial history of the simulated pipe fabrication operations. The result set (as in Table 3.4) contains a record of the activities performed on the corresponding entities (spools or welds)

utilizing the required resources. For each entity/activity/resource occurrence, the data contains a start date and time, an end date and time, and a number of resources utilized for the duration.

Scenario #	Spool ID	Weld No	Activity	Start Date and Time	End Date and Time	Resources
1304152	A140-A141-B92SL-15139D-S101	2	Welding	4/3/2013 11:01	4/3/2013 11:23	1
1304152	A140-A141-B92SL-15139D-S101	4	Welding	4/3/2013 11:01	4/3/2013 11:23	1
1304152	A140-A141-B92SL-15139D-S101	5	Welding	3/3/2013 15:01	3/3/2013 15:23	1
1304152	A140-A141-B92SL-15139D-S101		Painting	3/14/13 8:00	3/16/13 10:00	1

Table 3.4: Sample Simulation Outputs

3.3 Asphalt Paving Simulation Model

This sections explores the second of the three simulation models studied for commonalities. It describes the development of the asphalt paving simulation model and looks at the data requirements, the DES model which forms the basis of this simulator, and the outputs of this model.

3.3.1 Background

Asphalting operations are a main constituent of road construction projects. Asphalting operations involve numerous interactions between the many participants in the process including the paving machines, trucks, loaders, rollers, asphalt plants, and material sources. Loaders load trucks with aggregate material for the sub-base laying operation. Trucks transfer the sub-base layer material to the paving machine. Trucks form a queue at the paving machine and wait for their turn to unload into the paving machine while it is laying the fill material. Rollers follow the paving machine at an appropriate distance and compress the layers. The same operation is repeated for asphalt. Trucks are loaded with asphalt from the asphalt plants. The trucks then haul the material to the paving machines and form a queue, waiting for their turn to unload the material into the paving machine, which is laying

an asphalt base course or wearing course. Appropriate rollers follow the asphalting machine to compress the asphalt layer.

Many factors affect the operations and the interactions between the different resources. A main factor in asphalt paving operations is the number and the asphalt laying rate of the asphalt paving machines. The number of paving machines must be sufficient to meet the overall laying speed required to finish the operations on time. Asphalt plants operate at a typical asphalt production rate which governs the amount of asphalt available for the operation. A lower than required asphalt plant production rate will lead to delays in finishing the operation. The number of trucks that can be loaded at the same time, truck loading time, truck load size, truck travel speed, and the distance between the sources and the construction location are among the factors affecting the supply of material to the paving machines. Not enough trucks will lead to delays in the operation.

The most common uses of the asphalting simulator are estimating, planning and managing asphalt operations on a project. It helps determine: (1) the total time required for an asphalting operation; (2) the required asphalt plant production rate (3); the required number of different equipment involved (pavers, rollers, trucks, loaders); and (4) which type of equipment is acting as a bottleneck.

3.3.2 Simulator Design and Development

The simulator was developed to cater to the issues stated above. The first step was the abstraction of the real world situation into a simulation model representing asphalt paving operations including the product and process definitions.

3.3.3 Product Definitions

For product definitions, this included building up to three main courses: the sub-base course, the base course and the wearing course. First, we defined the overall length of the

paving operation. For each of the courses, we needed to define the width, the number of layers, and the thickness of each layer.

Figure 3.12: The Three Layers

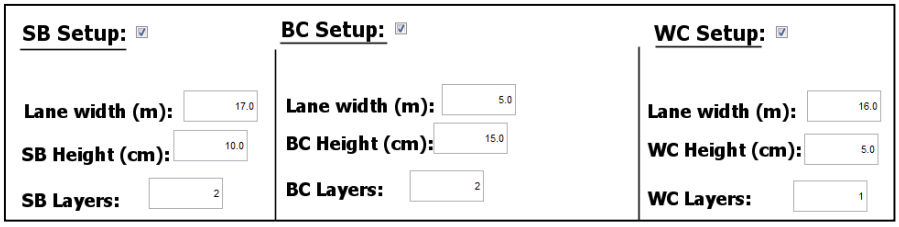

Figure 3.13: Properties of the three layers

3.3.4 Process Definitions

For the process definitions we needed to define the flow of the tasks required to build each of the products and the inter-relationships between the different processes. For each process we needed to identify the resources required (along with all their relevant properties), the material sources including production rates and distances from site, and the team compositions for each layer operation. A large amount of supplementary code was written behind each of the task flows to support the generic nature with which this model was being developed. Changing any of the flow sequences in the model requires changes to be applied to the model through the graphical user interface and to the supplementary code behind the scenes.

3.3.4.1 Sub-base Course

To represent the sub-base course laying operations, two discrete event simulator flows were implemented. The first is a material delivery flow handling trucking operations from the aggregate source(s) to site and back. The second flow depicts the aggregate laying operations for the x layers of sub-base course.

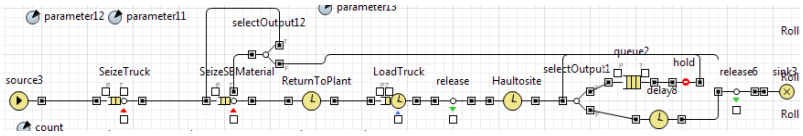

Figure 3.14: DES flow #1 - aggregate material delivery to site

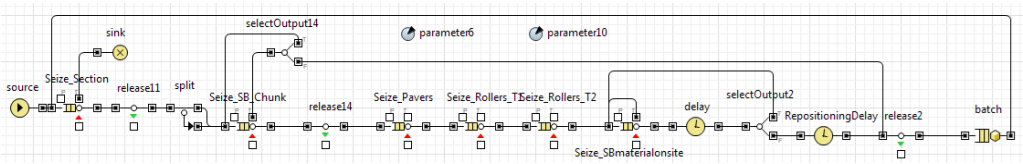

Figure 3.15: DES flow #2 - aggregate laying operation

3.3.4.2 Base Course

To represent the base course laying operations, two discrete event simulator flows were implemented. The first is a material delivery flow handling trucking operations from the asphalt plant to site and back. The second flow depicts the paving operations for the y layers of base course.

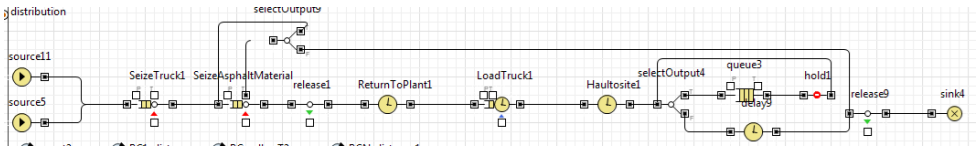

Figure 3.16: DES flow #3 - asphalt material delivery to site for base course operation

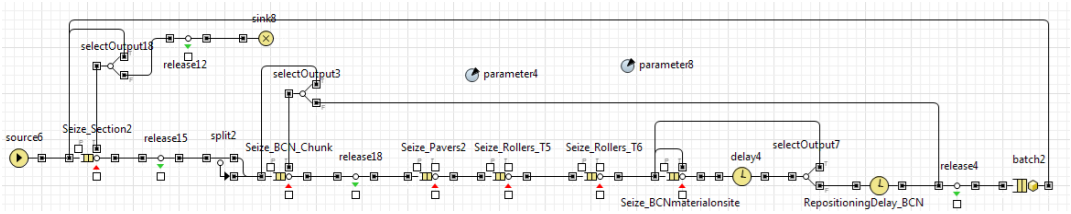

Figure 3.17: DES flow #4 - asphalt base course laying

3.3.4.3 Wearing Course

To represent the wearing course laying operations, two discrete event simulator flows were implemented. The first is a material delivery flow handling trucking operations from the asphalt plant to site and back. The second flow depicts the paving operations for the z layers of wearing course.

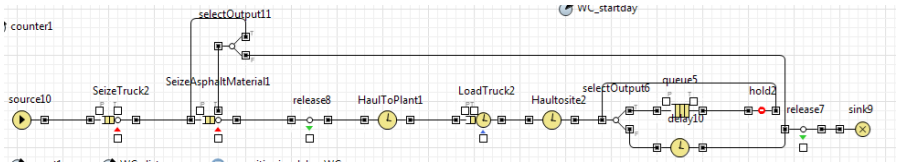

Figure 3.18: DES flow #5 - asphalt material delivery to site for wearing course operation

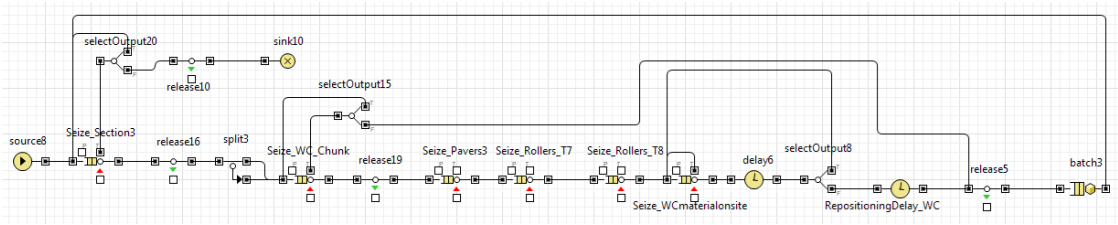

Figure 3.19: DES flow #6 - asphalt wearing course laying

3.3.4.4 Asphalt Plant Sources

The asphalt plant definition interface allows the user to add a record for each asphalt plant to be used in the simulator. Each record contains the asphalt plant production rate, the number and size of hot storages, the distance between the plant and site, and the percentage of total asphalt required to be drawn from this plant.

3.3.4.5 Aggregate Sources

The aggregate sources definition interface allows the user to add a record for each aggregate source to be used in the simulator. Each record contains the aggregate source production rate, the distance between the aggregate source and site, and the percentage of total required aggregate to be drawn from this source.

3.3.4.6 Equipment Selection

The equipment required for each of the processes is selected through the interface, assigning the appropriate relevant models such as loaders, trucks and pavers.

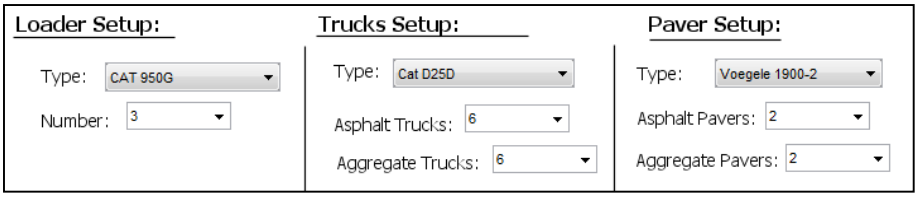

Figure 3.20: Selection of equipment

For each process, a model team (crew) of equipment is assembled listing the number of pavers and rollers required. A process may have one or more teams available to do the work.

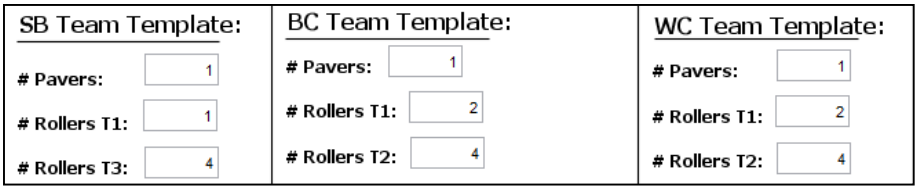

Figure 3.21: Crew building interface

3.3.5 Model Structure

The above model components come together as in Figure 3.22 below: (a) a main parameters input module, (b) a main process module, (c) an asphalt plant definition module, (d) an aggregate source definition module, (e) an equipment database module, (f) an outputs module, and (g) an animation module.

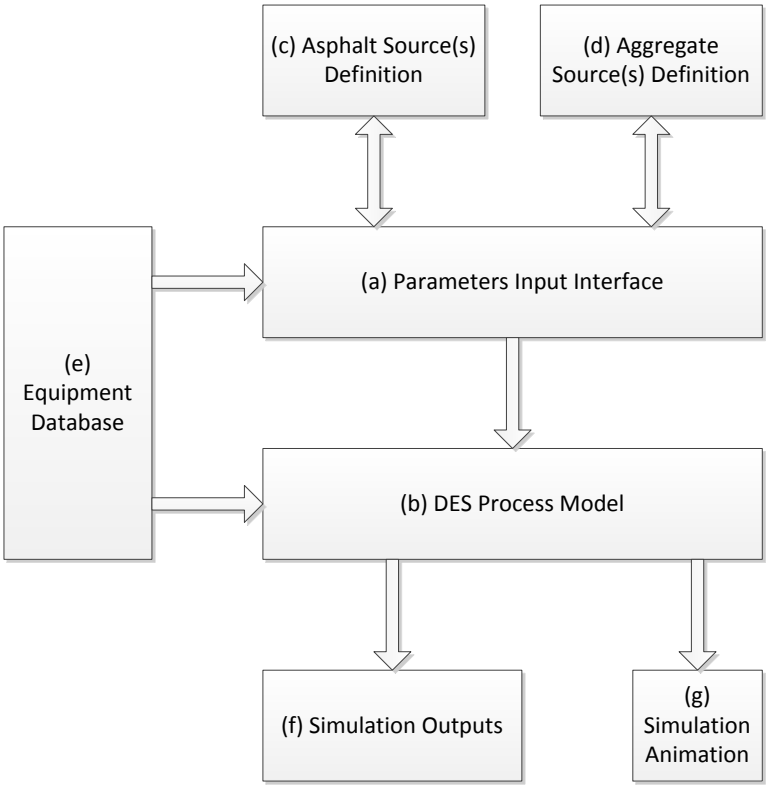

Figure 3.22: The overall model

The parameters input interface (a) allows the definition of the products and resources and links to the asphalt (c) and aggregate (d) sources definitions. The DES process model (b) incorporates all the simulator flows and supplementary code for the processes. The equipment database (e) contains information about all the relevant equipment types that can be selected for use within the simulator; each piece of equipment in the database carries its productivity norms for the simulator.

The main simulator outputs available during runtime and at the end of the simulation run appear in the simulation outputs (f) module. These include the time required to complete each layer, time required for total completion, equipment idle times, and equipment productivity.

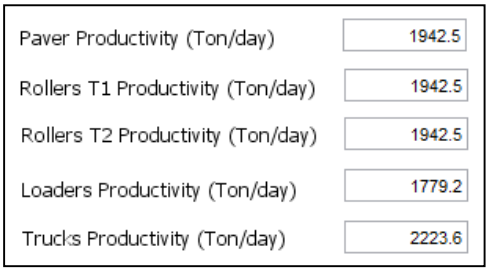

Figure 3.23: Productivity output measures

The simulator animation module aids in visualizing the progress of each layer and the actions of the equipment as they operate on site.

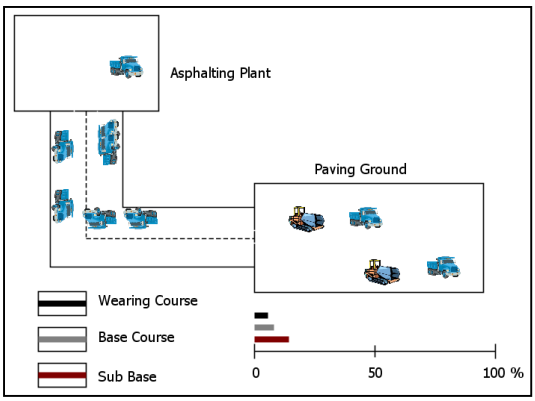

Figure 3.24: Process animation and layer progress

3.3.6 Simulation Outputs

The simulator produces, as its main output, a comprehensive set of data comprised of the artificial history of the simulated pipeline construction operations. The result is a data set showing the tasks performed on the corresponding products. For each product and task intersection, the data contains a start date and time and an end date and time. As the simulator runs and results are produced, they are summarized for the user using the following interface.

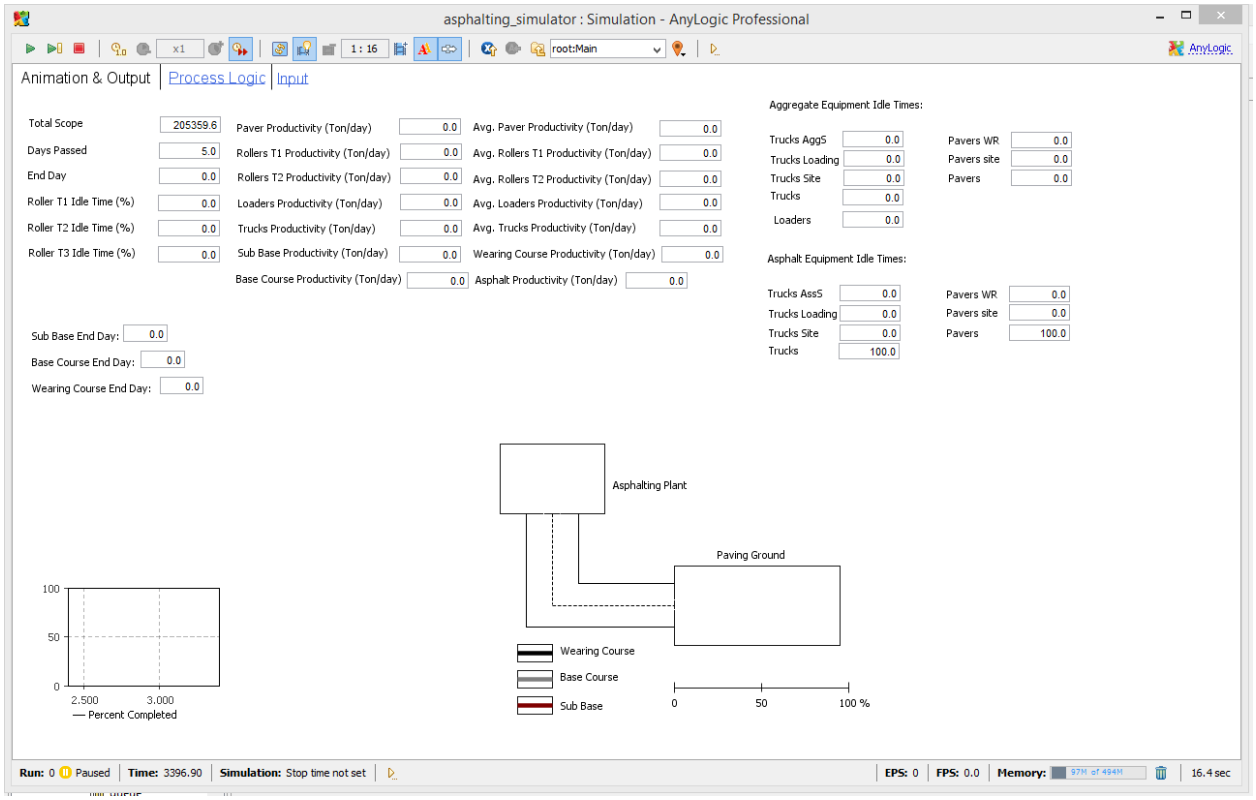

Figure 3.25: Asphalt Paving Simulator Outputs Interface

3.4 Pipeline Construction Simulation Model

This sections explores the third of the three simulation models studied for commonalities. It describes the development of the pipeline construction simulation model and looks at the data requirements, the DES model which forms the basis of this simulator, and the outputs of this model.

3.4.1 Background

Pipeline construction projects are by nature complex linear projects with dynamic properties that vary along the length and duration of the project. Although it is possible to use analytic techniques to plan and manage the performance of such projects, using simulation can provide us with an advantage in addressing the complexity and dynamicity involved in pipeline projects. A computer simulation of pipeline construction projects is a valuable predictive tool where we can vary inputs, collect and analyze outputs, and determine bottlenecks and sources of waste and delay. We can also determine the best preemptive measures to take to minimize risks of delays and cost overruns. It allows us to perform scenario-based planning and forecasting during execution. The target users of this simulator include project managers, planners and construction engineers. The pipeline construction simulator was developed with the aim of aiding stakeholders in simulating construction of pipelines at any point in the lifecycle of the construction project.

3.4.2 Simulator Design and Development

The first step was the creation of an abstract simulation model of the pipeline construction project covering the product and process definitions. This includes the major steps involved in pipeline construction such as receiving material, excavating sections, stringing, welding, lowering in, and backfilling.

3.4.3 Product Definitions

For this simulator, the pipeline construction project is broken down into pipeline sections, kilometers and pipes. Sections are made up of kilometers; kilometers are made up of individual pipes. The product definition process is straightforward as only those section, kilometer, and pipe characteristics relevant for simulating the direct pipeline construction activities are modeled. Figure 3.26 below depicts the pipeline product hierarchy.

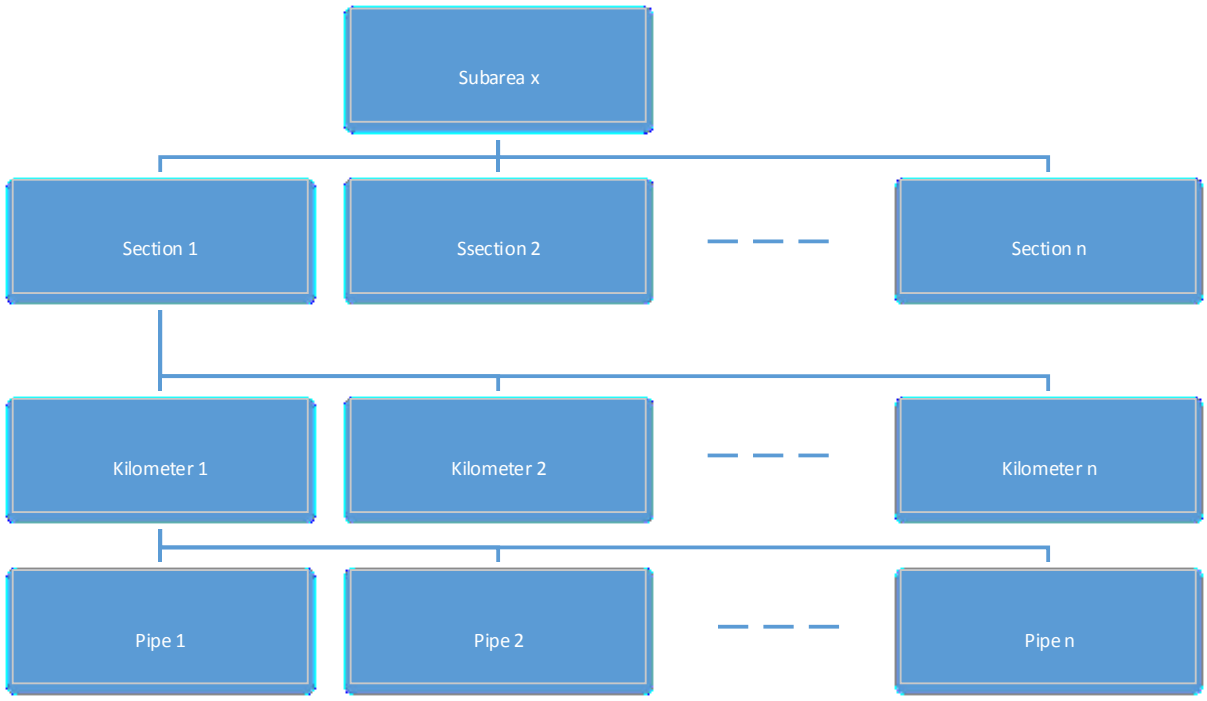

Figure 3.26: Pipeline Subarea/Section/Kilometer/Pipe Sample Hierarchy

3.4.4 Process Definitions

For process definitions the major activities and flows required for the simulator scope were defined. These include material handling activities and the main construction activities.

3.4.4.1 Material Handling

There are two material handling activities represented in the simulator, namely “material receiving” and “material transportation to site”. These are modeled using the DES flow in Figure 3.27 below. The source creates “material” entities based on the project shipping schedules. Material entities flow through the model going into the ReceiveMaterial task

first. This represents the material handling process where material is received, checked, and put into temp storage if required. From there the entities continue to the TransportToSite task which represents the process of delivering material from temporary storage to relevant site locations. This task takes into account truck availability, routes lengths, speeds, etc…

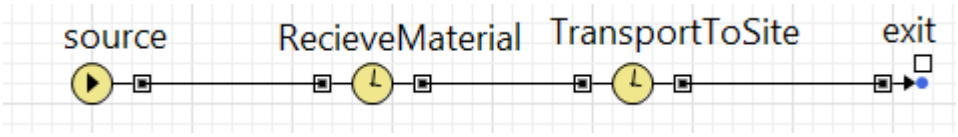

Figure 3.27: Material Handling DES Flow

3.4.4.2 Pipeline Construction Activities

The main construction activities modeled in the simulator include excavation, stringing, welding, lowering-in, and backfilling. The excavation activity is applied to the section level of the product hierarchy and is modeled in a separate flow shown in Figure 3.28 below. As it progresses, the excavation activity hands over excavated length of earth to the subsequent construction activities.

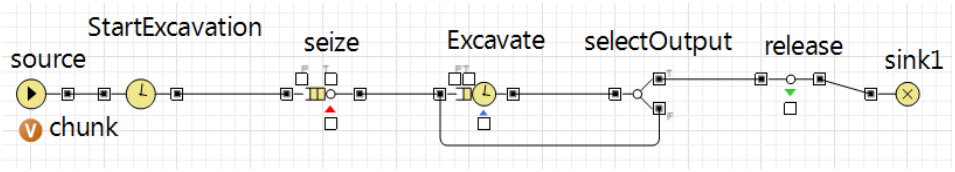

Figure 3.28: Excavation DES Model Flow

The remainder of the main construction activities of the pipeline simulator modeled here include stringing, welding, lowering in, and backfilling. Their DES model flow is depicted in Figure 3.29 below. Once sections are released from the excavation cycle they are handed over to the main construction activities in the form of their constituent pipes. From that point on, the entities traversing the flow of stringing, welding, lowering in and backfilling represent the pipes. Backfilling is applied to batches of pipes.

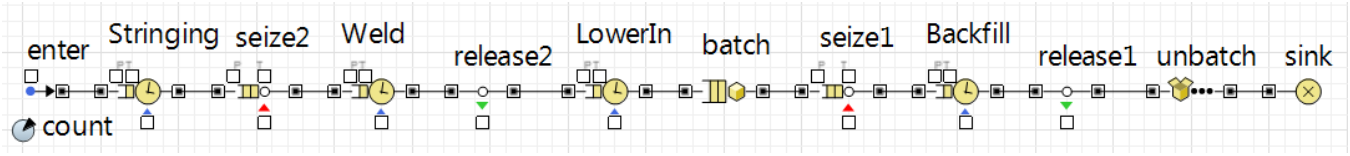

Figure 3.29: DES Model Flow of Balance of Tasks

3.4.5 Resource Definitions

Each of the pipeline construction activities modeled in the simulator is associated with a specific type of crew resource. These crews are composed of different types of workers. Table 3.5 shows typical crew compositions on a large pipeline construction project. Worker types are shared amongst the various crew types.

Crew Type	Worker Type 1	Worker Type 2	Worker Type 3	Worker Type 4	Equipment Type 1	Equipment Type 2	Equipment Type 3	Equipment Type 4
Excavation	1			1			1	1
Stringing	1	2		1	1	1		
Welding	1	2	2	1	1	1		
Lowering In	1	2		1				
backfilling	1			1				1

Table 3.5: Sample Crew Compositions

Table 3.6 depicts typical worker and equipment availability over time on a pipeline construction project. Workers and equipment available make up the required crews for the activities which are used as the resources to simulate the performance of a pipeline construction activity.

	Month							
Type	1	2	3	4	5	6	7	8
W1	21	23	25	25	25	25	25	25
W2	85	90	95	95	100	100	100	100
W3	80	82	85	86	104	104	104	104
W4	37	38	40	41	42	42	42	42
E1	1	1	2	3	3	3	3	3
E2	1	1	2	2	2	2	2	2
E3	1	1	1	1	1	2	2	2
E4	1	2	2	3	3	3	3	3

Table 3.6: Typical Worker Availability over Time

3.4.6 Model Structure

All the above pieces come together as in the structure shown in Figure 3.30 below.

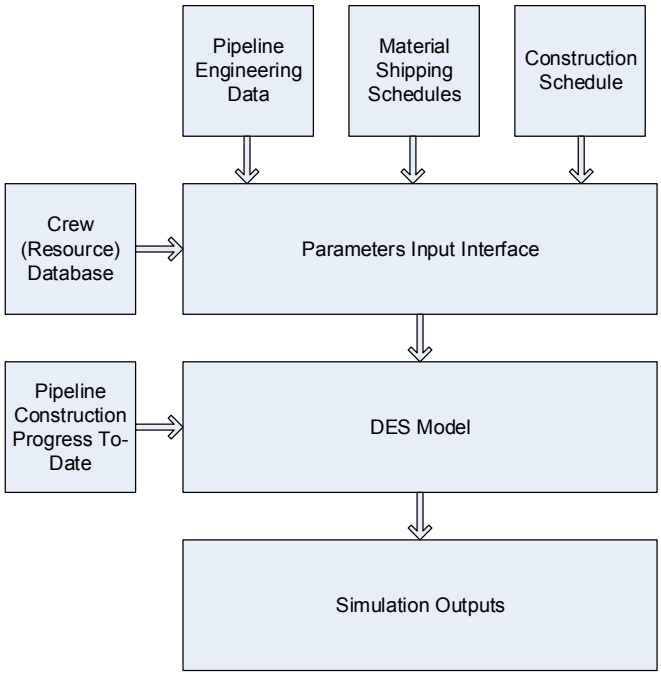

Figure 3.30: The Overall Model

The DES model was developed in the Anylogic simulation development environment. Figure 3.31 below depicts the DES model and the representative animated output graphics showing the application of the simulation model to sections of the pipeline.

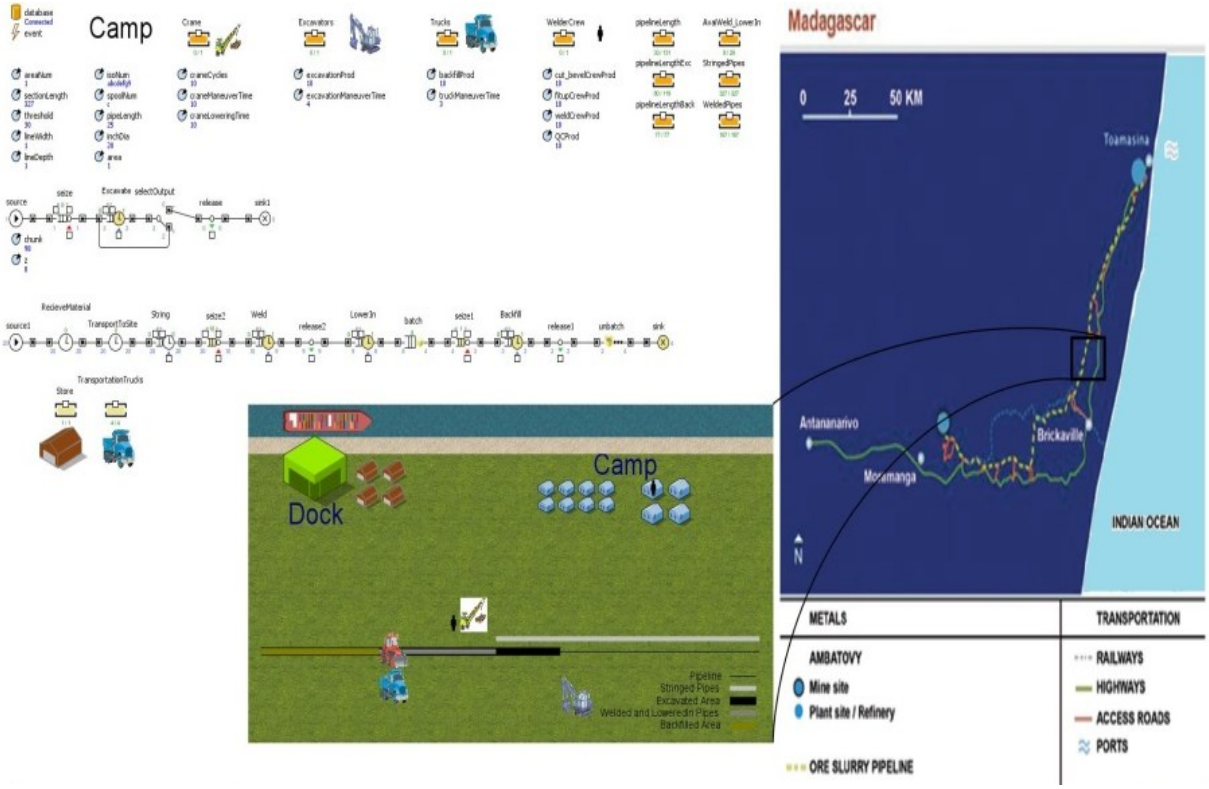

Figure 3.31: Pipeline Construction Simulator – DES Model

3.4.7 Simulation Outputs

The simulator produces, as its main output, a comprehensive set of data comprised of the artificial history of the simulated pipeline construction operations. The result set shown in Table 3.7 contains a sample data set showing a subset of the tasks (string, weld, lower-in) performed on the corresponding products. For each product and task intersection, the data contains a start date and time and an end date and time.

	A	B	C	D
1	Product	Task	SimStartTime	SimEndTime
2	0001/0101	String	2/10/12 7:55	2/10/12 8:05
3	0001/0101	Weld	2/10/12 8:05	2/10/12 8:15
4	0002/0102	String	2/10/12 8:05	2/10/12 8:15
5	0001/0101	LowerIn	2/10/12 8:15	2/10/12 8:25
6	0002/0102	Weld	2/10/12 8:15	2/10/12 8:25
7	0003/0103	String	2/10/12 8:15	2/10/12 8:25
8	0002/0102	LowerIn	2/10/12 8:25	2/10/12 8:35
9	0003/0103	Weld	2/10/12 8:25	2/10/12 8:35
10	0004/0104	String	2/10/12 8:25	2/10/12 8:35
11	0003/0103	LowerIn	2/10/12 8:35	2/10/12 8:45
12	0004/0104	Weld	2/10/12 8:35	2/10/12 8:45
13	0005/0105	String	2/10/12 8:35	2/10/12 8:45
14	0004/0104	LowerIn	2/10/12 8:45	2/10/12 8:55
15	0005/0105	Weld	2/10/12 8:45	2/10/12 8:55
16	0005/0105	LowerIn	2/10/12 8:55	2/10/12 9:05

Table 3.7: Sample Simulation Outputs

3.5 Verification and Validation

Credibility of a model which is expected to help manage construction projects is of utmost importance in order for stakeholders to accept and adopt the model. In order to verify the above models, both unit tests on each of the tasks within each model, and an overall system test were run at the time of development of each model. Outputs after the tests were compared with expected results based on predetermined inputs and ensured the models

and their components were correctly implemented. Validation of each of the models was done in two steps. First, each model's flow and logic were compared and confirmed against conceptual model design based on workflows and information collected from actual operations on construction projects. Subsequently, the models were each run with historical data from multiple projects and its outputs compared to historical results to ensure the models were behaving as per their design purposes.

3.6 Conclusion

This chapter presented the data requirements, the DES models developed for three different construction simulation models, and their data outputs. These simulation models helped stakeholders manage their activities and plan their resource requirements. The main benefits of the models are (1) predictive analysis of resource requirements, and (2) managing operations and forecasting resource and time requirements during project execution.

4 Construction Simulation Model Commonalities

4.1 Introduction

The preceding chapter explored three case studies that showcased the successful application of discrete event simulation to three distinct construction problems. Analysis of the three case-studies helped describe the basic common model components found in building and running such construction discrete event simulation models. Figure 4.1 illustrates how these common model components are organized and flow. The following sections detail the basic findings.

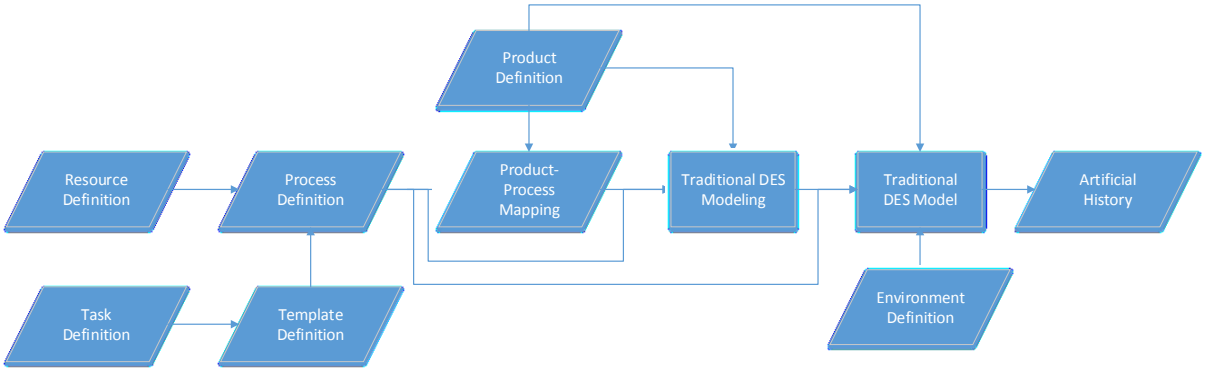

Figure 4.1: Conceptual Model of Common Model Structures

4.2 Product

In modelling a construction problem, one of the first questions to arise is "*What am I going to build?"* The answer to this, in our context, is a simple "*one or more products"*. To model building a specific product, it is necessary to identify the minimum relevant product properties required for representing the product in the simulation model context. In all three case studies, the basic common requirements were:

- Unique identifier
- Product Hierarchy
- Work Quantities

4.2.1 Product Hierarchy

Products in all three case study models were organized in a hierarchy starting from a top level product to be built and drilling down to various levels of detail depending on the model scope and analysis requirements.

- For the pipe fabrication simulation model, products had a hierarchical structure of spool – weld. To enable this capability, the model required a definition of the hierarchical relationship between the spool [parent] and the weld [child].

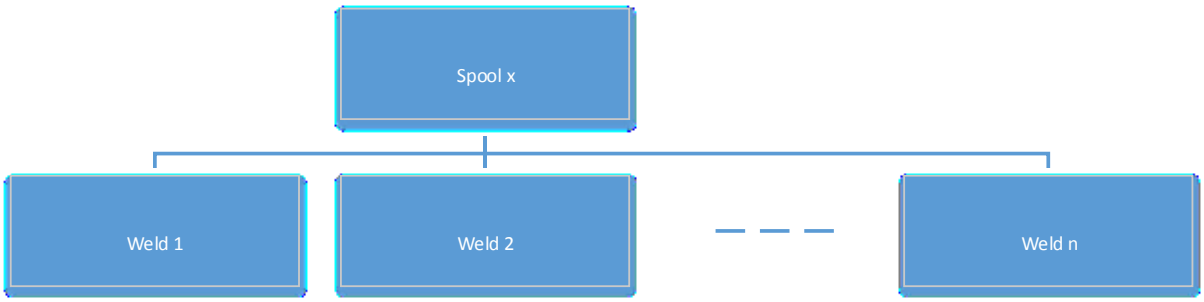

Figure 4.2: Pipe Spool Fabrication Spool/Weld Sample Hierarchy

- For the pipeline construction simulation model, products had a hierarchical structure of subarea – section – kilometer – pipe. To enable this capability, the model required a definition of the hierarchical relationship between the subarea [parent], section [child & parent], kilometer [child & parent] and the pipe [child].

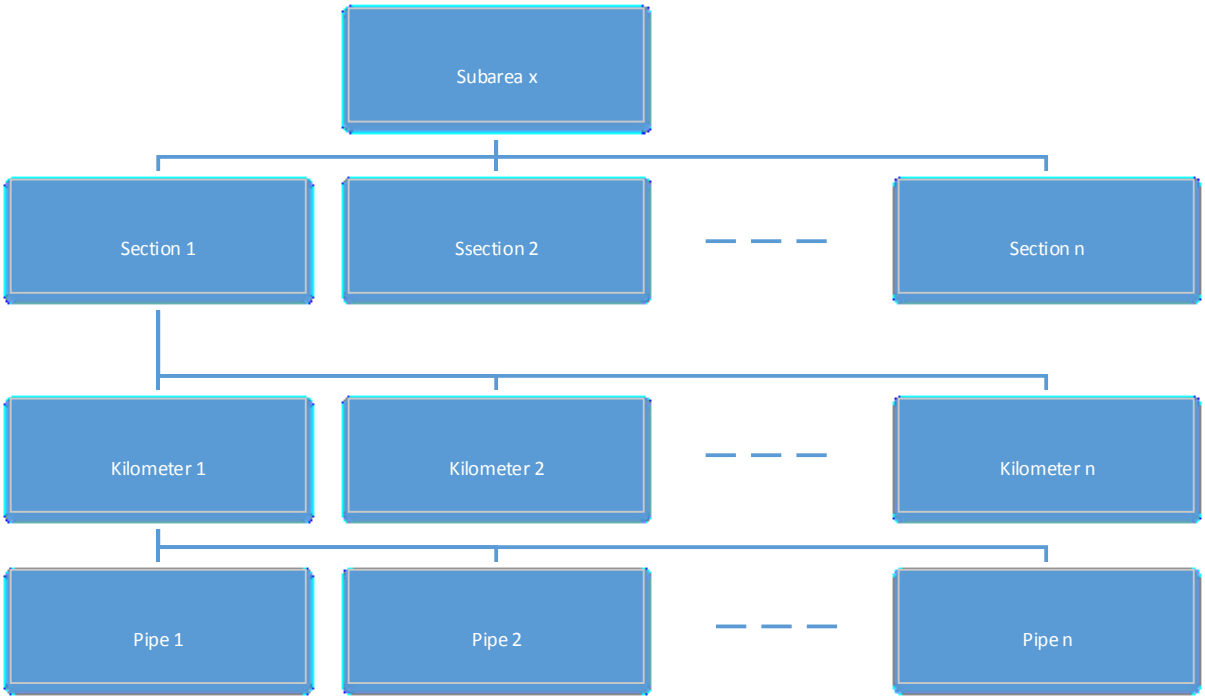

Figure 4.3: Pipeline Subarea/Section/Kilometer/Pipe Sample Hierarchy

- For the asphalt paving simulation model, the scope was defined by product layers (sub-base, base, wearing) and each of the layers had its constituent “lanes”. Internally lanes within a layer were broken down into sections based on the truck load sizes of aggregate or asphalt. This was a three level hierarchy with layers, lanes and sections. To enable this capability, the model required a definition of the hierarchical relationship between the layers [parent], lanes [child and parent] and sections [child].

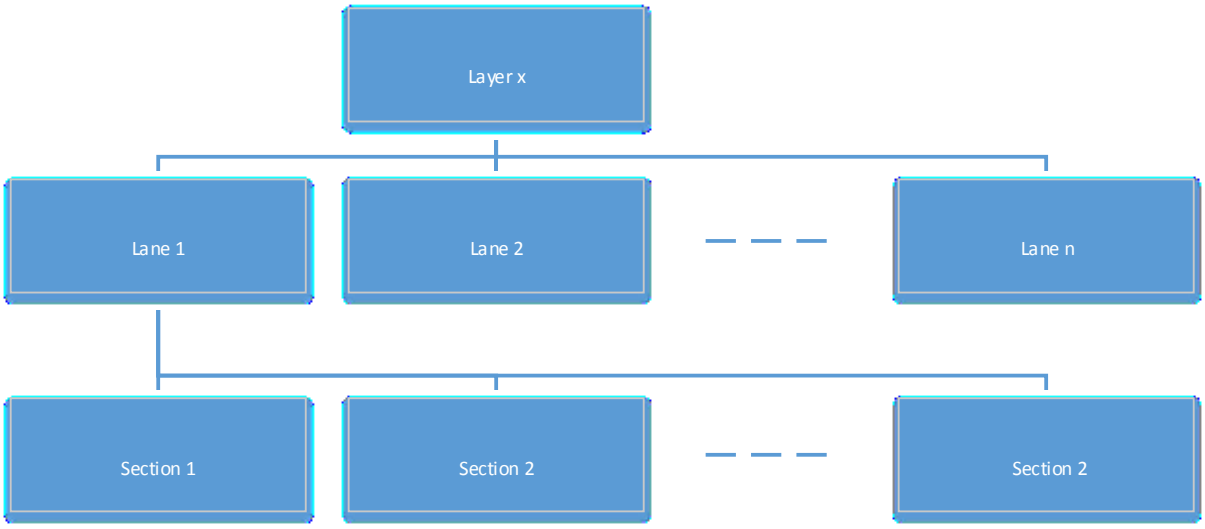

Figure 4.4: Asphalt Paving Subarea/Section/Kilometer/Pipe Sample Hierarchy

4.3 Process Definition

After identifying the products to be built, the next question that poses itself is "How do I build these products?" This is best answered by identifying the work that needs to be done to build the product and identifying who will do that work. For our simulation purposes, the work to be done can be described as a set of tasks that need to be performed by their corresponding crews.

4.3.1 Template of Tasks

This set of tasks is generally repeated for each specific product instance. These repeating construction tasks can be organized into a *template* comprising the tasks and their relevant properties, and the sequence in which they will occur. All three case study models demonstrated this behavior.

- In the pipe spool fabrication simulator, fabricating a spool is a process composed of several tasks that are repeated for the overwhelming majority of spools [i.e. certain spools are statistically exempt from PWHT and/or NDT requirements]: cut, bevel, fit-up, weld, QC release, PWHT, NDT, blasting, painting. This set of repeating tasks forms a template for pipe spool fabrication.

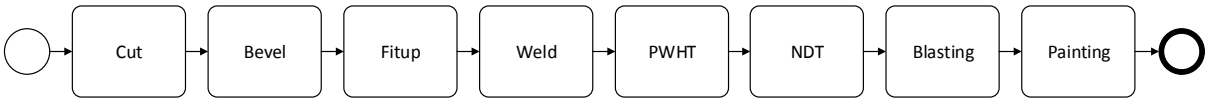

Figure 4.5: Pipe Spool Fabrication Tasks

- In the pipeline simulator, a repeating set of tasks is required for pipeline construction: excavate, string, weld, lower-in, backfill. This set of repeating tasks forms a template for pipeline construction.

Figure 4.6: Pipeline Construction Tasks

- In the asphalt paving simulator, laying the aggregate and asphalt is a repetitive process comprised of some basic tasks including: load material on trucks, haul material to site, lay it using paver, and roll it. This set of repeating tasks forms a template for asphalt paving.

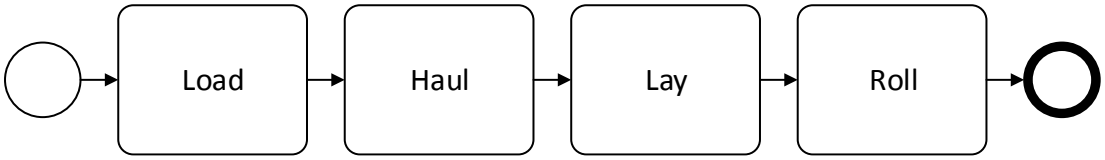

Figure 4.7: Asphalt Paving Tasks

4.3.1.1 Tasks

Tasks, in all three case studies, require a common basic set of properties to define them:

- Unique Identifier
- Descriptive Name: each task is described by a name specific to this type of activity or trade involved (excavation, welding, painting, etc…)

- Resource Type: to perform each task a specific type of resource is required – i.e. excavation requires earthworks crews, welding requires welding crews, painting requires painting crews, etc…
- Resource Quantity: a specific number (quantity) of the relevant resource is required for each task to be performed. In all three case studies, the assumption was to always have one crew assigned to perform a task on a product. In real life, such a crew would be composed of a group/team of resources required to perform the job. For our modeling purposes this team is abstracted as a single crew resource.

Following is a typical representative set of tasks with their relevant properties for each of the three case study models.

ID	Name	Resource Type	Resource Quantity
T1	Cut	CrewCut	1
T2	Bevel	CrewCut	1
T3	Fitup	CrewFitup	1
T4	Weld	CrewWelding	1
T5	PWHT	CrewPWHT	1
T6	NDT	CrewNDT	1
T7	Paint	CrewPainting	1

Table 4.1: Pipe Spool Fabrication Simulator – Tasks

ID	Name	Resource Type	Resource Quantity
T1	String	CrewString	1
T2	Excavate	CrewExcavate	1
T3	Weld	CrewWeld	1
T4	Lower-in	CrewLowerin	1
T5	Backfill	CrewBackfill	1

Table 4.2: Pipeline Construction Simulator – Tasks

ID	Name	Resource Type	Resource Quantity
T1	Load	CrewLoader	1
T2	Haul	CrewTruck	1
T3	Pave	CrewPaver	1
T4	Roll 1	CrewRoller1	2
T5	Roll 2	CrewRoller2	2

Table 4.3: Asphalt Paving Simulator – Tasks

4.3.1.2 Task Sequence

The second element of the task template, the task sequence, describes the logical sequencing dependencies between the tasks of the template as required to build the product. Task sequences can vary in complexity. Following is a typical sequence from the pipe spool fabrication model.

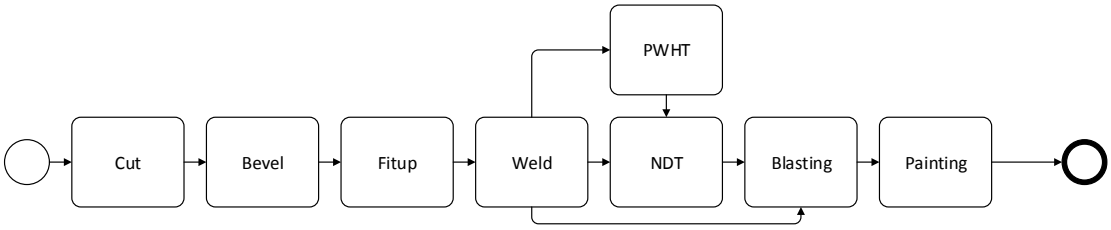

Figure 4.8: Pipe Spool Fabrication Simulator – Tasks Sequence Showing Welded Spool Proceeding on Three Different Paths to Completion

4.3.2 Resources

In all three case studies, resources require a common basic set of properties to define them.

- Unique Identifier
- Descriptive Name: each resource is described by a name specific to the type of work the resource performs (earthworks, welding, painting, etc…)
- Production Rate: each resource unit has a basic production capability per unit of time matched to the work requirement of the task applied to the product.
    a. In the asphalt paving simulator, an asphalt paver is set to lay a specific quantity of material per hour.

b. In the pipe spool fabrication simulator, a welding crew is able to weld a specific amount of inch-dia per hour.

c. In the pipeline construction simulator, a stringing crew is able to string a specific number of pipes per hour.

- Availability Schedule: availability of each resource type described either as a fixed number or as a varying number over time.

a. In the asphalt paving simulator, the number of asphalt paving crews was set as a fixed parameter for the duration of the simulation run

b. In the pipeline construction and pipe spool fabrication simulators, resources were available in varying numbers over the duration of the simulation run – i.e. welding crews: the number of available welding crews was either (1) set as a fixed number of crews available for the duration of the simulation run, or (2) increased gradually as the project reached its peak and then decreased steeply as the project reached its end.

ID	Resource	Available	Production Rate
APC1	Loaders	4	
APC2	Trucks	12	
APC3	Asphalt Pavers	3	
APC4	Rollers Type 1	6	
APC5	Rollers Type 2	6	
APC6	Rollers Type 3	6	

Table 4.4: Asphalt Paving Simulator - Resource List

ID	Resource	Available	Production Rate
PFC1	CrewCut	2	
PFC2	CrewFitup	2	
PFC3	CrewWelding	4	
PFC4	CrewPWHT	3	
PFC5	CrewNDT	4	
PFC6	CrewPainting	4	

Table 4.5: Pipe Spool Fabrication Simulator - Resource List

ID	Resource	Available	Production Rate
PLC1	CrewString	1	
PLC2	CrewExcavate	1	
PLC3	CrewWeld	1	
PLC4	CrewLowerIn	1	
PLC5	CrewBackfill	1	

Table 4.6: Pipeline Construction Simulator - Resource List

4.3.3 Product-Process Mappings

From the three case studies, it was found that there are two relevant basic product-process intersection definitions that are required: (1) the level at which the task will be applied to the product, and (2) the quantity of work for each product-task combination.

4.3.3.1 Product-Task Level

Products being modelled have tasks performed to complete them often at varying levels of detail. To better model and simulate building specific products, some tasks might be applied at the top level of the product hierarchy, while other tasks might require to be applied at a more detailed level of the constituents of the product. Accordingly, it was necessary in each of the case studies to have a definition of the proper level where a task is applied on a product. Moreover, since not all tasks were always required to be applied for building a product, the product-task mapping proved a viable structure to hold this information.

- In the pipe fabrication simulator, tasks in general were applied at the spool level except welding, which was applied at the weld level. For this model, not all spools required to have the PWHT and NDT tasks performed on them and this was reflected in the matrix mapping tasks to product levels as below as an empty space in the intersection box.

	Cut	Bevel	Fitup	Welding	PWHT	NDT	Painting
Spool #1	Spool	Spool	Spool	Weld	Spool	Spool	Spool
Spool #2	Spool	Spool	Spool	Weld		Spool	Spool
Spool #3	Spool	Spool	Spool	Weld			Spool
Spool #4	Spool	Spool	Spool	Weld	Spool	Spool	Spool
Spool #5	Spool	Spool	Spool	Weld	Spool		Spool
...	Spool	Spool	Spool	Weld	Spool	Spool	Spool
Spool #n	Spool	Spool	Spool	Weld	Spool	Spool	Spool

Table 4.7: Pipe Spool Fabrication Simulator – Spool-Task Level Mapping

- Similarly, in the pipeline construction simulator, tasks were applied at varying levels of product detail. Excavation was applied at the section level, while the remainder of the construction tasks were applied at the pipe level.

	Excavate	String	Weld	Lower-in	Backfill
Section #1	Section	Pipe	Pipe	Pipe	Pipe
Section #2	Section	Pipe	Pipe	Pipe	Pipe
Section #3	Section	Pipe	Pipe	Pipe	Pipe
Section #4	Section	Pipe	Pipe	Pipe	Pipe
Section #5	Section	Pipe	Pipe	Pipe	Pipe
...	Section	Pipe	Pipe	Pipe	Pipe
Section #n	Section	Pipe	Pipe	Pipe	Pipe

Table 4.8: Pipeline Construction Simulator – Section-Task Level Mapping

- In the asphalt paving simulator, scope definition was done at the level of the layer but task execution was simulated at the section level.

	Load	Haul	Pave	Roll
Section #1	Section	Section	Section	Section
Section #2	Section	Section	Section	Section
Section #3	Section	Section	Section	Section
Section #4	Section	Section	Section	Section
Section #5	Section	Section	Section	Section
...	Section	Section	Section	Section
Section #n	Section	Section	Section	Section

Table 4.9: Asphalt Paving Simulator – Section-Task Level Mapping

4.3.3.2 Product-Task Quantities

Each task to be performed on a product required a specific definition of the quantity of work relevant to the task to be performed in order for the simulator to depict the running of the

task. This is clearly the case in all three case studies. Following are three example representations based on the three case study models showing the quantity of work required for each product-task intersection.

	Cut	Bevel	Fitup	Welding	PWHT	NDT	Painting
Spool #1	Q-Cut-S1	Q-Bevel-S1	Q-Fitup-S1	Q-Welding-S1	Q-PWHT-S1	Q-NDT-S1	Q-Painting-S1
Spool #2	Q-Cut-S2	Q-Bevel-S2	Q-Fitup-S2	Q-Welding-S2		Q-NDT-S2	Q-Painting-S2
Spool #3	Q-Cut-S3	Q-Bevel-S3	Q-Fitup-S3	Q-Welding-S3			Q-Painting-S3
Spool #4	Q-Cut-S4	Q-Bevel-S4	Q-Fitup-S4	Q-Welding-S4	Q-PWHT-S4	Q-NDT-S4	Q-Painting-S4
Spool #5	Q-Cut-S5	Q-Bevel-S5	Q-Fitup-S5	Q-Welding-S5	Q-PWHT-S5		Q-Painting-S5
...	Q-Cut-S6	Q-Bevel-S6	Q-Fitup-S6	Q-Welding-S6	Q-PWHT-S6	Q-NDT-S6	Q-Painting-S6
Spool #n	Q-Cut-S7	Q-Bevel-S7	Q-Fitup-S7	Q-Welding-S7	Q-PWHT-S7	Q-NDT-S7	Q-Painting-S7

Table 4.10: Pipe Spool Fabrication Simulator – Spool-Task Quantity Mapping

	Excavate	String	Weld	Lower-in	Backfill
Section #1	Q-Excavate-S1	Q-String-S1	Q-Weld-S1	Q-Lower-in-S1	Q-Backfill-S1
Section #2	Q-Excavate-S2	Q-String-S2	Q-Weld-S2	Q-Lower-in-S2	Q-Backfill-S2
Section #3	Q-Excavate-S3	Q-String-S3	Q-Weld-S3	Q-Lower-in-S3	Q-Backfill-S3
Section #4	Q-Excavate-S4	Q-String-S4	Q-Weld-S4	Q-Lower-in-S4	Q-Backfill-S4
Section #5	Q-Excavate-S5	Q-String-S5	Q-Weld-S5	Q-Lower-in-S5	Q-Backfill-S5
...	Q-Excavate-S6	Q-String-S6	Q-Weld-S6	Q-Lower-in-S6	Q-Backfill-S6
Section #n	Q-Excavate-S7	Q-String-S7	Q-Weld-S7	Q-Lower-in-S7	Q-Backfill-S7

Table 4.11: Pipeline Construction Simulator – Section-Task Quantity Mapping

	Load	Haul	Pave	Roll
Section #1	Q-Load-S1	Q-Haul-S1	Q-Pave-S1	Q-Roll-S1
Section #2	Q-Load-S2	Q-Haul-S2	Q-Pave-S2	Q-Roll-S2
Section #3	Q-Load-S3	Q-Haul-S3	Q-Pave-S3	Q-Roll-S3
Section #4	Q-Load-S4	Q-Haul-S4	Q-Pave-S4	Q-Roll-S4
Section #5	Q-Load-S5	Q-Haul-S5	Q-Pave-S5	Q-Roll-S5
...	Q-Load-S6	Q-Haul-S6	Q-Pave-S6	Q-Roll-S6
Section #n	Q-Load-S7	Q-Haul-S7	Q-Pave-S7	Q-Roll-S7

Table 4.12: Asphalt Paving Simulator – Section-Task Quantity Mapping

4.3.4 Task Duration Calculation

Simulation task duration is derived from the quantity of work required to perform a task for a specific product and the production rate of the resource required for that specific task. Assuming T is the task duration, Q is the quantity of work required to be performed, and P is the production rate of the resource required for the task, the formula to calculate T is as follows:

$$T = \frac{Q}{P}$$

where

- the unit of measure of T is one which represents time (i.e. day, hour, minute, second, etc…)
- the unit of measure of Q is one which represents the type of work being performed (i.e. M3 for excavation, M2 for painting/tiling, inchdia for welding, etc…)
- the unit of measure of P is one which represents the type of work being performed per unit of time (i.e. M3/hour, M2/hours, inchdia/hours, inchdia/day, etc…)

with the provision that the same work unit of measure used in Q is used to describe the production rate per unit of time.

The model descriptive structures do not constrain the user to supply a unit of measure for the data being entered to describe the construction model. Instead the user is left to utilize the unit of measure of their choice with the stipulation that the unit of measure they use to describe the quantity of work matches the unit of measure describing the production rate of the resource required to perform the relevant task. It is also assumed that the user will utilize a unified unit of measure for time such that all tasks will have their task durations calculated using the same unit (i.e. day, hour, minute, second, etc…).

Two examples to illustrate the calculation of task duration

Example 1: Calculate the task duration of "beveling" a pipe spool in the pipe spool fabrication model:

Assume the following:

- Quantity of work (Q) for the bevel task for a spool is 45 inchdia
- Production rate (P) for the bevel resource is 10 inchdia per hour

Calculating the total "bevel" task duration for the given quantity of work and resource production rate:

$$T = \frac{Q}{P} = \frac{45\ inchdia}{10\ inchdia/hour} = 4.5\ hours$$

In the model descriptive data structures the user is expected to enter 45 for quantity of work and 10 for production rate without any need for units of measure.

Example 2: Calculate the task duration of "painting" a pipe spool in the pipe spool fabrication model:

Assume the following:

- Quantity of work (Q) for the paint task for a spool is 20 M2
- Production rate (P) for the painting resource is 4 M2 per hour

Calculating the total "paint" task duration for the given quantity of work and resource production rate:

$$T = \frac{Q}{P} = \frac{20\ M2}{4\ M2/hour} = 5\ hours$$

In the model descriptive data structures the user is expected to enter 20 for quantity of work and 4 for production rate without any need for units of measure.

4.4 Environment

In examining the three case study models, there were always external environment related elements that present as factors affecting the model but are not an inherent part of the

product or the process. Such factors might include construction schedules, work calendars as applied in different countries or seasons, varying shifts and work hours depending on location, holidays, seasonal productivity factors due to extreme weather conditions, labor/material/equipment supply fluctuations due to market conditions, cash flow issues, etc… These can all be modeled as add-ons to the basic simulation model based on need at time of implementing the model.

4.5 DES Model

Examining the creation of the discrete event model for each of the three case studies gave us two distinct types of findings. The first deals with the structure and flow of the DES model. The second deals with the modeling process and expertise required to develop the model.

4.5.1 DES Model Structure

The discrete event simulation (DES) model is the implementation of the product, process and product-process mapping components described above. The model depicts the task flow for building the products based on the template sequence defined. Each task is modeled with the relevant resource pool serving it. Products are represented by entities that traverse the flow and decompose to lower levels and recompose to higher levels as required. In all three case studies, a DES model was developed using a GUI-based DES modeling environment purposefully to describe the product and process at hand, and to be used specifically to answer questions about that situation. Following are the three DES models developed for the three case studies.

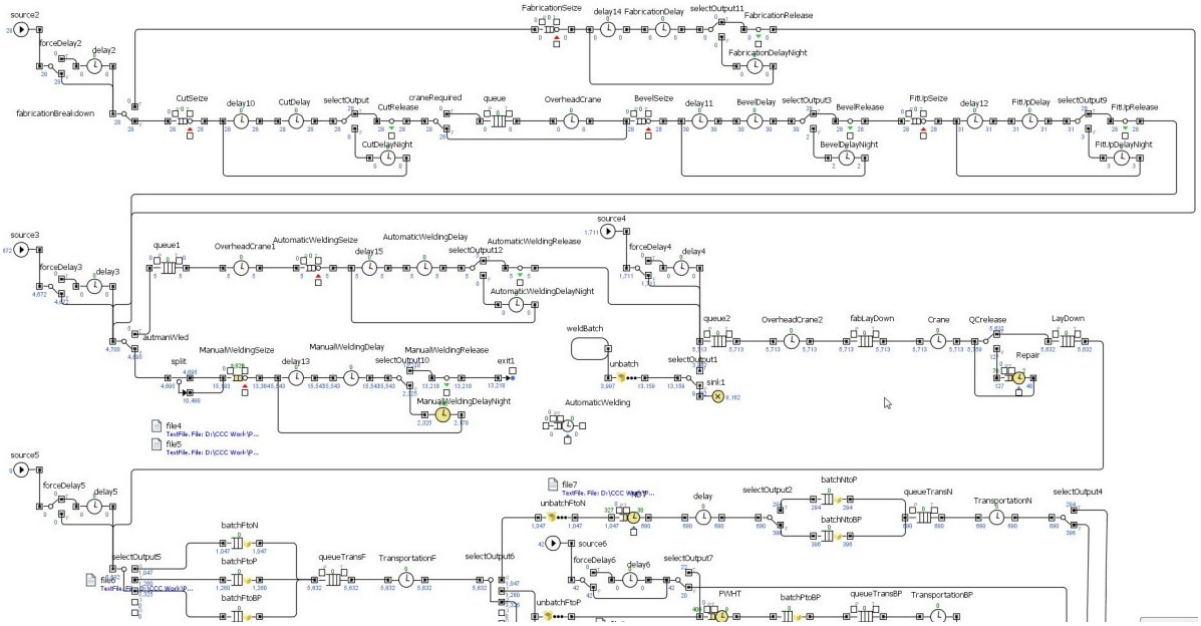

Figure 4.9: Pipe Spool Fabrication Simulator – DES Model

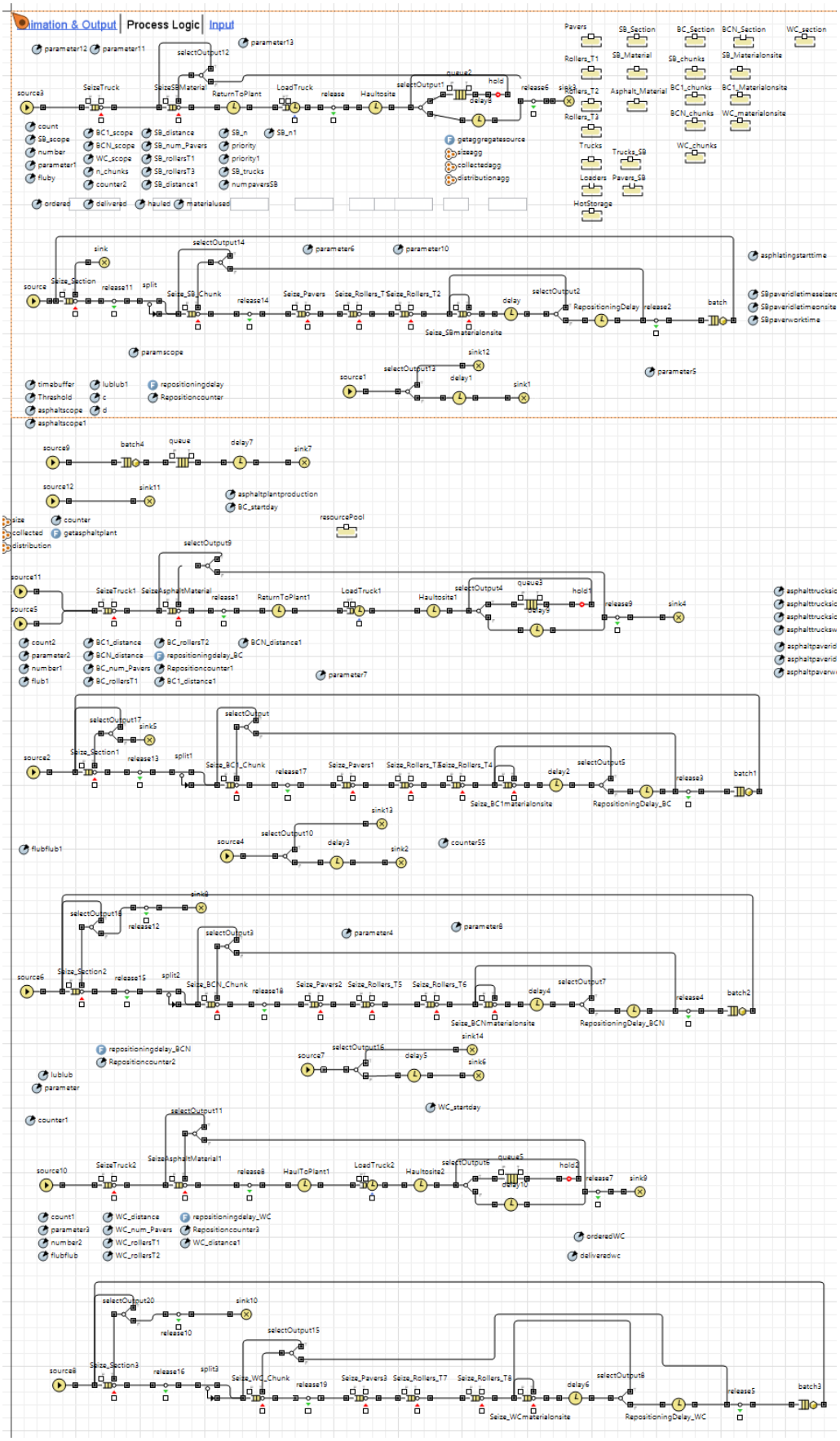

Figure 4.10: Asphalt Paving Simulator – DES Model

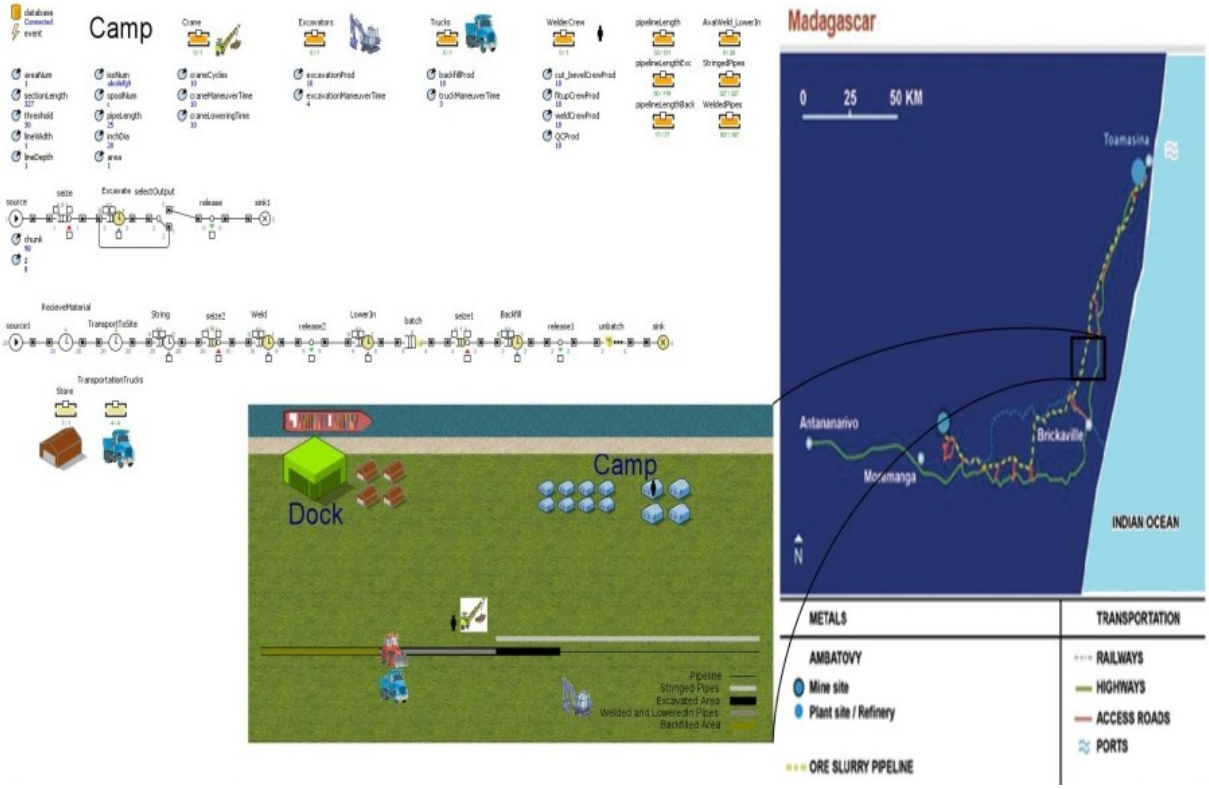

Figure 4.11: Pipeline Construction Simulator – DES Model

4.5.2 DES Model Development Requirements

Developing each of the above DES models required a simulation expert to take part in the effort. The simulation expert played a major role in transforming the information about the abstracted products and processes into the DES models with their structure and flow. Although stakeholders - end users including engineers and managers - on construction projects were able to provide the developer of the simulation models with all the functional requirements and describe the construction situation in detail, they were not able to take part in the actual model development process. They did not have the required modeling and simulation development experience required, nor the time and/or interest to learn the process.

4.6 Artificial History Produced by Simulation Run

In all three case studies, the simulation runs produce an artificial history which includes a data set with a row of data for each product-task combination at the product hierarchy level

where the task was applied. Each data row includes the product identifier, task identifier, task simulated start date and time, task simulated end date and time.

- In the pipe fabrication simulator, the artificial history includes data about the start and end of each of the tasks applied to the products at the spool level, except the weld where the task is applied at the weld level, and recorded accordingly. We can notice that the artificial history does not include any rows for tasks not being applied to a product such as PWHT for Spool #2, PWHT and NDT for Spool #3 and NDT for Spool #5.

Product ID	Task ID	Sim Start	Sim End
S1	Cut	S1 Cut Sim Start	S1 Cut Sim End
S1	Bevel	S1 Bevel Sim Start	S1 Bevel Sim End
S1	Fitup	S1 Fitup Sim Start	S1 Fitup Sim End
S1-A	Welding	S1-A Welding Sim Start	S1-A Welding Sim End
S1-B	Welding	S1-B Welding Sim Start	S1-B Welding Sim End
S1-C	Welding	S1-C Welding Sim Start	S1-C Welding Sim End
S1	PWHT	S1 PWHT Sim Start	S1 PWHT Sim End
S1	NDT	S1 NDT Sim Start	S1 NDT Sim End
S1	Painting	S1 Painting Sim Start	S1 Painting Sim End
S2	Cut	S2 Cut Sim Start	S2 Cut Sim End
S2	Bevel	S2 Bevel Sim Start	S2 Bevel Sim End
S2	Fitup	S2 Fitup Sim Start	S2 Fitup Sim End
S2-A	Welding	S2-A Welding Sim Start	S2-A Welding Sim End
S2-B	Welding	S2-B Welding Sim Start	S2-B Welding Sim End
S2	NDT	S2 NDT Sim Start	S2 NDT Sim End
S2	Painting	S2 Painting Sim Start	S2 Painting Sim End
S3	Cut	S3 Cut Sim Start	S3 Cut Sim End
S3	Bevel	S3 Bevel Sim Start	S3 Bevel Sim End
S3	Fitup	S3 Fitup Sim Start	S3 Fitup Sim End
S3-A	Welding	S3-A Welding Sim Start	S3-A Welding Sim End
S3-B	Welding	S3-B Welding Sim Start	S3-B Welding Sim End
S3	Painting	S3 Painting Sim Start	S3 Painting Sim End
S4	Cut	S4 Cut Sim Start	S4 Cut Sim End
S4	Bevel	S4 Bevel Sim Start	S4 Bevel Sim End
S4	Fitup	S4 Fitup Sim Start	S4 Fitup Sim End
S4-A	Welding	S4-A Welding Sim Start	S4-A Welding Sim End
S4-B	Welding	S4-B Welding Sim Start	S4-B Welding Sim End
S4-C	Welding	S4-C Welding Sim Start	S4-C Welding Sim End
S4	PWHT	S4 PWHT Sim Start	S4 PWHT Sim End
S4	NDT	S4 NDT Sim Start	S4 NDT Sim End
S4	Painting	S4 Painting Sim Start	S4 Painting Sim End
S5	Cut	S5 Cut Sim Start	S5 Cut Sim End
S5	Bevel	S5 Bevel Sim Start	S5 Bevel Sim End
S5	Fitup	S5 Fitup Sim Start	S5 Fitup Sim End
S5-A	Welding	S5-A Welding Sim Start	S5-A Welding Sim End
S5	PWHT	S5 PWHT Sim Start	S5 PWHT Sim End
S5	Painting	S5 Painting Sim Start	S5 Painting Sim End
…	…	…	…
Sn	Cut	Sn Cut Sim Start	Sn Cut Sim End
Sn	Bevel	Sn Bevel Sim Start	Sn Bevel Sim End
Sn	Fitup	Sn Fitup Sim Start	Sn Fitup Sim End
Sn-A	Welding	Sn-A Welding Sim Start	Sn-A Welding Sim End
Sn-B	Welding	Sn-B Welding Sim Start	Sn-B Welding Sim End
Sn	PWHT	Sn PWHT Sim Start	Sn PWHT Sim End
Sn	NDT	Sn NDT Sim Start	Sn NDT Sim End
Sn	Painting	Sn Painting Sim Start	Sn Painting Sim End

Table 4.13: Pipe Spool Fabrication – Artificial History Output Sample

- In the pipeline construction simulator, the artificial history includes data about the start and end of each of the tasks applied to each of the products at their appropriate levels – i.e. excavation is applied at the section level; welding is applied at the pipe level.

- In the asphalt paving simulator, the artificial history includes data about the start and end of each of the tasks applied to each of the products. The loading and hauling tasks are recorded in reference to material volumes being handled; asphalt laying is recorded in reference to sections of layers.

4.7 Conclusion

From the section above it can be observed that all three models examined displayed a common set of features that is always required in building a traditional DES model for solving construction problems. In addition to the common individual DES model features summarized in Table 4.14 below, it is to be noted that all three case studies showed a major commonality in nature where all three represented construction situations consisting of sets of repetitive construction activities applied to similar or diverse products. The pipe spool fabrication model depicted the application of a repetitive set of pipe fabrication activities to the various spools to be built; the asphalt paving model depicted the application of a set of activities over and over to similar road sections in the different, but abstractedly similar, road layers; the pipeline construction model depicted the application of a repetitive set of construction activities to multiple similar pipeline sections consisting of multiple similar constituents.

Category		Pipe Spool Fabrication	Asphalt Paving	Pipeline Construction	Comments
Product Definition	Product	X	X	X	All simulation models required a basic product definition
	Product Hierarchy	2 levels	3 levels	4 levels	Although product definitions might have multi-level hierarchies, tasks might still all apply at the same level
Process Definition	Tasks	X	X	X	The basic idea behind the methodology is that all the studied models were based on a set of repeating tasks with a specifc sequence
	Task Sequences	X	X	X	
	Resources	X	X	X	All simulation models required specific resource crews for each task. Still this may not always be necessary and some models might do away with resource requirements.
	Resource Availability	X	X	X	All simulation models required specific resource availability for each resource.
Product-Task Mappings	Task Level	Tasks apply at two levels	Tasks apply at lowest level	Tasks apply at two levels	This is an essential ingreditent defining the level at which tasks will apply on products
	Task Quantities	Task type related	Fixed - repeating	Task type related	This is an essential ingreditent defining the quantity of work required for each task applied o neach product
DES Model	Traditional DES Model				This is the basis of the simulation and is required for all simulators.
	Simulationist Required				
	New Project/Problem	X	X	X	A simulationist proved to be need through out the modeling process of a new project or problem to be solved.
	Changes	X	X	X	A simulationist was almost always required for any changes to the model design (i.e. task addition or deletion, change in sequence, change in resource type, etc...)

Table 4.14: Summary of Findings

5 Automated DES Model Creation Methodology

5.1 Introduction

Based on the findings in the preceding chapter, an automated DES model creation methodology for construction is being proposed to help overcome the threshold which exists between end users on construction projects and the use of simulation as a tool in answering questions and solving construction related problems. Current traditional practices make it hard for end users to adopt simulation on construction projects. A simulation practitioner is almost always required to develop a simulation model or adapt an existing one to suit evolving needs in a construction environment. The methodology proposed and being described in this work aims at enabling end users on construction projects to utilize simulation in solving some of their problems by eliminating the requirement of having a simulation practitioner on hand taking part in the problem solving process every time.

Using the new methodology, and instead of building DES models in the traditional manner, a simulation practitioner would now develop a DES model compiler system using which the required DES model is automatically created from data describing the process to be simulated. Developing the DES model compiler is an exercise that the simulation expert will perform only once. The compiler can then be re-used as needed by construction end users by simply identifying the data that is required to describe the model, entering it into the system, and running the system to automatically compile a DES simulation model of the process, run the simulation model, and produce the required artificial history. It will not be necessary to redevelop the system if the construction requirements change. Such construction related changes are now dealt with through the data describing the model, and not through amendments to a DES model as used to be the case with the traditional

approach. Using this methodology, end users are only dealing with data that they handle and use day to day, is very familiar to them, and which is required in a simple format in line with their daily work operations.

The common findings amongst all three case studies as detailed in the preceding chapter can be organized into two categories. The first category is the set of inputs and outputs of the simulation process. The inputs include the product, process, and environment definitions along with any mappings between them. The outputs are the result set produced by running the simulator. The second category includes the traditional DES model and the traditional modeling process required to produce that DES model. Following is a color coded conceptual model of the common components. The green components are the common model data structures needed for describing the construction model. The red components represent the traditional modeling process and the traditional DES model resulting from the traditional modeling process performed by the simulation expert every time a new model was needed, or an existing model needed to be changed. The environment component may be included in the development process if needed; for our purposes we will assume that the decision on structural definition and functional integration of the environment component within the developed system will be left to the developers and end users of the simulation system.

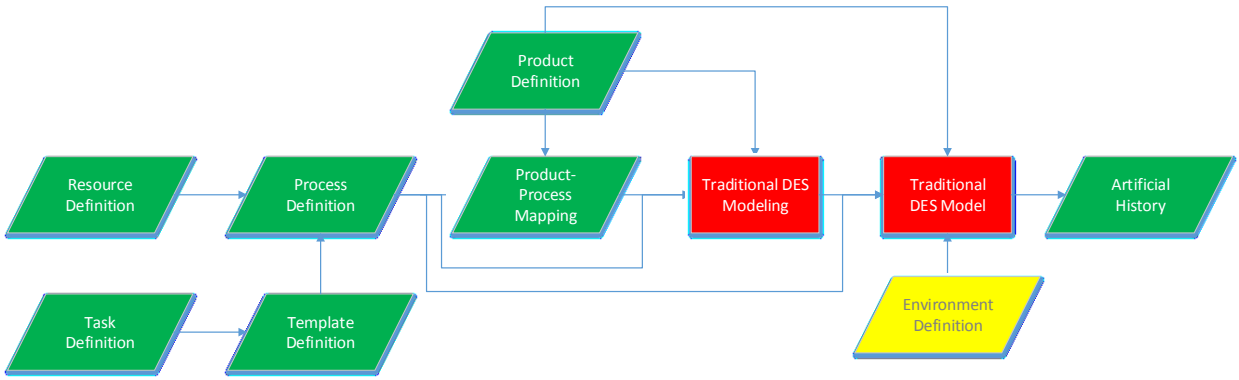

Figure 5.1: Conceptual Model of Traditional Modeling Methodology

The conceptual model of the new methodology is depicted in Figure 5.2 below.

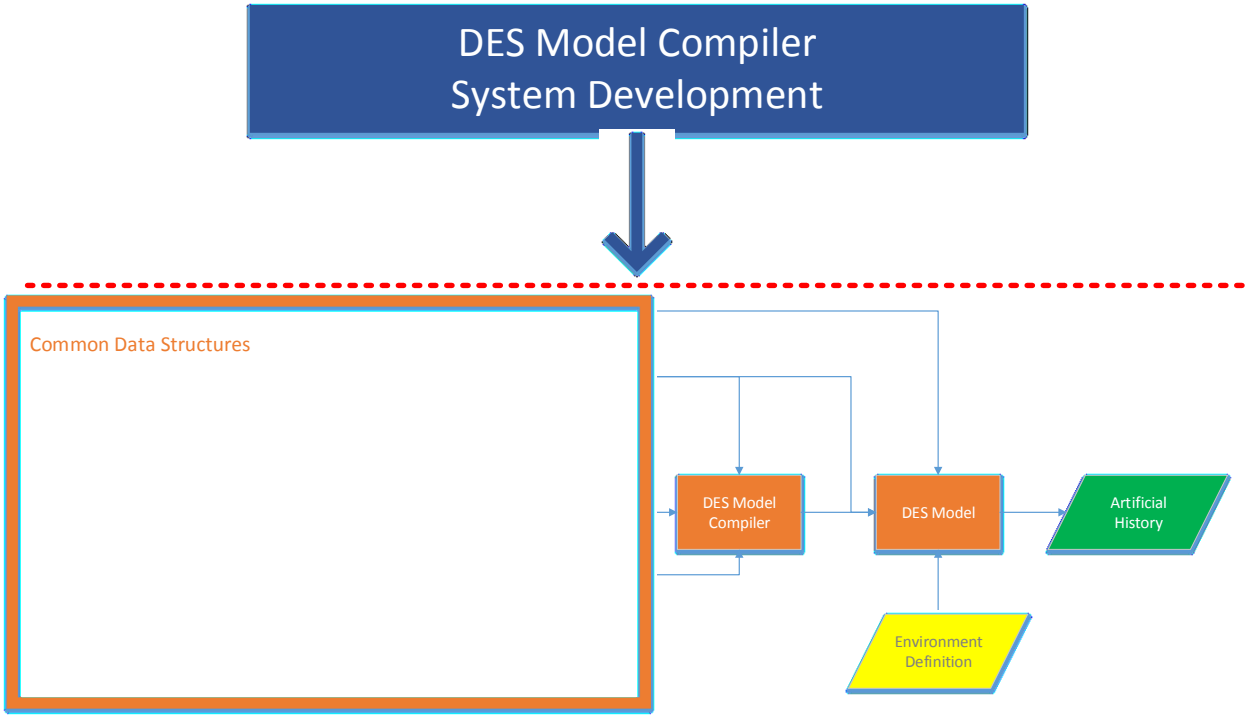

Figure 5.2: Conceptual Model of New Modeling Methodology

The "DES Model Compiler System Development" process is a software development exercise performed once to develop the system, including the required data structures and algorithms. The result is a set of data structures hosting the required common data components, and a set of classes and procedures which constitute the "DES Model Compiler" component. The data structures, classes and algorithms required for this methodology are described in detail in the sections that follow.

5.2 Automated DES Model Creation Methodology

The methodology being proposed in this work is a move from the traditional procedural method of construction of DES models to a declarative method of construction of DES models. The methodology involves the creation of a DES model compiler that utilizes model descriptive data structures to produce the relevant required DES simulation model. This is a process where, for each product-task intersection, a basic DES model structure is instantiated to represent that intersection. This basic DES structure is composed of a source element, a delay element, a sink element, and a resource pool element. Figure 5.3 depicts this basic DES model structure.

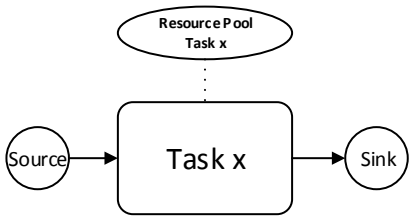

Figure 5.3: Basic DES Model Structure

Each instance of the basic DES Model represents the application of a task to a product at the appropriate product level as derived from the Product-Task-Level mapping. The resource pool element maps to the resource crew required by the task. The delay element represents the duration required for the resource crew to perform the quantity of work as derived from the Product-Task-Qty mapping. Accordingly, for each product to be built, a number of basic DES models are instantiated corresponding to the number of tasks required to build the product. Figure 5.4 below depicts this process.

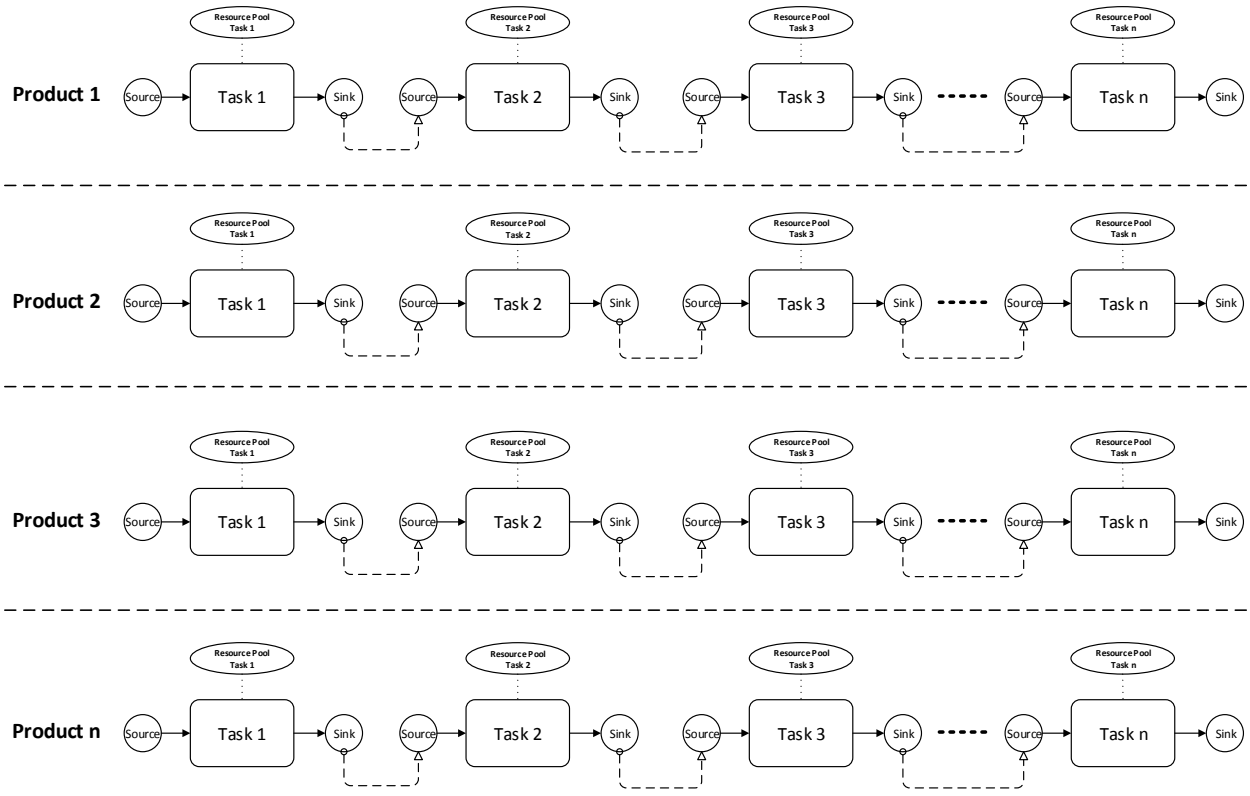

Figure 5.4: Virtual Basic DES Model Created for every Task for every Product to be Built

These basic DES models are not all run immediately. Instead, only the model corresponding to the first task in the task template has an arrival entity created for it and injected into its source. Once the entity reaches the delay element, an attempt is made to capture the resource. If resource capture is unsuccessful, the DES model will wait until the resource becomes available for capture. If, on the other hand, resource capture is successful, the delay is processed and the start time for the task is noted. Once the duration for the delay passes, the entity proceeds from the delay element to the sink element. Upon exiting the delay element, the end time of the task is noted and a data record representing the completion of the run of this DES mode containing the product name, task name, start time, and end time, is sent to the result set. Upon entering the sink element, a check is made as to

whether a next task exists in the task template for this product. If that check is positive, an arrival in the DES model representing the next task for this product is created and DES entity flow starts within that model. This process continues for all DES models until all have been processed and their data recorded in the result set. To facilitate the dynamic DES model creation process described above, a set of classes describing the data structures, dynamic object model structure, and algorithms is required. These classes are described in the following sections.

5.2.1 Data Structure Classes

The data structures host the different data items required including the product, process, task template, tasks, sequence, resources, product task mappings, and the simulation result set. For our purpose, these components are defined as classes using the UML standard. Following is a description of each component, and a UML diagram of its class definition.

5.2.2 Product

The “Product” class represents the abstracted products to be constructed. It has the basic fields required to identify a product for the purpose of the simulator. These fields include:

- Name: a string which identifies the product
- Nominal Quantity: a quantity which describes the major type of work of the finished product
- Children: an array which holds references to children of the current product in the hierarchy
- Parent: a string which identifies the parent of this product in the hierarchy

Figure 5.5: Product Class

5.2.3 Task Template

The "TaskTemplate" class represents the abstracted combination of tasks and their sequence required to build the "Product". It has the basic fields and methods required to identify the "TaskTemplate" for the purpose of the simulator. These fields and methods include:

Fields

- Name: a string which identifies the task template

Methods

- findFirst(): this method carries the code for finding the first task in the sequence for a specific product
- findNext(): this method carries the code for finding the next task in the sequence for a specific product and for a specific current task

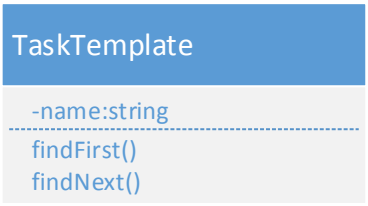

Figure 5.6: TaskTemplate Class

5.2.3.1 Task

The "Task" class represents the abstracted tasks to be performed to build the products. It has the basic fields required to identify a product for the purpose of the simulator. These fields include:

- Name: a string which identifies the task.
- Template ID: a string which identifies the template to which this task belongs
- Res_Qty: a number which identifies the number of resource crews required.
- Res_Type: a string which identifies the type of resource crew required for this task.

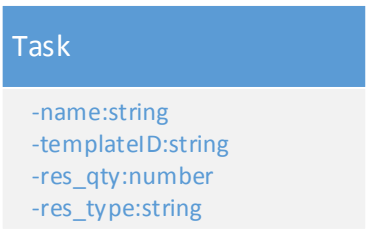

Figure 5.7: Task Class

5.2.3.2 Sequence

The "Sequence" class represents the abstracted logical order the tasks will follow through to product completion. It has the basic fields required to identify the predecessor and successor tasks for a template. These fields include:

- Template ID: a string which identifies the template to which this task sequence belongs
- Predecessor: a string which identifies the predecessor task
- Successor: a string which identifies the successor task

Figure 5.8: Sequence Class

5.2.4 Resource

The “Resource” class represents the abstracted resource crews required to perform the tasks. It has the basic fields required to identify the resource and its production capability. These fields include:

- Name: a string which identifies the resource crew
- Prod_Rate: a number which identifies the resource crew production rate over time

Figure 5.9: Resource Class

5.2.4.1 Resource Availability

The “ResourceAvailability” class represents the abstracted resource availability schedule. It has the basic fields required to describe the number of resource crews available over time of each type of resource. This may be used to describe a fixed crew availability for the duration of the simulation (i.e. by setting the “From” and “To” fields to null values). These fields include:

- Res_Name: a string which identifies the resource crew
- From: a datetime which identifies the start of the period being described
- To: a datetime which identifies the end of the period being described
- Qty: a number which describes how many crews of the said resource are available in the time window formed between the From and To datetime values

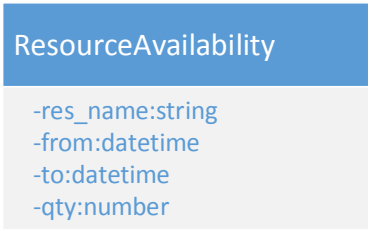

Figure 5.10: ResourceAvailability Class

5.2.5 Product-Task-Level Mapping

The "ProductTaskLevel" class represents the abstracted mapping between products and tasks describing the level of the product at which each task applies. The fields in this class include:

- Prod: a string which identifies the product highest level
- Task: a string which identifies the task
- Level: a string which identifies the product level within the product hierarchy

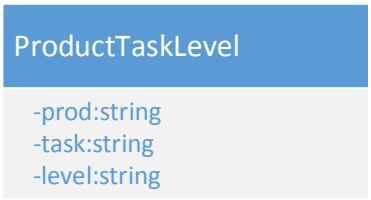

Figure 5.11: ProductTaskLevel Class

5.2.6 Product-Task-Quantity Mapping

The "ProductTaskQuantity" class represents the abstracted mapping between products and tasks describing the quantity of the product to be performed by the task at the appropriate product level. The fields in this class include:

- Prod: a string which identifies the product highest level
- Task: a string which identifies the task
- Qty: a number describing the quantity of work to be performed by the Task at the appropriate product level

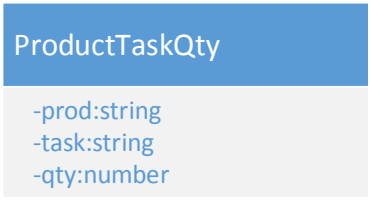

Figure 5.12: ProductTaskQty Class

5.2.7 Result Set

The "ResultSet" class represents the abstracted simulation run result set produced by running the dynamically create simulation model. It has fields that describe the data output of the simulator which include:

- Prod: a string which identifies the product
- Task: a string which identifies the task performed on the product
- StartTime: a datetime which identifies the start date and time of the task
- EndTime: a datetime which identifies the end date and time of the task

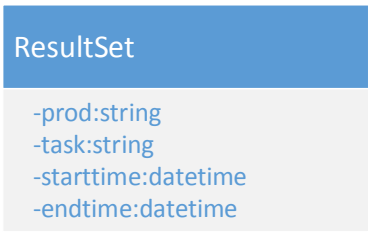

Figure 5.13: ResultSet Class

5.2.8 Basic DES Model Class

The "DESObject" class represents the abstracted dynamic discrete event simulation model for a product and a task being applied to it as shown in Figure 5.14. It has fields and methods including:

Fields

- Name: a string which identifies the product-task combination
- Prod: a string which identifies the product

- Task: a string which identifies the task performed on the product
- Res: a string which identifies the resource crew required
- Delay: a number which identifies the duration of the task
- StartTime: a datetime which identifies the start date and time of the task
- EndTime: a datetime which identifies the end date and time of the task

Methods

- createArrival():this method carries the code for creating a new arrival for the discrete event simulation flow within the DESObject

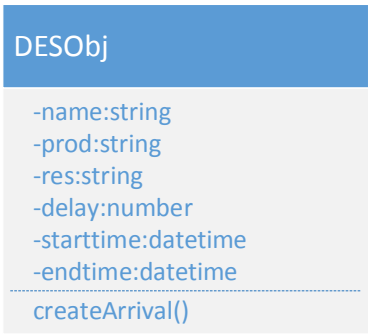

Figure 5.14: DESObject Class

5.2.9 Algorithms

The algorithms required include procedures for creating the DES models at the product task intersections, creating initial arrivals for the first-task-model for each product, handling progression between predecessor and successor tasks for each product, and recording result data in the result set structure. The following sections detail these procedures.

5.2.9.1 Creation of DES Models for Product Task Intersections

This procedure is executed at the beginning of the Dynamic DES Model run. It creates all the DES models for each product task intersection. It is composed of three nested loops. The outermost loop goes through every product in the collection of products being simulated.

The next loop goes through all the tasks that apply to this product and derives the appropriate level at which the task is to be applied to the product. The innermost loop goes through the product constituents as determined by the level at which the task is to be applied. If the task is applied at the top level, then the product top level is the only constituent, otherwise, its children at the application level are enumerated and used. Inside the three nested loops is a call to create a DES model representing the intersection of product and task and give it its properties including the DES model name, product name, task name, resource required and delay value. Figures 5.15 and 5.16 depict the pseudo-code and the workflow for this procedure.

```
For every Product
    For Every MatchedTask in ProductTaskLevelMap
        For every ProductConstituent in ProductHierarchy@MatchedTaskLevel
            Create DESObj in DynamicModel
            DESObj.Product = ProductConstituent.Name
            DESObj.Task = MatchedTask.Name
            DESObj.Name = Concatenate(DESObj.Product,DESObj.Name)
            DESObj.Resource = MatchedTask.Resource
            DESObj.Delay = ProductConstituent.Qty / Resource.ProductionRate
        Next
    Next
Next
```

Figure 5.15: Pseudo-Code for Creation of DES Models at Product task Intersections

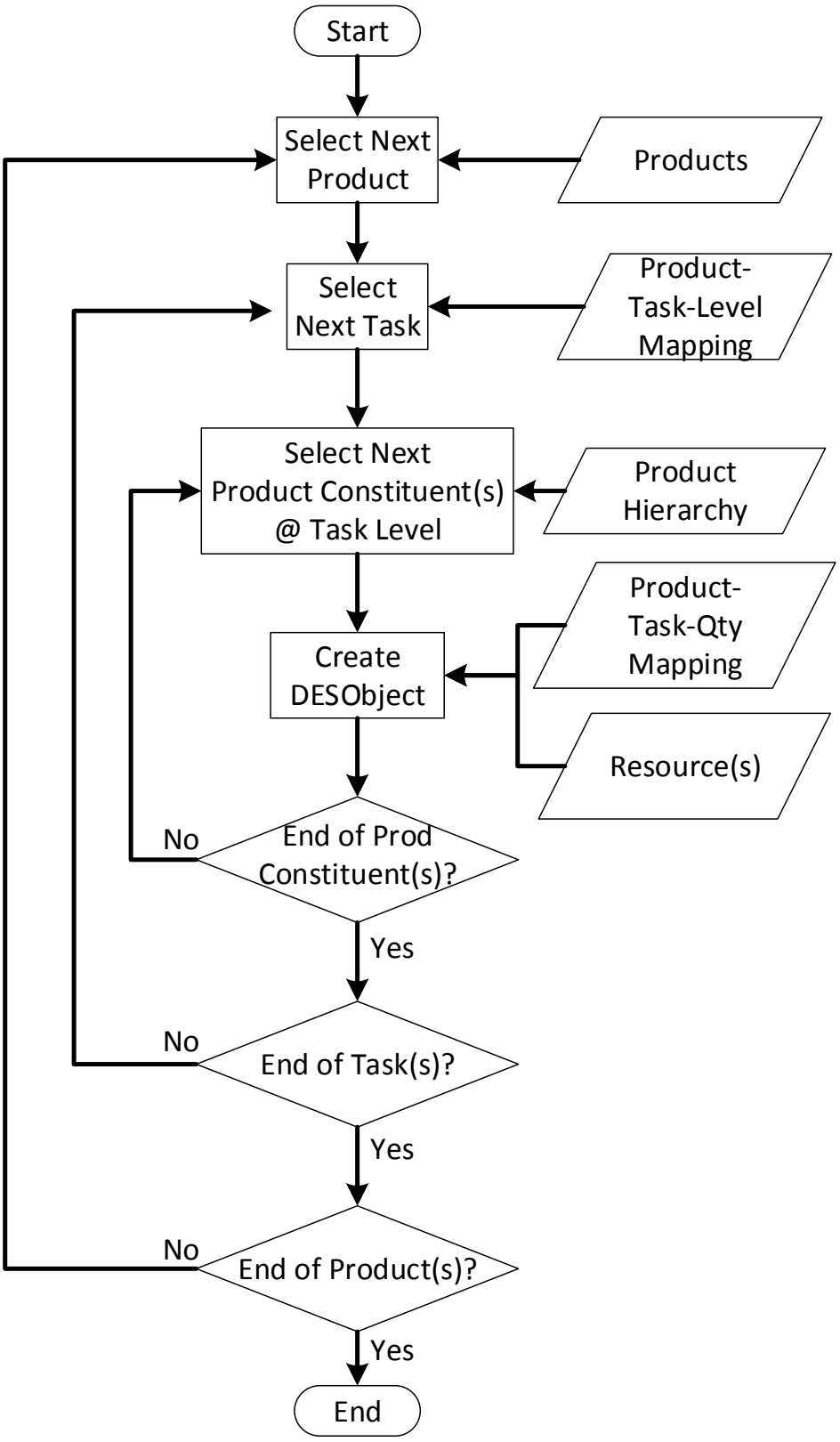

Figure 5.16: Workflow for Creation of DES Models at Product task Intersections

5.2.9.2 Creation of Initial Arrivals for the First-Task-Model for Each Product

This procedure is executed once all the DES models for the product task intersections have been created. It creates an arrival for each DES model representing the first task for each of the products as per the sequence in the task template. It is composed of a single loop that goes through all the DES models whose task value is equal to the first task value for their corresponding product, as derived from the task template, creating for each an entity arrival. Figures 5.17 and 5.18 depict the pseudo-code and workflow for this procedure.

```
For every DESObj in DynamicModel _
        where DESObj.Task = TaskTemplate.FindFirst(DESObj.Product)
        DESObj.CreateArrival
Next
```

Figure 5.17: Pseudo-Code for Creation of Initial Arrivals for the First-Task-Model for Each Product

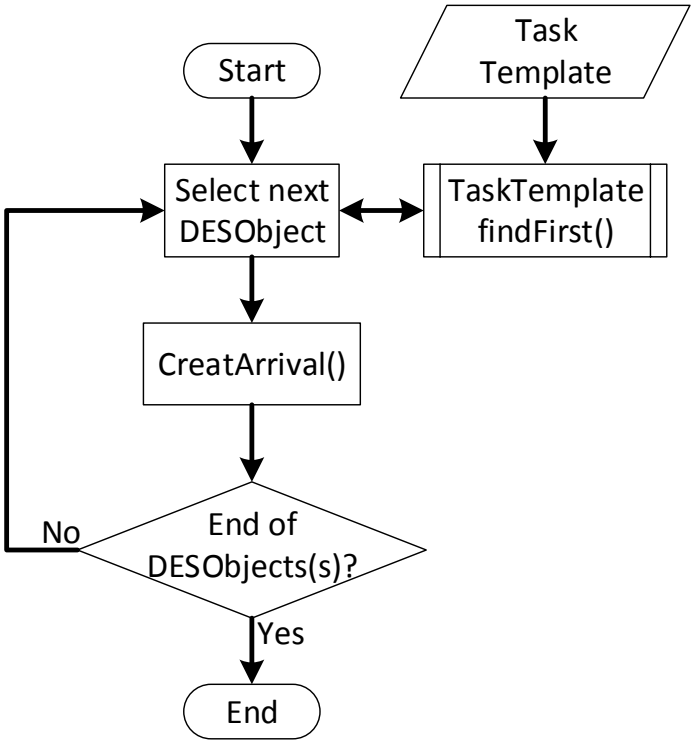

Figure 5.18: Workflow for Creation of Initial Arrivals for the First-Task-Model for Each Product

5.2.9.3 Record Start of Task

This procedure is executed in-line with the intrinsic DES model flow execution. It stores the start time of the execution of a task upon successful capture of the required resource from the relevant resource pool.

```
On DESObj.Delay.ResourceCaptureSuccessful
        DESObj.StartTime = Now()
End
```

Figure 5.19: Pseudo-code for Record End of Task and Create Arrival for Next Task

5.2.9.4 Record End of Task and Create Arrival for Next Task

This procedure is executed every time the entity exits the delay element and enters the sink elements inside the DESObj model. It notes the current time as the endtime for executing this task on the product, writes result data to the result set including product name, task name, start time and end time, and then finds the next task in the sequence and creates an arrival in its corresponding DESObj model.

```
On DESObj.Sink.Enter
        DESObj.EndTime = Now()
        With DESObj
                OutputToResultSet
                        .Product
                        .Name
                        .StartTime
                        .EndTime
        endwith
        NextDESObjName = TaskTemplate.FindNext(DESObj.Name, DESObj.Product)
        If Exists(NextDESObjName) then
                Set NextDESObj = DESObj where DESObj.Name = NextDESObjName
                NextDESObj.CreateArrival
        Endif
End
```

Figure 5.20: Pseudo-code for Record End of Task and Create Arrival for Next Task

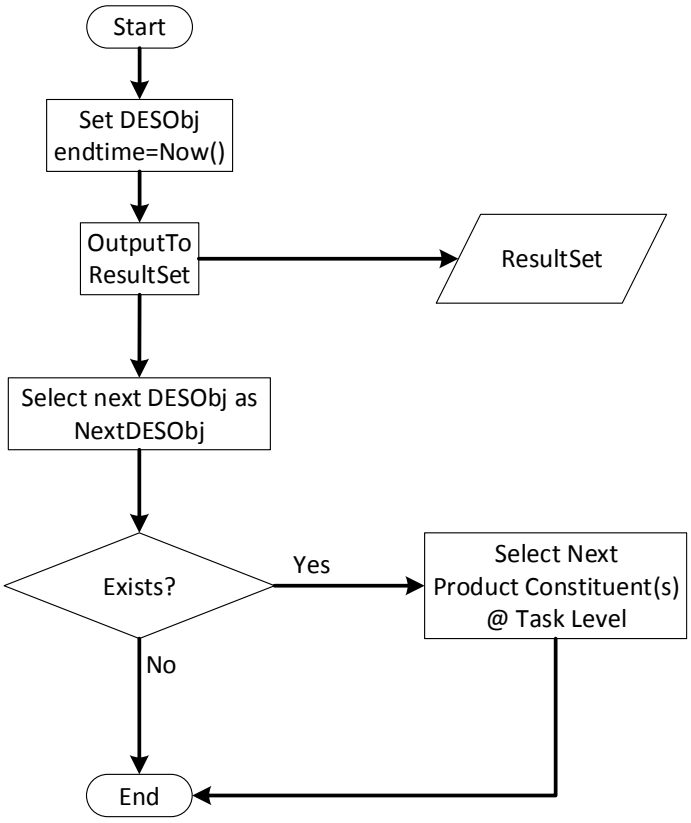

Figure 5.21: Workflow for Record End of Task and Create Arrival for Next Task

5.3 Methodology Prototype

The combination of structures and algorithms described above is a methodology aimed at enabling a simulation practitioner to develop a simulation system which allows end users to create and run simulations by entering data describing their model into the system. The following sections describe (1) prototyping of the methodology to produce a DES model compiler and applying it to model a simple construction example of building cast-in-place foundations; and (2) re-use of the prototype for the redevelopment of the pipe spool fabrication model explored in the case studies.

5.3.1 Simple Construction Example

For the purpose of demonstrating how the methodology comes together, we will assume a construction scenario where an end user on a construction project would like to validate the duration required for building a set of cast-in-place concrete foundations. Modeling the

building of the foundations is a simple application of the proposed methodology. The following sections depict the different data required to describe the model.

5.3.1.1 Product Definitions

First, we start with product definitions for the foundations. Table 5.1 shows the product names and nominal quantities for each of the foundations. For this modeling exercise the foundations do not have parents or children.

Product Name	Nominal Qty (Volume - m3)	Parent	Children
F1	4.00	-	-
F2	4.00	-	-
F3	6.00	-	-
F4	6.00	-	-
F5	4.00	-	-

Table 5.1: Product Definition

5.3.1.2 Task Template / List / Sequence

The task template definition includes the task list and task sequence information. For this exercise, the template ID is "FND1". Table 5.2 shows the task definitions.

Template ID	Task	ResourceRequired	ResQty
FND1	Excavation	CrewExcavation	1
FND1	FormworkInstall	CrewFormwork	1
FND1	RebarInstall	CrewRebar Install	1
FND1	ConcretePouring	CrewConcrete	1

Table 5.2: Task Definition

The task sequence is quite straightforward for this example. Table 5.3 below describes it in the methodology's predecessor/successor format.

Template ID	Predecessor	Successor
FND1	Excavation	FormworkInstall
FND1	FormworkInstall	RebarInstall
FND1	RebarInstall	ConcretePouring

Table 5.3: Sequence Definitions

5.3.1.3 Resources

Resources have two definition items – resource and availability. Tables 5.4 and 5.5 show a set of resource definitions and resource availability as required for the tasks detailed above.

Resources	ProductionRate	Unit
CrewExcavation	1	MHr/m3
CrewFormwork	4	MHr/m2
CrewRebarInstall	50	MHr/ton
CrewConcrete	4	MHr/m3

Table 5.4: Resource Definitions

Resources	From	To	Qty
CrewExcavation	-	-	1
CrewFormwork	-	-	1
CrewRebarInstall	-	-	1
CrewConcrete	-	-	1

Table 5.5: Resource Availability Definitions

5.3.1.4 Product Task Mappings

The product task mappings define the levels at which tasks are applied on products and the quantities of work for each task on each product. Following are the two mappings for the example model.

5.3.1.4.1 Product-Task-Level Mapping

Product Name	Excavation	Formwork Install	Rebar Install	Concrete Pouring
F1	F1	F1	F1	F1
F2	F2	F2	F2	F2
F3	F3	F3	F3	F3
F4	F4	F4	F4	F4
F5	F5	F5	F5	F5

Table 5.6: Product-Task-Level Mapping

5.3.1.4.2 Product-Task-Qty Mapping

Product Name	Excavation (m3)	Formwork Install (m2)	Rebar Install (Ton)	Concrete Pouring (m3)
F1	4.80	8.00	0.40	4.00
F2	4.80	8.00	0.40	4.00
F3	7.20	12.00	0.60	6.00
F4	7.20	12.00	0.60	6.00
F5	4.80	8.00	0.40	4.00

Table 5.7: Product-Task-Qty Mapping

5.3.2 Prototype Development

The methodology was explained to an undergraduate computer science student before leaving the student to develop their own adaptation of the DES model compiler, including all the required code and data structures. Within a very short time, and with no further help after explaining the methodology, the computer science student was able to develop a simple implementation of the DES model compiler with the ability to read model descriptive data from user-fillable Excel sheets. The following sections explore the different parts of the system developed.

5.3.2.1 Foundations Model Descriptive Data Structures

5.3.2.2 Product List

The product list for this example is represented using a basic Excel spreadsheet showing product name and product nominal quantity as depicted in Table 5.8.

	A	B
1	F1	4.00
2	F2	4.00
3	F3	6.00
4	F4	6.00
5	F5	4.00

Table 5.8: Product List showing Product Name and Nominal Quantity

5.3.2.3 Tasks and Resources

The task list encompassed the task template, task properties and task sequence. As this is a prototyping exercise left to the student to develop and produce a result, the task list used for the implementation was a simple set of repeating tasks where the sequence was deduced from the natural order the tasks appeared in the task list. It also carried the production rate expected from each resource crew. The task list structure developed is

depicted in Table 5.9 showing Task name, expected crew production rate, resource crew required, and number of crews required.

	A	B	C	D
1	Excavation	1	Excavation	1
2	FormworkInstall	0.25	FormworkInstall	1
3	RebarInstall	0.02	RebarInstall	1
4	ConcretePouring	0.25	ConcretePouring	1

Table 5.9: Task List showing Task name, expected crew production rate, resource crew required, and number of crews

The Excel sheet carrying resource availability is shown in Table 5.10. It is composed of two columns: (1) Resource name; and (2) Resource Quantity Available.

	A	B
1	Excavation	1
2	FormworkInstall	1
3	RebarInstall	1
4	ConcretePouring	1

Table 5.10: Resource Availability List showing Resource Name and Resource Quantity Available

5.3.2.4 Product-Task Mappings

The two required product-task mappings are depicted in Tables 5.11 and 5.12. Product-Task-Level mapping reflects that the tasks are to be applied at level “1” of the product. Product-Task-Qty mapping depicts the quantity of work required for each task.

	A	B	C	D	E
1		Excavation	FormworkInstall	RebarInstall	ConcretePouring
2	F1	1	1	1	1
3	F2	1	1	1	1
4	F3	1	1	1	1
5	F4	1	1	1	1
6	F5	1	1	1	1

Table 5.11: Product-Task-Level Mapping

	A	B	C	D	E
1		Excavation	Formwork Install	Rebar Install	Concrete Pouring
2	F1	4.80	16.00	0.40	4.00
3	F2	4.80	16.00	0.40	4.00
4	F3	7.20	24.00	0.60	6.00
5	F4	7.20	24.00	0.60	6.00
6	F5	4.80	16.00	0.40	4.00

Table 5.12: Product-Task-Qty Mapping

5.3.2.5 DES Compiler Development

At the core of the development of the simulation system is the DES compiler code. The student used the Visual Studio software development environment and the Simphony API to create the DES model compiler. A basic interface for the system allowing the user to select the relevant Excel sheets required to describe and run the model was developed and is shown in Figure 5.22.

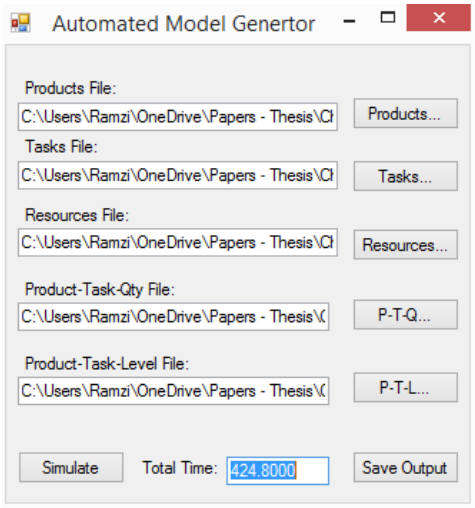

Figure 5.22: Prototype User Interface

The user would point the system to the corresponding Excel sheets describing the model and then press the “Simulate” button to run the system. Running the model results in the creation of a discrete event task for each of the tasks of each of the foundations. These are

then executed with execution passing from a predecessor task to a successor as each task terminates, taking into consideration DES model flows and resource availability and capture. The simulation run ends when the last task is executed and its result is written to the results set. The code developed by the student is a simple interpretation of the methodology and is shown in Figure 5.23.

```
private void ButtonSimulate_Click(object sender, EventArgs e)
{
            this.tasks = new List<SimTask>();
            this.entities = new List<SimEntity>();
            this.resourceQuantities = new Dictionary<string, int>();
            if (!FileReader.ReadTasksFileCSV(this.TaskFilePath, this.tasks) ||
            !FileReader.ReadResourcesFileCSV(this.ResourceFilePath, this.resourceQuantities) ||
            !FileReader.ReadEntitiesFileCSV(this.EntityFilePath, this.entities) || !this.ValidateInput())
            {
                        this.Simulated = false;
                        return;
            }
            using (new Hourglass())
            {
                        this.buttonSimulate.Enabled = false;
                        this.engine.InitializeEngine();
                        this.engine.TimeUnit = TimeUnit.Hour;
                        this.output = new List<string>();
                        this.output.Add("Product,Task,SimStartTime,SimEndTime");
                        foreach (var resourcePair in this.resourceQuantities)
                        {
                                    string resourceName = resourcePair.Key;
                                    int resourceQuantity = resourcePair.Value;
                                    Resource resource = new Resource(resourceName, resourceQuantity);
                                    WaitingFile waitingFile = new WaitingFile(resourceName + "Queue");
                                    resource.WaitingFiles.Add(waitingFile);
                                    this.engine.Resources.Add(resource);
                                    this.engine.WaitingFiles.Add(waitingFile);
                        }
                        this.engine.InitializeRun(0);
                        for (int i = 0; i < this.entities.Count; i++)
                        {
                                    this.engine.ScheduleEvent(this.entities[i], this.BeginTask, TimeSpan.Zero);
                        }
                        this.engine.Simulate();
                        this.engine.FinalizeRun(0);
                        this.Simulated = true;
                        foreach (var resourcePair in this.resourceQuantities)
                        {
                                    string resourceName = resourcePair.Key;
                                    Resource resource = this.engine.Resources.FirstOrDefault(x => x.Name == resourceName);
                                    var avgUtil = string.Format("{0:F4}", resource.Utilization.Mean);
                                    WaitingFile waitingFile = resource.WaitingFiles[0];
                                    var avgLength = string.Format("{0:F4}", waitingFile.FileLength.Mean);
                                    var avgWait = string.Format("{0:F4}", waitingFile.WaitingTime.Mean);
                                    this.output.Add("Resource: " + resourceName + "," + "Average Utilization: " + avgUtil +"," +
                                    "Average File Length: " + avgLength + "," + "Average Wait Time: " + avgWait);
                        }
                        this.textBoxTotalTime.Text = string.Format("{0:F4}", this.engine.TimeNow);
                        this.buttonSimulate.Enabled = true;
            }
}
```

Figure 5.23: Main Simulation Algorithm

The code in Figure 5.24 is executed in preparation for the start of each task. It requests a resource for the task and puts it into the resource queue.

```
private void BeginTask(SimEntity entity)
{
            int taskIndex = entity.TasksCompleted;
            SimTask task = this.tasks[taskIndex];
            if (task.ResourceName != string.Empty && task.ResourceAmount > 0)
            {
                        Resource resource = this.engine.Resources.FirstOrDefault(x => x.Name == task.ResourceName);
                        if (resource != null)
                        {
                                    this.engine.RequestResource(entity, resource, task.ResourceAmount, this.CapturedResource,
                                    resource.WaitingFiles[0]);
                        }
            }
            else
            {
                        double delay = 0D;
                        if (entity.Quantities[taskIndex] > 0)
                        {
                                    delay = entity.Quantities[taskIndex] / task.Productivity;
                        }
                        entity.StartTaskTime = this.engine.TimeNow;
                        this.engine.ScheduleEvent(entity, this.FinishTask, TimeSpan.FromHours(delay));
            }
}
```

Figure 5.24: Code Executed on BeginTask

The code in Figure 5.25 is executed each time a resource is captured. It calculates the total delay for the task, schedules its end, and stores the current time as the start time of this task.

```
private void CapturedResource(SimEntity entity)
{
            int taskIndex = entity.TasksCompleted;
            SimTask task = this.tasks[taskIndex];
            double delay = 0D;
            if (entity.Quantities[taskIndex] > 0)
            {
                        delay = entity.Quantities[taskIndex] / task.Productivity;
            }
            entity.StartTaskTime = this.engine.TimeNow;
            this.engine.ScheduleEvent(entity, this.FinishTask, TimeSpan.FromHours(delay));
}
```

Figure 5.25: Code Executed on CapturedResource

When the task completes for a product it outputs the required results [product name, task name, From, and To] into the “Results” as shown in the code snippet in Figure 5.26 below. After which, a check is made if there exists a successor to this task. If the check is positive, the successor is scheduled to start.

```
private void FinishTask(SimEntity entity)
{
            int taskIndex = entity.TasksCompleted;
            SimTask task = this.tasks[taskIndex];
            var resourcesUsed = 0;
            if (task.ResourceName != string.Empty && task.ResourceAmount > 0)
            {
                        resourcesUsed = task.ResourceAmount;
                        Resource resource = this.engine.Resources.FirstOrDefault(x => x.Name == task.ResourceName);
                        if (resource != null)
                        {
                                    this.engine.ReleaseResource(entity, resource, resourcesUsed);
                        }
                        }
            entity.TasksCompleted++;
            var startTime = entity.StartTaskTime;
            var finishTime = this.engine.TimeNow;
            var taskName = task.Name;
            this.output.Add(entity.Name + "," + task.Name + "," + startTime + "," + finishTime);
            if (entity.TasksCompleted < this.tasks.Count)
            {
                        this.engine.ScheduleEvent(entity, this.BeginTask, TimeSpan.Zero);
            }
}
```

Figure 5.26: Code Executed on FinishTask

5.3.2.6 Result Set

The simulation run outputs of the example model are shown in Table 5.13 below. For each foundation (product) and task a simstarttime and simendtime time are recorded.

	A	B	C	D
1	Product	Task	SimStartTime	SimEndTime
2	F1	Excavation	-	4.80
3	F2	Excavation	4.80	9.60
4	F3	Excavation	9.60	16.80
5	F4	Excavation	16.80	24.00
6	F5	Excavation	24.00	28.80
7	F1	FormworkInstall	4.80	68.80
8	F1	RebarInstall	68.80	88.80
9	F1	ConcretePouring	88.80	104.80
10	F2	FormworkInstall	68.80	132.80
11	F2	RebarInstall	132.80	152.80
12	F2	ConcretePouring	152.80	168.80
13	F3	FormworkInstall	132.80	228.80
14	F3	RebarInstall	228.80	258.80
15	F3	ConcretePouring	258.80	282.80
16	F4	FormworkInstall	228.80	324.80
17	F4	RebarInstall	324.80	354.80
18	F4	ConcretePouring	354.80	378.80
19	F5	FormworkInstall	324.80	388.80
20	F5	RebarInstall	388.80	408.80
21	F5	ConcretePouring	408.80	424.80

Table 5.13: Simulation Result Set

5.3.3 Redevelopment of Pipe Spool Fabrication Model

The value of the proposed methodology lies in the capability of end users to model a new construction situation without the need for resorting to simulation programming to construct a new model or amend an existing model to suit the new application. This exact capability is demonstrated in the sections below where the same Excel sheets describing the foundations example model are re-used to describe the pipe spool fabrication model which was explored as a case study with real project data.

5.3.3.1 Pipe Spool Fabrication Model Descriptive Data Structures

5.3.3.1.1 Product List

The product list for this model is represented using a basic Excel spreadsheet showing product name and product nominal quantity as depicted in Table 5.14 below.

	A	B
1	A112-B12GSO-1008/S03	46
2	A112-B12GSO-1008/S06	52
3	A112-B12GSO-1008/S09	52
4	A112-B12GSO-1008/S10	104
5	A112-B12GSO-1008/S11	52
6	A112-B12KA-1091/S01	106
7	A112-B12KA-1091/S02	20
8	A112-B12KA-1091/S03	20
9	A112-B12KA-1091/S04	80
10	A112-B12KA-1091/S05	20
11	A112-B12KA-1091/S06	20
12	A112-B12KA-1091/S07	80
13	A112-B12KA-1091/S08	20
14	A112-B12KA-1091/S09	20
15	A112-B12KA-1091/S10	80
16	A112-B12KA-1091/S11	20
17	A112-B12KA-1091/S12	20
18	A112-B12KA-1091/S13	52
19	A112-B12KA-1091/S18	26
20	A112-B12KA-1091/S19	26

Table 5.14: Product List showing Product Name and Nominal Quantity

5.3.3.1.2 Tasks and Resources

The task list structure is populated with data from the pipe spool fabrication model in Table 5.15 below showing Task name, expected crew production rate, resource crew required, and number of crews required.

	A	B	C	D
1	Cut	1	Cutters	1
2	Bevel	1	Bevelers	1
3	Fitup	1	Fitters	1
4	Weld	1	Welders	1
5	PWHT	1	PWHT	1
6	NDT	1	NDT	1
7	Blast	1	Blaster	1
8	Paint	1	Painters	1

Table 5.15: Task List showing Task name, expected crew production rate, resource crew required, and number of crews required

The Excel sheet carrying resource availability is shown in Table 5.16 below. It is composed of two columns: (1) Resource name; and (2) Resource Quantity Available.

	A	B
1	Cutters	1
2	Bevelers	1
3	Fitters	1
4	Welders	1
5	PWHT	1
6	NDT	1
7	Blaster	1
8	Painters	1

Table 5.16: Resource Availability List showing Resource Name and Resource Quantity Available

5.3.3.1.3 Product-task Mappings

The two required product-task mappings are depicted in Tables 5.17 and 5.18 below. Product-Task-Level mapping reflects that the tasks are to be applied at level “1” of the product. Product-Task-Qty mapping depicts the quantity of work required for each task. For this prototype model, all tasks are applied at the top product level as shown in the product task level mapping.

	A	B	C	D	E	F	G	H	I
1		Cut	Bevel	Fitup	Weld	PWHT	NDT	Blast	Paint
2	A112-B12GSO-1008/S03	1	1	1	1	1	1	1	1
3	A112-B12GSO-1008/S06	1	1	1	1	1	1	1	1
4	A112-B12GSO-1008/S09	1	1	1	1	1	1	1	1
5	A112-B12GSO-1008/S10	1	1	1	1	1	1	1	1
6	A112-B12GSO-1008/S11	1	1	1	1	1	1	1	1
7	A112-B12KA-1091/S01	1	1	1	1	1	1	1	1
8	A112-B12KA-1091/S02	1	1	1	1	1	1	1	1
9	A112-B12KA-1091/S03	1	1	1	1	1	1	1	1
10	A112-B12KA-1091/S04	1	1	1	1	1	1	1	1
11	A112-B12KA-1091/S05	1	1	1	1	1	1	1	1
12	A112-B12KA-1091/S06	1	1	1	1	1	1	1	1
13	A112-B12KA-1091/S07	1	1	1	1	1	1	1	1
14	A112-B12KA-1091/S08	1	1	1	1	1	1	1	1
15	A112-B12KA-1091/S09	1	1	1	1	1	1	1	1
16	A112-B12KA-1091/S10	1	1	1	1	1	1	1	1
17	A112-B12KA-1091/S11	1	1	1	1	1	1	1	1
18	A112-B12KA-1091/S12	1	1	1	1	1	1	1	1
19	A112-B12KA-1091/S13	1	1	1	1	1	1	1	1
20	A112-B12KA-1091/S18	1	1	1	1	1	1	1	1

Table 5.17: Product-Task-Level Mapping

	A	B	C	D	E	F	G	H	I
1		Cut	Bevel	Fitup	Weld	PWHT	NDT	Blast	Paint
2	A112-B12GSO-1008/S03	46	46	46	46	2	2	29.64	29.64
3	A112-B12GSO-1008/S06	52	52	52	52	2	2	46.84	46.84
4	A112-B12GSO-1008/S09	52	52	52	52	2	2	47.72	47.72
5	A112-B12GSO-1008/S10	104	104	104	104	3	3	71.21	71.21
6	A112-B12GSO-1008/S11	52	52	52	52	2	2	47.72	47.72
7	A112-B12KA-1091/S01	106	106	106	106	10	10	27.16	27.16
8	A112-B12KA-1091/S02	20	20	20	20	3	3	5.9	5.9
9	A112-B12KA-1091/S03	20	20	20	20	3	3	5.9	5.9
10	A112-B12KA-1091/S04	80	80	80	80	9	9	26.89	26.89
11	A112-B12KA-1091/S05	20	20	20	20	3	3	5.9	5.9
12	A112-B12KA-1091/S06	20	20	20	20	3	3	5.9	5.9
13	A112-B12KA-1091/S07	80	80	80	80	8	8	25.85	25.85
14	A112-B12KA-1091/S08	20	20	20	20	3	3	5.9	5.9
15	A112-B12KA-1091/S09	20	20	20	20	3	3	5.9	5.9
16	A112-B12KA-1091/S10	80	80	80	80	9	9	26.89	26.89
17	A112-B12KA-1091/S11	20	20	20	20	3	3	5.9	5.9
18	A112-B12KA-1091/S12	20	20	20	20	3	3	5.9	5.9
19	A112-B12KA-1091/S13	52	52	52	52	5	5	10.54	10.54
20	A112-B12KA-1091/S18	26	26	26	26	3	3	23.03	23.03

Table 5.18: Product-Task-Qty Mapping

5.3.3.1.4 Result Set

Using the same system and pointing to the above described spreadsheets as inputs, the compiler was run and the simulation model executed with the following result set.

	A	B	C	D
1	Product	Task	SimStartTim	SimEndTime
2	A112-B12GSO-1008/S03	Cut	0	46
3	A112-B12GSO-1008/S03	Bevel	46	92
4	A112-B12GSO-1008/S06	Cut	46	98
5	A112-B12GSO-1008/S03	Fitup	92	138
6	A112-B12GSO-1008/S09	Cut	98	150
7	A112-B12GSO-1008/S06	Bevel	98	150
8	A112-B12GSO-1008/S03	Weld	138	184
9	A112-B12GSO-1008/S03	PWHT	184	186
10	A112-B12GSO-1008/S03	NDT	186	188
11	A112-B12GSO-1008/S09	Bevel	150	202
12	A112-B12GSO-1008/S06	Fitup	150	202
13	A112-B12GSO-1008/S03	Blast	188	217.64
14	A112-B12GSO-1008/S03	Paint	217.64	247.28
15	A112-B12GSO-1008/S10	Cut	150	254
16	A112-B12GSO-1008/S09	Fitup	202	254
17	A112-B12GSO-1008/S06	Weld	202	254
18	A112-B12GSO-1008/S06	PWHT	254	256
19	A112-B12GSO-1008/S06	NDT	256	258
20	A112-B12GSO-1008/S06	Blast	258	304.84

Table 5.19: Simulation Result Set

5.4 Discussion

The prototyping exercise demonstrated the ease with which the methodology can be applied. A DES model compiler was developed by a computer science student given only an explanation of the requirements. The system developed, although simple in nature, was able to read model descriptive data from Excel spreadsheets that could effortlessly be filled by end users with construction data about the problem they want to solve. For both the simple foundations example and the redevelopment of the pipe spool fabrication example,

all the user needs to do is fill the required data describing the model. The data is pure construction related information about the products including their quantities, required tasks to build them and the relevant required resources. No software development or simulation expertise is required by the end user to utilize the developed prototype.

5.5 Conclusion

The proposed methodology presented in this chapter is an effort towards overcoming the threshold standing in the way of the use of simulation by end users on construction projects to quickly solve problems and answer. At the heart of the methodology is a change in the process of building simulation models for construction where end users will deal with data they know and understand well and use it to populate a pre-developed DES compiler which will in turn read that data, compile the required DES model, and run it producing the required results.

Observing three different simulation development efforts and recording the commonalities between them in terms of information and simulation modeling requirements helped form an understanding of the basic components required to create the new methodology. The new architecture requires a set of data structures to host the data describing the products and processes for modeling the construction situation, and the data set resulting from running the simulation model. It also requires a set of algorithms that would initially use the data describing the construction model to compile a DES model from it, and then run the virtual model and produce its associated artificial history.

Further, in this chapter, the proposed methodology was prototyped where a DES compiler was developed by an undergraduate computer science student using Visual Studio and the Simphony API. The prototype system was initially successfully tested by the student using a

simple example construction situation. Subsequently, the prototype was successfully tested by redeveloping the pipe spool fabrication model explored in the case studies chapter using real project data.

6 Implementing the Methodology

6.1 Introduction

The main objective behind the proposed methodology is aiding in the adoption of simulation as a go-to problem solving technique on construction projects. A prototype illustrating the feasibility of developing the DES model compiler system was explored in the previous chapter where an undergraduate computer science student was able to develop a DES model compiler system and to use it to model a simple construction situation. The same system was then use to quickly redevelop the pipe spool fabrication simulator explored in the case studies illustrating the ease and swiftness with which a construction situation can be modeled using the proposed methodology .

A further verification of the feasibility of using the methodology comes in the shape of applying it to develop a production version of the system proposed by the methodology in an enterprise environment using commercially available tools. The first step towards that goal is to develop the DES model compiler system and its supporting data structures. The second step is to use this system to model real construction models from actual construction projects: (1) the pipe spool fabrication model from the case studies will be redeveloped; and (2) a new model for simulating building finishes activities will be developed.

6.2 DES Model Compiler System Development

There are two constituents to the system proposed in the methodology, the DES model compiler and the data structures used to describe the model. The first step is the development of the DES model compiler component which will be responsible for compiling the required DES models by utilizing the model descriptive data. Anylogic, a commercially available simulation development environment, will be used to develop the DES compiler

and act as host for running the system. The second step is building the data structures to support the methodology. A combination of Microsoft SQL Server and Microsoft Excel will be used to hold the data.

6.2.1 DES Compiler

The DES compiler requires two basic constituents, the basic DES model class which will be instantiated for every product task intersection, and the compiler algorithms.

6.2.1.1 Basic DES Model Class

The development of the basic DES model in Anylogic is depicted in Figure 6.1 below. We can see a Source element, a NetworkEnter element [the implementation is making use of network modeling in Anylogic], a ResourceSeizeQueue element, a Task element, a ResourceRelease element, a NetworkExit element, a Sink element, and an overall Exit element. This DES model will form the basis used by the algorithm to replicate the source-task-sink basic model as needed. This DES model flow will be traversed by an entity representing the product at its appropriate level.

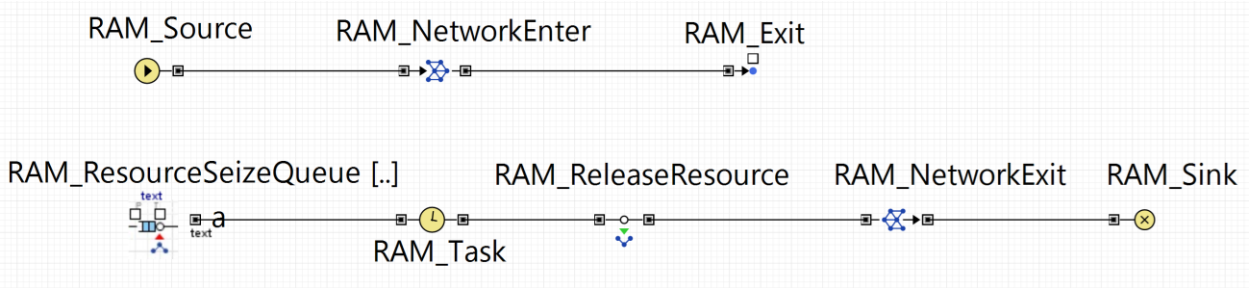

Figure 6.1: Anylogic DES Model

6.2.1.2 Algorithms

Using Anylogic network DES model components allowed the developers to replicate an instance for every product-task intersection. Following is the code required to populate the product and task related data in Anylogic and inject them through the network DES model.

6.2.1.2.1 Create Instance of Basic DES Model for every Product-Task Intersection

```
ResultSet result;
result = database.getResultSet("Select ProductName, Type,ID, MaxTasks
"+ordername+",Parent, Average_distance, QTY from Products where
ParentID = '"+Parentname+"' "+orderby+(orderby.compareTo("")==0 ? "" :
order));
int index=hallwayindex;
String lastordernum="";
double runningqty=0;
while(result.next())
{
      String lsProdName = result.getString("ProductName");
      String Type= result.getString("Type");
      String ID= result.getString("ID");
      int MaxTasks = result.getInt("MaxTasks");
      double Parent11=result.getDouble("Parent");
      double Average_distance11 = result.getDouble("Average_distance");
      double qty11=result.getDouble("QTY");
      add_Volume(MaxTasks, lsProdName, Type, Parent11, qty11);
}
```

Figure 6.2: Populate Product Data

```
ResultSet result= get_Main().database.getResultSet("Select * from
Activities_view where ParentID = '"+ParentName+"' AND ProductName =
'"+ProductID+"'");
ResultSet result2;
int index=0;
while(result.next()){
actname=result.getString("ActivityType").trim();
actqty=result.getDouble("Qty");
ID=result.getString("ID").trim();
Activity.add(new ActivityClass(actname,actqty,ID));
result2=get_Main().database1.getResultSet("select Predecessor, Successor
from [relations] where predecessor='"+actname+"' group by Predecessor,
Successor");
while(result2.next())
{
Activity.get(index).Successors.add(result2.getString("Successor"));
}
result2.close();
result2=get_Main().database1.getResultSet("select count(predecessor)as
numpredecessors from [relations]  where Successor = '"+actname+"' group
by Successor");

Predecessorcount.add(new Activitylist(actname,((result2.next()) ?
result2.getInt("numpredecessors")+1 : 1),0));
source.inject(1);
index++;
}
```

Figure 6.3: Populate Task and Sequence Data for Each Product

```
ResultSet result= database.getResultSet("SELECT *  FROM
Resources_quantity");
int index=0;
while(result.next())
{
add_Activity_resources();
resourcesnames.add(result.getString("Resource"));
Activity_resources.get(index).set_capacity(result.getInt("Quantity"));
index++;
}
ResultSet result1= database.getResultSet("SELECT *  FROM
Resources_quantity");
while(result1.next())
{
resourceQ.add(result1.getInt("Quantity"));
}
```

Figure 6.4: Populate Resource Data for Activities

6.2.1.2.2 Record Results and Call Successors

```
entity.effectivestart=date();
```

Figure 6.5: Store Product-Task Start Date in Entity Traversing the Product-Task DES Model

When the product entity exits the task element it outputs the required results [product name, task name, From, and To] into the "Results" as shown in the code snippet in Figure 6.6.

```
get_Main().database.modify("INSERT INTO results
Values('"+ProductName+"','"+entity.Name+"','"+java.text.DateFormat.getDateTimeInsta
nce().format(entity.effectivestart)+"','"+java.text.DateFormat.getDateTimeInstance(
).format(date())+"','"+get_Main().sessionNumber+"')");
```

Figure 6.6: Output Result of Product-Task DES Model Run and Call Successor

When the entity enters the sink element, the code in Figure 6.7 is executed to find the successor task (if it exists) and inject a product entity into its source element.

```
int size=Activity.get(actindex).Successors.size();
for(int i=0;i<size;i++){
String sucessor=Activity.get(actindex).Successors.get(i);
int index2=getactivityindex(sucessor);
Predecessorcount.get(index2).count++;
if(Predecessorcount.get(index2).count==Predecessorcount.get(index2).Target)
	{
		actname=Activity.get(index2).Name;
		actqty=Activity.get(index2).Qty;
		ID=Activity.get(index2).ID;
		indexinarraylist=index2;
		RAM_Source.inject(1);
	}
}
```

Figure 6.7: Inject Entity into Successor Product-Task DES Model

6.2.2 Data Structures

The data structures to support the methodology are developed as a relational database in Microsoft SQL Server. Each required data structure maps to a dedicated table in SQL Server. Users are presented with Excel spreadsheets to enter the relevant data in a predetermined format, with a specific spreadsheet mapping to each of the data structure components required for the methodology. These spreadsheets are then imported into SQL Server - with a single press of the button - to make them available for the simulation engine. Following are the Excel spreadsheets and their corresponding SQL tables for each of the data sets (Figures 6.8 through 6.15).

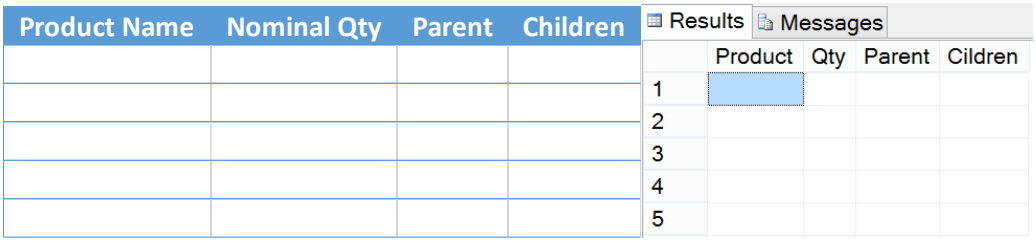

Figure 6.8: Spreadsheet and Corresponding SQL Server Data Structure for Product Definitions

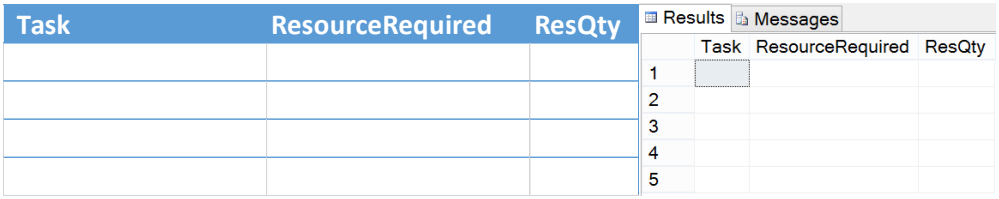

Figure 6.9: Spreadsheet and Corresponding SQL Server Data Structure for Task Definitions

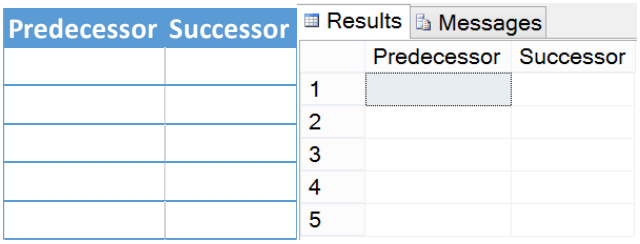

Figure 6.10: Spreadsheet and Corresponding SQL Server Data Structure for Predecessor-Successor Relationships

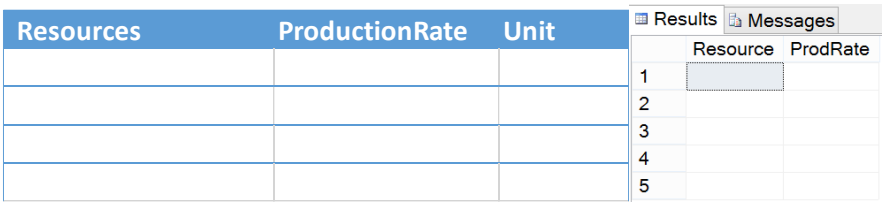

Figure 6.11: Spreadsheet and Corresponding SQL Server Data Structure for Resource Definition

Resources	From	To	Qty

Results | Messages

	Resource	From	To	Qty
1				
2				
3				
4				
5				

Figure 6.12: Spreadsheet and Corresponding SQL Server Data Structure for Resource Availability Definitions

Product Name	Task 1	Task 2	- - -	Task n
(Product Level)				

Results | Messages

	Product	Task	Level
1			
2			
3			
4			
5			

Figure 6.13: Spreadsheet and Corresponding SQL Server Data Structure for Product-Task-Level Mapping

Product Name	Task 1	Task 2	- - -	Task n
(Product Qty)				

Results | Messages

	Product	Task	Qty
1			
2			
3			
4			
5			

Figure 6.14: Spreadsheet and Corresponding SQL Server Data Structure for Product-Task-Qty Mapping

Results | Messages

	ProductID	TaskID	SimStart	SimEnd
1				
2				
3				
4				
5				

Product ID	Task ID	Sim Start	Sim End

Figure 6.15: SQL Server Data and Corresponding Spreadsheet Structure for Simulation Result Set

6.3 Redevelopment of the Pipe Spool Fabrication Model

The pipe spool fabrication simulator from Chapter 3 models the main pipe spool fabrication tasks and crews required to build pipe spools. Redeveloping the model using the new methodology requires describing the model using the newly developed structures.

6.3.1 Product Definitions

Product definition for the spools is a straightforward process for our purposes. Spools have a basic two level hierarchy as shown in Figure 6.16 and a depiction of the product list is shown in Table 6.1.

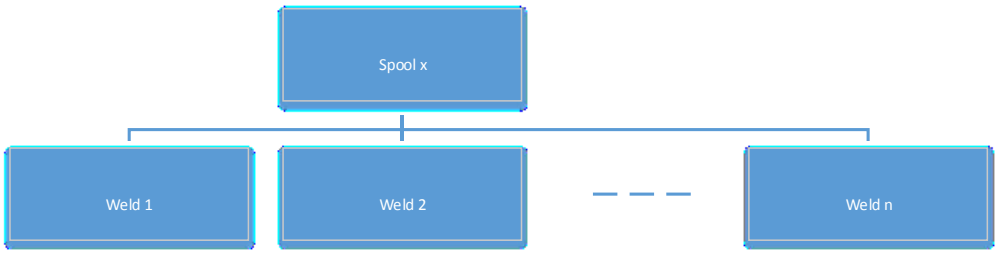

Figure 6.16: Pipe Spool Fabrication Spool-Weld Basic Two-Level Hierarchy

	A	B	C
1	**Product**	**Nominal Qty**	**Parent**
2	A111-B16GN-1087/S01	63	
3	A111-B16GN-1087/S01/1	24	A111-B16GN-1087/S01
4	A111-B16GN-1087/S01/2	1	A111-B16GN-1087/S01
5	A111-B16GN-1087/S01/4	24	A111-B16GN-1087/S01
6	A111-B16GN-1087/S01/701	14	A111-B16GN-1087/S01
7	A111-B16GN-1087/S04	24	
8	A111-B16GN-1087/S04/11	24	A111-B16GN-1087/S04
9	A111-B16GN-1087/S14	24	
10	A111-B16GN-1087/S14/22	24	A111-B16GN-1087/S14
11	A111-B16GN-1087/S18	24.75	
12	A111-B16GN-1087/S18/35	24	A111-B16GN-1087/S18
13	A111-B16GN-1087/S18/36	0.75	A111-B16GN-1087/S18

Table 6.1: Pipe Spool Fabrication Simulator – Product List

6.3.2 Task Definitions

A set of repeating steps was applied to spools to build them. These steps include Cut, Bevel, Fitup, Weld, PWHT, NDT, Blasting, and Painting. Figure 6.17 shows those steps. Table 6.17 below shows the Task definition table used to feed the Anylogic model with the data.

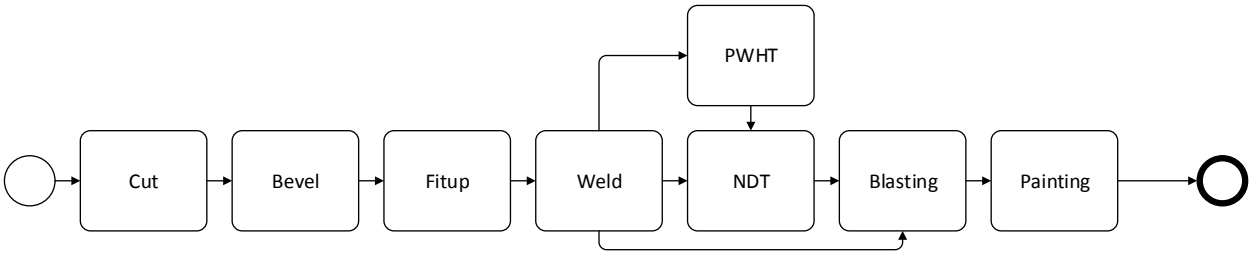

Figure 6.17: Pipe Spool Fabrication Simulator – Tasks Sequence Showing Welded Spool Proceeding on Three Different Paths to Completion

	A	B	C
1	**Activity**	**Resource**	**Res_Qty**
2	cut	cut	1
3	bevel	bevel	1
4	fitup	fitup	1
5	welding	welding	1
6	NDT	NDT	1
7	PWHT	PWHT	1
8	blasting	blasting	1
9	painting	painting	1

Table 6.2: Pipe Spool Fabrication Simulator – Task List

6.3.3 Sequence

Predecessor-Successor relationships for the tasks are depicted in Table 6.3 below. As this implementation utilizes a single template, no template name is required in the data.

	A	B
1	**Predecessor**	**Successor**
2	cut	bevel
3	bevel	fitup
4	fitup	welding
5	welding	NDT
6	NDT	PWHT
7	PWHT	blasting
8	blasting	painting

Table 6.3: Pipe Spool Fabrication Simulator – Task Sequence

6.3.4 Resources

Resource requirements and availability are represented in a single table (Table 6.4) below. This is due to the fact that for this implementation the resources are fixed flat for the duration of the simulation and the developers decided to combine the two tables into one.

	A	B	C
1	**Resource**	**Quantity**	**ProdRate**
2	cut	2	
3	bevel	2	
4	fitup	2	
5	welding	5	
6	NDT	10	
7	PWHT	5	
8	blasting	4	
9	painting	10	

Table 6.4: Pipe Spool Fabrication Simulator – Resource List

6.3.5 Product-Task-Level Mapping

Table 6.5 depicts the mapping of the level at which a task is applied to a product. Each task is applied to the spool itself except the welding task which is applied to each weld in the spool.

	A	B	C	D	E	F	G	H	I
1	**Product**	**cut**	**bevel**	**fitup**	**welding**	**NDT**	**PWHT**	**blasting**	**painting**
2	A111-B16GN-1087/S01	Spool	Spool	Spool	Weld	Spool	Spool	Spool	Spool
3	A111-B16GN-1087/S04	Spool	Spool	Spool	Weld	Spool	Spool	Spool	Spool
4	A111-B16GN-1087/S14	Spool	Spool	Spool	Weld	Spool	Spool	Spool	Spool
5	A111-B16GN-1087/S18	Spool	Spool	Spool	Weld	Spool	Spool	Spool	Spool

Table 6.5: Pipe Spool Fabrication Simulator – Product-Task-Level Mapping

6.3.6 Product-Task-Quantity Mapping

Table 6.6 depicts the mapping of the quantity for each task for each product. Cut, bevel, fitup and weld all have inch-dia values; NDT and PWHT have number of joints, and blasting and painting deal with surface area.

	A	B	C	D	E	F	G	H	I
1	**Product**	**cut**	**bevel**	**fitup**	**welding**	**NDT**	**PWHT**	**blasting**	**painting**
2	A111-B16GN-1087/S01	63.00	63.00	63.00	63.00	4	4	8.37	8.37
3	A111-B16GN-1087/S04	24.00	24.00	24.00	24.00	1	1	23.48	23.48
4	A111-B16GN-1087/S14	24.00	24.00	24.00	24.00	1	1	10.12	10.12
5	A111-B16GN-1087/S18	24.75	24.75	24.75	24.75	2	2	26.16	26.16

Table 6.6: Pipe Spool Fabrication Simulator – Product-Task-Qty Mapping

6.3.7 Result Set

Running the redeveloped pipe spool fabrication model using the new methodology and the above data structures defining the model we get the following result set (Table 6.7).

	A	B	C	D
1	**Product**	**Activity**	**Simulationstartdate**	**Simulationenddate**
2	A111-B16GN-1087/S01	cut	10/16/13 8:10	10/16/13 9:59
3	A111-B16GN-1087/S01	bevel	10/16/13 8:47	10/16/13 10:58
4	A111-B16GN-1087/S01	fitup	10/16/13 9:30	10/16/13 12:09
5	A111-B16GN-1087/S01/1	welding	10/16/13 10:20	10/16/13 12:54
6	A111-B16GN-1087/S01/2	welding	10/16/13 10:20	10/16/13 12:54
7	A111-B16GN-1087/S01/4	welding	10/16/13 11:10	10/16/13 13:44
8	A111-B16GN-1087/S01/701	welding	10/16/13 11:10	10/16/13 13:44
9	A111-B16GN-1087/S01	NDT	10/16/13 13:05	10/18/13 5:05
10	A111-B16GN-1087/S01	PWHT	10/17/13 8:10	10/18/13 16:10
11	A111-B16GN-1087/S01	blasting	10/17/13 16:20	10/18/13 0:20
12	A111-B16GN-1087/S01	painting	10/18/13 8:10	10/22/13 8:10
13	A111-B16GN-1087/S04	cut	10/16/13 9:24	10/16/13 9:58
14	A111-B16GN-1087/S04	bevel	10/16/13 10:13	10/16/13 10:53
15	A111-B16GN-1087/S04	fitup	10/16/13 11:10	10/16/13 11:58
16	A111-B16GN-1087/S04/11	welding	10/16/13 12:09	10/16/13 15:18
17	A111-B16GN-1087/S04	NDT	10/16/13 15:28	10/17/13 1:28
18	A111-B16GN-1087/S04	PWHT	10/17/13 8:10	10/17/13 16:10
19	A111-B16GN-1087/S04	blasting	10/17/13 16:20	10/17/13 18:20
20	A111-B16GN-1087/S04	painting	10/18/13 8:10	10/19/13 8:10
21	A111-B16GN-1087/S14	cut	10/16/13 9:24	10/16/13 9:58
22	A111-B16GN-1087/S14	bevel	10/16/13 10:13	10/16/13 10:53
23	A111-B16GN-1087/S14	fitup	10/16/13 11:10	10/16/13 11:58
24	A111-B16GN-1087/S14/22	welding	10/16/13 13:05	10/16/13 16:14
25	A111-B16GN-1087/S14	NDT	10/16/13 16:24	10/17/13 2:24
26	A111-B16GN-1087/S14	PWHT	10/17/13 16:20	10/18/13 0:20
27	A111-B16GN-1087/S14	blasting	10/18/13 8:10	10/18/13 10:10
28	A111-B16GN-1087/S14	painting	10/18/13 10:20	10/19/13 10:20
29	A111-B16GN-1087/S18	cut	10/16/13 10:08	10/16/13 10:43
30	A111-B16GN-1087/S18	bevel	10/16/13 11:03	10/16/13 11:44
31	A111-B16GN-1087/S18	fitup	10/16/13 12:09	10/16/13 12:59
32	A111-B16GN-1087/S18/35	welding	10/16/13 13:05	10/16/13 14:42
33	A111-B16GN-1087/S18/36	welding	10/16/13 13:54	10/16/13 15:32
34	A111-B16GN-1087/S18	NDT	10/16/13 14:52	10/17/13 10:52
35	A111-B16GN-1087/S18	PWHT	10/17/13 16:20	10/18/13 8:20
36	A111-B16GN-1087/S18	blasting	10/18/13 8:10	10/18/13 12:10
37	A111-B16GN-1087/S18	painting	10/18/13 10:20	10/20/13 10:20

Table 6.7: Pipe Spool Fabrication Simulator – Simulation Result Set

6.4 Development of the Building Finishes Model

The building finishes simulator serves to aid in validating the schedule and in low level resource planning for finishing activities applied to rooms in buildings. The basic product is a room and the basic tasks to be applied are finishing activities as required for each room. This model presents itself as a typical application of the new methodology as it can be described as a set of products, repeating tasks and resources as described in the sections below.

6.4.1 Product Definitions

Product definition for the rooms is a straightforward process for our purposes. Rooms have only one level and for our purposes do not have parents or children. Table 6.8 depicts the product list.

	A	B	C
1	**Product**	**Nominal Qty**	**Parent**
2	P1-L1-S150/2200	12.81	
3	P1-L1-S150/613	3.31	
4	P1-L1-S151/2201	12.81	
5	P1-L1-S151/2202	21.28	
6	P1-L1-S151/319	3.94	
7	P1-L1-S151/606	19.04	
8	P1-L1-S151/609	3.31	
9	P1-L1-S151/611	4.41	
10	P1-L1-S151/614	1224.22	
11	P1-L1-S151/615	14.24	
12	P1-L1-S151/637	12.74	

Table 6.8: Building Finishes Simulator – Product List

6.4.2 Task Definitions

A set of repeating steps is applied to each room to finish it. These tasks are depicted in Table 6.9.

	A	B	C
1	**Activity**	**Resource**	**Res_Qty**
2	Anti Static Floor Tiling	Anti Static Floor Tiling	1
3	CABLE CONTAINMENT	CABLE CONTAINMENT	1
4	CABLE TRAYS & TRUNKING	CABLE TRAYS & TRUNKING	1
5	Ceiling Work	Ceiling Work	1
6	Ceramic Floor Tiles	Ceramic Floor Tiles	1
7	Concrete Screed	Concrete Screed	1
8	Doors	Doors	1
9	Epoxy Resin Floor Coating	Epoxy Resin Floor Coating	1
10	Final Painting	Final Painting	1
11	Initial Painting	Initial Painting	1
12	LIGHTING FIXTURES INSTALLATION	LIGHTING FIXTURES INSTALLATION	1
13	Stone Flooring	Stone Flooring	1
14	Wall Plastering	Wall Plastering	1
15	WIRING	WIRING	1

Table 6.9: Building Finishes Simulator – Task List

6.4.3 Sequence

Predecessor-Successor relationships for the tasks are depicted in Table 6.10 below. As this implementation utilizes a single template no template name is required in the data.

	A	B
1	**Predecessor**	**Successor**
2	CABLE TRAYS & TRUNKING	Initial Painting
3	Initial Painting	Stone Flooring
4	Stone Flooring	Final Painting
5	Final Painting	Anti Static Floor Tiling
6	Anti Static Floor Tiling	CABLE CONTAINMENT
7	CABLE CONTAINMENT	Ceramic Floor Tiles
8	Ceramic Floor Tiles	Concrete Screed
9	Concrete Screed	WIRING
10	WIRING	Wall Plastering
11	Wall Plastering	Ceiling Work
12	Ceiling Work	Doors
13	Doors	Epoxy Resin Floor Coating
14	Epoxy Resin Floor Coating	LIGHTING FIXTURES INSTALLATION

Table 6.10: Building Finishes Simulator – Task Sequence

6.4.4 Resources

Resource requirements and availability are represented in a single table (Table 6.11). This is due to the fact that for this implementation the resources are fixed flat for the duration of the simulation and the developers decided to combine the two tables into one.

	A	B	C
1	**Resource**	**Quantity**	**ProdRate**
2	Anti Static Floor Tiling	10	
3	CABLE CONTAINMENT	10	
4	CABLE TRAYS & TRUNKING	10	
5	Ceiling Work	10	
6	Ceramic Floor Tiles	10	
7	Concrete Screed	10	
8	Doors	10	
9	Epoxy Resin Floor Coating	10	
10	Final Painting	10	
11	Initial Painting	10	
12	LIGHTING FIXTURES INSTALLATION	10	
13	Stone Flooring	10	
14	Wall Plastering	10	
15	WIRING	10	

Table 6.11: Building Finishes Simulator – Resource List

6.4.5 Product-Task-Level Mapping

Table 6.12 depicts the mapping of the level at which a task is applied to a product. All the tasks for this model are applied at the room level.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Product	Anti Static Floor Tiling	CABLE CONTAINMENT	CABLE TRAYS & TRUNKING	Ceiling Work	Ceramic Floor Tiles	Concrete Screed	Doors	Epoxy Resin Floor Coating	Final Painting	Initial Painting	Lighting Fixtures Installation	Stone Flooring	Wall Plastering	WIRING
2	P1-L1-S150/2200	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room
3	P1-L1-S150/613	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room
4	P1-L1-S151/2201	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room
5	P1-L1-S151/2202	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room
6	P1-L1-S151/319	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room
7	P1-L1-S151/606	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room
8	P1-L1-S151/609	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room
9	P1-L1-S151/611	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room
10	P1-L1-S151/614	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room
11	P1-L1-S151/615	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room
12	P1-L1-S151/637	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room	Room

Table 6.12: Building Finishes Simulator – Product-Task-Level Mapping

6.4.6 Product-Task-Qty Mapping

Table 6.13 depicts mapping of the quantity for each task for each product. For this model a uniform quantity equivalent to the duration required for each crew to perform the required task on a room was directly entered into the product-task-qty mapping.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Product	Anti Static Floor Tiling	CABLE CONTAINMENT	CABLE TRAYS & TRUNKING	Ceiling Work	Ceramic Floor Tiles	Concrete Screed	Doors	Epoxy Resin Floor Coating	Final Painting	Initial Painting	Lighting Fixtures Installation	Stone Flooring	Wall Plastering	WIRING
2	P1-L1-S150/2200	-	99.07	137.21	21.72	-	-	21.72	21.72	354.95	211.91	30.46	-	27.02	192.84
3	P1-L1-S150/613	-	25.60	35.45	5.61	-	-	5.61	5.61	91.72	54.76	7.87	-	6.98	49.83
4	P1-L1-S151/2201	-	1.41	2.47	0.27	-	-	0.27	0.27	1.13	2.28	0.28	-	0.31	2.58
5	P1-L1-S151/2202	-	2.34	4.10	0.44	-	-	0.44	0.44	1.88	3.79	0.46	-	0.52	4.28
6	P1-L1-S151/319	-	0.43	0.76	0.08	-	-	0.08	0.08	0.35	0.70	0.08	-	0.10	0.79
7	P1-L1-S151/606	-	2.09	3.67	0.40	-	-	0.40	0.40	1.68	3.40	0.41	-	0.46	3.83
8	P1-L1-S151/609	-	0.36	0.64	0.07	-	-	0.07	0.07	0.29	0.59	0.07	-	0.08	0.67
9	P1-L1-S151/611	-	0.48	0.85	0.09	-	-	0.09	0.09	0.39	0.79	0.09	-	0.11	0.89
10	P1-L1-S151/614	-	134.58	235.98	25.43	-	-	25.43	25.43	107.91	218.30	26.36	-	29.77	246.21
11	P1-L1-S151/615	-	1.57	2.74	0.30	-	-	0.30	0.30	1.26	2.54	0.31	-	0.35	2.86
12	P1-L1-S151/637	-	1.40	2.46	0.26	-	-	0.26	0.26	1.12	2.27	0.27	-	0.31	2.56

Table 6.13: Building Finishes Simulator – Product-Task-Qty Mapping

6.4.7 Result Set

Running the building finishes model using the new methodology and the above data structures defining the model we get the following result set.

	A	B	C	D
1	Product	Activity	Simulationstartdate	Simulationenddate
11	P1-L1-S150/613	CABLE TRAYS & TRUNKING	1/18/14 8:10 AM	2/15/14 12:36 PM
12	P1-L1-S150/2200	CABLE TRAYS & TRUNKING	1/18/14 8:10 AM	2/15/14 12:36 PM
85	P1-L1-S150/613	Initial Painting	4/14/14 4:20 PM	8/25/14 11:20 PM
86	P1-L1-S150/2200	Initial Painting	4/14/14 4:20 PM	8/25/14 11:20 PM
132	P1-L1-S150/613	Stone Flooring	8/26/14 8:10 AM	8/26/14 8:10 AM
133	P1-L1-S150/2200	Stone Flooring	8/26/14 8:10 AM	8/26/14 8:10 AM
134	P1-L1-S150/613	Final Painting	8/26/14 8:20 AM	4/6/15 3:20 PM
135	P1-L1-S150/2200	Final Painting	8/26/14 8:20 AM	4/6/15 3:20 PM
136	P1-L1-S150/2200	Anti Static Floor Tiling	4/6/15 3:30 PM	4/6/15 3:30 PM
137	P1-L1-S150/613	Anti Static Floor Tiling	4/6/15 3:30 PM	4/6/15 3:30 PM
138	P1-L1-S150/2200	CABLE CONTAINMENT	4/6/15 3:40 PM	6/7/15 10:40 PM
139	P1-L1-S150/613	CABLE CONTAINMENT	4/6/15 3:40 PM	6/7/15 10:40 PM
140	P1-L1-S150/613	Ceramic Floor Tiles	6/8/15 8:10 AM	6/8/15 8:10 AM
141	P1-L1-S150/2200	Ceramic Floor Tiles	6/8/15 8:10 AM	6/8/15 8:10 AM
142	P1-L1-S150/2200	Concrete Screed	6/8/15 8:20 AM	6/8/15 8:20 AM
143	P1-L1-S150/613	Concrete Screed	6/8/15 8:20 AM	6/8/15 8:20 AM
144	P1-L1-S150/2200	WIRING	6/8/15 8:30 AM	10/7/15 3:30 PM
145	P1-L1-S150/613	WIRING	6/8/15 8:30 AM	10/7/15 3:30 PM
146	P1-L1-S150/2200	Wall Plastering	10/7/15 3:40 PM	10/24/15 3:40 PM
147	P1-L1-S150/613	Wall Plastering	10/7/15 3:40 PM	10/24/15 3:40 PM
148	P1-L1-S150/2200	Ceiling Work	10/24/15 3:50 PM	11/7/15 5:50 AM
149	P1-L1-S150/613	Ceiling Work	10/24/15 3:50 PM	11/7/15 5:50 AM
150	P1-L1-S150/2200	Doors	11/7/15 8:10 AM	11/21/15 12:10 AM
151	P1-L1-S150/613	Doors	11/7/15 8:10 AM	11/21/15 12:10 AM
152	P1-L1-S150/2200	Epoxy Resin Floor Coating	11/21/15 8:10 AM	12/5/15 12:10 AM
153	P1-L1-S150/613	Epoxy Resin Floor Coating	11/21/15 8:10 AM	12/5/15 12:10 AM
154	P1-L1-S150/2200	LIGHTING FIXTURES INSTALLATION	12/5/15 8:10 AM	12/24/15 12:10 PM
155	P1-L1-S150/613	LIGHTING FIXTURES INSTALLATION	12/5/15 8:10 AM	12/24/15 12:10 PM

Table 6.14: Sample Result Set from Building Finishes Implementation using the Proposed Methodology

6.5 Verification and Validation of the Models

As stated previously, credibility of a model which is expected to help manage construction projects is of utmost importance in order for stakeholders to accept and adopt the model. Accordingly, both verification and validation of the production version of the system were performed. Verification, in simulation, allows the determination of whether a model is performing as intended or not (Law and Kelton, 1991) and its technical correctness (Sargent 1991). Model verification was performed throughout the development process of the production version of the system based on the proposed modeling methodology. Verification covered the coding of the discrete event simulation (DES) compiler to ensure it contained no errors. The compiled models produced by the DES compiler were also checked for simulation logical flow and timing, entity flow within each product-task intersection and

in-between the various product-task intersections. Unit tests on each of the tasks within each model, and an overall system test were run at the time of development of the model. Outputs after the tests were compared with expected results based on predetermined inputs and ensured the models and their components were correctly implemented. Moreover, a simulation trace file was created (Kleijnen 1995) to examine intermediate simulation outputs comparing simulation results with user expected values. Validation of a simulation model ensures that "the model is sufficiently accurate for the purpose at hand" (Carson 1986). The flow and logic of the production version of the system were compared and confirmed against conceptual model design. The models developed using the production version of the system were each run with historical data from real projects and their outputs compared to historical results to ensure the models were behaving as per their design purposes. Furthermore, face validation (Lucko and Rojas 2010) was performed where a real project scenario is simulated using the newly developed system and the expectation is that the simulation outputs "be sufficiently similar to the real project outcomes" (Robinson 1997). The scenarios previously applied through the case study models were re-applied using the newly developed models in the production version of the system. Scenarios and their results were then examined by domain experts who agreed that they are representative of what happens in reality.

6.6 Conclusion

This chapter explored the application of the proposed methodology to two enterprise construction simulation models. The first step was the development of the DES model compiler system. This was implemented in a combination of Microsoft SQL Server for data storage and manipulation, Microsoft Excel spreadsheets for user input data and output analysis, and Anylogic as the discrete event simulation environment for developing the

required system. A set of SQL Server data structures was created to host the model descriptive data to be provided by end users along with a set of matching Excel spreadsheets for an easy user experience using tools end users are already familiar and comfortable with. This set of data structures followed the findings and proposed methodology of the two preceding chapters, categorizing the required data into product definition, process definition, product-task mappings, and simulator outputs.

Once the DES model compiler was developed in Anylogic and the data structures were ready in SQL Server with their matching preformatted spreadsheets, the system was used to implement two different construction simulation models in two diverse construction disciplines.

The first implementation was one in the industrial sector where the same pipe spool fabrication simulator used as part of the commonalities analysis was redeveloped using the new methodology. Data describing the pipe spool fabrication process was entered into the preformatted Excel spreadsheets by the end users and the system was run to produce the required artificial history of the simulation model.

The second implementation was one in the commercial buildings sector where a building finishes simulator was developed using the new methodology. Data describing the building finishes process was entered into the preformatted Excel spreadsheets by the end users and the system was run to produce the required artificial history of the simulation model.

The two models constructed using the DES model compiler developed in Anylogic, and the two prototype models constructed using the DES model compiler developed using Simphony were all performed following the basic principles set out in the proposed methodology. Table 6.15 compares all four models.

	Prototype				Production Version			
	Cast-in-place Foundations		Pipe Spool Fabrication		Pipe Spool Fabrication		Building Finishes	
Methodology Component	Functional Description	Implementation Technique	Functional Description	Implementation Technique	Functional Description	Implementation Technique	Functional Description	Implementation Technique
Product								
Product Definition	Foundations	Preformatted Excel Sheet	Spools	Preformatted Excel Sheet	Spool - Weld	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation	Rooms	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation
Hierarchy Definition	1 level	Preformatted Excel Sheet	1 level	Preformatted Excel Sheet	2 levels	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation	1 level	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation
Process								
Tasks and Sequences	Foundation Building Tasks	Preformatted Excel Sheet	Spool Fabrication Tasks	Preformatted Excel Sheet	Pipe Spool Fabrication Tasks	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation	Room Finishing Activities	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation
Resources	Each task mapped to a crew	Preformatted Excel Sheet	Each task mapped to a crew	Preformatted Excel Sheet	Each task mapped to a crew	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation	Each activity mapped to a crew	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation
Resource Availability	Crew availability fixed over time	Preformatted Excel Sheet	Crew availability fixed over time	Preformatted Excel Sheet	Crew availability fixed over time	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation	Crew availability fixed over time	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation
Product-Task Mappings								
Task Level	Foundation	Preformatted Excel Sheet	Spool	Preformatted Excel Sheet	Spool / Weld	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation	Room	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation
Task Quantities	Qty matrix per each task for each foundation	Preformatted Excel Sheet	Qty matrix per each task for each spool	Preformatted Excel Sheet	Qty matrix per each task for each spool	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation	Qty matrix per each task for each room	Preformatted Excel spreadsheet for user input + SQL Server Database for data storage and manipulation
DES Model Generator	Simphony DES API + Visual Studio				Anylogic Network DES Model + Java Code			
Simulator Outputs	Product-Task-SimStart-SimEnd	Preformatted Excel Sheet	Product-Task-SimStart-SimEnd	Preformatted Excel Sheet	Product-Task-SimStart-SimEnd	SQL Server Database for data storage and Excel spreadsheet for user viewing and analysis	Product-Task-SimStart-SimEnd	SQL Server Database for data storage and Excel spreadsheet for user viewing and analysis

Table 6.15: Prototype and Production Version Construction Simulation Models

7 Conclusion

7.1 Research Contributions

The main contribution of this research is the development of a modeling methodology to make simulation more accessible to end users in the construction industry. The proposed methodology delivers on this by offering end users a means by which they can model a construction problem by simply entering model descriptive data about the construction situation instead of resorting to the traditional simulation modeling techniques. With this methodology, end users enter the model descriptive data into a set of preformatted data structures and run a DES model compiler. Running the DES compiler will produce a DES model using the entered data, run the model, and yield a simulation result set. The DES compiler and the supporting data structures are developed once only by a simulation programmer and re-used as needed for modeling different construction situations. The proposed methodology describes the required components including (1) model descriptive data structures; (2) simulator output structures; and (3) DES model compiler algorithms. The model descriptive data structures comprise (1) product definitions, which is a list of the products and their nominal quantities; (2) process definitions, which include tasks to be performed to build the products, task sequences defining the logical flow of the tasks to complete the products, resource requirements for each task, and resource availability; and (3) product-task mappings, which include the product-task-quantity matrix identifying the quantity of work for each task applied to each product, and the product-task-level matrix identifying the level at which a task is applied to a product. The simulator output structures is a result data set where each data record maps to an intersection of a product and a task to be performed on that product, with relevant data fields to identify that intersection along with a simulation start date and time and a simulation end date and time. The DES model

compiler algorithms describe the programming logic required to compile a DES model from the model descriptive data structures. It portrays the process of creating a dedicated task flow for each product task intersection taking into consideration the quantity of work required for that product-task intersection, the level at which the task applies to that product, and the resource required to perform the task. The algorithms also describe the start of product-task processing, the flow control between product task intersections, and recording of the simulation result set.

In order to arrive at the proposed methodology, a study of the commonalities in developing traditional construction simulation models for three different construction situations was performed. Simulation models were developed for each of the three construction situations and applied at multiple construction projects to aid in low level resource planning and schedule validation. The study performed acted as a learning experience where the common product definitions, process definitions, simulation outputs and modeling requirements found while developing the three models were identified and catalogued.

In addition to identifying and cataloguing the requirements for building the three models, the proposed methodology describes a technique for building a system making the use of simulation more accessible to end users in construction. It does so by eliminating the traditional modeling requirement where in addition to a construction domain expert, a simulation practitioner was needed for developing or amending a construction model. A system developed following the proposed methodology will enable the end user to model the required construction situation by providing construction domain descriptive data about the situation and enter them into a set of preformatted data structures. This methodology means the construction end user will no longer require simulation expertise for construction

simulation modeling and will only have to deal with domain familiar information and data for that purpose without the need to organize the necessary data in structures required by simulation systems which are sometimes too different from the way they are found in the daily work process. To test the methodology, a proof-of-concept prototype was developed using the proposed methodology by a computer science student not familiar with the methodology. The prototype was developed using Simphony Simulation Services and Microsoft Visual Studio. The student was able to develop a basic system representing the methodology and use it to quickly model a simple construction problem. The prototype was further tested by using it to redevelop the pipe spool fabrication model - one of the case study simulation models - again very quickly. A further test of the methodology was then performed by implementing the methodology to develop an enterprise production version of the system using the commercially available simulation modeling system Anylogic and Microsoft SQL Server. The developed system was then used to (1) redevelop the case study pipe spool fabrication model; and (2) develop a new model of for building finishing activities. Both models were developed fairly quickly. For each model only the domain knowledge for the model was required. No simulation expertise was required to model the situations as was required before using the traditional modeling techniques.

7.2 Limitations

The aim behind this research is to make simulation more accessible to construction end users by eliminating the need for developing a traditional simulation model or amending one on a construction project for every new situation encountered. The proposed methodology will allow construction end users to model various construction situations which are repetitive in nature (i.e. roads, buildings, pipelines, industrial fabrication, etc…) successfully as long as the situation can be described using data that will have a functional

fit within the model descriptive structures defined, and where relevant output analysis can be usefully made from simulation result data structured as per the output structures of the methodology being proposed. A construction simulation model can be produced using the proposed methodology only if (1) its product properties can map to the proposed methodology's product definition structures; (2) its process properties can map to the proposed methodology's process definition structures including tasks, task sequences, resources, resource availability; (3) its quantities of work for each task to produce the product are available and can map to the proposed methodology's product-task-qty; (4) in the case of tasks applying to different product levels, a matrix defining task applicability to product level can map to the proposed methodology's product-task-level matrix; and (5) the result set produced by the methodology can map back to the end user's requirements and aid in answering the question for which the simulation model is required.

7.3 Recommendations for Future Research and Development

Application of the proposed methodology results in a simulation system composed of input and output data structures and a DES compiler. As shown through the prototype and the enterprise production version instances of the methodology, it may be developed using different tools and in different environments which makes it adaptable to being implemented and integrated within varying corporate and academic ecosystems. One such integration maybe with corporate construction ERP systems where simulations maybe run seamlessly from within the ERP system using the ERP system's data and used to return data for analysis. With this type of integration, model descriptive data sets supporting the simulation system would be constructed in a manner which maps to the proposed methodology's structures. ERP system users would then be able to call the simulator from within the ERP system natively with specific parameters that would define the products to

be built linked to their process information. The integration module would automatically fill the model descriptive structures using ERP data, run the DES compiler, produce a model, run it, and return the data in a transparent manner. The returned data would then act as a basis for analysis and reporting from within the ERP system or using external analysis tools. Such integration would make simulation even more accessible to construction end users as it becomes a native part not only of their domain expertise but also of their day to day corporate ERP tools.

The proposed methodology addresses a need within the construction industry to make simulation more accessible to end users. The proposed methodology may also be applied to different industries where it may help in low level resource planning and schedule validation. Although the product definitions, process definitions, product-task mappings, and simulator outputs of the methodology were developed based on the construction industry, these may be adapted to various industries by changing the structures to map to an intended industry use.

References

AbouRizk, S. (1995). Automating the Process of Building Simulation Models. *Proceedings of the Winter Simulation Conference .* Washington, D.C.

AbouRizk, S. (1998). Simulation. *Fifth Canadian Construction Research Forum*, (pp. 55-68).

AbouRizk, S. (2009). Framework for Highly Integrated, Interoperable Construction Simulation Environments. *ICCEM-ICCPM.* Jeju, Korea.

Abourizk, S. (2010). Role of Simulation in Construction Engineering and Management. *Journal of Construction Engineering and Management*, 1140-1153.

AbouRizk, S. M. (2006). Synthetic Environments for Construction Planning and Control. NSERC Industrial Research Chair Research Program.

AbouRizk, S., & Hajjar, D. (1998). A Framework for Applying Simulation in the Construction Industry. *Canadian Journal of Civil Engineering*, 604-617.

AbouRizk, S., & Mohammad, Y. (2000). Simphony—An Integrated Environment for Construction Simulation. *Proceedings of the 2000 Winter Simulation Conference*. Orlando, Florida.

Abourizk, S., Halpin, D., Mohamed, Y., & Hermann, U. (2011). Research in Modeling and Simulation for Improving Construction Engineering Operations. *Journal of Construction Engineering and Management*, 843.

AbouRizk, S., Shi, J., McCabe, B., & D., H. (1995). Automating the Process of Building Simulation Models. *Proceedings of the Winter Simulation Conference* , (pp. 1032-1038). Washington, D.C.

Anylogic. (2013, April 14). *The Big Book of Anylogic.* Retrieved from Anylogic: http://www.anylogic.com/the-big-book-of-anylogic

Balkany, A., Birmingham, W., Maxim, B., Runkel, J., & Tommelein, I. (1994). DIDS: Rapidly Prototyping Configuration Design Systems. *of Intelligent Manufacturing*, (5) 33-45.

Barrie, D., & Paulson, B. (1992). *Professional Construction Management, Including C.M., Design Construct, and General Contracting* (3rd ed.). New York: McGraw-Hill, Inc.

Caterpillar. (2001). *Caterpillar Performance Handbook.* Caterpillar Inc.

Chang, D. Y., & Carr, R. I. (n.d.). RESQUE: A resource oriented simulation system for multiple resource constrained processes. *Proc., 1987 PMI Seminar/Symp.*, 4-19.

Department of Defense (DoD) Modeling and Simulation Coordination Office. (2013, July 14). *Verification, validation, & accreditation (VV&A) recommended practices guide (RPG).".* Retrieved from Department of Defense (DoD) Modeling and Simulation Coordination Office (M&SCO): http://msco.mil/VVA_RPG.html

Gaarslev, A. (1969). *Stochastic models to estimate the production of material handling systems in the construction industry .* Stanford University, Dept. of Civil Engineering.

Hajjar, D., & AbouRizk, S. (1999). Simphony: An Environment for Building Special Purpose Construction Simulation Tools. *Proceedings of the 1999 Winter Simulation Conference.* Phoenix, Arizona.

Hajjar, D., & AbouRizk, S. (2000). Application Framework for Development of Simulation Tools. *Journal of Computing in Civil Engineering*, 14: 160-167.

Hajjar, D., & AbouRizk, S. (2002). Unified Modeling Methodology for Construction Simulation. *Journal of Construction Engineering and Management*, 128: 174-185.

Halpin, D. W. (1973). An investigation of the use of simulation networks for modeling construction operations. Dissertation, Department of Civil Engineering, University of Illinois, Urbana-Champaign, Illinois.

Halpin, D. W. (1977). CYCLONE: Method for modeling of job site processes. *J. Constr. Div.*, 103(3), 489–499.

Halpin, D. W. (1990). *MicroCYCLONE User's Manual.* West Lafayette, Indiana: Division of Construction Engineering and Management, Purdue University.

Halpin, D., & Woodhead, R. (1976). *Design of Construction and Process Operations.* New York, N.Y.: John Wiley and Sons, Inc.

Halpin, D., Jen, H., & Kim, J. (2003). A construction process simulation web service. *Proceedings of the 2003 Winter Simulation Conference* , (pp. 1503-1509). New Orleans, LA.

Hassan, M., & Gruber, S. (2008). Simulation of concrete paving operations on Interstate-74. *Journal of Construction Engineering and Management*, 134(1): 2-9.

Hu, D., & Mohamed, Y. (2011). Effects of pipe spool sequences in industrial construction processes. *Proceedings of CSCE 2011 Construction Specialty Conference* (pp. CN144-1 – CN144-10). Ottawa, ON: Canadian Society of Civil Engineers.

Huang, R.-Y., & Halpin, D. W. (1994). Visual Construction Operation Simulation: The DISCO Approach. *Computer-Aided Civil and Infrastructure Engineering*, 9: 175–184.

Institute of Electrical and Electronics Engineers. (2000). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.* New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ioannou, P. G. (1989). *UM-CYCLONE User's Guide.* Ann Arbor, Michigan: Department of Civil Engineering, The University of Michigan.

Ioannou, P. G., & Martínez., a. J. (1996). Simulation of complex construction processes. *Proceedings of the 1996 Winter Simulation Conference* , (pp. 1321-1328). Coronado, California.

Kleijnen, J. P. (1995). Verification and validation of simulation models. *Eur J Oper Res*, 82(1), 145-162.

Kuhl, F., Weatherly, R., & Dahmann, J. (1999). *Creating Computer Simulation Systems. An Introduction to the High Level Architecture.* Prentice Hall.

Labban, R., AbouRizk, S., & Hague, S. (2009). Demonstrating the Usefulness of the High Level Architecture in Simulating Construction. *Proceedings of the CSCE Annual Conference.* St. John's, NL.

Labban, R., AbouRizk, S., Haddad, Z., & El-Sersy, A. (2013). A Discrete Event Simulation Model of Asphalt Paving Operations. *Proceedings of the 2013 Winter Simulation Conference.* Washington, D.C.

Labban, R., AbouRizk, S., Haddad, Z., & El-Sersy, A. (2013). A pipe spool fabrication simulation model. *Proceedings of the 25th EMSS.* Athens, Greece.

Law, A., & Kelton, W. (1991). *Simulation Modeling and Analysis.* New York, N.Y.: McGraw-Hill.

Lingineni, M., Caraway, B., Benjamin, P. C., & Mayer., R. J. (1995). A tutorial on PROSIMTM: A knowledge-based simulation model design tool. *Proceedings of the 1995 Winter Simulation Conference* , (pp. 408-412). Arlington, Virginia.

Liu, L. Y., & Ioannou, P. G. (1992). Graphical object-oriented discrete event simulation system. *Proceedings of the 1992 Winter Simulation Conference* , (pp. 1285–1291). Arlington, Virginia.

Lucko, G. R. (2010). Research Validation in the Construction Domain: Challenges and Opportunities. *Journal of Construction Engineering and Management*.

Lucko, G., Benjamin, P. C., & Madden, M. G. (2008). Harnessing the power of simulation in the project management / decision support aspects of the construction industry. *Proceedings of the 2008 Winter Simulation Conference* , (pp. 2479-2487). Miami, Florida.

Lucko, G., Swaminathan, K., Benjamin, P., & Madden, M. (2009). Rapid deployment of simulation models for building construction applications. *Proceedings of the 2009 Winter Simulation Conference*, (pp. 2733-2744). Austin, Texas.

Manavazhi, M., & and AbouRizk, S. (1997). Configuration-based Simulation Modeling. *CSCE Annual Canadian Conf.*, (pp. 147-154). Sherbrooke, Quebec.

Martinez, J., & Ioannou, P. G. (1994). General purpose simulation with Stroboscope. *Proceedings of the 1994 Winter Simulation Conference* , (pp. 1159-1166). Orlando, Florida.

McCahill, D. F., & Bernold, L. E. (1993). Resource Oriented Modeling and Simulation in Construction. *Journal of Construction Engineering and Management*, 590-606.

Mohamed, Y., & AbouRizk, S. (2001). Simulation Made Easy and Effective. *29th Annual Conference of the Canadian Society for Civil Engineering*, (pp. A-45). Victoria, BC.

Mohamed, Y., & AbouRizk, S. (2005). Framework for Building Intelligent Simulation Models of Construction Operations. *Journal of Computing in Civil Engineering*, 277-291.

Mohamed, Y., & AbouRizk, S. (2006). A Hybrid Approach for Developing Special Purpose Simulation Tools. *Canadian Journal of Civil Engineering*, 1505-1515.

Mohammed, Y., AbouRizk, S., & and Hajjar, D. (2000). Creating Special Purpose Simulation Tools with Simphony. *Proceedings of the Construction Congress VI*, (pp. 87-96). Orlando, Florida.

Odeh, A. M., Tommelein, I. D., & Carr, R. I. (1992). Knowledge based simulation of construction plans. *Proceedings of the 8th Conference on Computing in Civil Engineering*, (pp. 1042-1049). New York.

Oloufa, A. (1993). Modeling operational activities in object-oriented simulation. *J. Comput. Civ. Eng.*, 94-106.

Oloufa, A. (1994). Construction Simulation of Buildings with a User-Oriented Approach. *Microcomputers in Civil Engineering*, 425-433.

Paulson, B. C. (1987). Construction operation simulation by microcomputer. *J. Constr. Eng. Manage.*, 302-314.

Peurifoy, R. L., & Schexnayder, C. (2001). *Construction Planning, Equipment, and Methods.* Columbus, OH: McGraw-Hill.

Pritsker, A. (1986). *Introduction to Simulation and SLAM II* (2nd ed.). New York, NY, and West Lafayette, IN: Wiley and Pritsker Associates.

Sadeghi, N., & Fayek, A. R. (2008). A framework for simulating industrial construction processes. *Proceedings of the 2008 Winter Simulation Conference* , (pp. 2396-2401). Miami, Flordia.

Sargent, R. G. (1991). Simulation model verification and validation. *Proceedings of the 1991 Winter Simulation Conference* , (pp. 37–47). Phoenix, Arizona.

Sawhney, A. (1994). Simulation-based planning for construction. Thesis (PhD). University of Alberta.

Sawhney, A., & AbouRizk, S. (1993). An Overview of Construction Simulation and Its Applications. *Journal of Construction Management*, 93-100.

Sawhney, A., & AbouRizk, S. (1996). A Computerized Tool for Hierarchical Simulation Modeling. *Journal of Computing in Civil Engineering*, 115-124.

Shahin, A., AbouRizk, S., Mohamed, Y., & Fernando, S. (2007). A Simulation-based Framework for Quantifying the Cold Regions Weather Impacts on Construction Schedules. *Proceedings of the 2007 Winter Simulation Conference.* Washington, D.C.

Shahin, A., Mohamed, Y., & AbouRizk, S. (2006). Quantifying Weather Impacts on Construction Activities – An Application on HDPE Pipeline Installation. *Proceedings of the International Construction Specialty Conference.* Calgary, Alberta.

Shewchuk, J., & Chang, T. (1991). An approach to object-oriented discrete-event simulation of manufacturing systems. *Proceedings of the 1991 Winter Simulation Conference*, (pp. 302-311). Phoenix, Arizona.

Shi, J., & AbouRizk, S. (1997). Resource-Based Modeling for Construction Simulation. *Journal of Construction Engineering and Management*, 26-33.

Shi, J., & AbouRizk, S. (1998). An Automated Modeling System for Simulating Earth-moving Operations. *Journal of Computer-Aided Civil and Infrastructure Engineering*, 121-130.

Song, L., Wang, P., & AbouRizk, S. (2006). A virtual shop modeling system for industrial fabrication shops. *Simulation Modeling Practice and Theory* , 649-662.

Teicholz, P. (1963). A simulation approach to the selection of construction equipment.

Tommelein, I., Carr, R., & Odeh, A. (1994). Assembly of Simulation Networks using Designs, Plans, and Methods. *J. of Constr. Engrg. and Mgmt.*, 796-815.

Wales, R., & AbouRizk, S. (1996). An Integrated Simulation Model for Construction. *Simulation Practice and Theory*, 401-420.

Wang, P., & AbouRizk, S. (2009). Large-scale Simulation Modeling System for Industrial Construction. *Canadian Journal of Civil Engineering* , 1517-1529.

Wang, P., Mohamed, Y., AbouRizk, S., & Rawa, A. (2009). Flow Production of Pipe Spool Fabrication: Simulation to Support Implementation of Lean Technique. *Journal of Construction Engineering Management*, 1027-1038.

Zayed, T. M., & Halpin, D. W. (2001). Simulation of concrete batch plant production. *Journal of Construction Engineering and Management* , 132-141.

Zayed, T. M., & Halpin, D. W. (2004). Simulation as a tool for pile productivity assessment. *Journal of Construction Engineering and Management* , 394-404.

Zeigler, B. P., Praehofer, H., & Kim, T. G. (2000). *Theory of Modelling and Simulation* (2nd ed.). Academic Press.