

University of Alberta

**Incorporating the Effects of Complex Dynamic Interactions in the
Construction Decision Making Process**

By

Amin Alvanchi

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

Doctor of Philosophy

in

Construction Engineering and Management

Department of Civil and Environmental Engineering

©Amin Alvanchi

Fall 2011

Edmonton, Alberta

Permission is hereby granted to the University of Alberta Libraries to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only. Where the thesis is converted to, or otherwise made available in digital form, the University of Alberta will advise potential users of the thesis of these terms.

The author reserves all other publication and other rights in association with the copyright in the thesis and, except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatsoever without the author's prior written permission.

Examining Committee

Dr. Simaan AbouRizk, Civil and Environmental Engineering

Dr. SangHyun Lee, Civil and Environmental Engineering, University of Michigan

Dr. Aminah Robinson Fayek, Civil and Environmental Engineering

Dr. Marwan El-Rich, Civil and Environmental Engineering

Dr. Eleni Stroulia, Computing Science

Dr. Lucio Soibelman, Civil and Environmental Engineering, Carnegie Mellon University

Dedication

This thesis is dedicated with love, admiration, and respect

to my kind mother, dear father, my lovely wife

and my beloved brothers;

Abstract

Construction projects involve complex interactions among operational components such as labour, materials and equipment and context or organizational components such as worker morale and organizational policies. Interactions among different components of construction projects form byzantine chains of cause and effect, and determining their final impact on project behaviour can be beyond human capabilities. Specialized tools that can capture these interactions and provide perspective on the outcomes of construction managers' prospective decisions are needed. In this research I proposed and applied a modeling approach that uses a hybrid model of System Dynamics (SD) and Discrete Event Simulation (DES), combining the capabilities of these two powerful modeling tools to foresee a construction project's ultimate outcome as the result of changes in the different components.

The first stage of the research was to recognize different aspects of the hybrid SD-DES modeling approach and to assess the current and potential challenging issues which might affect hybrid model developments in the construction domain. A customized hybrid modeling framework and architecture targeted construction projects and was developed to address the previously described challenges. The hybrid modeling framework is meant to assist hybrid model developers during the design phase; the framework provides and suggests a set of tools that can be used during the implementation phase of hybrid model development.

The proposed hybrid model framework and architecture provided the foundation used in the next stage of the research: applying hybrid SD-DES modeling approach to complex construction decision making problems. Two common decision and policy making problems found in construction projects – identifying improved working hour arrangements and human resource policies – were analyzed in this stage, and original hybrid models were developed to assist construction managers in finding the best answers. The models developed for both applications were then validated through real construction projects.

In sum, in this research I introduced and validated a new hybrid approach which can be used for improving complex dynamic construction decision making processes by capturing feedbacks between operational level and organizational level effective factors within construction projects. Furthermore, the research contributes to two prevalent construction decision making problems (construction working hour arrangement and human resource policy improvement) by developing hybrid models which can be used for improving the decision making processes.

Acknowledgement

I would like to express my deepest gratitude to my beloved family who taught me how to live. My sincere thanks to my lovely parents for their never ending support. I am but a product of your dreams and sacrifices. Although I have been away from you, your prayers have always paved my road to success. This journey could not be finished successfully without the help, support, and warm companionship of my dearest lovely wife Soreh. How her kind companionship can be appreciated! Thank you for your continuous love, inspiration, and patience. Words cannot express my gratitude and everlasting love toward you.

This research work would not have been done without the kind support of my supervisors, Dr. Simaan AbouRizk and Dr. SangHyun Lee. I have been truly fortunate to have both of them as my mentors. Dr. AbouRizk guided me with his keen advice, great insights, and confidence throughout the whole dissertation process. Dr. SangHyun Lee also inspired me with his wisdom, knowledge and effective guidance. I thank both of them and will be in their debt forever.

I would like to acknowledge Dr. Lucio Soibelman for serving as my external examiner and providing supportive advice. I am also grateful to my doctoral committee chair Dr. Aminah Robinson, and other committee members Dr. Eleni Stroulia, and Dr. Marwan El-Rich, for their thoughtful review and invaluable suggestions.

I would like to take this opportunity to thank all the individuals, colleagues and friends who assisted me during this research project. I extend my sincere thanks to Paul Zubick and Jim Kanerva, the Managers at Waiward Steel Fabricators Ltd., for their support and collaboration during this research project. I also appreciate the assistance of Dr. Yasser Mohammed, Brenda Penner, Holly Parkis, Stephen Hague, Maria Al-Hussein and other faculty members and staff in the Construction Engineering and Management Program who assisted me in various aspects of this research.

And finally my special thanks go to my dear brother Hosein who was always an honest help, a good colleague and a true friend during this journey to me.

Table of Contents

Chapter 1.	Introduction	1
1.1.	Problem Statement	1
1.2.	Research Objectives	4
1.3.	Scope of Research	6
1.4.	Implementation Environment	8
1.5.	Terminology	10
1.6.	Thesis Organization	12
Chapter 2.	Hybrid Framework and Architecture	14
2.1.	Introduction	14
2.2.	Hybrid Simulation Challenges	18
2.2.1.	Lack of Modeling Framework	18
2.2.2.	Time Advancing	20
2.2.3.	Communication Architecture	21
2.3.	Hybrid Modeling Framework	23
2.3.1.	Basic Hybrid Structures	23
2.3.2.	Different Forms of Interacting Variables	29
2.4.	Proposed Hybrid Architecture	31
2.4.1.	Time Advancing Assessment	32
2.4.2.	Communication Architecture	36
2.5.	Experimental Model	39
2.5.1.	Case Description	39
2.5.2.	Fabrication Shop Hybrid Model Development	40
2.5.3.	Performance Test	54
2.5.4.	Expandability Test of The Model	59
2.6.	Chapter Conclusion	62
Chapter 3.	Dynamics of Working Hours in Construction	65
3.1.	Introduction	65

3.2.	Dynamics of The Working Hours.....	67
3.2.1.	Dynamics of Prolonged Working Hours.....	67
3.2.2.	Dynamics of Time of Day.....	84
3.2.3.	Dynamics of Overtime Work.....	87
3.2.4.	Critical Parameters for Worker Capability	92
3.3.	Model Testing	94
3.4.	Sensitivity Analysis of Model Behavior.....	98
3.5.	Hybrid Model of Integrated Working Hour Dynamics.....	101
3.6.	Case Study	103
3.6.1.	Case Specification.....	104
3.6.2.	Base Model Experiment.....	105
3.6.3.	Shift Alternatives	109
3.6.4.	Case Analysis.....	110
3.7.	Chapter Conclusion.....	112
Chapter 4.	Construction Workforce Evolution Dynamics	114
4.1.	Introduction.....	114
4.2.	Conceptual Model of The Workforce Dynamics.....	116
4.2.1.	Workforce Dynamics in Literature	116
4.2.2.	Conceptual Model of Workforce Dynamics in Construction	119
4.2.3.	Model Analysis	125
4.3.	Inclusive Model of Construction Workforce Dynamics.....	132
4.3.1.	Core Dynamic Model of Workforce Evolution	132
4.3.2.	Workload Dynamics	136
4.3.3.	Overtime Dynamics	138
4.3.4.	Dynamic Data Collecting.....	141
4.3.5.	Model Validity	143
4.4.	Hybrid Model of Construction Worker Dynamics	149
4.5.	Case Study	150
4.5.1.	Case Specification.....	151

4.5.2. Base Model Experiment.....	152
4.5.3. Alternative Policies Simulation	154
4.5.4. Case Analysis.....	158
4.6. Chapter Conclusion.....	159
Chapter 5. Conclusions and Recommendations	161
5.1. Research Summary	161
5.2. Research Contributions.....	165
5.3. Lessons Learned.....	167
5.4. Recommendations for Future Research.....	170
References.....	173
Appendix A. Meaningful Level of Change in Hybrid Simulation for Construction Analysis.....	184
Appendix B. Programming Details of the Simulation Model Used for Hybrid Framework and Architecture Test.....	200
Appendix C. Programming Details of The Simulation Model Used for Working Hours Dynamics	252
Appendix D. Programming Details of the Simulation Model Used for Construction Workers Evolution	266

List of Tables

Table 2-1. Brief assessment of the interacting variables of basic hybrid models.	44
Table 2-2. Steel Construction Federation Object Model Template	48
Table. 2-3. Comparison of the results of the proposed and base hybrid models ..	56
Table. 2-4. Simulation time for different developed models	57
Table 3-1. Average performance indexes during day and night shift.....	85
Table 3-2. Performance indexes in different hours of the day.....	86
Table 3-3. Summary of applied validation tests	96
Table 3-4. Productivity ratio (%) and fabrication duration in calendar days (cd) and working days (wd) achieved in simulation runs of different shift alternatives	110
Table 4-1. Summary of applied validation tests	144
Table 4-2. Base values considered for the test model's exogenous variables and constant parameters.....	145
Table 4-3. Simulation results for employment and overtime extreme policies ..	146
Table 4-4. Simple alternative policies applied to the case.....	155
Table 4-5. Results achieved for simple alternative policies	156

List of Figures

Figure 2-1. SD-dominant hybrid structure of capital level in a fabricating company	25
Figure 2-2. DES-dominant hybrid structure of a fabrication shop	27
Figure 2-3. Parallel SD and DES to model fabrication operations and labor employment.....	28
Figure 2-4. Turning the conceptual model to HLA based design.....	37
Figure 2-5. Shop structure and material flow of the experimental case	40
Figure 2-6. Sample managerial and context-level feedback loops	41
Figure 2-7. Sample basic hybrid interactions used in the model	42
Figure 2-8. Top-level architecture of the developed hybrid model	46
Figure 2-9. Interface of the calendar federate.....	49
Figure 2-10. Interface of the Data Management federate.....	50
Figure 2-11. Main interface of the DES federate.....	51
Figure 2-12. Stations specification form in the DES federate	52
Figure 2-13. Mid-buffers or storages specification form in the DES federate	52
Figure 2-14. Movers specification form in the DES federate.....	52
Figure 2-15. Interface of the SD federate	53
Figure 2-16. Interface of the Visualization federate	60
Figure 2-17. A snap shot from Tekla during the model run	61
Figure 2-18. A schematic view of integration between the developed federation and RFID technology.....	62
Figure 3-1. The dynamics of physical fatigue as a result of prolonged high physical involvement	68
Figure 3-2. The dynamics of mental fatigue as a result of prolonged sustained attention.....	68
Figure 3-3. The dynamics of performance factor changes based on biological clock.....	85
Figure 3-4. The dynamics of overtime working and performance	89
Figure 3-5. The effect of different types of fatigue on workers' productivity ratio	97

Figure 3-6. Deviations in the efficient working hours by changing length of working periods	99
Figure 3-7. The effects of start time on productivity ratio in physical and mental tasks.....	100
Figure 3-8. The effects of overtime on weekly efficient working hours	100
Figure 3-9. The effects of worker's strength on productivity ratio in Physical and mental tasks.....	101
Figure 3-10. Modified dynamics of fatigue models in hybrid model.....	103
Figure 3-11. Comparison between completed fabrication in simulation and actual case.....	108
Figure 4-1. Conceptual dynamic model of experience chain	117
Figure 4-2. Conceptual model of construction workforce	119
Figure 4-3. Four-level dynamic model for construction Workforce.....	125
Figure 4-4. Workers fluctuation in construction workforce evolution and experience chain dynamics	127
Figure 4-5. Fluctuation in experienced workers percentage over the course of the project	129
Figure 4-6. Weekly workload distribution for different work skills during the project	130
Figure 4-7. Effects of skill distinction on the number of workers	131
Figure 4-8. Core dynamic model of the workforce evolution in a two-level skill/wage organization	133
Figure 4-9. Supporting dynamics on the required workload	137
Figure 4-10. Overtime dynamics in construction projects.....	140
Figure 4-11. Built in mechanism for collecting the cost information.....	141
Figure 4-12. Sensitivity analysis on the maximum number of new workers	148
Figure 4-13. Comparison between the results from simulation model and the actual results.....	154

CHAPTER 1. Introduction ¹

1.1.Problem Statement

The operational details involved in construction projects, the variable nature of construction projects over the course of time, and the significant impacts of human factors on the project progress add to the complexity of construction projects. For completing a construction project a complex combination of dependent tasks takes place using a variety of workers with different types and levels of expertise equipped with a range of tools with various functionalities and capabilities. Missing any operational detail in the project might have a significant effect on the project. For example just shipping out of order material to the project's job site can cause long delays in the project completion.

Many construction projects continue for several years during which different effective factors in the project dynamically fluctuate. Many of these fluctuations in the project happen as the results of changes in project performance over time. For example, worker skill continuously grows as a project goes on and the workers gain experience because of their involvement in the project, or equipment performance is diminished as it wears out as a result of the assigned workload. The managers' decisions and organizational policies also contribute to project

¹ Parts of this chapter have been published in the Proceeding of the Construction Research Congress, Seattle, April 2009, pp. 1290-1299.

fluctuations over time. For example, workload thresholds set for hiring new or firing redundant workers, incentive policies put in place for workers, and decisions made to outsource a part of the project affect the pace of project progress and its future condition. Changes imposed by the project's environment are another group of changes affecting project's fluctuations. Weather changes, job market fluctuations and material market conditions are some examples of fluctuations which may affect project condition.

Unlike the tool based and repetitive nature of manufacturing, every construction project is unique and requires an enormous amount of human communication and judgment during the project implementation. Therefore, the way that the project crew reacts in response to its assigned tasks might be different from one project to the other. In fact, the level of the productivity expected from a group of workers working on the same type of project during winter, spring, summer and fall can be quite different; so can the workers' fatigue from different types of tasks.

Mutual interactions between organizational effective factors and construction operation form different effective feedback loops within the construction projects which prevent construction managers from tracking the ultimate effects of their decisions on the project's productivity and final cost using traditional project management tools. This causes construction managers to remain dependent on their past experiences and common sense during their daily decisions, which are subject to the human mind's inference limitations. Some examples of such

decision making situations in construction projects can be 1) whether to set overtime, to set a new shift, to outsource the job or to hire new workers as the response to over-capacity workload assigned to the project's crew, 2) what is the best working hour arrangement for the new project assigned? or 3) what combination of the best incentive and penalty policies results in the highest productivity for the project? Some of these decisions, especially when they form a project's policies or determine trends or structures to be followed perpetually, can have substantial impacts on the final project outcome.

In this research I have addressed this concern of construction project managers and have proposed and developed a framework and architecture which can capture the effects of complex internal and external interactions among different organizational and operational effective factors on construction project outcomes. This framework and architecture will assist construction managers by tracing the effects of their alternative decisions and improve their project performance by allowing them to select the alternative which gives the improved result.

The proposed framework and architecture contains a hybrid modeling approach which is formed by integrating system dynamics (SD) and discrete event simulation (DES) modeling tools. Both SD and DES are well-known simulation-based decision support tools which have been used in many construction simulation cases (e.g., Halpin 1973, Paulson 1987, Martinez et al. 1994, AbouRizk and Hajjar 1998 have used DES and Sterman 1992, Park and Pena-

Mora 2003, Ford et al. 2004, Lee et al. 2006 have applied SD to construction simulation cases). While SD is an appropriate tool for capturing the effects of system control variables which affect the system's behavior through chains of cause and effect and create causal feedback loops, DES is useful for modeling operational details of the systems. Although most real world systems are combinations of feedback and sequential processes, context or organizational - level and operation-level parameters, and continuous and discrete changing variables, most simulation-based analyses are still facilitated by the use of either SD or DES, with the modeller's selection based on which tool can capture more aspects of the problem. This conventional type of analysis may force system analysts to make simplifying assumptions and to accept a lower level of accuracy. Hybrid SD and DES modeling framework and architecture provides a set of tools that use the capabilities, while improving upon the disadvantages, of these two approaches. The result of this is a stronger modeling tool which can capture more aspects of the construction project; however, there are some implementation challenges which should be addressed as well.

1.2. Research Objectives

The main objective of this research is to introduce a new framework and architecture to construction project managers with which they will be able to follow the effective feedback loops between different organizational and operational effective factors within construction project. The introduced

framework and architecture let them to trace the ultimate effects of their decisions on the final construction project output and pick the choices that improve their projects the most. The functionality and applicability of the proposed hybrid SD-DES modeling framework and architecture are tested and validated to make sure that the proposed hybrid SD-DES modeling framework and architecture can be used for assisting construction managers in their complex dynamic decision making problems.

The research also elaborates on two applications of using a hybrid modeling approach in construction projects. These applications are firstly meant to serve as examples for the applicability and usefulness of the proposed hybrid framework and architecture and, secondly, aim to introduce new interactions within construction projects which have not been looked into in the past, since traditionally used tools have limited capabilities for capturing them. Therefore, the use of a hybrid SD-DES modeling approach represents a new area, still unexplored, and where new insight on these interactions within the construction domain can be found. With this perspective, another objective of the research is introducing new hybrid interactions within the construction domain which result in a better understanding of the effective mechanisms of construction.

1.3.Scope of Research

Although research on using hybrid SD-DES simulation models for modeling the behavior of the complex systems has been around since the early 2000s (Rus et al. 1999; Zeigler et al. 2000; Martin and Raffo 2001; Choi et al. 2006), there are some challenging issues which have prevented them from being applied to construction cases. In the first phase of this research I have looked into these challenging issues and proposed a hybrid SD-DES modeling framework and architecture to address them. The proposed hybrid modeling framework introduces a set of hybrid concepts, terminologies, and guidelines to be used and followed when designing the hybrid models. The proposed architecture suggests a group of tools to be used for implementing hybrid simulations. The applicability and usefulness of the proposed hybrid framework and architecture were then tested by applying the framework and architecture in a real construction case. A detailed explanation of the hybrid framework and architecture is given in Chapter 2.

The proposed framework and architecture in the first phase of the research created a strong foundation upon which to base the second phase of my research into exploring and capturing different aspects of complex construction project behaviours. Improving the working hour arrangement for construction projects by exploring and capturing the effects of alternative arrangements on the final output of the construction projects was the first application explored. This is a decision

making problem that perpetually affects construction projects. However, complex interactions among a variety of organizational and operational effective parameters in every working hour arrangement do not allow construction managers to evaluate final effects of different working hour alternatives using conventional decision making tools. Some examples of these effective parameters are work start time and finish time, duration of the work and rest periods, shift work and the amount of overtime during the week. Many studies have measured the effects of changes in single aspects of the working hours on worker performance levels, such as shift work and time of the work (e.g., Dijk et al. 1992; Folkard and Tucker 2003; Baltter and Cajochen 2007), work duration (e.g., Taylor 1911; Rohmert 1973b; Oglesby et al. 1989) or overtime (e.g., Homer 1985; Sterman 2000; RSMMeans 2010). However, in addition to the limited aspects they cover, these types of research efforts typically result in proposing general guidelines for working hour adjustments, rather than introducing a tool that can be customized based on some system parameters, then applied to different projects with different specifications. A hybrid SD-DES modeling approach was used in this part of the research to introduce a new inclusive approach for improving working hour arrangements for construction projects. The results achieved in a variety of studies into different aspects of the working hours first were compiled into dynamic models and then were integrated in the form of a hybrid SD-DES model of construction working hours. It can be dynamically applied to different types of construction projects with different operational details and specifications

to track project behaviour with different working hour adjustments. A complete explanation of this application is provided in Chapter 3.

As another application of the hybrid SD-DES modeling approach I modeled workforce evolution dynamics within construction projects. This application was meant to propose a method for modeling the effects of human resource policy adjustment in construction projects which is a major concern in many construction companies. A generic SD model customized for construction projects was first developed in this part of the research for capturing the causal effects of human policy adjustments and then was integrated with the DES model of a construction project in a hybrid manner. The model was then used to improve the human resource policies of a real construction case. Chapter 4 provides a detailed explanation of this application.

1.4.Implementation Environment

The proposed hybrid SD-DES modeling framework and architecture as well as its applications have been implemented and validated in real construction cases. A different set of computer software packages were used for these model implementations, explained below:

- Microsoft Visual Studio.NET was used as the programming environment during the first phase of the research. All externally used software packages in phase one of the research were either implemented directly (e.g., the SD

part of the model) or controlled and managed by Visual Studio.NET (e.g., DES and supporting packages)

- The Symphony.NET 3.5 modeling engine (<http://irc.construction.ualberta.ca/Symphony35/>) was used for implementing the DES part of the hybrid framework. The Symphony.NET 3.5 modeling engine provided a pre-developed set of required DES classes and functions to facilitate DES model implementation in my experiment in the first phase of the research.
- The Construction Synthetic Environment (COSYE) (AbouRizk and Hague 2009) facilitated data communication between different parts of the hybrid model and provided a framework through which different parts of the previously developed hybrid models could be implemented independently and run in a distributed manner on multiple computers.
- Microsoft Access (MS Access) was used as the database medium in the entire research project. The construction project data used in the research either were directly stored in MS Access and used as the data fed into the simulation model or were directed to the simulation model through the links to the remote server (where the original data were stored) provided by MS Access (run on the Windows XP operating system).
- AnyLogic 6.4 (<http://www.xjtek.com/>) was the main simulation environment in the second phase of the research used for implementing the experimental cases of the hybrid SD-DES modeling applications. AnyLogic

provided SD, DES and the communication channels required for the hybrid model at this phase of the research.

1.5. Terminology

Different terms associated with the hybrid SD-DES modeling approach are frequently used throughout this document. Since the use of the hybrid SD-DES modeling approach is a new experience in the construction domain, and its related terminology still is not commonly agreed upon and documented, to eliminate any points of confusion related to the terminology, the following definitions are provided for the major terms used in the research. These definitions do not necessarily match definitions provided for similar terms in other areas of research or disciplines.

Effective Factor/Parameter: When any changes made to an internal (indigenous) or external (exogenous) element, component or aspect can affect the system behaviour, this element, component or aspect is called an effective factor/parameter for that system. The number of project resources, level of skill in the workers, and the environmental condition on the job site are some examples of effective factors/parameters in construction projects.

Operation Level of the Construction System: The operation level “is concerned with the technology and details of how construction is performed.” (Halpin 2011, p17) The operation level aspects of construction systems can be measured directly

by using the prevalent measuring tools. The capacity and number of project resources, the durations of construction tasks, and the hauling distance of the truck are in the operation level of the system.

Context/Organizational/ Non-operation Level of the Construction System:

Different aspects of the system which cannot be measured directly by using common measuring apparatus are categorized in the context or organizational level of the system. These aspects of the system are usually linked to human behaviour. Level of fatigue, level of skill, and satisfaction level of the workers are some examples of aspects of the system in the context or non-operation level.

Hybrid or Combined Interaction in the System: Hybrid or combined interaction in the system happens when continuously changing variables (e.g., by reaching a threshold) trigger changes in discretely changing system variables or, conversely, when discretely changing variables affect values of continuously changing variables in the system (Pritsker et al. 1997). Reduction in worker fatigue (a continuously changing variable) as a result of change in work status from work period to rest break (a discretely changing variable) and employing new workers (discretely changing variable) as a result of reaching a certain level of work delay (a continuously changing variable) are two examples of hybrid interaction.

1.6. Thesis Organization

The organization of the remainder of this thesis is as follows: Chapter 2 provides an introduction to hybrid simulation and its related literature and elaborates on the hybrid framework and structure proposed in the research. A major part of the discussion provided in this chapter has been published in the Journal of Computer Aided Civil and Infrastructure Engineering (Alvanchi et al. 2011a) and some other parts have been published in the Proceedings of the Construction Research Conference, Seattle, April 2009 (Alvanchi et al. 2009a) and the Proceedings of the 10th International Conference on Construction Applications of Virtual Reality, Sendai, Miyagi, Japan, Nov 2010 (Alvanchi et al. 2010). Chapter 3 presents the research into exploring new hybrid interactions related to construction working hour adjustments and human fatigue. The development of the related hybrid model has been explained in this chapter as the first application of the hybrid SD-DES modeling approach. The majority of the material presented in this chapter is published in the ASCE Journal of Construction Engineering and Management (Alvanchi et al. 2011b). Chapter 4 explains the hybrid model developed for the dynamics of workforce evolution customized for construction projects. This is another example (out of a variety of hybrid model applications) of how the hybrid SD-DES modeling approach can be used for improving construction projects. Parts of the material presented in this chapter have been published in the Proceedings of the Annual Conference of Canadian Society for Civil Engineering, Ottawa, Canada, June 2011 (Alvanchi et al. 2011c) and some

other parts are in preparation to be submitted in the form of a paper to the journal of Automation in Construction (Alvanchi et al. 2011d). Chapter 5, the research conclusion, includes the research summary, research contributions, lessons learned, and, finally, recommendations for future research work. Appendix A presents a paper on the “Meaningful Level of Change in Hybrid Simulation for Construction Analysis” published in the Proceedings of the Winter Simulation Conference, Austin, Texas, USA, Dec 2009 (Alvanchi et al. 2009b). This paper elaborates on the Meaningful Level of Change concept, introduced in Chapter 2, as a part of the proposed hybrid modeling framework and architecture.

CHAPTER 2. Hybrid Framework and Architecture²

2.1.Introduction

Construction systems are influenced by operation level and context level variables (Lee et al. 2007). System behaviors may be affected by physical and tangible aspects of system components (i.e. operation level), such as the number of laborers and equipment capacity, as well as non-physical aspects of system components (i.e. context level), such as laborer skill level and organizational policies. The system behavior over time results from interactions among different system components (Sterman, 2000, p. 28). These interactions might remain at the operation level (i.e. among operation variables) or context level (i.e. among context variables), or may be interactions among operation and context variables. Such interactions between operation and context can occur in a bidirectional manner; that is, while the changes in operation variables have effects on the context level variables, the context level variables in turn affect operation level variables.

The main problem with most modeling approaches in this regard is that they can only capture system interactions at either the operation level or context level of the system. However, during system modeling, it should be taken into account

² Parts of this chapter have been published in Computer-Aided Civil and Infrastructure Engineering Journal, 2011, Volume 26, Issue 2, pp. 77-91; the Proceeding of the Construction Research Conference, Seattle, April 2009, pp. 1290-1299; and the Proceeding of the 10th International Conference on Construction Applications of Virtual Reality, Sendai, Miyagi, Japan, Nov 2010, pp. 283-290.

that without considering the effects of feedback between the context level and operation level variables inside that system, the complicated behaviors of construction systems cannot be properly captured, especially over the long-term life cycle of that system.

In an effort to provide decision-makers with more reliable system analysis, a system modeling approach that considers both operation level and context level system variables, while capturing their evolution through feedback and sequential interactions, has been introduced by the researchers. This is a hybrid simulation modeling approach that combines System Dynamics (SD) and Discrete Event Simulation (DES). Hybrid SD-DES simulations attempt to integrate these simulation approaches in order to create a unified modeling approach in which the deficiencies of each approach are compensated by the other approach's capabilities. In the SD modeling approach, the dynamic behaviors of systems are derived from system structures (Richardson and Pugh 1981). SD attempts to capture system structures through chains of cause and effect variables which model the controlling behaviors of systems through causal feedback loops. On the other hand, the DES modeling approach tries to model system behavior as its state evolves over time by following the system events, which are recognized as the change initiators in the state of the system. While SD can capture the effects of context level variables in feedback systems, DES is extremely useful in modeling the effects of operation level variables in sequential systems (for further information, see Borshchev and Filippov, 2004).

The first attempts to develop and use a hybrid SD-DES system originate from the software industry in the late 1990s and early 2000s. At that time, a booming market in the software industry and fears of widespread Y2K software crashes led developers to seek ways to improve the software development process and gain knowledge on how to design and develop more powerful hybrid simulation tools.

Rus et al. (1999) have tried to incorporate feedback loops into discrete-based activities in the software development process, and this work was built upon by Zeigler et al. (2000), Martin and Raffo (2001), and Choi et al. (2006). These research efforts focused on improving the software development process by providing more accurate evaluation strategies required for important decision-making (e.g. staff hiring, training, and firing policies). For example, Martin and Raffo (2001) proposed their hybrid model for the industrial software development process and validated this model through an aerospace software development project. They attempted to address important problems that face software project managers, problems that deal with both the context level and operation level of software development projects, including the effects of staffing and training on the operation, the impact of increased overtime on quality, and the effects of experienced and inexperienced staff arrangements on the operation. In this study, a DES model was used to capture operation details of the software development process and SD was utilized to model the project policies. Most research studies run in the software industry were followed by producing simulation software

applications that can support hybrid simulation models such as ExtendSim, SimuLog and AnyLogic.

More recently, scholars from disciplines other than software engineering, such as manufacturing (Venkateswaran et al. 2004; Rabelo et al. 2005) and construction (Lee et al. 2007; Pena-Mora et al. 2008; Lee et al. 2009) have begun work on hybrid SD-DES simulation. A literature review indicated that research on hybrid simulation can be divided into two main categories: (1) works focused on improving hybrid architecture development (e.g. Zeigler et al. 2000; Borshchev et al. 2002; Venkateswaran et al. 2004; Choi et al. 2006); and (2) works proposing improvements for system modeling by integrating previously neglected hybrid interactions to increase modeling accuracy (e.g. Martin and Raffo 2001; Rabelo et al. 2005; Lee et al. 2007; Pena-Mora et al. 2008; Lee et al. 2009).

So far, hybrid research efforts within the construction industry have been focused on the potential benefits the construction industry could gain by using hybrid simulation; the efforts done in other disciplines are mainly kept within academia. However, we are still missing hybrid models that are fully developed and operational for real construction systems. This can affirm the existence of challenging issues which hinder the use of hybrid SD-DES models within the construction domain. To facilitate the hybrid SD-DES model development process, this research investigates these challenging issues, including the lack of a modeling framework and the time advancing and communication issues, and

proposes a hybrid modeling framework and model development architecture for construction related systems that can be used to address these issues. This chapter consists of the following 5 sections: Section 2.2, an overview of hybrid simulation and its related challenges; Section 2.3, the introduction of the proposed hybrid modeling framework; Section 2.4, the introduction of the hybrid model development architecture; Section 2.5, a case study that applies the proposed framework and architecture at an actual steel fabrication shop to investigate the applicability and functionality of the proposed model; and finally Section 2.6, a brief conclusion.

2.2. Hybrid Simulation Challenges

As indicated by existing hybrid SD-DES research efforts, there remain several challenging issues that create deficiencies and slow down the model development process. These challenging issues have been divided into three main categories: (1) lack of modeling framework; (2) time advancing; and (3) communication architecture. Further explanation of each challenge will be provided in the rest of the section.

2.2.1. Lack of Modeling Framework

The term “hybrid modeling framework” refers to the set of basic elements and concepts that help hybrid model developers during the conceptual design stage of hybrid model development. Construction hybrid modeling aims at capturing the

complex behaviors of construction systems. However, the lack of basic modeling elements and concepts—which could assist hybrid model developers in describing the system and conceptualizing the “big picture” of the model—is a significant hindrance on hybrid model development. In the absence of such a modeling framework, it is possible for model developers to lose view of the scope and main purpose of their built model, or to over-simplify it. In this case, it is more likely that completing such a hybrid model will cost a considerable amount of extra money for model developers, and that the model capabilities will not meet the developers’ expectations.

As demonstrated by different existing hybrid modeling studies, this area of research is still missing a set of modeling elements and concepts that can be used during the conceptual stages of hybrid model development. Although researchers within the software industry have introduced several modeling concepts in this area, these concepts are mainly customized for the software industry and generally attempt to address the implementation details rather than framing and conceptualizing the model. For example, one of the most widely utilized concepts in hybrid modeling development is hybrid formalism, proposed by Zeigler et al. (2000). This concept is specialized for software development projects which follow a phase-to-phase process, whereas in the construction industry, based on the selected construction project delivery system (e.g. design-bid-build, design-build, lump sum, and cost plus), the construction process may have diverse variations.

Therefore, to capture complex behaviors of construction systems, the hybrid modeling framework should have the capacity to consider all types of interactions inside the hybrid model and to help model developers create presentable, well-defined, and comprehensible hybrid model designs for those responsible for model implementation.

2.2.2. Time Advancing

In an SD based model, the time advancing step is a preset interval in which any significant changes can be captured; however, in a DES model, time advancing is based on scheduled events which are created dynamically during the simulation run as a result of previously occurring events. A method to synchronize different interacting simulation models, which can be either continuous or discrete, has been well documented: the High Level Architecture framework (Kuhl et al. 1999). However, another challenging issue can occur, because of the effect of SD and DES interactions on time advancement. While the effects of any interactions between the DES and SD components are considered during the next time step of SD calculations, the interactions originating from the SD components may instantly change and cause successive changes to the timing of the pre-scheduled events of the DES model.

This time advancing issue is amplified when there is a continuously changing variable within an SD related component, which affects some DES modules. In this case, every updated value from the SD module can cause new rescheduling in

the Future Event List (FEL) of the affected DES module. These actions and reactions might radically increase the simulation time required for updating and rescheduling the FEL.

The time advancing issue in hybrid continuous and discrete simulation techniques has also been addressed in other areas besides hybrid SD-DES simulation, such as high speed or concurrent animation and visual interactive simulation (Rekapalli et al. 2009; Rekapalli and Martinez1 2007) in construction related cases. The proposed time advancing methods in these research efforts can also be employed for prospective hybrid SD-DES models when hybrid SD-DES model developers want to add concurrent animation or real time user interactions (regardless of interaction medium) as supporting capabilities.

2.2.3. Communication Architecture

Communication among the different parts of a hybrid model, particularly among the SD and DES components, is another challenge for the development of a hybrid model for construction related systems. Most proposed hybrid architectures have been developed by software industry researchers trying to deal with existing issues in the software industry (e.g., Zeigler et al. 2000; Borshchev et al. 2002; Choi et al. 2006). However, there are fundamental differences between construction-related works and software-related works. While the software industry is considered to be a labor-consuming industry, the construction industry combines industrial equipment and labor. Furthermore, environmental,

safety, material, and equipment issues are some of the most significant factors involved in construction-related jobs (and effectiveness) that do not have a comparable level of importance in the software industry.

If one considers these fundamental differences between the software and construction industries, it is reasonable to expect more hybrid interactions in construction-based systems. Current data communication architectures propose using specified input and output ports that can communicate with specified output and input ports in other parts of the model. While this approach may be effective for software development projects because of their sequential nature and the limited number of interactions between context and operation variables, as illustrated by the research of Choi et al. (2006), it creates increasingly complicated communication combinations when dealing with larger scale and more interrelated systems, as is expected in construction related systems. For example, with n different interacting components in a system, the minimum number of required communication channels, achieved through a sequentially interacting system, is $(n-1)$. However, the maximum required communication channels, as occurring in a fully interrelated hybrid system, will go up to $n*(n-1)$. As a result, even in two systems with the same number of interacting components, the number of required communication ports may be exponentially different.

2.3. Hybrid Modeling Framework

To help hybrid model developers during the conceptual design phase of model development, a hybrid model framework is proposed here. This modeling framework aims to assist hybrid model developers in establishing a confined scope for their work and an outline for the prospective hybrid model. Two main concepts are introduced in this modeling framework: (1) the possible types of basic hybrid structures; and (2) the different types of interacting interfaces, or interface variables. Additional explanation of these concepts follows.

2.3.1. Basic Hybrid Structures

The term “hybrid structure” refers to any arrangement of different interacting SD and DES modeling parts that consists of at least one SD and one DES model. I call a hybrid structure a “basic hybrid structure” if there is only one SD model and one DES model participating in the structure. Before developing a hybrid simulation model, the model developer should have an informed understanding of the hybrid system. That is, the developer should be aware of the different possible types of basic hybrid structures and the interactions among them. To identify the possible basic hybrid structures within complex construction systems, different construction systems, such as steel construction, pipeline construction, and spool construction, have been investigated. Consequently, three major basic hybrid structure types have been identified: (1) SD dominant, (2) DES dominant, and (3)

parallel SD-DES structures. These can all be used to capture the different components of hybrid systems.

Therefore, a hybrid SD-DES model can be presented as a combination of these three basic structures, and hybrid model developers will be able to capture every hybrid interaction inside a system by mapping it to one of the introduced SD and DES interacting basic hybrid structures. The development of each of the recognized SD and DES parts will be pursued via the methods established for SD and DES modeling tools.

1) SD Dominant Hybrid Modeling

Consider a system in which interacting components at the top level of the system form a feedback loop, while several effective variables are internally affected by a set of sequential interactions at the operation level. In SD dominant hybrid modeling, the top level feedback interactions are modeled through SD, while DES is employed to simulate the sequential interactions and track the values of operation based variables. The direction of the interactions in an SD dominant model will be from the DES part to the SD part while the results of calculations within the DES part will be used in the SD part.

Figure 2-1 depicts the dynamics of the capital level of a fabricating company at the top, modeled by an SD model, which has a supportive DES model of production beneath it. The plus sign (+) on the arrows at the SD part of the model indicates that two variables are directly related (i.e. an increase in the first

variable results in an increase in the second). Use of the minus sign (-) on the arrow resembles the inverse relation between the variables (i.e. increase of the first variable causes decrease in the second). Arrows on the DES part of the figure show the direction of the flow of material.

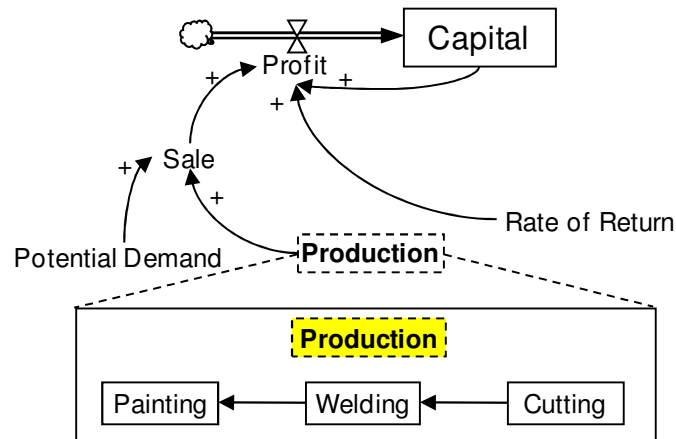


Figure 2-1. SD-dominant hybrid structure of capital level in a fabricating company

According to Figure 2-1, changes in the capital level during the current period are based on the current net profit flow, which is a function of the sale and the rate of return during the current period, and the total capital during the last period. The annual sales depend on the current Potential Demand and Production. The DES model here is for more accurately estimating the amount of Production over time by following the fabrication operations. I place the Production in a dashed rectangle with a bold font in order to distinguish Production as the contact point between the DES and SD parts.

2) DES Dominant Hybrid Modeling

DES dominant hybrid models are commonly used for system structures in which there are sequential interactions between different system components at the top level, while the values of several effective system variables are changed through the feedback interactions between other system components at the context level. When modeling this system with a DES structure, context level variables are generally assumed to be constant during the simulation runs. However, when using a DES dominant hybrid structure, the top level of the system is modeled through DES, and the SD approach is applied to track the values of the context-level variables during simulation to increase the model's accuracy. The direction of interactions in a DES dominant model will be from the SD part to the DES part since the results of the calculations inside the SD part will be used in the DES part.

Figure 2-2 shows a basic schematic of a DES dominant hybrid model inside a fabrication shop in which the welding operation duration is set by the fatigue feedback loop presented in the causal loop diagram. DES is used to simulate the fabrication shop operation interactions, while SD is the tool selected to capture the effects of the feedback loop between worker fatigue level and welding duration. It should be noted that the same basic DES dominant hybrid model can be used at the cutting and painting stations of the shop.

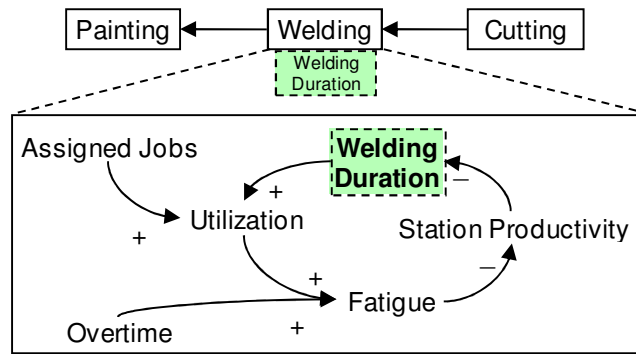


Figure 2-2. DES-dominant hybrid structure of a fabrication shop

3) *Parallel Hybrid SD-DES Modeling*

In many systems, there are mutual effects between sequentially interacting components and the interacting components of the feedback loop. Generally, to model such systems, separate models, either DES or SD, might be selected. In such cases, the interacting variables between sequential and feedback components are assumed to be constant or pre-estimated values. However, when using a parallel SD-DES basic hybrid structure, model accuracy will be improved by the addition of the capacity to track interacting variables during the entire life cycle of a system. For example, the evolved hybrid structure can be used to perform a hybrid simulation of a fabrication shop's operations and labor employment (Figure 2-3). While SD is used to model the company's labor employment process, DES modeling is utilized to simulate the fabrication shop operations. The number of laborers set during the SD process determines the working resources in the DES structure. Furthermore, the schedule delay variable, which is determined by the DES components, plays an integral role in decision-making regarding labor employment. Unlike the two previously mentioned basic hybrid interactions, this

type of basic hybrid structure has bidirectional interactions (i.e. from the SD part to the DES part as well as from the DES part to the SD part). Dashed arrows have been used to demonstrate the contact points and directions of the hybrid interactions for this type of hybrid structure.

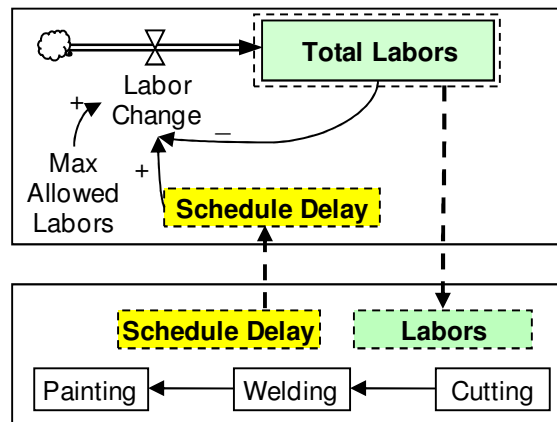


Figure 2-3. Parallel SD and DES to model fabrication operations and labour employment

4) Guidelines to Develop the Basic Hybrid Structures

Generally, hybrid systems can be modeled in different ways and with different combinations of the basic model structures to meet various purposes. However, I will introduce several modeling guidelines that model developers can follow during model development to facilitate the hybrid model design process.

After recognizing the basic hybrid structures of the model, a critical step is to determine the modeling parts that will stay at the top-center level of the entire hybrid model. A Top-Center Level modeling Part(s) (TCLP) is a part that participates as the dominant part in basic hybrid interactions. It may also have several parallel hybrid interactions, but it does not participate in basic hybrid

interactions in which another part of the system is dominant. Sometimes, prospective hybrid SD-DES models improve upon, or act as substitutes for, previously developed SD or DES models. Thus, in these cases, the TCLP will usually be the previously developed SD or DES model. After determining the TCLP of the proposed hybrid model, it is also possible to drive new rounds to recognize new basic hybrid structures, determine new TCLPs, and link newly recognized hybrid structures together to come up with a more comprehensive hybrid model.

2.3.2. Different Forms of Interacting Variables

With a closer look at the interactions among different feedback and operation components of hybrid systems, the system variables, namely interface variables, can be identified. These are the main contact points (between the SD and DES parts) during the hybrid interactions. For example, when changes in interface variables in one part depend on changes in interface variables in another part of the system, or when feedback and operational components in the system share an interface variable, these two interacting structures affect each other during the system's life cycle.

When model developers use non-hybrid modeling techniques, they are usually forced to simplify the interactions among interface variables as constants or as a series of roughly estimated values during the system's life cycle. However, in hybrid models such as hybrid SD-DES models, the values of the interface

variables are calculated at interaction initiator parts and monitored by the interaction receiver parts to set their own interface variables. Thus, variables that have initially been assumed to be static or pre-defined in non-hybrid models can be defined as dynamically changing interface variables, which may change continuously or discretely in the hybrid models.

Furthermore, interface variables are recognizable at the contact points of each (previously identified) basic hybrid structure. However, it must be taken into account that the ways in which different interface variables interact with each other can vary. Thus, a comprehensive understanding of the possible types of interaction, and their expected behaviors, can assist model developers in predicting the appropriate interaction channels for the model.

According to the changing behavior of each system variable, different forms of interaction can occur among interface variables. In 1997, Pritsker et al. (1997) classified all of the possible interaction combinations (i.e. the fundamental interactions) between two continuously changing and discretely changing variables into three main process types: (1) a discrete change in a variable may cause a discrete change in other continuous variables; (2) a continuous change in a variable, by reaching a threshold, may cause a discrete change in interacting variables; and (3) a discrete change in a variable may change the functional description of a continuously changing variable.

Interface variables in hybrid SD-DES modeling are derived from the previously determined static values of the SD or DES components. However, shifting from static values to dynamically changing variables does not necessitate that the interacting variables be derived from the differently changing modes (i.e. continuous and discrete). Thus, in addition to the three previously mentioned variable interaction forms (Pristker et al. 1997), I also used two more forms of interface variable interaction: (4) when both interacting interface variables are continuously changing variables, and continuous changes in one variable causes continuous changes in its related variable; and (5) when two discrete variables interact and the changes in one variable cause discrete changes at the other. The results of this variable classification for five different forms of interface variables will be used in the proposed model architecture in Section 2.4.

2.4. Proposed Hybrid Architecture

The term “hybrid modeling architecture” refers to the set of methods and tools which help hybrid model developers during the detailed design and implementation phases of hybrid model development. A hybrid architecture, which addresses the challenges involved in hybrid modeling (see Section 2.2), is proposed in this section. This architecture involves two main processes: (1) driving a time advancing assessment, and (2) developing a proper simulation architecture while considering data communication issues. These processes are discussed below.

2.4.1. Time Advancing Assessment

Time advancing may become an issue when the updating rate of an interaction initiator variable in the SD part of the model becomes significantly faster than the normal changing rate of the interaction receiver variable in the DES part. As a result, after recognizing the different interface variables, a comparison should be conducted between the expected updating rates of interface variables on both sides of every interaction. In cases in which the interaction initiator variables of the SD model have a significantly faster updating rate, or when the SD time step is shorter than the normal updating rate for the interaction receiver variable in the DES model, the hybrid model may be overloaded with an enormous number of calculations by repetitively rescheduling previously scheduled events in the FEL, which subsequently slows down the simulation runs.

For example, a change in work station utilization (or a busy portion of a station's working time) causes change in the fatigue level of the welding station's working crews; in turn, this continuously affects the station's productivity and correspondingly impacts welding duration (Figure 2-2). Using a conventional DES model to simulate the fabrication shop, normally the finish time of each welding operation is scheduled once, as soon as each welding operation commences. Based on the type of welding operation, the finish time scheduling occurs every several minutes or hours. However, with the effect of worker fatigue on the station's productivity, while the fatigue level constantly changes in

response to continuous changes in station utilization level, the finish time of each operation keeps changing and requires numerous instances of rescheduling the welding operation finish time in the FEL. This adds a considerable number of extra calculations during the simulation runs.

In the literature, two different approaches can be found that try to reduce the number of interactions and thus decrease the negative effects of the time advancing issue. One of these approaches considers adapting criteria for SD time steps, based on the fourth-order Runge-Kutta method, to adjust the length of time steps according to the chronological rates of change (Fehlberg 1970). Proposed by Venkateswaran et al. (2004), the other approach limits all required data exchanges among different SD and DES parts within the SD-DES hybrid models to the set time intervals.

The adjusted time steps in the first approach are based on the recent trend of the changes in the SD part, which will reduce the number of null interactions (i.e., the interactions that send the same value as the previously sent value). However, if the interacting variables in the SD part are changed at every short time interval, this approach cannot help in improving the simulation time. On the other hand, by adjusting the length of time steps based on the chronological rates of change, there is a possibility that the unexpected fluctuations of the SD variables over a short period of time will be neglected.

In the second approach, the SD and DES model parts work separately according to their regular solving methods, and a time step is set for the data exchanges among the different parts. The set time step in this approach should be big enough to cause no interruptions to the DES parts due to sent interactions from the SD parts. On the other hand, the time step should be small enough to be able to capture all significant changes within the SD model parts. While the system behavior and rate of change in different parts of the system may vary during the system life cycle, by setting a constant time step for hybrid interactions among different model parts, this approach may not efficiently capture all hybrid interactions during the system life cycle.

To address this issue, this research introduces a concept called the Meaningful Level of Changes (MLC), which prevents the overloading of hybrid models caused by excessive calculations. This concept proposes setting a meaningful level of changes for interface variables on the SD parts which may cause time advancing issues. The MLC prevents the interface variable from initiating hybrid interactions if the magnitude of change is assumed to be trivial (i.e. less than the defined MLC). For example, if the last reported productivity level is 90% and the MLC for productivity is 1%, the productivity is reported if the value of productivity crosses 89% or 91%.

To be able to set an appropriate value for the MLC, first, a range of MLC values for the variables with the potential time advancing issues is estimated based on

the accepted level of inaccuracy provided by the construction manager. In the second step, the effects of different chosen values for the MLC on the final simulation results are investigated through a sensitivity analysis. This can be done after model development by observing the achieved results and the occurred inaccuracies for different set MLC values. The proper MLC values for the interface variables can be determined by the users of the hybrid model and based on their accepted level of accuracy for the final results.

The MLC concept introduced here plays the same role as the set thresholds in quantization based filtering in distributed discrete event simulation discussed by Zeigler et al. (2002). Zeigler et al. (2002) proposed utilizing the quantization approach to reduce the number of interactions among interacting variables in distributed DES by limiting them to the situation in which their values cross the preset thresholds. However, the MLC concept should reduce the number of interactions in hybrid SD-DES simulation, rather than DES, and will reduce the number of interactions from the SD parts to the DES parts. Applying the MLC concept in hybrid models will minimize the number of interactions while the model continues to be updated, reporting major and consequential changes. Different aspects of the MLC have been investigated and tested in detail; the result of this part of the research was published in a form of conference paper in the 2009 Winter Simulation Conference (Alvanchi et al. 2009b) and is presented in Appendix A.

2.4.2. Communication Architecture

In this study, the communication architecture of the proposed hybrid model is based on the High Level Architecture (HLA). The HLA is a framework for performing distributed simulation modeling developed in the 1990s by the United States Department of Defense (DoD) to simulate military systems (Kuhl et al. 1999). There are three main elements in an HLA-based simulation program: (1) federates; (2) runtime infrastructure (RTI); and (3) object model template (OMT). An HLA-based computer simulation program, called a federation, includes these three elements to simulate the system of interest in a distributed manner. A federate is an independently implemented and run simulation program that represents some parts of the system and interacts with other federates inside the federation. The RTI provides the data communication among different federates, and the OMT defines the structure and formation of the shared data inside the federation (for further information, see [Kuhl et al. 1999]).

Every recognized SD and DES sub-model in a conceptual model, regardless of its level of hierarchy, may be implemented as a federate in a hybrid simulation federation (Figure 2-4). However, according to the level of complexity inside each identified sub-model, these sub-models can be implemented in more than one federate. It is also possible that several sub-models employing a similar modeling approach (i.e. SD or DES) can be merged into one federate while they follow the same method of implementation. The shared data among different

federates are reflected in the OMT, while the recognized interface variables constitute the major part of it. Federates which contain interaction initiator variables are registered in the OMT as the data publisher for those pieces of data; federates that are affected by initiated interactions are registered as subscribers to those variables to be informed by every change made to those variables.

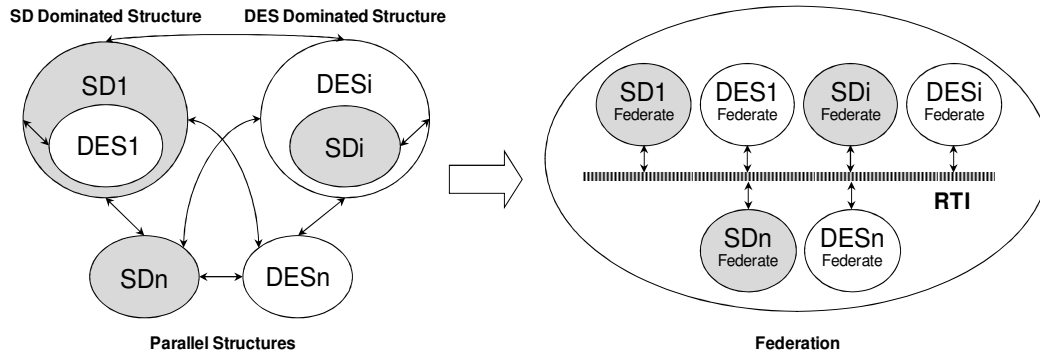


Figure 2-4. Turning the conceptual model to HLA based design

Furthermore, the HLA-based approach to hybrid modeling development eliminates the need for dedicated data import-export communication channels. In addition, each federate simply subscribes to the interacting variables from other parts of the system and publishes these variables, while the RTI handles all of the issues related to additionally required communication processes.

Many hybrid systems are complex systems and their development phases are labor-intensive and require different sorts of expertise. Also, their simulation runs need huge amounts of calculations and a long time on a single computer. The HLA supports distributed model development by groups of experts working separately, thus decreasing the total model development time, as well as supporting distributed simulation on multiple computers, thereby dividing

simulation time among various computers. This capability of HLA-based development also enables hybrid model developers to develop their federation in different stages and time frames. Accordingly, a federation can begin with a limited number of federates and gradually grow by adding new federates as they are implemented.

In addition, by using an HLA based architecture hybrid simulation, developers will be capable of adding prevalent construction supporting services, such as 2D/3D visualization capabilities (Rekapalli and Martinez 2007; Rekapalli et al. 2009), automated material and equipment tracking capabilities (Chi et al. 2009; Azimi et al. 2009; Skibniewski and Jang 2009), and construction cost and schedule optimization (Adeli and Karim 1997; Karim and Adeli 1999) to hybrid simulation as supporting federates for the federation. The distributed model development capability of the HLA framework will also facilitate implementation of future changes in the methods and technologies utilized within specific areas of the construction industry by just adjusting those specific areas in the affected federates.

Every SD, DES, or supporting federate in an HLA federation can use a different development package; however, the selected package should be able to communicate with the RTI, and it should be flexible enough to include all the proposed steps in the hybrid architecture. The detailed methods and requirements

for designing and implementing an HLA-based simulation program have been explained by Kuhl et al. (1999).

2.5. Experimental Model

An experimental hybrid model has been developed for a structural steel fabrication shop. The main purpose of this model is to investigate whether the proposed hybrid framework and architecture can address the aforementioned hybrid modeling challenges (Section 2.2) when applied to a real scale construction related system. Therefore, the assessment of the experimental model is mainly focused on testing the performance and functionality of the model rather than its analysis capabilities. Further elaboration on the experimental case, model development stages, and model verification follows in the remainder of this section.

2.5.1. Case Description

The fabrication shop observed for this study fabricates approximately 50,000 tons of structural steel annually, and in 2008 it completed 30 different projects involving 3,000 divisions and more than 100,000 pieces. The five different operation types executed in the fabrication shop are cutting, fitting, welding, inspection, and painting. In the case study shop, all of the fabricated pieces sequentially pass the cutting, fitting, welding, and inspection operations, while only approximately 40% of the pieces require the final fabrication operation:

painting. This fabrication shop contains a main cutting shop which serves all kinds of pieces; three fitting/welding shops, each of which have been designed to serve one of the three types of pieces (i.e. heavy, average, and light); one inspection station; and one painting station. There are also three areas for storing in-progress pieces and internal movers (i.e. rail-based carts and cranes) that handle all the required movements inside the fabrication shop (see Figure 2-5).

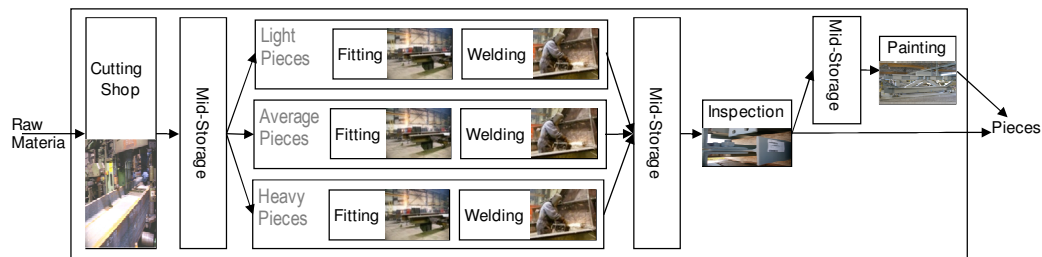


Figure 2-5. Shop structure and material flow of the experimental case

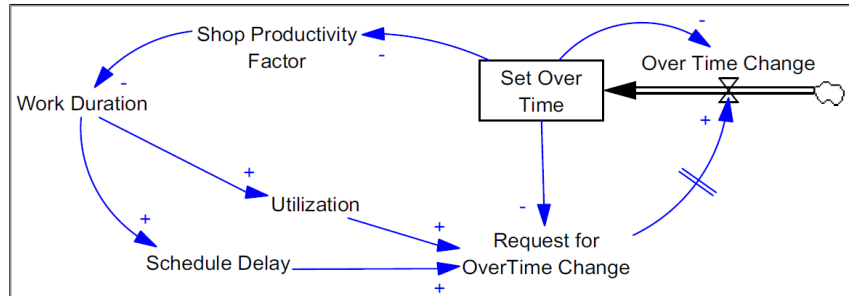
2.5.2. Fabrication Shop Hybrid Model Development

The fabrication shop hybrid model development has been performed based on the modeling framework and modeling architecture introduced in Sections 3 and 4. In the rest of this section, different steps of the fabrication shop model development have been explained. During the explanation of the model development process, several terms related to the fabrication shop are utilized.

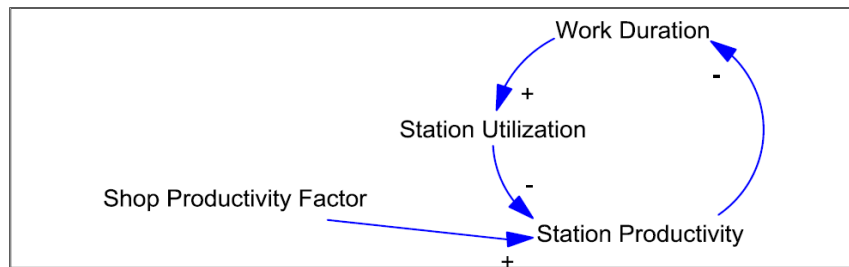
1) Conceptual Design of Fabrication Shop

The operation parts of the fabrication shop, as mentioned in Section 2.5.1, are simulated by the DES model. Additionally, several managerial and context-level feedback loops have been considered within the fabrication shop and are modeled

through SD. Furthermore, hybrid interactions among the feedback loops and operation parts of the fabrication shop have been captured through basic hybrid structures. Figure 2-6 demonstrates two examples of the utilized feedback loops for the fabrication shop, and Figure 2-7 shows their related basic hybrid structures while feedback loops interact with the operation part of the fabrication shop.

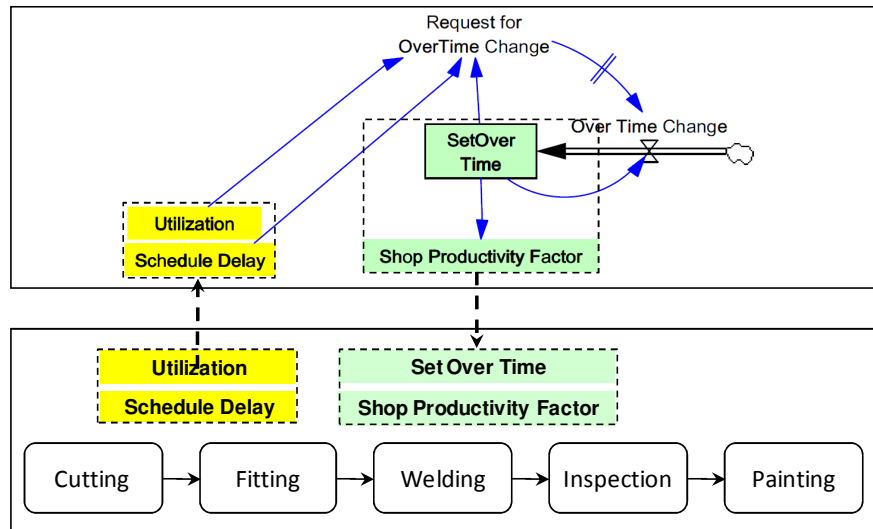


a) Overtime Loop

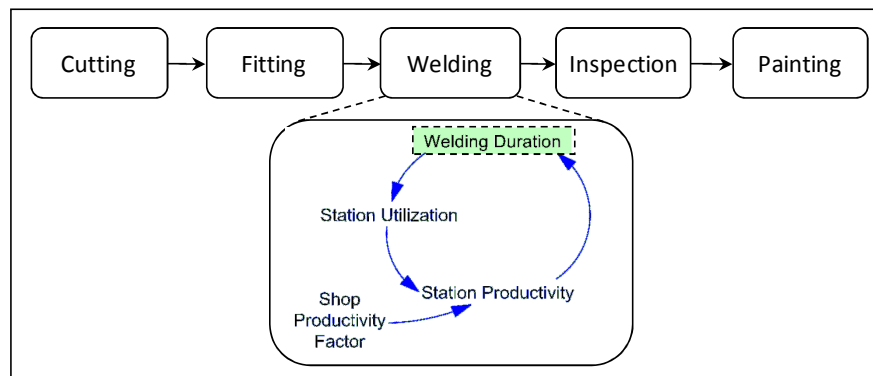


b) Station Workload Pressure

Figure 2-6. Sample managerial and context-level feedback loops



a) Parallel interaction between overtime loop and shop operation



b) DES dominated interaction of station workload pressure

Figure 2-7. Sample basic hybrid interactions used in the model. Bold fonts are used for interface variables.

In Figures 2-6 and 2-7, schedule delay shows the total number of days that different divisions of a project are behind the schedule; utilization level shows the portion of time that a station has been busy with serving the assigned jobs; and overtime is the additional working hours that fabrication shop managers have set for the fabrication shop to compensate for project delays.

In model (a) (Figure 2-7), the overtime setting policy exhibits a parallel interaction with the fabrication shop operation. According to the schedule delay and utilization level, which are calculated and sent by the DES part of the basic hybrid structure, shop managers decide whether to increase or decrease the overtime. The set overtime affects the operating hours of the fabrication shop. Additionally, the fatigue level that results from the set overtime will have its influence on the productivity factor of the fabrication shop (see [Sterman, 2000, p. 581] for more information on effects of set overtime on productivity). Model (b) (Figure 2-7) is a DES-dominant basic hybrid model which contains autonomously set feedback within every single station. This model captures the direct effect of the utilization level on station productivity at every single station.

Among the different basic hybrid structures utilized, the DES model of the fabrication shop operations in its entire hybrid interactions participated in parallel or as a dominant part to all other related parts. Thus, it was considered to be the Top-Center Level modeling Part (TCLP) of the experimental case. Additionally, there were no SD dominant basic structures recognized for the experimental case. The main reason is because the DES modeling part of the fabrication shop was put at the highest level of interest for the case study, and because all the other SD modeling part were involved as supplements to it.

2) *Fabrication Shop Time Advancing Assessment*

Table 2-1 presents a brief assessment of the interacting variables related to the basic hybrid structures (a) and (b) presented in Figure 2-7. According to the data presented, there are a total of six contact points of hybrid interactions: three initiated from the SD model, and three initiated from the DES model. To determine potential challenges regarding the time advancing issue, an assessment of the interacting variables identified was performed.

Table 2-1. Brief assessment of the interacting variables of basic hybrid models

Initiator					Receiver				
Variable	Model	Variable Type	Updating Rate	Variable	Model	Variable Type	Updating Rate	Time Issue	
Overtime	SD	Discrete	Daily	Operation Duration	DES	Continuous	Per Minute	No	
Shop Productivity	SD	Continuous	Daily	Request for Overtime Change	SD	Discrete	Daily	No	
Schedule Delay	DES	Discrete	Daily					No	
Shop Max. Utilization	DES	Continuous	Per Second					No	
Station Utilization	DES	Continuous	Per Second	Station Productivity	SD	Continuous	Per Second	No	
<i>Station Productivity</i>	<i>SD</i>	<i>Continuous</i>	<i>Per Second</i>	<i>Operation Duration</i>	<i>DES</i>	<i>Continuous</i>	<i>Per Minute</i>	Yes	

Based on the five forms of interacting variables (see Section 2.3.2), the only critical time advancing issue occurs in model (a), where the station productivity is updated every second based on the station utilization rate and the shop productivity factor. However, duration updates for a station operation are normally performed every several minutes. The hybrid interactions at this point might require sixty extra rescheduling occurrences of the finish time for every

scheduled operation. Following the proposed MLC approach (Section 2.4.1), an MLC of 1% is set for the station productivity.

Interactions at all the other interface variables are not critical; this means that the interaction initiators either have longer updating rates, or they originated from the DES part of the hybrid model. For example, the “Overtime” variable in the SD part of the hybrid model (a) is updated every day and has an effect on “Operation Duration” which is updated every minute. In this case, even if Overtime occurs every day, it will only change one pre-scheduled event out of hundreds of scheduled and occurring events. Thus, although the Operation Duration in the DES component will be impacted, it will have a minimal effect on the total system simulation time.

3) *Fabrication Communication Architecture*

While the estimated number of calculations required for the SD parts was lower than the DES-related calculations, all of the SD-based models have been assigned to one federate. Thus, the hybrid simulation of the fabrication shop federation consists of one SD and one DES federate. There are also two supporting federates recognized for the hybrid model: (1) a federate for communicating with the company’s main data server, and (2) a calendar federate to regulate the current time and date within the federation. Figure 2-8 shows the top level architecture for the federation and how different federates (i.e. SD, DES, Calendar and Data

Management) communicate only to the RTI which works as a bridge for all required communications.

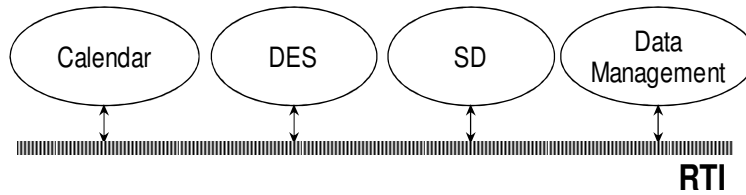


Figure 2-8. Top-level architecture of the hybrid model

4) *Fabrication Implementation*

The Construction Synthetic Environment (COSYE) (AbouRizk and Hague 2009), an HLA-based framework developed especially for construction engineering and management researchers at the University of Alberta, has been employed for the hybrid modeling. For more flexibility in implementing the proposed architecture and because of its compatibility with COSYE, Microsoft Visual Studio (VS) 2008 was used as a general programming tool to implement the hybrid model. SD components were characterized by discrete difference equations (i.e. a set of equations connecting differences between consecutive values of functions) and were coded directly in VS. However, to implement the DES component of the model, the DES-related Dynamic Link Library (DLL) files provided by Symphony.NET 3.5 (<http://irc.construction.ualberta.ca/Simphony35/>), developed by construction engineering and management researchers at the U of A, have been employed. Furthermore, MS Access 2007 was used as an interim database to

handle the required data link between the simulation program and the collaborative company's SQL server database.

The four federates and the Object Model Template (OMT) were implemented as different projects of a VS solution in order to be able to distribute them among different computers. The OMT contains the object classes and the attributes (i.e., interacting variables) which are shared among multiple federates. Table 2-2 presents the OMT that has been used in the steel construction federation. According to HLA terminology, when a federate is responsible for updates to an attribute, the federate is Publishing that attribute (represented by P in the table); when a federate uses the updated values of an attribute, the federate is Subscribing to that attribute (represented by S in the table). In addition to shared object classes and their attributes, the OMT should be aware of the Publishing and Subscribing federates. Every attribute should have at least one Publishing federate and one Subscribing federate to be eligible for OMT inclusion. The shared OMT contains four object classes, ShopProductivity, StationProductivity, Calendar, and PieceEntity, which handle all required data exchanges inside the hybrid federation. ShopProductivity and StationProductivity objects contain the recognized interface variables (Table 2-1), while the main hybrid interacting variables in the Calendar and PieceEntity objects contain supporting variables to enable communication among the supporting federates with the SD and DES federates. Because the supporting federates, i.e., Data Management and Calendar,

provide services to the other federates they will do more Publishing than Subscribing; for the main federates, i.e., DES, and SD, I expect more Subscribing.

Table 2-2. Steel Construction Federation Object Model Template

Object Class	Attribute	Type	Federates*				
			DES	SD	Data Management	Calendar	
Calendar	StartDate	Date	S		P		
	CurrentDate	Date	S	S	P/S	P/S	
	CurrentShiftHours	Integer	S	S	P	P	
	CurrentShiftType	Integer	S	S	P	P	
	DayNo	Integer	S	S	S	P	
	DesireOvertime	Double		P		S	
	MaxOvertime	Double		S		P	
	SetOverTime	Double		S	P	P	
Entity (Piece)	PieceID	Integer	S		P		
	PieceStartDate	Date	S		P		
	Weight	Integer	S		P		
	Cutting Man Hour	Double	P/S		P/S		
	Fitting Man Hour	Double	P/S		P/S		
	Welding Man Hour	Double	P/S		P/S		
	Inspection Man Hour	Double	P/S		P/S		
	Painting Man Hour	Double	P/S		P/S		
	CuttingStart	Date	P/S		P/S		
	CuttingFinish	Date	P/S		P/S		
	FittingStart	Date	P/S		P/S		
	FittingFinish	Date	P/S		P/S		
	WeldingStart	Date	P/S		P/S		
	WeldingFinish	Date	P/S		P/S		
	InspectionStart	Date	P/S		P/S		
	InspectionFinish	Date	P/S		P/S		
	PaintingStart	Date	P/S		P/S		
	PaintingFinish	Date	P/S		P/S		
	Shop Productivity	DelayRate	Double		S	P	
		TotalDelay	Double	S	S	P	
Accuracy (MLC)		Double	S	P	S		
Station Productivity	ID	Integer	P	S			
	CurOperator	Integer	P/S	P/S			
	MaxOperator	Integer	P	S			
	MinOperator	Integer	P	S			
	Productivity	Double	S	P			
	State	Integer	P	S			
	Utilization	Double	P	S			

* P stands for Publisher of an attribute and S stands for Subscriber to an attribute.

In the federation, the Calendar federate is responsible for advancing the date and determining the working hour arrangements (day shifts, night shifts and

overtime). Users can enter upcoming holidays, and the model will take care of set holidays during the simulation. The Calendar federate publishes the date information to the federation to be used by other federates. Figure 2-9 presents the interface form of the calendar federate. The calendar interface has three main parts. The first part determines the required information for linking to the RTI, which is similar in all federates. The second customizes the working hours, and the third provides a place for setting the holiday schedule.

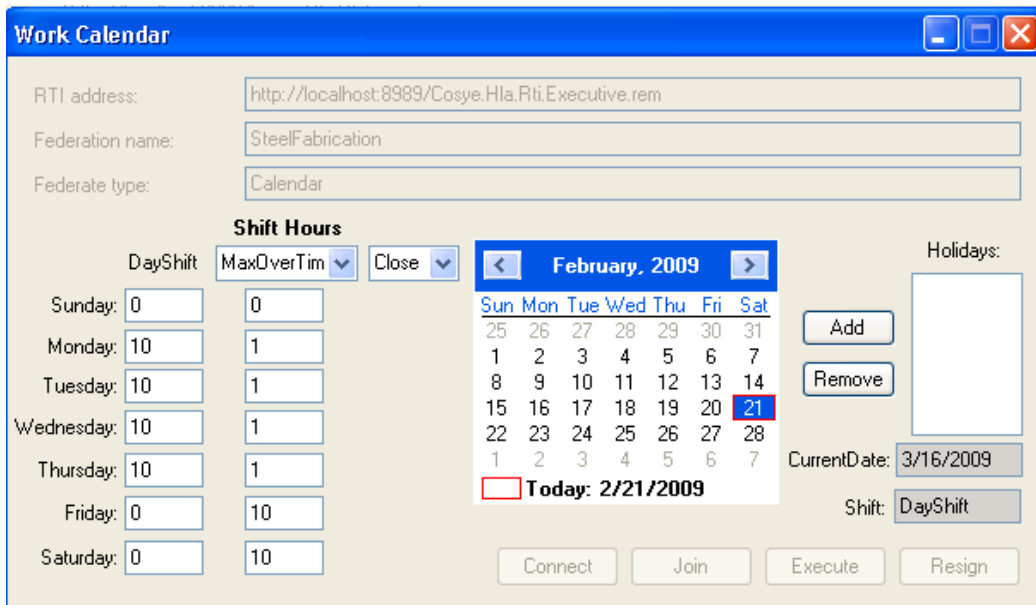


Figure 2-9. Interface of the calendar federate.

The Data Management federate provides database communication services for different federates. This federate retrieves the piece information from the database, directs the piece information to the fabrication shop (i.e., DES and SD federates) and reports the fabrication completion of the pieces to the database. Figure 2-10 presents the interface form of the Data Management federate. The federate interface allows the user to set the simulation start time and the duration

of the shop simulation as the main constraints for running related queries in the database; other inputs to this federate are the importance weights of different fabrication operations. The federate reports some managerial project information through the interface – such as delays, number of completed pieces, and finish time – and writes aggregated reports on project performance to the database.

Activity Percentage of the Work:			
	Mod%	Min%	Max%
Cutting:	5	4	10
Fitting:	40	30	50
Welding:	40	30	50
Inspection:	3	2	5
Painting:	8	5	10

Figure 2-10. Interface of the Data Management federate.

The Discrete Event Simulation (DES) federate captures the operational part of the fabrication process. The fabrication operation starts by sending the fabrication orders and their related materials to the fabrication shop. The DES federate then simulates the flow of materials in the fabrication shop from one station to the other and sends the pieces out when the required set of operations is completed. Figure 2-11 presents the main interface of this federate. Three buttons on the left side of the form open the detailed forms for entering the specifications of stations

(Figure 2-12), mid buffers or storage (Figure 2-13) and movers including cranes and rail cars (Figure 2-14) within the fabrication shop. The pink and green buttons on the middle of the form represent different stations, including cutting, fitting, welding, inspection and painting. The user can set the number of dedicated stations for each operation type, e.g., welding, by selecting the proper number of stations from the drop down combo box on the top of each series of stations. The buttons are green when stations have no job to do and are pink with the number of the piece written on them when they are busy serving pieces. The initial number of in-progress pieces at each station is also listed in the list boxes at the bottom of the form.

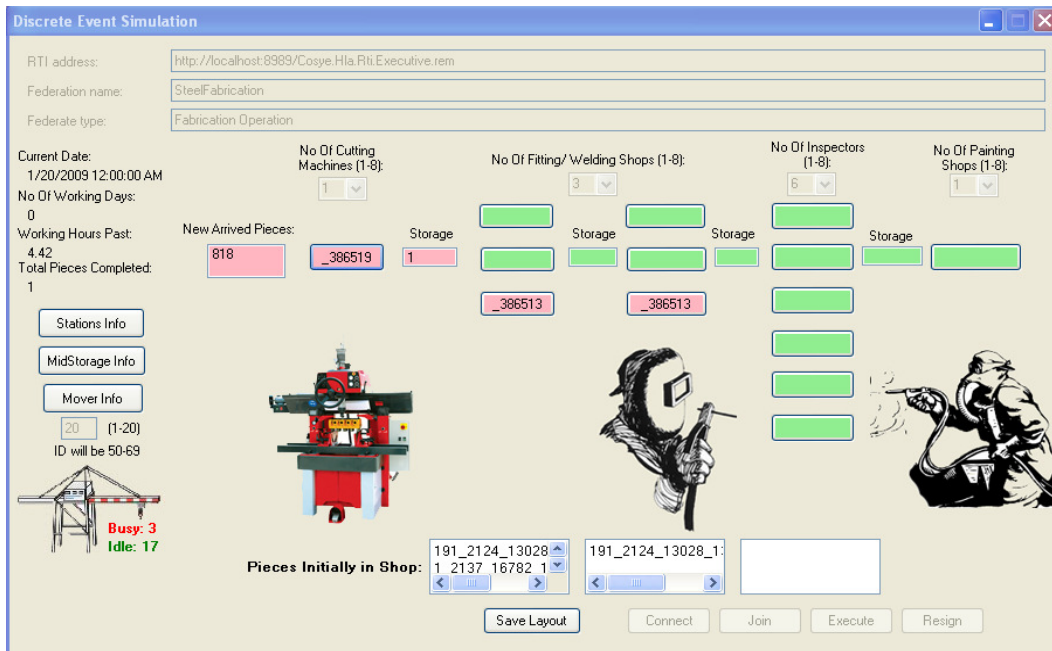


Figure 2-11. Main interface of the DES federate.

The StationInfo dialog box is titled "StationInfo" and contains the following configuration options:

- ID: 1 (dropdown)
- Function: Cutting (text field)
- ImportMidBufferID: 70 (text field)
- OutputMidBufferID: 71 (text field)
- Buffer Capacity: 1 (text field)
- Type: NoType (dropdown)
- Min. Operator: 8 (text field)
- Max. Operator: 10 (text field)
- Cur. Operator: 9 (text field)
- Import: X 1 Y 1 (checkboxes)
- Output: X 1 Y 10 (checkboxes)
- Back linked stations are limited to: (list of stations 1-23, Add/Remove buttons)
- Front linked stations are limited to: (list of stations 1-23, Add/Remove buttons)
- Duration Factor(%):
 - Num. Operator: (dropdown)
 - Dist. Type: (dropdown)
 - Parameters: (text field containing "8.Triangular,10,15,12,5", "9.Triangular,8,14,12", "10.Triangular,8,12,11", Add/Remove buttons)
- Close (button)

Figure 2-12. Station specification form in the DES federate.

The MidBufferInfo dialog box is titled "MidBufferInfo" and contains the following configuration options:

- ID: 71 (dropdown)
- Capacity: 214748364 (text field)
- Unlimited (checkbox)
- Import: X 1 Y 15 (checkboxes)
- Output: X 1 Y 15 (checkboxes)
- Back linked stations: (list of stations 1-23, Add/Remove buttons)
- Front linked stations: (list of stations 1-23, Add/Remove buttons)
- Close (button)

Figure 2-13. Mid-buffers or storage specification form in the DES federate.

The MoverInfo dialog box is titled "MoverInfo" and contains the following configuration options:

- ID: 50 (dropdown)
- Waiting List Length: 1000000 (text field)
- LoadedSpeed (m/Min/10Tons): 100 (text field)
- UnLoadedSpeed (m/Min): 100 (text field)
- Dist. Type: (dropdown)
- Parameters: (text field)
- Loading Time (Min/10Tons): Constant (dropdown), 0.2 (text field), 0 (text field), 0 (text field)
- UnLoading Time (Min/10Tons): Constant (dropdown), 0.2 (text field), 0 (text field), 0 (text field)
- Loading Zones: (list of station coordinates, Add/Remove buttons)
- Unloading Zones: (list of station coordinates, Add/Remove buttons)
- Close (button)

Figure 2-14. Movers specification form in the DES federate.

Finally the System Dynamics (SD) federate captures the effects of non-operational mechanisms on the fabrication shop's productivity. The non-operational mechanisms which I included in the SD model are: fatigue, skill level, hiring and firing, and the work balance (detailed supporting equations used in the program for the SD model are presented in Section B.2 of Appendix B). Figure 2-15 presents the interface of this federate. The user can enter the marginal inaccuracy that is acceptable for calculating and reporting the productivity rate through different feedback loops in the model. Different types of feedback loops have been put in the tabular forms on the main form and the user can browse through them during the simulation run and see the changes in the non-operational mechanisms of the fabrication shop.

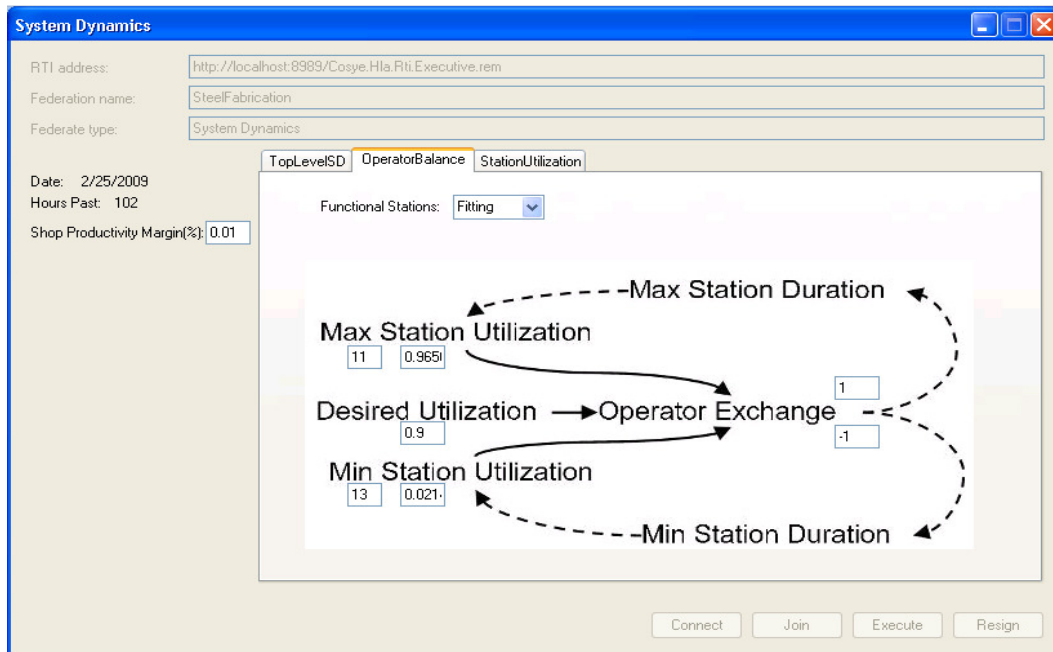


Figure 2-15. Interface of the SD federate.

The Visual Basic codes used for developing different federates are presented in Appendix B.

2.5.3. Performance Test

The proposed modeling framework was efficiently employed to design different parts of the steel fabrication hybrid model, whereas the model implementation was subsequently conducted based on the proposed architecture. One of the most significant achievements of the proposed architecture was the introduction of the interface variables to the entire federation by easily implementing an OMT project as the container of all the interacting variables inside the HLA-based architecture.

To test the usefulness of the hybrid architecture, two types of quantitative evaluation have been conducted: (1) investigating the effects of the MLC concept in the achieved simulation results, and (2) comparing the effects of distributed simulation (based on the HLA framework) and MLC concept on the simulation time of the fabrication shop model. In each evaluation, two types of models were compared: (1) the proposed hybrid modeling, which utilizes the MLC concept, and (2) base hybrid modeling, in which all variable updates are immediately reflected in the interaction receiver components of the model. Other aspects of the modeling approach stay the same.

The MLC concept should improve the total simulation time without affecting the final achieved results of the simulation compared to the base model. Thus, the first evaluative test was conducted to compare the achieved results of the two (i.e. proposed and base) models. The hybrid simulation models are developed to examine the effects of the latest changes or different scenarios in the modeled system and are meant to be regularly used during the job. Hence, any enhancements that keep their simulation time within a reasonable range (i.e. no more than several hours) will make the hybrid models more applicable to real cases. The second evaluative test was conducted to assess how introducing the distributed simulation and MLC concepts would affect the reduction of the hybrid models' simulation time. Simulation time of hybrid simulation models can significantly affect the applicability of the hybrid SD-DES models in the real construction decision making problems. The complexity involved in such hybrid simulation models requires longer simulation runs compared to the conventional models; multiple decision alternatives usually should be run and tested to be able to improve the final decision; construction managers are usually under time pressure to decide and make their final decisions. In this perspective reducing the simulation time in many hybrid modeling applications can affect the construction managers' choice on using or not using a hybrid modeling approach in their decision making process.

All of the simulation runs were based on three months of material feed to the fabrication shop from January 20, 2009 to April 20, 2009. During this period, the

steel materials required for fabricating (approximately 20,000 pieces) were fed through the steel fabrication shop. The simulation runs were completed when all of the pieces were fabricated. Sample input data-tables, containing piece data, and output data-tables, used for storing the model reports, are presented in Appendix B.

1) MLC Verification

Table 2-3 presents the duration required to complete the total assigned fabrication jobs, which is derived from five different runs of the two developed hybrid models. The difference between duration time achieved in the proposed and base hybrid models is 0.41%, which is less than one percent of the accepted level of inaccuracy (i.e. the established value for the MLC). Normally such a level of inaccuracy within the construction industry is considered minimal and does not affect the system analysis and final decision-making processes in construction.

Table. 2-3. Comparison of the results of the proposed and base hybrid models

	Duration (Day)		
	Average	Standard Deviation	Avg. Difference
Proposed Model	144.2	2.9	0.41%
Base Model	144.8	3.0	

As a result, this test verifies the MLC concept by affirming the trivial difference between the results of the base hybrid model and the proposed hybrid model. However, it should be noted that this test was not conducted to evaluate the

accuracy of the two models, but was intended to simply depict how adopting the MLC concept can affect the expected results of hybrid models.

2) *Simulation Time*

To assess the effects of applying the MLC to the hybrid models, both the proposed and base hybrid models were run in a centralized manner using a single computer. Additionally, the models were run in a distributed manner, using two computers, to evaluate the concurrent effects of the COSYE framework and the MLC concept on simulation time. The computers that were used each have 3.06 GHz CPU and 1 GB RAM, and the speed of the main local network was 1 gigabit per second (Gbps). Table 2-4 shows the achieved average simulation time for different simulation runs, which is calculated based on the five different observations for each case.

Table. 2-4. Simulation time for different developed models

Model	Hybrid Interactions			Computer Employed	Average Simulation Time (minute)
	SD to DES	DES to SD	Total		
Proposed hybrid model run	19,500	42,000	61,000	One	126
				Two	127
Base hybrid model run	35,000	58,000	93,000	One	281
				Two	323

As was expected, by using the MLC, the number of hybrid interactions significantly decreased from the base hybrid model to the proposed hybrid model. As presented in Table 2-4, although MLC's direct effect is on the interaction reduction from the SD part of the model to the DES part, the reduction has happened at both directions and approximately at the same level (i.e. close to

15,000 interactions at each side). This shows while MLC eliminates insignificant interactions from the SD part of the model, the consequent changes within the DES parts which accordingly would cause hybrid interactions originated from the DES part also have been eliminated.

As a result, the proposed hybrid model had a shorter simulation time, 55% shorter than that of the base hybrid model. Furthermore, when comparing the centralized and distributed simulation of each type of simulation model, it can be seen that distributing the simulation run causes a small increase in the proposed hybrid model and a 15% increase in simulation time in the base model. The main reason for this contradictory result is related to the number of data interactions in the base model, where every hybrid interaction is reported to the RTI to be considered in simulation. Thus, the number of data transmissions through the employed network will be much higher than in the proposed hybrid model simulation. Further, although data communication between computers via a computer network is slower than the communication within a computer, the communication delays involved in base hybrid simulation offset the time saved by distributing the simulation between two computers.

The first distributed tests were held in a 100 Mbps local network that caused a longer simulation time even for the proposed hybrid model. Changing the local network speed to 1 Gbps resulted in an improved simulation time for the proposed hybrid model but not for the base hybrid model (see Table 2-4). These tests

indicated that simulation time is highly sensitive to the way in which different federates are distributed in different computers. Because network speed is a major constraint for distributed simulation speed, better results will be achieved when highly interacting federates are located on one computer. So, to reduce the simulation time in this federation the SD and DES federates, which share more interactions, were run on one computer, and two supporting federates were run on another computer.

2.5.4. Expandability Test of the Model

To investigate the expandability capability of the models, another supporting federate, called the Visualization federate, was added to the federation several months after the development of the first group of federates. The Visualization federate visualizes the progress of the fabrication shop using 3D models of structural steel projects. This federate lists the current steel divisions that are under way in the fabrication shop. The user can select any division to load the related 3D model. The completed pieces in the fabrication shop are found and highlighted inside the 3D model. Different colors have been used to visually illustrate the cost and time performance of the completed pieces (resulting in a 5D model). Figure 2-16 presents the main interface of this federate. In-progress steel divisions are listed in the list box on the left side of the form. The user can select every in-progress or completed division from this list box and push the Change the Division button; the federate will then show the progress and performance

indices of related pieces in the list box on the right. Because the visualization process slows down the simulation, the default option in the visualization federate is set to just update the progress in the text. The user can also select the Show in Tekla radio button on the form and ask the federate to run Tekla, a structural steel 3D detailing package (Tekla Corporation, Finland, <http://www.tekla.com>), and reflect the progress concurrently in the 3D model. The color coded structural steel 3D model in Figure 2-17 shows a snap shot from Tekla during the model run (for further information about the visualization federate please refer to Azimi et al. 2011)

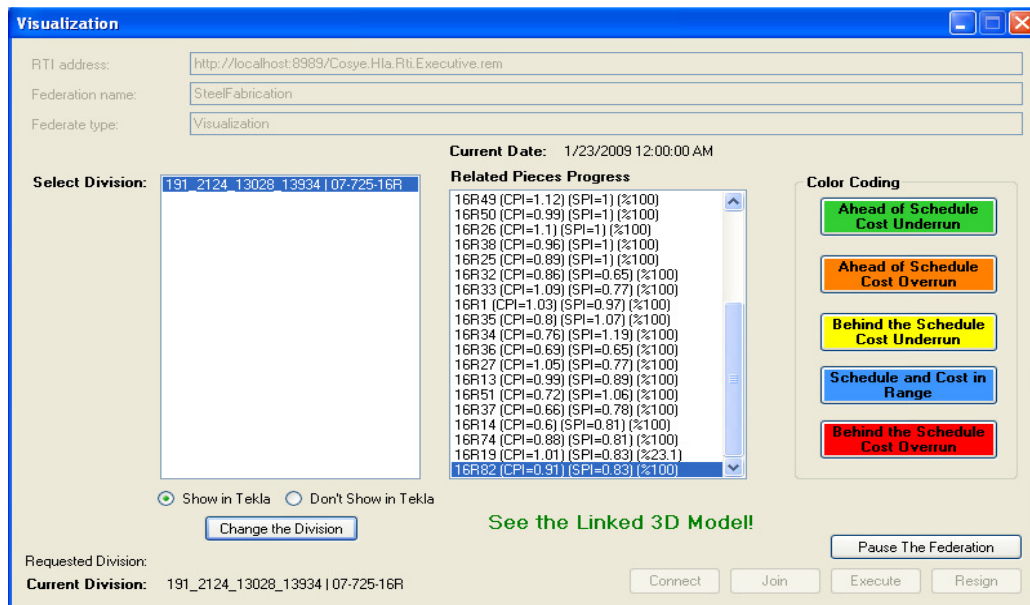


Figure 2-16. Interface of the Visualization federate

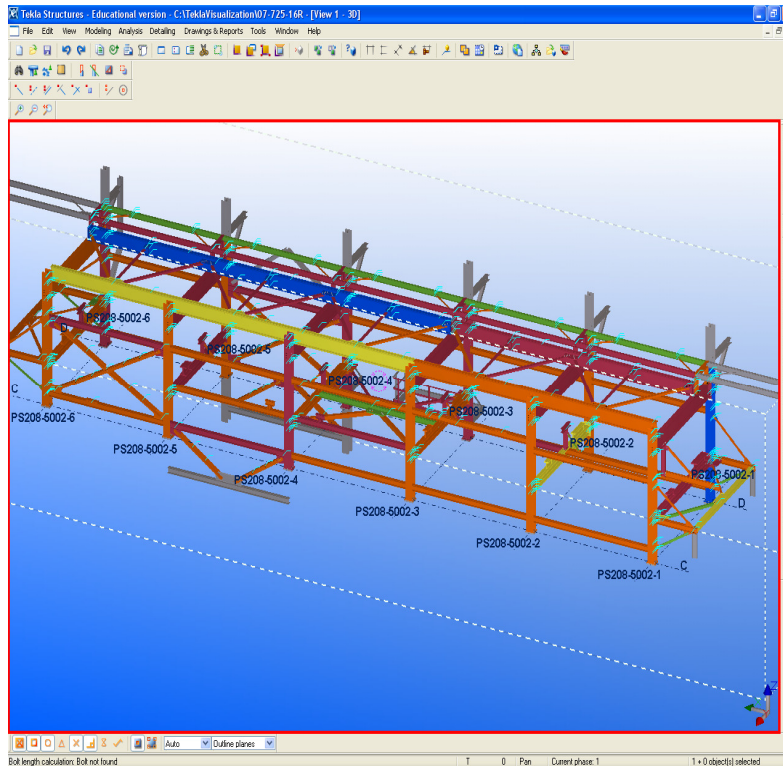


Figure 2-17. A snap shot from Tekla during the model run

To further test expandability of the model, I successfully integrated the steel federation with a process control system based on RFID technology. The RFID tags were linked to the pieces and used for tracking the current location of the piece within the shop floor and updating the most recent operations done on the pieces. These data then were used for setting up the initial condition of the fabrication shop and using the most current data for the simulation (see Azimi et al. 2011 for the detail information). Figure 2-18 presents a schematic view of the way that the simulation federation was integrated with RFID technology.

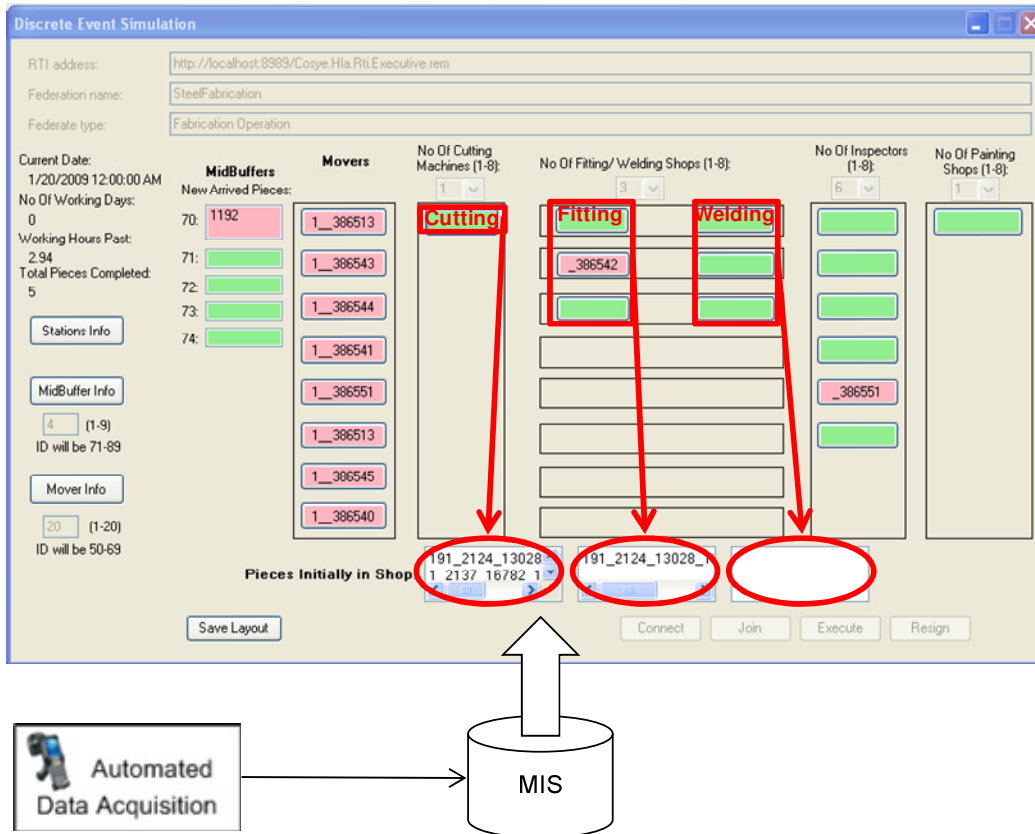


Figure 2-18. A schematic view of integration between the developed federation and RFID technology

2.6. Chapter Conclusion

The hybrid SD-DES modeling approach is a new approach introduced for dynamically capturing both operation and context levels of complex systems. Using such a powerful tool for modeling complex construction systems requires a robust modeling framework and architecture to assist hybrid model developers in the design and implementation stage of the construction hybrid model development. In this part of the research I introduced and validated a new hybrid framework and architecture which addresses challenging issues involved in the

method of model development and presents a step by step guide for construction hybrid model developers.

Introducing different types of the basic hybrid models and the interface variable concept provides a guideline to model developers during the design phase of the hybrid model. The MLC concept was introduced as a response to the computational problem made because of the mutual effects of SD and DES model components to increase the efficiency of the simulation runs. This research also proposes use of the HLA framework to create equally accessible communication channels for all of the model components in order to eliminate the need to create bilateral communication channels between every two interacting variables, and to facilitate model expansion over time.

To test the applicability and functionality of the proposed hybrid framework and architecture for real world construction related systems, a case study was conducted using an actual structural steel fabrication shop. The development of the case study affirmed the appropriateness of the proposed hybrid framework and its related concepts to be used for the conceptual design phase of hybrid model development. The computational results of the evaluation also confirmed that the proposed hybrid model can capture a system as realistically as the base hybrid model can, with significantly faster simulation runs and an easier implementation of its communication channels, which is made possible by adopting the HLA

framework. The use of HLA and its distributed implementation capability showed its benefits during the expansion of the model which was developed.

The implementation of the hybrid model requires significant effort, but it should be considered that the set of tools, methods, and procedures provided by the HLA can facilitate the entire implementation process. In addition, I found implementation of the DES federate to be the most challenging part of the implementation process; it is expected that using an HLA based simulation package, which provides a visual approach to DES modeling, may reduce the duration of implementation for DES federates.

One reason for adopting the hybrid simulation modeling approach is to capture real, complex construction systems in greater detail than traditional modeling tools. The proposed hybrid modeling approach can be used for exploring new hybrid interactions in the construction industry in order to enhance system analysis capabilities for construction projects. These new models are expected to provide new types of analysis for construction managers which were not available prior to development of such models. Two different applications of hybrid SD-DES models within the construction industry are described in the rest of the thesis.

CHAPTER 3. Dynamics of Working Hours in Construction³

3.1. Introduction

Construction project owners generally would like their projects to be finished and operable as quickly as is practical. To achieve this, one typical method is increasing construction working hours, using evening and night shifts as well as overtime. However, this may not be effective: diminished performance during night and overtime work has been addressed in many studies (e.g, Cohen and Muehl 1977; Homer 1985; Vidacek et al. 1986; Rosa et al. 1998; Folkard and Tucker 2003). Research has shown that the adverse effects of overtime or night shifts on work performance can negate any positive effects of additional working hours.

Performance reduction due to fatigue during continuous or prolonged work and the positive effects of a rest allowance (i.e., the length of the rest break divided by the length of the preceding working period) on performance have been studied extensively (e.g, Taylor 1911; Rohmert 1973b; Oglesby et al. 1989; Smit et al. 2004; Helton and Warm 2008). For example, Taylor, in the late 19th century, studied a material handling job in which laborers were loading pig-iron into gondola cars. The job output increased from 12.5 to 47 long tons per day per man when Taylor set 58 percent of their time for rest (Oglesby et al. 1989, p. 245).

³ Parts of this chapter have been accepted for publication in the Journal of Construction Engineering and Management, ASCE.

This is an example of how construction managers can use work and rest period arrangements to maximize productivity.

However, past research into the effects of working hour arrangements on worker performance shares three main issues which may prevent results from being used widely in the construction industry: (1) studies are limited to one or two aspects of work hours (i.e., overtime, time of day, work length, or prolonged working hours); (2) the results are too narrow, involving specific types of work and levels of workers with no instructions for generalizing the results; and (3) most are static, where just one type of work has been assigned to the workers during the study. In contrast, construction jobs are usually project based and the type of work that is assigned to the workers is dynamic.

To address these issues, I have developed a computer simulation model that integrates System Dynamics (SD) and Discrete Event Simulation (DES) to account for the effects of working hours on performance (i.e., a hybrid SD-DES model). The SD model continuously sets the level of performance in construction jobs and captures feedback associated with working hour arrangements; the DES model follows the construction operation details and provides the SD model with operational updates, such as worker status (i.e., busy or idle), type of assigned work and amount of work done. Using this hybrid model, I have completed a series of quantitative assessments on the effects of working hour arrangements on work performance, which may assist management personnel in finding working

hour arrangements resulting in improved performance and cost savings. Furthermore, the hybrid model developed in this chapter is basically used as a validation test for the proposed hybrid framework and architecture in Chapter 2 and demonstrates an instance for the benefits that it can bring to the construction industry.

3.2.Dynamics of the Working Hours

In this section I describe the SD model development, which accounts for the effects of a variety of factors impacting performance, in accordance with the literature. My main focus for these effects is productivity and quality. In this part of the research, I use the productivity ratio to measure productivity. I consider the productivity ratio as 100% for a normal skilled person in regular working conditions, though fluctuations in the productivity ratio can occur due to changes in the worker and job conditions. To measure quality, reliability (Lee et al. 2005) is used, an index which indicates the probability of deficiency occurrence.

3.2.1. Dynamics of Prolonged Working Hours

Performance decrease because of fatigue during a prolonged or a continuous period of working has been attributed to two main causes: (1) decreased muscular strength because of physical work (e.g., Taylor 1911; Rohmert 1973b; Oglesby et al. 1989); and (2) increased strain because of over-capacity mental stress (e.g., Nuechterlein et al. 1983; Matthews et al. 1990; Smit et al. 2004; Helton and

Warm 2008). The recognized dynamics of these two causes have been summarized in Figures 3-1 and 3-2 and are discussed below. Circle arrows labeled “B” and “R” in the figures refer, respectively, to the balancing and reinforcing feedback loops in System Dynamics (SD). In a balancing loop, an increase in a variable later on causes a reduction in that variable through a causal feedback loop; in a reinforcing loop, this increase is followed by a further increase (Sterman 2000).

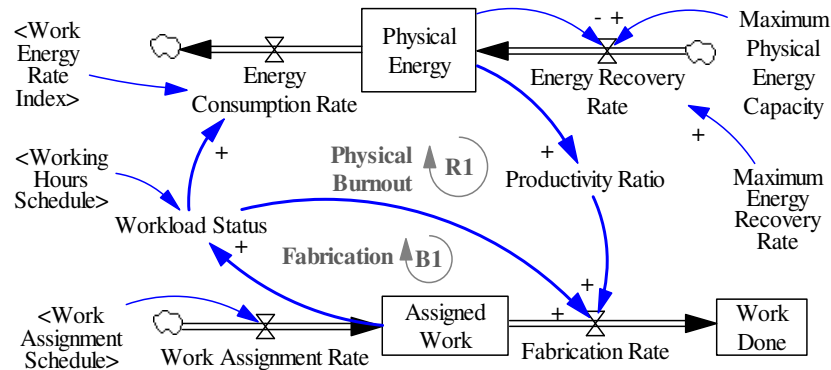


Figure 3-1. The dynamics of physical fatigue as a result of prolonged high physical involvement

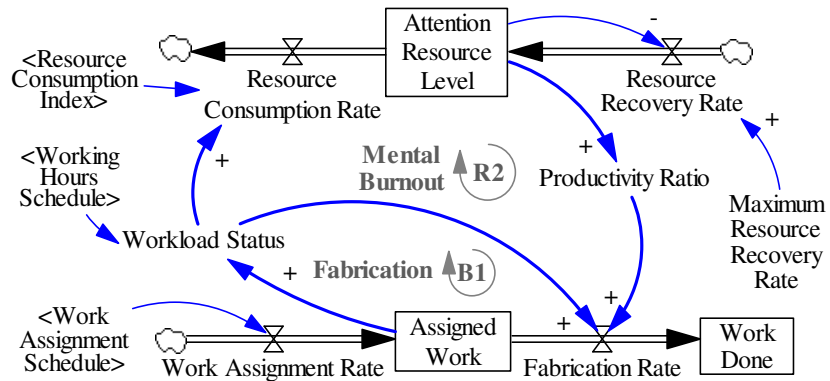


Figure 3-2. The dynamics of mental fatigue as a result of prolonged sustained attention

1) Dynamics of Physical Fatigue

As a complement to Taylor (1911) and Rohmert (1973a and 1973b), Oglesby et al. (1989, pp. 240-251) developed a fatigue model based on human energy consumption. According to Oglesby et al. (1989, pp. 240-251), a typical 25-year-old healthy man has a maximum energy reservoir of 25 kcal and a recovery rate of 5 kcal/minute. With no additional activity, the human body requires 1 kcal/minute for basal metabolism. As a result, work which requires an additional energy consumption rate of less than or equal to 4 kcal/minute can be done continuously for a long period of time with no fatigue. This 4 kcal/minute energy consumption resembles the 15% of maximum strength mentioned by Rohmert (1973b), below which one can do static muscular work with no fatigue. According to the Oglesby et al. model (1989, pp. 240-251), energy required above the available 4 kcal/minute is provided by the energy reservoir until it is completely depleted, resulting in sudden pain and weakness and a drastic decrease in performance.

Figure 3-1 illustrates the stock and flow diagram of the SD model for physical fatigue as a result of high physical involvement based on the human energy consumption theory. In SD, stocks, which appear in rectangles, are the system variables whose values are accumulated or depleted over time at the rate of the flow variables linked to the stock; the flow variables are represented by a double line arrow with a “valve” (an hourglass shape) in the middle. Other variables are

called “auxiliary variables” which are represented by plain text. The auxiliary variables surrounded by angle brackets (“< >”) are read from data tables. Single line arrows show the cause and effect relations between two variables. The plus (+) and minus (-) signs on the arrows indicate direct and inverse relations, respectively, between the variables at the tail and the head of the arrow.

A reinforcing feedback loop, Physical Burnout (R1), is present in the model. The worker will be in busy status if he/she has Assigned Work to do and if, based on the Working Hours Schedule, he/she is in his/her working period. Workload Status value of a worker is 1 if the worker is busy and 0 if the worker is idle. The value of Workload Status for a group of workers (e.g., working in a station) is calculated by dividing the number of busy workers by total number of workers. If the worker is busy, he/she is consuming energy. The Energy Consumption Rate will be set based on the type of work, using the Work Energy Rate Index, which represents the energy consumption rate for a range of basic and construction related activities (Oglesby et al. 1989, p. 248). The level of Physical Energy in the working period is the result of the gap between Energy Consumption Rate and Energy Recovery Rate during the last working period. Thus, an Energy Consumption Rate higher than the Energy Recovery Rate will result in depleted Physical Energy over time and ultimately a reduced Productivity Ratio. The effect of Physical Energy depletion on the Productivity Ratio has been considered using the results of the research done by Rohmert (1973b). A decreased Productivity

Ratio reduces the Fabrication Rate, keeps the level of Assigned Work higher, and consequently will increase the Workload Status of the workers.

Furthermore, as a result of balancing causal feedback loop of Fabrication (B1), a larger Workload Status will result in an increased Fabrication Rate and a correspondingly lower level of Assigned Work, resulting in a reduced Workload Status. In addition to the model variables which are set internally through the recognized feedback loops, there are two variables which are set externally and affect dynamics of physical fatigue: Work Assignment Schedule and Working Hours Schedule. The Work Assignment Schedule contains the list of work assigned to workers or work stations on a daily or hourly basis. Different factors, such as scope of contracts, suppliers, financial condition, and construction managers' preferences, affect the Work Assignment Schedule. The first three factors are set externally; therefore, the construction manager's preference is the main control mechanism for this variable. The Working Hours Schedule refers to the shift start time, the length and arrangement of the work and rest periods within the shifts, and the amount and location of overtime. Setting the Working Hours Schedule is another control mechanism that construction managers can use to influence the physical fatigue dynamics. The set of supporting definitions, equations, descriptions and initial values related to the objects used in the SD model of physical fatigue are presented below:

1. Maximum Physical Energy Capacity:

Type: Constant

Units: Kilocalories (kcal)

Equation:

-

Description: Presents the maximum energy reservoir that can be stored in the muscles.

Initial Value: 25 kcal (Oglesby et al., 1989, p. 249)

2. Maximum Energy Recovery Rate:

Type: Constant

Units: Kilocalories per minute (kcal/minute)

Equation:

-

Description: Presents the maximum energy recovery rate that can be done as compensation to the consumed energy.

Initial Value: 5 kcal/minute (Oglesby et al., 1989, p. 249)

3. Work Energy Rate Index:

Type: Constant

Units: Kilocalories per minute (kcal/minute)

Equation:

Chosen based on the type of work; read from the table provided by Oglesby et al. (1989, p. 248 for more information)

Description: Represents how much energy is going to be consumed according to the type of task assigned to the worker.

Initial Value: Depended on the type of the task might be a value from 1 kcal/minute to 20 kcal/minute.

4. Energy Consumption Rate:

Type: Flow

Units: Kilocalories per minute (kcal/minute)

Equation:

$$\text{EnergyConsumptionRate}_t = \text{Min}((\text{Work Energy Rate Index} - 1) * \text{WorkloadStatus}_t + 1, \frac{\text{PhysicalEnergy}_t}{\Delta t} + \text{EnergyRecoveryRate}_t)$$

Description: Presents the rate of muscular energy consumption in the worker's body and is related to the current task performed by the worker.

Initial Value: 1 kcal/minute

t: current time

Δt : length of time interval (= 1 minute)

5. Energy Recovery Rate:

Type: Flow

Units: Kilocalories per minute (kcal/minute)

Equation:

$$\text{EnergyRecoveryRate}_t = \text{Min}(\text{MaximumEnergyRecoveryRate}, \frac{\text{MaximumPhysicalEnergyCapacity} - \text{PhysicalEnergy}_t}{\Delta t} + \text{EnergyConsumptionRate}_t)$$

Description: Presents the rate of muscular energy recovery in the worker's body.

Initial Value: 1 kcal/minute

Note: EnergyConsumptionRate and EnergyRecoveryRate are mutually dependent variables as is shown in the equations for these objects (i.e., items 4 and 5); mathematically these equations might form a circular link error. But while the dependency of these variables is conditional (as a part of minimum functions), circular link does not happen case here. In the equation for EnergyConsumptionRate, EnergyRecoveryRate only contributes to the value of EnergyConsumptionRate where the value of PhysicalEnergy approaches its lower limit (i.e., 0 kcal), at which point the second part of the minimum function becomes smaller than the first part.

Conversely, in the EnergyRecoveryRate equation, the value of EnergyConsumptionRate participates in the value of EnergyRecoveryRate where the value of PhysicalEnergy approaches its upper limit (i.e., 25 kcal) at which point the second part of the minimum function becomes smaller than the first part. So, these variables will not concurrently affect each other and there will be no circular link error for these equations.

6. Physical Energy:

Type: Stock

Units: Kilocalories (kcal)

Equation:

PhysicalEnergy_t =

PhysicalEnergy_{t-Δt} + (EnergyRecoveryRate_{t-Δt} - EnergyConsumptionRate_{t-Δt})Δt

Description: Represents the level of physical energy in the worker's muscles

Initial Value: 25 kcal

7. Productivity Ratio (Physical Fatigue Factor):

Type: Auxiliary Variable

Units: Percentage (%)

Equation:

ProductivityRatioFactor *_t

$$= \begin{cases} 100\%; & \text{if PhysicalEnergy}_t = 0 \text{ and } 0 \leq \text{Work Energy Rate Index} \leq 5.2 \\ 90\%; & \text{if PhysicalEnergy}_t = 0 \text{ and } 5.2 < \text{Work Energy Rate Index} \leq 5.4 \\ 80\%; & \text{if PhysicalEnergy}_t = 0 \text{ and } 5.4 < \text{Work Energy Rate Index} < 5.8 \\ 65\%; & \text{if PhysicalEnergy}_t = 0 \text{ and } 5.8 \leq \text{Work Energy Rate Index} < 7.5 \\ 0.5; & \text{else} \end{cases}$$

* Equation is based on the research done by Rohmert (1973b)

Note: The final Productivity Ratio in the model is calculated as a product of four Productivity Ratio Factors: (1) physical fatigue, (2) mental fatigue, (3) overtime and (4) time of day.

Description: Productivity ratio represents the speed of the work done by the worker. I consider the productivity ratio as 100% for a normal skilled person in regular working conditions, though fluctuations in the productivity ratio can occur due to changes in the worker and job conditions.

Initial Value: 100%

8. Fabrication Rate:

Type: Flow

Units: Man-hours per minute

Equation:

Is set in discrete event simulation (DES) model and is sent to the SD model. In this case study the entities (representing pieces in the DES model) which have been served and have left an activity (or work station in the DES model) during the last time interval (minute) determine the Fabrication Rate during that time interval. The value of fabrication rate is a function of Productivity Ratio, which affects the service time, and the time that the activity is busy serving the entities.

Description: Represents the rate of the work done over the time interval.

Initial Value: 0 man-hours per minute.

9. Work Assignment Rate:

Type: Flow

Units: Man-hours per minute

Equation:

Is set in discrete event simulation (DES) model and sent to the SD model. In this case the entities arriving to an activity (or work station in the DES model) during the last time interval (minute) determine the Fabrication Rate during that time interval. The entities' arrival is a function of Work Assignment Schedule and prior activities' performance.

Description: Represents the rate of the work assigned to an activity over the time interval.

Initial Value: 0 man-hours per minute.

10. Work Assignment Schedule:

Type: Exogenous Variable

Units: Man-hours per day

Equation:

Is read from the schedule developed for the project on the daily basis. In the DES model, all of the assigned work from the schedule for the day is released to the related activity (or work station, in the DES model) in the first minute (or interval) of the day.

Description: Represents the daily work assigned to an activity. This is usually set for the first activities on the chains of activities (tasks) of the project.

Initial Value: 0 man-hours per day.

11. Work Done:

Type: Stock

Units: Man-hours

Equation:

$$\text{WorkDone}_t = \text{WorkDone}_{t-\Delta t} + (\text{FabricationRate}_{t-\Delta t})\Delta t$$

Description: Represents the total man-hours spent for the completed tasks.

Initial Value: 0 man-hours.

12. Assigned Work:

Type: Stock

Units: Man-hours

Equation:

$$\text{AssignedWork}_t = \text{AssignedWork}_{t-\Delta t} + (\text{WorkAssignmentRate}_{t-\Delta t} - \text{FabricationRate}_{t-\Delta t})\Delta t$$

Description: Represents the total man-hours to be served in an activity (or work station in the DES model).

Initial Value: 0 man-hours.

13. Working Hours Schedule:

Type: Exogenous Variable

Units: No unit

Equation:

Is read from the working hours schedule in every time interval (minute). The time interval might be working (equal to 1) or non-working (equal to 0).

Description: Determines whether workers in the activity are present on their related activity (or work station in the DES model) or off the activity.

Initial Value: 0

14. Workload Status:

Type: Auxiliary Variable

Units: No unit

Equation:

Workload status is determined in DES model. In the DES model the value of Workload Status in an activity, with a number of resources (or workers) assigned to, is calculated by dividing the number of busy resources by the total number of the resources. The value of Workload Status is a function of Assigned Work (or entities) to the activity (or work station in the DES model), which determines whether there are jobs to be done, and the Working Hours Schedule, which determines whether it is working or non-working time.

Description: Represents the fraction of the activity which is busy serving assigned entities (or pieces in the DES model).

Initial Value: 0

2) *Dynamics of Mental Fatigue*

The “mindlessness” theory and the limited resource theory are two rival theories for describing the origins of mental fatigue (Helton and Warm 2008). The mindlessness theory states that the worker’s mind becomes filled with unrelated thoughts and daydreams during prolonged mental tasks; responses to infrequent signals deteriorate and more errors occur (Healy et al. 2004; Steinborn et al. 2009). In the limited resource theory, the human mind has limited resources for thoughtful processing of assigned tasks. There is a recovery rate for the resource reservoir and a consumption rate for mental activities. In prolonged sustained attention tasks, where the resource consumption rate exceeds the resource recovery rate, the resource reservoir decreases over time. This results in a reduction in the worker’s perceptual sensitivity and a decline in the productivity ratio. Smit et al. (2004) and Helton and Warm (2008) show that the mindlessness theory cannot justify the worker’s reaction time increase (or, in this research’s terminology, productivity ratio decrease) during prolonged working hours for mentally demanding tasks. I have therefore based the model of mental fatigue during prolonged working hours on the limited resource theory.

Figure 3-2 presents the dynamics for mental fatigue as a result of sustained attention tasks based on the limited resource theory. Here, again, mental fatigue

dynamics consist of a Mental Burnout reinforcing feedback loop (R2) and a Fabrication balancing feedback loop (B1). Both feedback loops follow similar causal chains as the physical fatigue dynamics. As in the physical fatigue dynamics, Work Assignment Schedule and Working Hours Schedule are two external variables which construction managers can set. The set of supporting definitions, equations, descriptions and initial values related to the objects used in the SD model of mental fatigue are presented in below:

1. Maximum Resource Recovery Rate:

Type: Constant

Units: Percent per minute (%/minute)

Equation:

-

Description: Presents the maximum resource recovery rate that can occur as compensation for the consumed resources.

Initial Value: 1 %/minute

2. Resource Consumption Index:

Type: Constant

Units: Percent per minute (%/minute)

Equation:

-

Description: Determines the mental resource consumption rate according to the type of task assigned to the worker.

Initial Value: 1.1 %/minute for sustained attention tasks and 1%/minute for non-sustained attention tasks.

3. Resource Consumption Rate:

Type: Flow

Units: Percent per minute (%/minute)

Equation:

$$\text{ResourceConsumptionRate}_t = \text{Min}((\text{ResourceConsumptionIndex} - \text{MaximumResourceRecoveryRate}) * \text{WorkloadStatus}_t * \text{SustainedAttention}, \frac{\text{AttentionResourceLevel}_t}{\Delta t} + \text{ResourceRecoveryRate}_t)$$

t: current time

Δt : length of time interval (= 1 minute)

Description: Presents the rate of mental resource consumption in the worker which is related to the current task performed by the worker.

Initial Value: 0 %/minute

4. Resource Recovery Rate:

Type: Flow

Units: Percent per minute (%/minute)

Equation:

$$\text{ResourceRecoveryRate}_t = \text{Min}(\text{MaximumResourceRecoveryRate} * (1 - \text{WorkloadStatus}_t), \frac{100\% - \text{AttentionResourceLevel}_t}{\Delta t} + \text{ResourceConsumptionRate}_t)$$

Description: Presents the rate of mental resource recovery in the worker.

Initial Value: 0 %/minute

Note: ResourceConsumptionRate and ResourceRecoveryRate are mutually dependent variables as is shown in the equations for these objects (i.e., items 3 and 4); mathematically these equations might form a circular link error. But while the dependency of these variables is conditional (as a part of minimum functions), circular link does not happen here. In the equation for ResourceRecoveryRate, ResourceConsumptionRate only contributes to the value of ResourceRecoveryRate where the value of AttentionResourceLevel approaches its lower limit (i.e., 0%), at which point the second part of the

minimum function becomes smaller than the first part. Conversely, in the ResourceConsumptionRate equation, the value of ResourceConsumptionRate participates in the value of ResourceConsumptionRate where the value of AttentionResourceLevel approaches its upper limit (i.e., 100%) at which point the second part of the minimum function becomes smaller than the first part. So, these variables will not concurrently affect each other and there will be no circular link error for these equations.

5. Attention Resource Level:

Type: Stock

Units: Percentage (%)

Equation:

$$\text{AttentionResourceLevel}_t = \text{AttentionResourceLevel}_{t-\Delta t} + (\text{ResourceRecoveryRate}_{t-\Delta t} - \text{ResourceConsumptionRate}_{t-\Delta t})\Delta t$$

Description: Represents the level of availability of mental resources in the worker compared to the maximum possible level of the worker's mental resources.

Initial Value: 100%

6. Productivity Ratio (Mental Fatigue Factor):

Type: Auxiliary Variable

Units: Percentage (%)

Equation:

$$\text{ProductivityRatioFactor}_t = \text{AttentionResourceLevel}_t$$

* Equation is an implicit result from the research done by Rohmert (1973b) and Smit et al. (2004)

Note: The final Productivity Ratio in the model is calculated as a product of four Productivity Ratio Factors: (1) physical fatigue, (2) mental fatigue, (3) overtime and (4) time in the day.

Description: The Productivity ratio represents the speed of the work done by the worker. I consider the productivity ratio as 100% for a normal skilled person in regular working conditions, though fluctuations in the productivity ratio can occur due to changes in the worker and job conditions.

Initial Value: 100%

7. Remaining Objects

The rest of the objects, including Fabrication Rate, Work Assignment Rate, Work Assignment Schedule, Work Done, Assigned Work, Workload Status and Working Hours Schedule, have been explained in the dynamics of Physical Fatigue.

One issue that arises when developing a dynamic model of mental fatigue based on the limited resource theory is that its effective parameters, including Maximum Resource Capacity, Resource Consumption Index and Maximum Resource Recovery Rate, have not yet been thoroughly quantified. I used the commonalities among different empirical case studies and used relational or percentage based values for quantifying different parameters in limited resource theory. One commonality among empirical case studies is that the productivity ratio decreases nearly linearly (e.g. Nuechterlein et al. 1983; Matthews et al. 1990; Smit et al. 2004; and Helton and Warm 2008). To estimate the slope of the productivity ratio reduction, I used Smit et al. (2004)'s study, in which the effects of mental fatigue were isolated via a controlled environment. In addition, continuous task duration

in the study was 50 minutes, which is closer to a typical construction working period duration than task durations used in other similar studies (Nuechterlein et al. 1983, 5 to 10 minutes; Matthews et al. 1990, 5 to 10 minutes; Helton and Warm 2008, 12 minutes). Smit et al. (2004) found no significant effects on performance in low-demand mental tasks, but found performance decreases during high-demand (or sustained attention) tasks by 0.1% per minute, because of the gap between resource consumption and resource recovery rates. During rest time the mental resource “reservoir” is refilled. According to Rohmert (1973b), a rest allowance of 10% is required for recovery from fatigue caused by mental work. This means the required recovery period is one tenth of the working period, so the resource recovery rate is 10 times the gap between Resource Consumption Rate and Resource Recovery Rate during work (i.e., 0.1% per minute), or 1% per minute.

For the mental fatigue dynamics I postulated that availability of all (or 100% of) mental resources shows the maximum mental resource capacity, regardless of the actual value; the value of Attention Resource Level is set as a percentage of the maximum mental resource capacity. The mental resource value component of all other parameters in the limited resource theory is also determined as a percentage of the maximum mental resource capacity. So, the Maximum Resource Recovery Rate will be 1% per minute, and the Resource Consumption Index simply returns 1.1% per minute for all sustained attention tasks. For non-sustained attention tasks, which do not consume mental resources over time, Resource Consumption

Index is equal to or less than Maximum Resource Recovery Rate. For the sake of simplification I assumed the Resource Consumption Index for all non-sustained attention tasks was equal to 1% per minute (equal to Maximum Resource Recovery Rate) since this will result in a mental resource consumption-recovery balance similar to reality. While the rate of resource consumption and resource recovery are set based on the changes in worker performance presented in Rohmert (1973b) and Smit et al. (2004), there is a one-to-one relation between changes in the Attention Resource Level and Productivity Ratio.

3.2.2. Dynamics of Time of Day

Typically, performance is higher during the day and lower at night (Dijk et al. 1992; Folkard and Tucker 2003; Baltter and Cajochen 2007); however, performance level fluctuates hourly. Interestingly, both work speed and work quality follow almost the same 24-hour trend (Baltter and Cajochen 2007). Because work performance fluctuates throughout the day, it is expected that work start and finish time can affect final work performance. Figure 3-3 illustrates the dynamics of performance factor changes over 24 hours. Hour in the Day Index reflects the hourly fluctuations in the performance indexes and is the main change driver in the model. It is set based on the normalized circadian performance changes provided by Folkard and Tucker (2003). For example, using the normalized performance changes and the work specification, productivity ratio in the case study fluctuates from 73% at 3 am to 107% at 10 am.

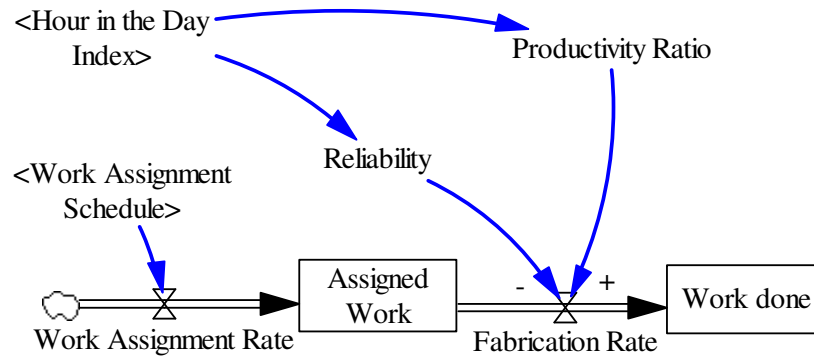


Figure 3-3. The dynamics of performance factor changes based on biological clock

Details of the supporting definitions, equations, descriptions and initial values for objects used in the dynamic model of the time of day are presented below:

1. Hour in the Day Index:

Type: Exogenous Variable

Units: Percentage

Equation:

For calculating the Hour in the Day Index in the case study, first I averaged the provided normalized fluctuations by Folkard and Tucker (2003) during the day and night shifts of the case. Then, I acquired the average productivity ratios and reliabilities of the day shift and night shift through a series of interviews with the managers and superintendents at the collaborative company. The results are summarized in Table 3-1.

Table 3-1. Average performance indexes during day and night shift

	Average Normalized Values*	Average Productivity Ratio **	Average Reliability**
Day Shift Average (5:30am to 4:00pm)	0.29	100%	1.0%
Night Shift Average(4:30pm to 2:30am)	-0.13	95%	1.5%

* Based on estimated values from the normalized diagram provided by Folkard and Tucker (2003)

** Based on interviews with the shop managers in the collaborative company

Then, hourly fluctuations in the productivity ratio and reliability were calculated through a relative equation between actual and normalized averages for the day shift and the night shift by using the following formula:

$$PIF_t = PIFNS + \frac{PIFDS - PIFNS}{NVDS - NVNS} \times (NV_t - NVNS)$$

PIF_t: Performance Index Factor at time *t*

PIFDS: Performance Index Factor average for the Day Shift

PIFNS: Performance Index Factor average for the Night Shift

NVDS: Normalized Values (from Folkard and Tucker, 2003) average for the Day Shift

NVNS: Normalized Values (from Folkard and Tucker, 2003) average for the Night Shift

NV_t: Normalized Values (from Folkard and Tucker, 2003) at time *t*

Some calculated performance index factors for hours of the day are presented in the Table 3-2.

Table 3-2. Performance indexes in different hours of the day

Hour in the day	Normalized Values*	Productivity ratio Factor	Reliability Factor
1	-1.18	83%	2.7%
6	-0.09	96%	1.4%
12	0.24	99%	1.1%
18	0.88	107%	0.3%
24	-0.74	88%	1.3%

* Based on estimated values from the normalized diagram provided by Folkard and Tucker (2003)

The Hour in the Day Index is updated in every time interval (i.e., every minute in out model minute).

Description: Reflects the fluctuations in the workers' performance as a result of biological change in the human body over 24 hours of the day.

Initial Value: 100%

2. Productivity Ratio (Circadian Factor):

Type: Auxiliary Variable

Units: Percentage (%)

Equation:

As explained for the first item (Hour in the Day Index)

Description: Productivity ratio represents the speed of the work done by the worker. I consider the productivity ratio as 100% for a normal skilled person in regular working conditions, though fluctuations in the productivity ratio can occur due to changes in the worker and job conditions.

Initial Value: 100%

3. Reliability:

Type: Auxiliary Variable

Units: Percentage (%)

Equation:

As explained for the first item (Hour in the Day Index)

Description: Reflects the probability of deficiency occurrence. In the DES model, in case of any deficiencies in the served entities (or fabricated pieces) the deficiencies are recognized in the inspection activities and the entities are directed to the previous activity where rework is done and their problems are fixed. The defective entities are not counted as fabricated entities.

Initial Value: 1%

3.2.3. Dynamics of Overtime Work

The main idea of scheduling overtime is to compensate for delays by working additional hours. However, fatigue from overtime can adversely affect job

performance. According to Homer (1985) and Sterman (2000, pp. 577-583), while weekend breaks relieve the fatigue accumulated during the week, overtime hours will add to accumulated fatigue such that complete relief cannot be achieved during the weekend, so some fatigue from the previous week is transferred to the next. The adverse effects of overtime on construction productivity are also reflected in the RSMMeans (2010) reference tables.

The overtime dynamics are mainly affected by a balancing feedback loop, Increased Working Hours (B2), and a reinforcing feedback loop, Overtime Burnout (R3) (Figure 3-4). Similar feedback loops were used by Lyneis and Ford (2007) as control feedback loops in their research on project schedule performance. In the Increased Working Hours loop, a higher Set Overtime for the week increases the amount of working hours during the week and will increase the Fabrication Rate and ultimately the actual Work Done. This will decrease the gap between actual Work Done and Scheduled Work Finish and will reduce the Required Amount of Overtime for the next week. The reinforcing Overtime Burnout loop starts with the calculated Amount of Overtime Required as a result of difference between actual Work Done and Scheduled Work Finish divided by Number of Workers. According to the calculated Required Amount of Overtime, the Maximum Overtime and the company's overtime policy, Set Overtime for the following week is determined. Any adverse effect of Set Overtime on the Productivity Ratio will occur a week later. The value of the Productivity Ratio is a function of Set Overtime for the last week, based on Sterman's Overtime

Productivity Index (2000, p. 581). A reduced productivity ratio will reduce the Fabrication Rate and Work Done will increase more slowly, resulting in a bigger gap between scheduled work and actual work done. The main external effective factor in the dynamic model that can be adjusted by construction project managers is Maximum Overtime, or any other company overtime policies.

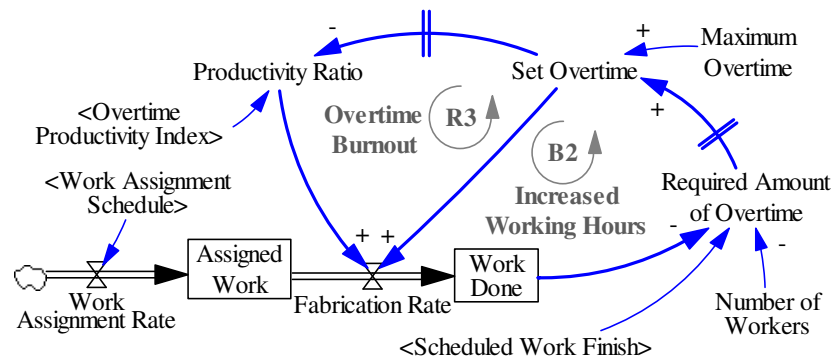


Figure 3-4. The dynamics of overtime working and performance

The complete set of supporting definitions, equations, descriptions and initial values for overtime dynamics are presented in below:

1. Scheduled Work Finish:

Type: Exogenous Variable

Units: Man-hours

Equation:

Read from the developed work finish schedule for the project on a daily basis.

Description: Represents the expected daily work progress for an activity.

Initial Value: 0 man-hours

2. Number of Workers:

Type: Exogenous Variable

Units: Workers

Equation:

Set in the DES model and sent to the SD model. Workers are represented by resources in the DES model developed for the case.

Description: Number of workers assigned to an activity.

Initial Value: Varies from one activity to the other.

3. Required Amount of Overtime:

Type: Auxiliary Variable

Units: Hours

Equation:

$$\text{RequiredAmountofOvertime}_w = \text{Max}\left(\frac{\text{ScheduledWorkFinish}_w - \text{WorkDone}_w}{\text{Number of Workers}}, 0\right)$$

w: current time (in week)

Description: Presents the amount of overtime in hours that is required for the workers assigned to the activity to advance the work progress to what was originally scheduled.

Initial Value: 0 hours

4. Maximum Overtime:

Type: Constant

Units: Hours per week (h/w)

Equation:

Is set based on the organization's policy and labor code.

Description:

Initial Value: 20 h/w

5. Set Overtime:

Type: Auxiliary Variable

Units: Hours per week (h/w)

Equation:

$$\text{SetOvertimefortheWeek}_{w+\Delta w} = \text{Min}(\text{RequiredAmountofOvertime}_w, \text{MaximumOvertime})$$

Δw : length of time interval (= 1 week)

Description: Refers to the set overtime for week. The set overtime for the week contributes directly to the Fabrication Rate by increasing the working hours of the week and affects the Fabrication Rate indirectly as a result of occurred fatigue and decreased productivity ratio for the next week.

Initial Value: 0 h/w

6. Overtime Productivity Index:

Type: Exogenous Variable

Units: No unit

Equation:

Is based on the empirical research study presented by Sterman (2000, p. 581).

Description: Reflects the fluctuations in the current week's productivity ratio as a result of set overtime for the last week. The index shows the instruction for calculating Productivity Ration (Overtime Factor) in different Set Overtime conditions.

Initial Value: -

7. Productivity Ratio (Overtime Factor):

Type: Auxiliary Variable

Units: Percentage (%)

Equation:

$$\text{ProductivityRatioFactor}_{w+\Delta w}^* = \begin{cases} 1; & \text{if SetOvertime}(W) = 0(\text{hours}) \\ 80\% + \frac{20 - \text{SetOvertime}_w}{20} * 20\%; & \text{if } 0\text{h} < \text{SetOvertime}_w \leq 20\text{h} \\ 40\% + \frac{40 - \text{SetOvertime}_w}{20} * 40\%; & \text{if } 20\text{h} < \text{SetOvertime}_w \leq 40\text{h} \\ 10\% + \frac{60 - \text{SetOvertime}_w}{20} * 30\%; & \text{if } 40\text{h} < \text{SetOvertime}_w \leq 60\text{h} \\ \frac{140 - \text{SetOvertime}_w}{40} * 10\%; & \text{if } 60\text{h} < \text{SetOvertime}_w \leq 100\text{h} \\ 0; & \text{else} \end{cases}$$

* Extracted from the table provided by Sterman (2000, p. 581)

Note: The final Productivity Ratio in the model is calculated as a product of four Productivity Ratio Factors related to (1) physical fatigue, (2) mental fatigue, (3) overtime and (4) time in the day.

Description: Productivity ratio represents the speed of the work done by the worker. I consider the productivity ratio as 100% for a normal skilled person in regular working conditions, though fluctuations in the productivity ratio can occur due to changes in the worker and job conditions.

Initial Value: 100%

3.2.4. Critical Parameters for Worker Capability

The effective parameter values within the dynamic models presented here (i.e., the base values) represent the behaviors of a range of workers with a certain level of capabilities. Deviations of some of the model parameters (critical parameters) should therefore be taken into account when determining the effects of working hours on productivity for different groups of workers.

In physical fatigue dynamics, two main effective factors may vary among different working groups according to capability: Maximum Physical Energy

Capacity and Maximum Energy Recovery Rate (see Figure 3-1). During a rest period, a weak worker takes longer to recover from occurred fatigue than a strong worker. One possibility is increasing Maximum Physical Energy Capacity to adjust the model to a weak worker. While this increases recovery time, it is not realistic, as a weak worker typically has a lower-capacity energy reservoir. The other possibility is decreasing the Maximum Energy Recovery Rate, which also increases recovery time, but conforms to the expected decreased recovery rate in weak workers. The critical effective parameter here is therefore the Maximum Energy Recovery Rate.

In the mental fatigue model, because there was no comprehensive quantification found for the effective parameters in this model, I have set the maximum resource capacity as a base value equal to 100%, regardless of its absolute value. The Resource Recovery Rate is the only effective parameter that can be changed to adjust for the worker's capability (Figure 3-2), and is therefore the critical parameter. Likewise, in the overtime fatigue model, the Overtime Productivity Index is the only effective parameter that can be changed (see Figure 3-4), and therefore the fatigue effect of overtime will be adjusted by increasing the slope of Overtime Productivity Index function.

Finally, for circadian dynamics, the base values for the circadian effects on worker performance are from the normalized diagram prepared by Folkard and Tucker (2003). These indexes are customized for every group of workers

according to their capabilities. For example (as indicated in Section 3.2.2), I have used the average of performance index factors for the day shift and the night shift to customize the provided normalized values. It is expected that weak workers' productivity will decrease more during the night shift compared to strong workers.

The weak group of workers in the research was created by deviating -20% in the critical parameters in the dynamic models, corresponding to the Oglesby et al.'s estimated maximum energy recovery rate reduction in 60-year-old male workers (1989, p. 250). The base model parameters were assumed to be a representation of strong workers.

3.3. Model Testing

All dynamics mentioned in Section 3.2 work in parallel in actual construction work. To be able to capture different aspects of working hour dynamics more precisely, I merged all the previously described dynamic models into one SD model. The time interval of all SD sub-models in the integrated model is one minute, except for the overtime dynamics, which adopts one week as its time interval. Therefore, the set time interval for the integrated model is one minute. However, in the overtime dynamic model minute-by-minute updates for the overtime are meaningless, because in reality it is set week by week. To resolve these issues in the overtime dynamics, although the value of Work Done in the integrated model is updated every minute, its effects on the Required Amount of

Overtime are considered on a weekly basis by counting the number of minutes within the week. Values of other variables down the causal chains of the overtime dynamics therefore stay constant during the week. In the case that variables from the overtime dynamic participate as causal variables in the equations together with variables from other parts of the model (e.g., in the productivity ratio equation), the variables from overtime dynamics stay constant during the week, regardless of the minute-by-minute changing nature of the rest of variables.

Since the developed SD models are heavily dependent on well-established theories, according to Cronbach and Meehl (1955) a construct validity test is suitable type of validation to test whether the SD models are proper representations of the adopted theories. For this purpose, I followed the methods suggested specifically for dynamic models by Sterman (2000, pp. 843-858) (Table 3-3). However, it should be noted that Sterman indicates that because of the measurement errors, abstractions, aggregation, and simplifications, true model validation for the developed SD models is impossible. Rather, these tests are to demonstrate the model's usefulness by revealing its capabilities, limitations, and flaws, to assist prospective model users in properly applying the model to their applications. Most of the structure validation tests were applied concurrently during the model development process and have been mentioned in the model descriptions in Section 3.2. The following are some examples of behavioral tests that were also conducted. All test models were implemented in AnyLogic 6.4; the details of the test models have been presented in Appendix C.

Table 3-3. Summary of applied validation tests

Test	Purpose of the Test	Summary of Test Process
Boundary adequacy	To ensure that the important concepts have been included in the model.	The dynamic models have been developed based on established theories in their related area of research (i.e., effect of working hours on workers' productivity). All variables in the theories are included in the models and the stock and flow diagrams (Figure 3-1 to 3-4).
Structure assessment	To test the model structure consistency with the relevant declarative knowledge of the system.	The model structure is built in accordance with the theory descriptions in the literature (Section 3.2). The descriptions of the dynamic models (Figure 3-1 to 3-4) affirm this.
Dimensional consistency	To test whether all used variables have meanings in the real world and all equations are consistent in the dimension of their used variables.	The meaning of each model variable and its counterpart in the real system has been described in Section 3.2. Dimensional analysis of the equations has been conducted to check consistency of the variables.
Parameter assessment	To test if values of model parameters can represent different aspects of the system.	Critical parameters which could represent different aspects of construction working groups were determined and explained in Section 3.2.4.
Extreme conditions	To test reasonable behavior of the model in response to extreme values for model inputs.	Extreme values were set in the model critical parameters (Section 3.2.4) and the model was run. The model behaved plausibly in response to all extremely set values.
Integration error	To check the sensitivity of the model to different time steps.	Different time steps (including 0.05, 0.1, 0.2 and 0.5) were tested in the model and no substantial differences were observed.
Behavior reproduction	To test whether model can generate different possible behaviors of the system.	Systems with different working hours conditions, types of work and start and finish time and overtime policies were successfully modeled. Some results are presented in Figure 3-5 and discussed in Section 3.3.
Sensitivity analysis	To assess how reasonable changes in some uncertain or adjustable assumptions can affect the final conclusion.	Sensitivity analysis conducted for the model (Section 3.4).

As a test for behavior reproduction capacity, I simulated worker behavior during physical and mental tasks. The shifts for the test were 11.25 hours long; the workers were assumed to be busy during the working periods, and each period was followed by a 15-minute rest break. I also set one rest break as a lunch break, 30 minutes in duration. The productivity ratio changes were simulated in the base

model for the physical, mental and combined physical and mental tasks (Figure 3-5). In accordance with the energy consumption theory, physical fatigue appeared right after energy depletion and caused a sudden decrease in the worker's productivity ratio (Figure 3-5a). Mental fatigue caused a gradual decrease in the productivity ratio, as predicted by the limited resource theory (Figure 3-5b). The combined physical and mental tasks showed both sudden and gradual productivity decrease (Figure 3-5c).

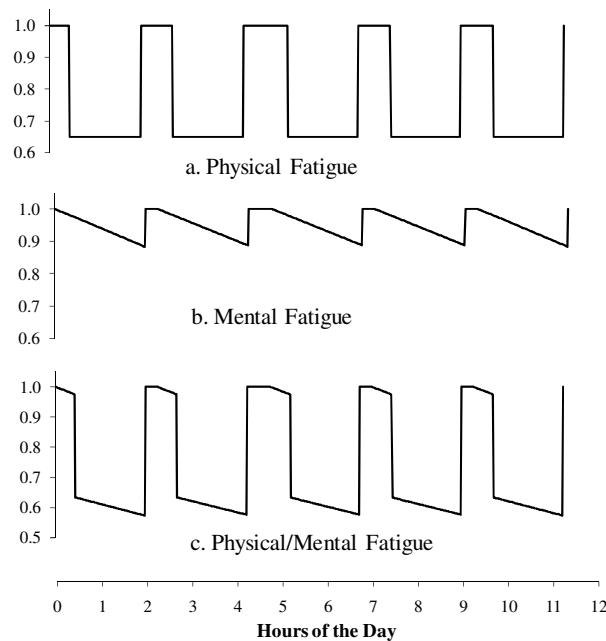


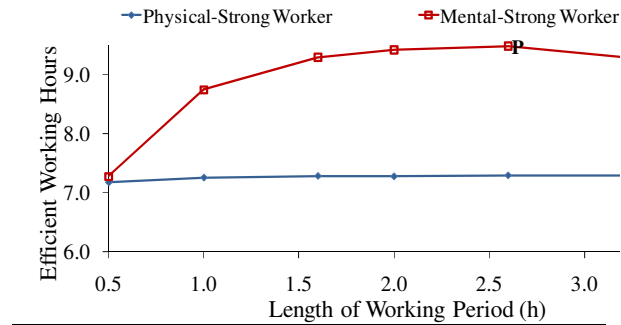
Figure 3-5. The effect of different types of fatigue on workers' productivity ratio

An extreme condition test was performed by creating extremely strong workers through deviating +100% in the critical parameters from the dynamic models (see Section 3.2.4). I exposed this group of the workers to the regular physical and mental tasks. As expected, the model indicated no fatigue and no reduction in the productivity ratio for the extremely strong workers. Extremely weak workers,

who only had enough energy and resource recovery rates to maintain their basal metabolisms, naturally had no productivity.

3.4.Sensitivity Analysis of Model Behavior

I investigated the extent of the effects of fatigue on the worker productivity ratio in different effective factors, concentrating on the effective factors which are adjustable through work policies. First, a series of simulation runs were conducted by varying work period length from 0.5 to 3.5 hours for different types of tasks in the base model. Although the length of the shift was the same (i.e., 11.25 hours), total working and rest hours changed, as shorter work periods resulted in more rest periods. Figure 3-6 shows the simulation results in different categories; the table below the graph shows total working and rest hours for each different work period length. The effect on efficient working hours of changing the work period length can be as substantial as 2.2 hours for mental tasks, or as minimal as 0.1 hours a shift for physical tasks within the 11.25 hour shift.



Length of Working Periods (h)	0.5	1	1.6	2	2.6	3.5
Working Hours (h)	7.375	9	9.75	10	10.25	10.5
Rest Hours (h)	3.875	2.25	1.5	1.25	1	0.75
Shift Duration(h)	11.25	11.25	11.25	11.25	11.25	11.25

* P letters in the curve indicate the achieved Peak point for the different types of works with different types of workers

Figure 3-6. Deviations in the efficient working hours by changing length of working periods

I also ran a series of sensitivity analyses to investigate the effects of changes in time of day (Figure 3-7), overtime (Figure 3-8), and worker strength (Figure 3-9) on worker performance. In sum, the observed productivity sensitivity to the length of working periods and shift start time is noteworthy here because they are adjustable with no additional investment or change in the workers (Figures 6 and 7). The dynamic model can also be used to improve overtime, another control mechanism (Figure 3-8). Finally, the model can investigate the effects of worker strength and different types of jobs, a control mechanism which may be useful when construction managers are recruiting new workers (Figure 3-9).

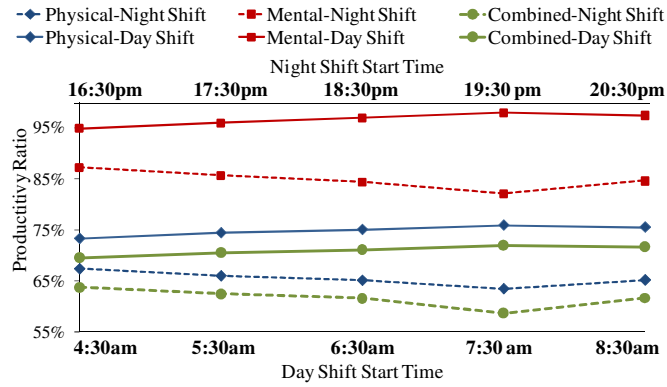


Figure 3-7. The effects of start time on productivity ratio in physical and mental tasks

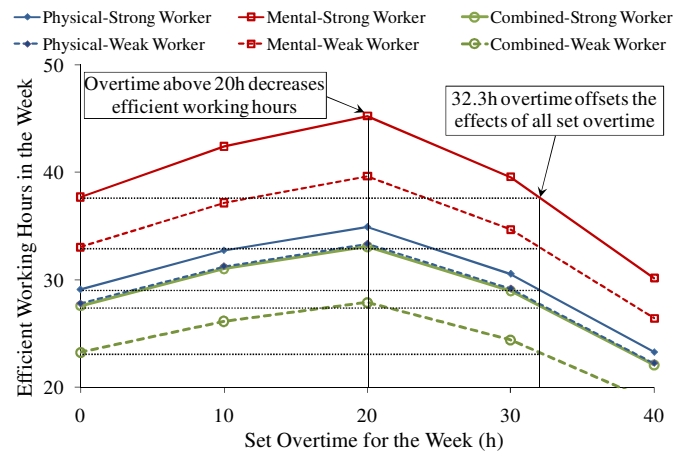


Figure 3-8. The effects of overtime on weekly efficient working hours

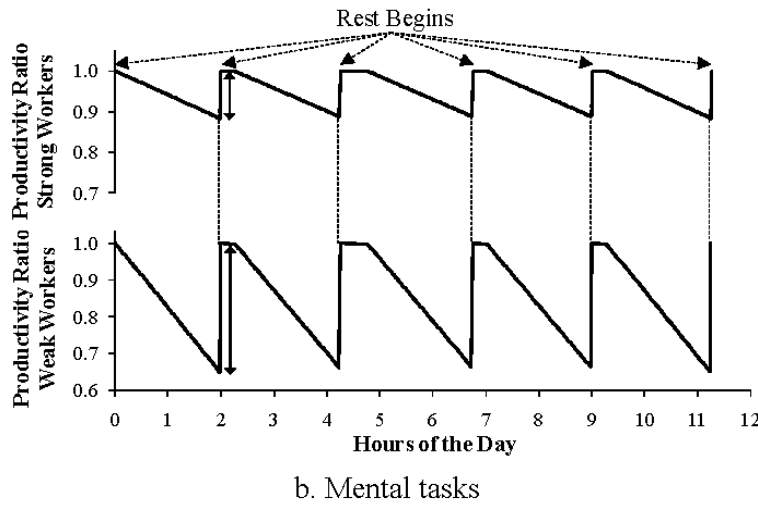
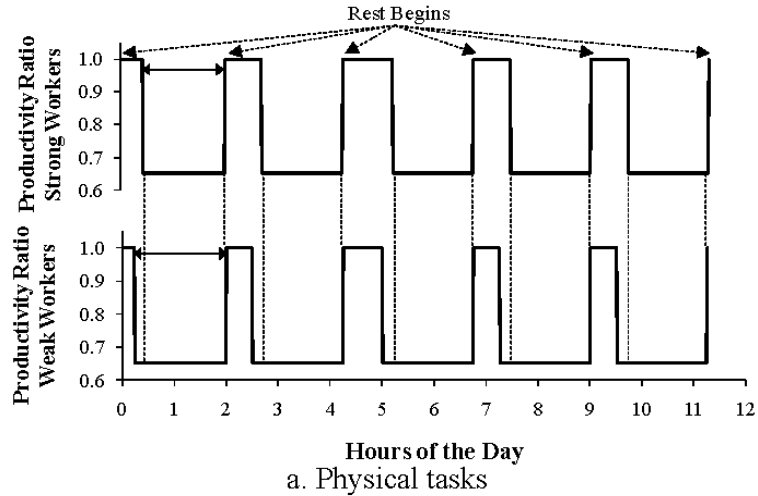


Figure 3-9. The effects of worker’s strength on productivity ratio in Physical and mental tasks

3.5. Hybrid Model of Integrated Working Hour Dynamics

Several simplifying but unrealistic assumptions were used in the sensitivity analyses in Section 3.4 (e.g., continuous work during work periods and no change in task type). More realistic assumptions would involve capturing operational structure: keeping track of the work schedule and flow of material within jobs

and running the model based on actual changes in worker status and types of assigned tasks. To accomplish this, I use Discrete Event Simulation (DES) for modeling the operational and SD for modeling the non-operational parts of the system (i.e., feedback); this combination is called the Hybrid SD-DES modeling approach. In the hybrid SD-DES modeling approach, one model's capabilities cover the other's shortcomings to provide more accurate results for system analysis (hybrid SD-DES modeling approach has been discussed in detail in Chapter 2).

The values for Work Assignment Rate, Assigned Work, Workload Status, Fabrication Rate and Work Done (Figures 3-1 to 3-4) will be determined in the DES part of the model. The methods for calculating the value of the mentioned parameters, developed for the case study, are presented in Section 3.6.2. Conversely, the Productivity ratio and Reliability are calculated and sent from the SD part to the DES part (more details the mechanism of the interactions between DES and SD parts are provided in Chapter 2). This causes all of the feedback loops described in Section 3.2 to have their cause and effect chains continued in the DES part of the model. Figure 3-10 illustrates this integrated hybrid model. The variables with the highlighted background and bold fonts in the figure represent the variables which send/receive updated values to/from other parts of the model. The dashed arrows show the direction of communication between the SD and DES modeling parts. Supporting equations for the SD part of the model have been presented in different parts of Section 3.2.

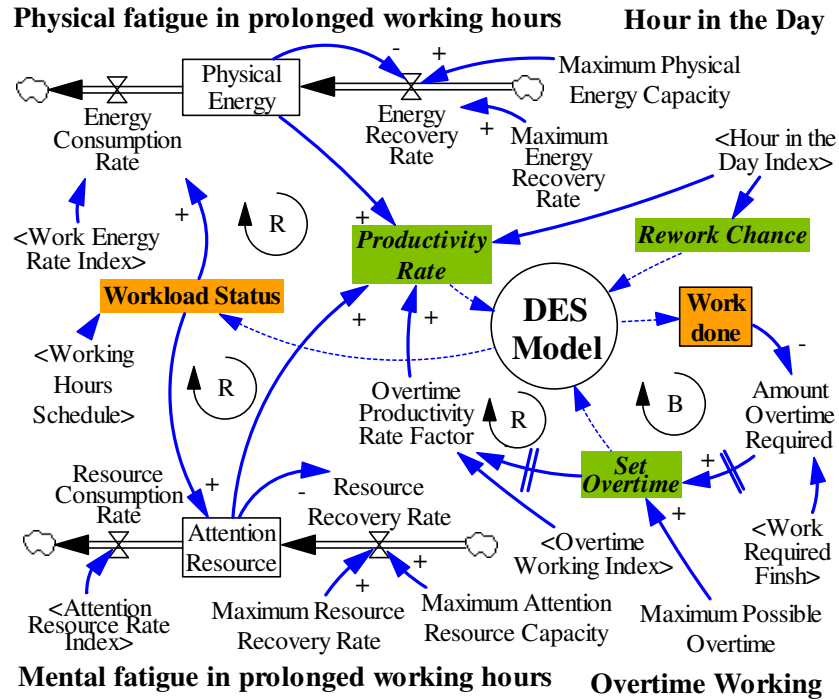


Figure 3-10. Modified dynamics of fatigue models in hybrid model

3.6. Case Study

To observe the effects of different working hour arrangements and policies on final productivity in a real construction case, I have developed a hybrid SD-DES model of working hours in a structural steel fabrication shop in a collaborative company. Off-site fabrication has been introduced to construction project processes due to the expected improvement in productivity compared to traditional on-site construction (Eastman and Sacks 2008). Unlike the repetitive work procedures and similar outputs of manufacturing work stations, jobs assigned to the fabrication shop stations fluctuate over time. Although fabrication shop stations are specialized for one type of operation (e.g., welding, cutting, or

painting), as time passes they serve different projects with different specifications and work scopes; even within a project they serve different parts of the project which are not necessarily similar to the others. I have tested the effects of different overtime policies, working period durations and rest break arrangements on the shop throughput for 3 months of operation. Data used in the case are real data collected as part of the fabrication shop's tasks and used for actual daily planning and control tasks in the shop.

3.6.1. Case Specification

Five major operation types are done in the fabrication shop: cutting, fitting, welding, inspection, and painting. There is one centralized cutting shop which serves the entire fabrication shop; three shops for fitting, welding and required inspections (called Shop AB, Shop C, and Shop D); and finally one painting shop. The cutting operation is almost completely mechanized, but the remaining operations are done mainly by hand tools. Shop AB is used for heavy pieces, and Shop D for light pieces, which require a high number of man-hours for the fitting and welding operations on each piece. Shop C is designed for pieces which require a small number of man-hours; the assigned operations here are usually less complicated and more routine than the other shops.

According to the dynamic models presented in Figure 3-10, Work Energy Rate Index and Attention Resource Rate Index are two inputs to the SD models. These indexes were set based on the station work type. I used the table provided by

Oglesby et al. (1989, p. 248) as the reference for the Work Energy Rate for different categories of construction tasks. For example, in this table the Energy Consumption Rate of male workers doing carpentry is 4 kcal/minute; continuous sawing and hammering is 8.1 kcal/minute; and average construction work is 6 kcal/minute. Most work is done with electrically powered tools, so I used a Work Energy Rate of 4.8 Kcal per minute, except for welding of heavy pieces (shop AB) which requires more manual effort and is set to 6 Kcal per minute. Since no references which have provided an Attention Resource Rate Index for construction operations could be found, to classify construction operations into low and high demand mental tasks in the case study (Section 3.2.1 part 2) I based the classification on the shop observation and interviews with the shop workers. The Required Attention Resource Rate for welding, which requires concentrated attention on the welding point, was set as one; it was set as zero for all other operation types.

3.6.2. Base Model Experiment

Currently the standard working hours for the fabrication shop in this collaborating company consist of a day shift (10.5 hours, 5:30 am – 4:00 pm) and night shift (10 hours, 4:30 pm – 2:30 am), Monday to Thursday. Every shift consists of 5 2-hour working periods, followed by 15 minutes rest, except the last working period, which is 1.25 hours for the day shift and 0.75 for the night shift. The second rest

break is half an hour long for the lunch break. When overtime is required, the company's policy is to set all of Friday and/or Saturday as overtime.

A DES model was developed for simulating the fabrication shop operation. In the DES model, entities represent steel pieces, workers are represented by resources and each station (a place where a group of workers perform the same type of operation on the received pieces) is represented by an activity (for a detailed description on building elements of DES, see Banks, 2010). Work Assignment Rate, Assigned Work, Workload Status, Fabrication Rate and Work Done are the parameters in the causal feedback loops (Figures 1 to 4) which are calculated in the DES model. The Values of Work Assigned Rate and Fabrication Rate are calculated based on the entities' (or pieces') flow from one activity (or station) to the other. The Productivity Ratio, calculated from the SD component of the model, here participates as a factor for determining the time that each entity spends to be served in an activity. The entities which arrived at an activity during a time interval determine the Work Assign Rate during that time interval and the entities which have left an activity during the last time interval determine the Fabrication Rate. The accumulation of the Fabrication Rate determines the Work Done and the accumulation of the Work Assign Rate minus Work Done determines the value of Assigned Work to an activity. The value of Workload Status in an activity (or station), which has a number of resources (or workers) assigned to it, is calculated by dividing the number of busy resources by the total

number of the resources. The implementation details of the model have been presented in Appendix C.

Every station in the DES model has a supporting SD model which captures the effect of different types of fatigue on the productivity ratio and reliability (if applicable), as explained in Section 3.5 and illustrated in Figure 3-10. The DES model is linked to the collaborative company's database and releases the steel piece fabrication orders to the fabrication shop according to the scheduled start dates, read from the database. I simulated a period of 3 months of material feed, from January 5th to April 5th, with around 30,000 pieces and scheduled working hours of 250,000 man-hours. The simulation was run until the last piece fed to the shop was fabricated. According to the simulation results fabrication will be completed on April 15th and the average productivity ratio will be 88.8%.

As illustrated in Figure 3-11, the simulated completion curve almost follows the actual completion curve. The only exceptions are at the beginning and at the end. The actual completion starts at a lower rate because of previously assigned work. After two weeks the progress slopes are very close, until the end of 3 months of material feed. At this stage, because the actual fabrication shop continues to receive new orders but the simulated shop does not, the simulated progress rate surpasses the actual progress. The simulation also does not take material shortage into account, which postponed fabrication completion for the actual fabrication shop. To assess the compatibility of the model results with the real shop output,

tests have been run on the first 3 months of the simulation results following the behavior reproduction tests presented by Sterman (2000, pp. 874-880). The actual data and simulation results showed a correlation coefficient (r) of 0.989 and coefficient of determination (R^2) of 0.979 which indicates a strong relation between the reproduced and actual data. For investigating systematic errors in the behavior reproduction, I used Theil's inequality statistics (Theil 1966). The test showed a bias index of 0.008, unequal variation index of 0.342, and unequal co-variation of 0.651. According to the interpretation method provided by Sterman (2000, p. 876), the results indicate unsystematic error, which affirms the goodness of behavior reproduction.

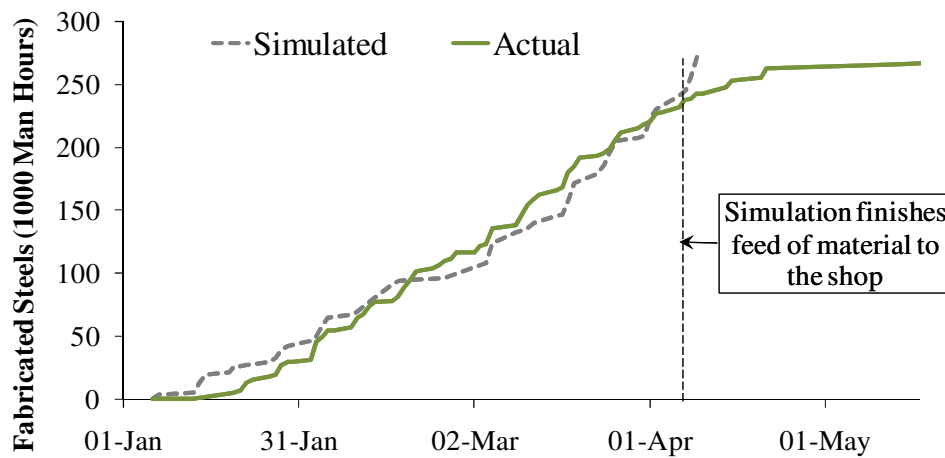


Figure 3-11. Comparison between completed fabrication in simulation and actual case

3.6.3. Shift Alternatives

To investigate the effects of deviations to the base working hours schedule on the final productivity of the fabrication shop, I developed 22 possible working shifts, described below:

Alternative 1: Shifted work schedule. 6 working schedules were formed by setting the start time 1, 2 or 3 hours before or after the regular work start time (5:30 am).

Alternative 2: Work schedule with evenly distributed time breaks. 7 working schedules were formed by setting the work periods to 1.85 hours for the day shift and 1.75 hours for the night shift, then different working schedules were created by setting the work start time 1, 2 or 3 hours before or after the current work start time.

Alternative 3: Work schedule with shortened working periods to 1 hour. 7 working schedules were formed by reducing the work periods to 1 hour and shifting the start time 1, 2 or 3 hours before or after the current work start time.

Alternative 4: Work schedule with overtime. 2 working schedules were formed by setting 1 or 2 permanent overtime days for the current working schedule.

3.6.4. Case Analysis

The average productivity ratio and fabrication completion were calculated through the simulation model for all of the working shift alternatives. As presented in Table 3-4, the best productivity ratio, 94.4%, was achieved through a work schedule with working periods shortened to 1 hour (shift alternative 3) and a work start time of 5:30 am. The least productivity ratio, as expected, was the work schedule with 1 and 2 days overtime (shift alternative 4), respectively with productivity ratios of 79.7% and 70.8%. The improved productivity ratio shows a potential improvement of 6% compared to the current work schedule, which has longer work periods (i.e. 1.75 hours).

Table 3-4. Productivity ratio (%) and fabrication duration in calendar days (cd) and working days (wd) achieved in simulation runs of different shift alternatives

Shift Alternative	Work Start Time						
	2:30 am	3:30 am	4:30 am	5:30 am	6:30 am	7:30 am	8:30 am
Current work schedule*				<u>88.3%</u> <u>94 cd</u> <u>55 wd</u>			
1. Shifted work schedule	<u>87.5%</u> <u>98 cd</u> <u>56 wd</u>	<u>88.5%</u> <u>94 cd</u> <u>55 wd</u>	<u>88.8%</u> <u>94 cd</u> <u>55 wd</u>		<u>87.5%</u> <u>98 cd</u> <u>56 wd</u>	<u>86.9%</u> <u>98 cd</u> <u>57 wd</u>	<u>85.7%</u> <u>99 cd</u> <u>58 wd</u>
2. Work schedule with evenly distributed time breaks	<u>87.4%</u> <u>98 cd</u> <u>56 wd</u>	<u>88.6%</u> <u>93 cd</u> <u>55 wd</u>	<u>88.8%</u> <u>94 cd</u> <u>55 wd</u>	<u>88.5%</u> <u>94 cd</u> <u>55 wd</u>	<u>87.5%</u> <u>94 cd</u> <u>56 wd</u>	<u>87.1%</u> <u>94 cd</u> <u>56 wd</u>	<u>86.5%</u> <u>98 cd</u> <u>57 wd</u>
3. Work schedule with shortened working periods to 1h	<u>89.4%</u> <u>93 cd</u> <u>54 wd</u>	<u>91.4%</u> <u>92 cd</u> <u>54 wd</u>	<u>91.5%</u> <u>92 cd</u> <u>54 wd</u>	<u>94.4%</u> <u>92 cd</u> <u>54 wd</u>	<u>89.8%</u> <u>92 cd</u> <u>54 wd</u>	<u>89.8%</u> <u>93 cd</u> <u>54 wd</u>	<u>89.4%</u> <u>92 cd</u> <u>54 wd</u>
4. Work schedule with overtime				<u>79.7%</u> <u>92 cd</u> <u>66 wd</u>			
				<u>70.8%</u> <u>92 cd</u> <u>79 wd</u>			

* Results from the base model experiment are reflected for the current work schedule.

Shift alternatives 3 and 4 had the shortest fabrication finish times with 92 calendar day, 2 days shorter than the current work schedule. Shift alternative 3 achieves this improved productivity even though the total duration of actual working periods is decreased from 18 hours (in current work schedule) to 16.5 hours a day. Although shift alternatives 4 increase weekly working hours by 25% and 50% (i.e., adds 10 and 20 hours a week, respectively, to the 40 standard working hours), interestingly the final fabrication time was the same as shift alternative 3, indicating that no overtime payment will be required if this work schedule is used. Finally the results indicate that shift alternative 3 uses the fewest working days (54 days); shift alternative 4 uses the most, respectively with 66 and 79 working days for 1 and 2 days of overtime. In current work schedule, fabrication is finished in 55 days (i.e., 1 days longer than improved alternative).

Achieving the improved result by shortening the length of working periods (i.e., shift alternative 3) indicates that the driving feedback loops in the current work schedule of the fabrication shop are Physical Burnout (R1) (Figure 3-1) and Mental Burnout (R2) (Figure 3-2) since reduction in the working periods and therefore workload status has resulted in increases in the productivity ratio and fabrication rate (Sections 2.1.1 and 2.1.2). The fabrication balancing feedback loop (B1) (Figures 1 and 2) is a non-driving loop, while decrease in working period and therefore in the workload status of the workers has shown in increase in fabrication rate.

3.7. Chapter Conclusion

I have introduced a new approach for improving final productivity in construction jobs by adjusting working hours. Worker productivity and accuracy vary according to the type and number of assigned tasks, length of work and rest periods, overtime, and time of the day, and these factors can be arranged for maximum productivity and accuracy. I first developed SD models for these factors using the literature. Next, I ran a sensitivity analysis based on different effective factors, and finally, I developed a hybrid model of system dynamics and discrete event simulation for a real structural steel fabrication shop, and explored different types of working shifts.

Although many construction managers may have noticed the potential effects of working hour arrangements on construction workers, the complex mechanism by which working hour arrangements affect workers' behavior usually prevents construction managers from considering adjusting working hours as an effective method for improving work productivity. However, the SD model of the working hour dynamics developed in this part of the research can provide a better understanding of the way this mechanism affects construction workers. For example, the model can be used to investigate how shortening the work periods can improve productivity in physically or mentally demanding tasks, how the contrary effects of overtime can be prevented, and how work start time can contribute to final productivity.

By developing a hybrid SD-DES model of working hour dynamics for an experimental case of a structural steel fabrication shop, I showed how tuning the working hours can result in productivity improvement in real construction jobs. This can stand as an example for construction managers on how they can improve their project's productivity by adjusting their project work schedule. A hybrid simulation model which captures the sequences of the project's tasks (e.g., from the schedule prepared for the project) and considers the working hour dynamics can be developed prior to the implementation phase and used to adjust the worker productivity ratio over the course of the project. Improving the productivity ratio, even by a few percent, decreases the final project's cost and can increase the final profit of the construction company considerably. This approach does not require special equipment or much additional investment from the company, since the main driver is changing the work schedule. Additionally, the dynamics of working hours, fatigue and productivity provide examples of how operational parts and non-operational, or context, parts of construction jobs can affect each other. In the future, as an expansion to the current model, dynamic models of other aspects of construction industry, such as skill level and changes in work assignment, can also be integrated with the model.

CHAPTER 4. Construction Workforce evolution Dynamics ⁴

4.1. Introduction

The project based nature of the construction industry makes it one of the most unpredictable industries for the amount of the work on hand and the number of the workers needed to do the job. Completion of each phase of a construction project may force the company to lay off workers; conversely, when new projects begin the company might need to recruit new workers. Furthermore, a down economy situation creates a conservative attitude in the market which encourages private investors to suspend expansions or new investment plans. This directly affects the number of construction projects in the market. The effect of a growing economy is, however, a greater tendency for expansion and new projects to meet the growing demands in the market, which introduces more construction projects to the market.

Additional cost spent for human resource departments is the result of this project-based nature of the construction industry, required to handle these workforce fluctuations. Annually, construction companies spend a considerable amount of money on their human resource departments to find the new workers required and to evaluate them. Experienced workers are usually hard to find and expect high wage levels; inexperienced workers typically have reduced performance and need

⁴ Parts of this chapter has been accepted for publication in 2011 CSCE Annual General Meeting and Conference

training courses to build up their capabilities. However, the money spent for hiring the new workers and the training provided for them during their service is lost when the workers leave. Any adjustments to human resource policies (e.g., hiring/firing policies, training policies, and overtime policies) can affect the fluctuations in the number and quality of the workers and contribute to the final cost. However, the complexity involved in the mechanism by which different effective factors (e.g., level of wage, level of skill and performance, and the workload) dynamically affect the final cost is a challenging issue which does not allow construction managers to track the results expected from any adjustments made to human resource policies. For example, the ultimate cost and benefit for hiring either fully experienced or inexperienced workers in a one-, two- or three-year project cannot be determined even by experienced construction managers using traditional project planning tools.

In this part of the research I have introduced and validated a new approach for improving human resource policies in the construction industry using the capabilities provided by system dynamics (SD) and discrete event simulation (DES) in a hybrid manner, where the SD parts track the dynamic changes in the organizational characteristics (e.g., hiring and firing policies, changes to the level of skill and wage in the workers, and overtime policies) following the causal feedback loops between organizational and operational level system variables and DES parts capture the operational details in the project over time (e.g., work dependencies, material flow, and stations status) since they incorporate

organizational updates modeled by SD parts. This chapter of the thesis presents another successful application of the way that the hybrid SD-DES modeling approach can contribute to solving complex decision making problems in the construction industry. This chapter includes 6 sections. Section 4-2 discusses the conceptual models of workforce dynamics and introduces and validates a customized conceptual model of workforce dynamics for construction projects. Section 4-3 explains different aspects of the construction worker dynamics and presents the development and validation steps of the inclusive model of construction worker dynamics. Section 4-4 highlights the main aspects of the prospective human resource hybrid SD-DES model that can be applied to different construction projects. The steps and the results of applying this hybrid model to a real structural steel fabrication case are explained in Section 4-5. The highlights and results have been summarized as a conclusion in Section 4-6.

4.2. Conceptual Model of the Workforce Dynamics

4.2.1. Workforce Dynamics in Literature

Tracking human resource skill evolution and workforce dynamics (i.e., changes in the quantity and quality of workers over time) within organizations for purposes of adjusting organizations' hiring/firing policies has been a major part of many SD models developed for analyzing organizational behaviours. For example, the SD dynamic model developed by Forrester (1958) helped a manufacturing company to recognize the origins of the employment disorder in the company;

since then, workforce dynamic models have been addressed in a variety of disciplines such as high-tech firms (Packer 1964), software companies (Abdel-Hamid and Madnick 1989), the service industry (Oliva 1996), academia (Sterman 2000), and the military (Wang 2005).

Because of the similarities between human resource dynamics in many industries, most previously developed dynamic models follow a general structure for developing their conceptual models. The human resource dynamic model presented by Sterman (2000) (hereafter called the experience chain) is one of the most cited models in the literature (e.g., Wang 2005; An et al. 2007; Koshio and Akiyama 2008). In the research I have selected this model, as a typical conceptual human resource dynamic model, and elaborated on its specifications; then I have tuned it to the work conditions within construction projects to develop a conceptual model of the workforce dynamics. Figure 4-1 depicts the stock and flow diagram representing the conceptual model of workforce dynamics.

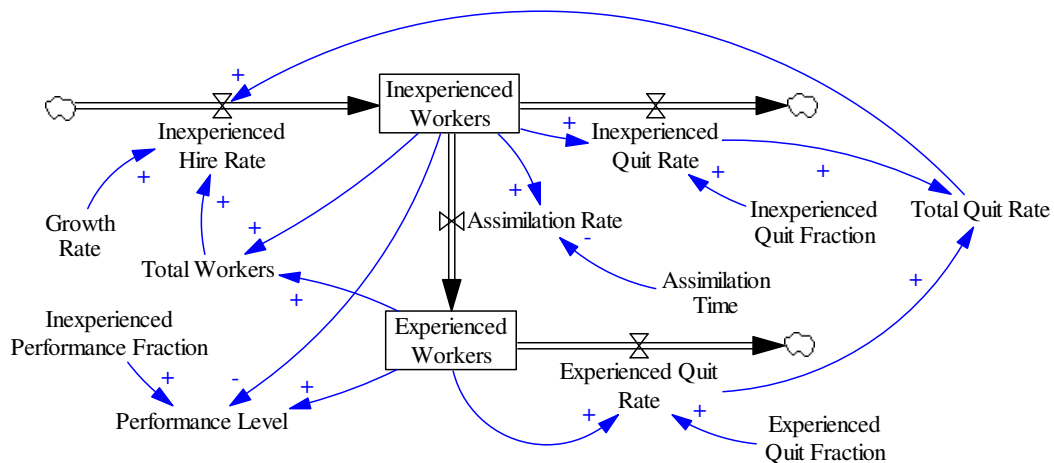


Figure 4-1. Conceptual dynamic model of experience chain (adapted from Sterman 2000)

Sterman (2000, pp. 490-493) has explained the experience chain model details and the attendant formulas; for a detailed explanation of the model, the reader is referred to Sterman (2000). Some aspects of the experience chain are highlighted below;

1. The model captures human resource dynamics in businesses with special expertise which is not available or is scarce in the job market; all new workers introduced to the organization are assumed to be inexperienced.
2. Types of expertise are either unique or easily exchangeable in the entire organization. For example large portion of assembly line workers in manufacturing companies can be moved from one station to the other, or bank tellers in service industry can be sent from one branch to the other.
3. The main decision making factor related to the number of workers is the growth rate estimated for the organization. Therefore, there are no major operational changes in business processes that can affect the rate of employment.

These aspects of Sterman's model are in fact the major points where workforce dynamics in construction projects diverges; the conceptual workforce dynamic model developed in this part of research for construction projects is formed based on these divergence points.

4.2.2. Conceptual Model of Workforce Dynamics in Construction

Unlike the scarce types of skill in the experience chain in the job market (Section 4.2.1, aspect 1), the skills required in construction projects, such as welding, painting, carpentering, fitting and crane operating, are usually well recognized and established in the construction job market. A variety of apprenticeship programs at different levels of expertise are offered across the country (e.g., EI Group 2011); the graduates from these programs can gain the skills that are used in construction projects. In addition, it is quite possible that experienced construction workers are laid off after completion of a construction project and hired for a new construction project. So, in the customized model for construction projects I am going to add another inflow to the Experienced Workers stock variable (presented as Experienced Hire Rate in Figure 4-2) directly originating from out of the system. The chance of hiring experienced workers is shown by Experienced Workers Fraction in the model (Figure 4-2).

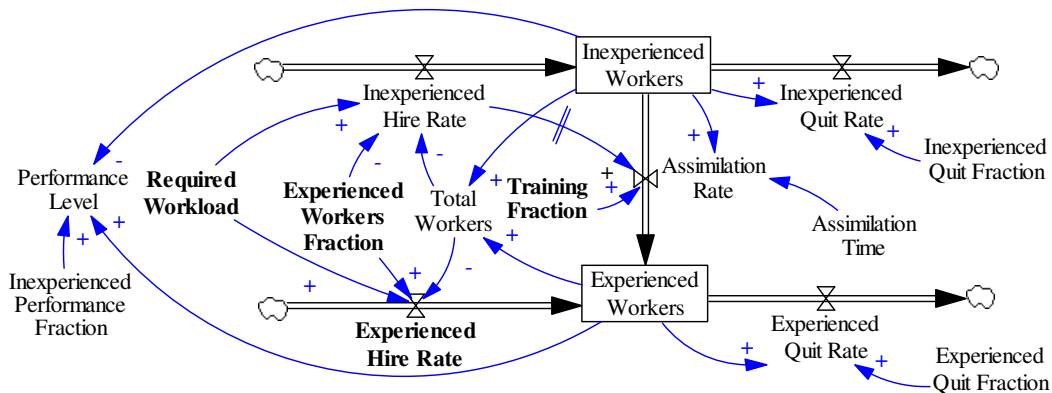


Figure 4-2. Conceptual model of construction workforce (new variables are bold)

Different types of skills in the construction are not very exchangeable, unlike what is presumed in the experience chain (Section 4.2.1, aspect 2). For example, construction managers cannot assign an idle and available welder to a painting job while he/she has not been trained for the job. As a result, modeling workforce dynamics within construction projects or a construction based organization in a single workforce dynamic model mixing all types of skill together might not be good practice. Instead, more relevant results are expected from a model formed out of a collection of separate workforce dynamic models, capturing workers with different types of skills.

It is true that the growth rate in the construction organizations is highly tied to the economic growth and it is rational for construction companies to keep their workers from one project to the next, as long as their new construction projects are within one region. But even with this perspective on construction projects, the growth rate in construction projects does not mean an equivalent growth in demand for all types of workers. From one construction project to another the portion of concrete, steel, wood or earth moving work might differ and correspondingly the amount of required skills will fluctuate. The interesting point here is that even within a project the distribution of required expertise might be different. For example, in a high-rise building project, the project starts with more demand for welders or concrete workers, while approaching the end of project, painters, carpenters, plumbers, and electricians are more in demand.

In mass production types of work, the required workforce growth rate depends on many effective factors such as labour cost, company revenue, supply and demand, and competitors' positions (Sullivan 2002) and is highly random. However, in construction projects, because of the limited and pre-defined scope of the work, the level of randomness drops sharply when estimating future workload and the required workforce over the course of projects. Usually, before a construction project reaches the stage in which construction workers can start working on the project, there is a long list of preparations that the project should go through. The design, engineering, procurement and mobilization processes of construction projects are usually time consuming, and give construction managers enough time to develop a reasonable construction plan and workload flow to the end of project. As such, rather than an estimation of the workforce growth rate, workload required (or number of workers required for the next interval) is the main indicator for determining the number of workers required in construction projects (Required Workload in Figure 4-2).

Time interval or adjustment time is another potential divergence in the model of construction workforce compared to the experience chain model. The workforce adjustment time – i.e., when the company makes decisions about hiring/firing its workers – in manufacturing based industries is considered on an annual or sometimes a semi-annual or monthly basis. Construction project progress is usually controlled on weekly basis; however, depending on the project type and the rate of progress, it might be controlled on daily, bi-weekly or monthly basis as

well. The difference between annual and weekly adjustment time creates some differences in the level of detail in the dynamic model. For example, although a delay of one week or two in the employment process for new workers is negligible in a model with a time adjustment of one year, this is an effective factor in a dynamic model with weekly time adjustments. Another example of the pace difference is the change in employment policies. It is unlikely that a manager loads an annual increase of 30% in workloads for the workers; he or she employs additional workers required for the job. However, due to the weekly nature of workload fluctuation in construction projects (compared to the mass-production work environment). For example, construction managers might decide to set 20 hours of overtime for their workers in addition to the 40 hours of standard working time as a response to a 50% workload increase on a weekly basis.

In reality, some part of the skills gained by construction workers come from apprenticeship programs which are usually held during a short period of time (e.g., several weeks). These apprenticeship programs generally increase workers' skill levels in a short period of time. These training programs are considered in the construction workforce dynamics model by introducing a training fraction which determines the extent of the effect of training on worker skill enhancement, compared to experiencing real construction work during the time of assimilation. Every inexperienced worker hired passes this training program in a short period of time during the assimilation period (e.g., at the end of the assimilation period).

Training Fraction in Figure 4-2 returns the share (as a percentage) that training has in enhancing the level of worker skill from inexperienced to experienced.

The following equations describe the changes applied in the Workforce model compared to the experience chain dynamics, presented by Sterman (2000).

- $$\text{ExperiencedWorkers}_t(\text{Worker}) = \text{ExperiencedWorkers}_{t-\Delta t}(\text{Worker}) + \left(\text{AssimilationRate}_{t-\Delta t} \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) + \text{ExperiencedHireRate}_{t-\Delta t} \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) - \text{ExperiencedQuitRate}_{t-\Delta t} \right) \times \text{WorkerTimeUnit} \times \Delta t$$

(4-1)

- $$\text{ExperiencedHireRate}_t \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) = \left(\frac{\text{RequiredWorkload}_{t+\Delta t}(\text{Worker} \cdot \text{Hour})}{\Delta t(\text{TimeUnit})} \times \frac{1(\text{Hour}) - \text{TotalWorkerstWorker} \Delta t(\text{TimeUnit})}{\text{ExperiencedWorkersFraction}} \right)$$

(4-2)

- $$\text{InexperiencedHireRate}_t \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) = \left(\frac{\text{RequiredWorkload}_{t+\Delta t}(\text{Worker} \cdot \text{Hour})}{\Delta t(\text{TimeUnit})} \times \frac{1(\text{Hour}) - \text{TotalWorkerstWorker} \Delta t(\text{TimeUnit})}{(1 - \text{ExperiencedWorkersFraction})} \right)$$

(4-3)

- $$\text{AssimilationRate}_t \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) = \text{Min} \left(\left(\frac{\text{InexperiencedWorkers}_t(\text{Worker})}{\text{AssimilationTime}(\text{TimeUnit})} \times (1 - \text{TrainingFraction}) + \text{InexperiencedHireRate}_{t-\text{AssimilationTime}} \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) \times \text{TrainingFraction} \right), \text{InexperiencedWorkers}_t \right)$$

(4-4)

t: Current time

Δt: Length of time interval

In equation 4-1, Experienced Hire Rate is the new flow variable added to the original equation for Experienced Workers in the experience chain. Equations 4-2 and 4-3 represent the new methods for calculating Hiring Rates as functions of the Required Workload. For calculating the Assimilation Rate in equation 4-4, first the portion of training related to the assimilation rate (or Training Fraction) has been deducted from the effect of the assimilation period. Then, the effect of the

crash training course has been involved in the equation by considering its effect on inexperienced workers hired at the end of the assimilation period. The minimum function added to this equation is to prevent the value of Inexperienced Workers from going below zero in the case that some inexperienced workers leave the job prior to the assimilation period.

There are different levels of certificates that construction workers in different types of skills can receive by attending apprenticeship programs and passing the tests. For example, levels 1 (flat welding), 2 (horizontal welding), 3 (vertical welding) and 4 (overhead welding) certification positions are granted to welders who were able to complete the related training and pass the qualification tests (refer to Go Welding.Org 2011 for more details on welding certifications). Such strict distinctions among different construction workers in one type of skill might justify introduction of several levels of experience to the dynamic model, instead of maintaining just two levels of experience (i.e., inexperienced and experienced). According to the body of research done since the 19th century in cognitive psychology (refer to Ritter and Schooler 2002 for a literature review), it is commonly agreed that the learning rate of workers is subject to changes over time. As such, assuming a flat rate for the progression from an inexperienced to an experienced worker (i.e., the assimilation rate) — especially when there are different levels of worker experience — over long period of time adds to the model inaccuracy. A multi-level dynamic model of the Workforce can be created by repeating structures similar to the two level experience model (as presented in

Figure 4-2) between every two consecutive experience models. As an example of a multi-level Workforce model, Figure 4-3 depicts a four-level Workforce dynamic.

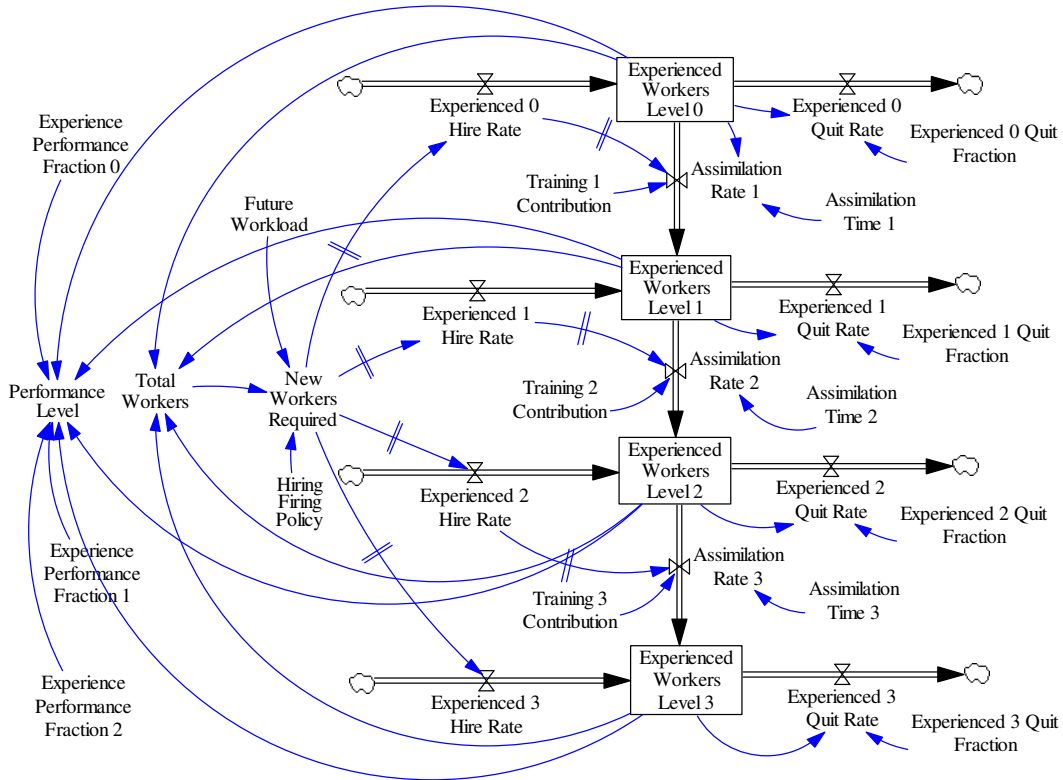


Figure 4-3. Four-level dynamic model for construction Workforce

4.2.3. Model Analysis

A fabricated example of the construction of a high-rise building is discussed in this section. The effects of the changes made in the experience chain structure for developing the construction workforce evolution model have been observed through simulation runs of this example. The high-rise building project is scheduled for three years or 150 weeks of construction. For the sake of

simplification I considered just two levels of expertise (i.e., experienced and inexperienced workers) for the project. The assimilation time for an inexperienced worker to become experienced is assumed to be one year, or 50 weeks. It is supposed that inexperienced workers participate in a crash training course at the end of the first year and will receive certification. This crash course will contribute to their experience improvement by 20%. It is assumed that experienced workers have a performance level of 100% and the performance level of inexperienced workers is 50%. It is also assumed that 70% of the available workers in the market are inexperienced and 30% are experienced; workloads within 50% over the standard working capacity are covered by overtime set during the week (e.g., 40 standard working hours and 20 hours of overtime) and no new workers are employed; workloads below 20% of the standard working capacity are also tolerated and no worker is laid off. The quit rate for experienced workers is 20% per year and the annual quit rate of inexperienced workers is 10%. Implementation details of the SD model have been presented in Appendix D.

Analyzing the effects of different modeling structures on the final results of the model simulation occurred in two steps. For the first step, I focused on differences between the results achieved from the construction workforce evolution dynamics model and the experience chain model. For the second step, I investigated the effects of skill distinction in the final model simulation results. Explanations of each step are found in the rest of the section.

1) Effects of Changes in the Experience Chain Model

Normally the control process of such construction projects is done on a weekly basis. The time interval in the model of construction workforce evolution in the project is one week. However, since growth rate is the main control factor in the experience chain and longer intervals are normally used when considering growth rates as a control factor in addition to the weekly intervals, monthly and half-yearly time intervals are also used for the experience chain model. At this step no distinction is supposed among different construction workers. The simulation results from different models are shown in Figure 4-4.

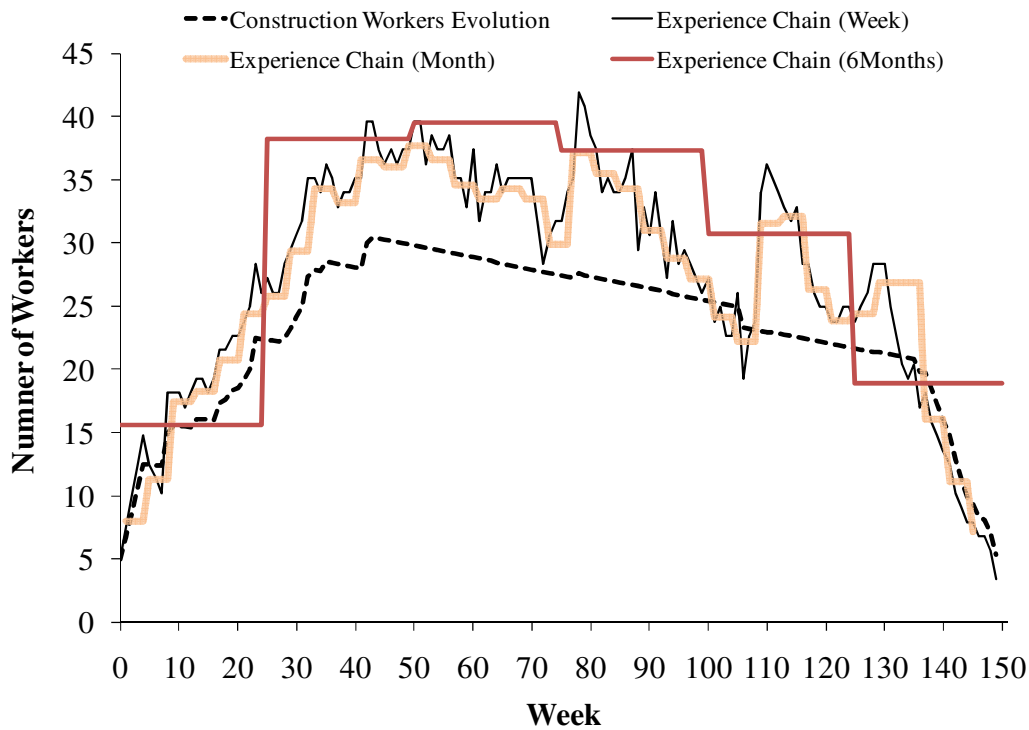


Figure 4-4. Workforce fluctuation in construction workforce evolution and experience chain dynamics

The use of the experience chain model at all time interval alternatives shows an inflated and highly fluctuating number of workers compared to the construction workforce evolution model. The total number of worker-weeks over the course of the project in the experience chain model with weekly time intervals becomes 4086; for monthly time intervals it becomes 4035, and for 6-monthly time intervals become 4510. It becomes 3435 worker-weeks in the construction evolution dynamic model. The number of workers in the experience chain model with weekly time intervals nearly follows all weekly fluctuations in the workload, which is not true in real construction work, where small fluctuations in workloads are handled by overtime or reduced working hours rather than continual hiring/firing. This behaviour from the experience chain model, which may not reflect construction work practices, is mainly because the experience chain is meant for modeling long term (e.g., annual) worker fluctuation and related policies, and it does not support organizational workforce policy with short term (e.g., weekly) periods of effect. Because of their larger time intervals, models with monthly and 6-monthly time intervals skip weekly fluctuations, but rapid fluctuations still are present from one time interval to the other. For example, the number of workers in the 6-monthly experience chain model during the first 6 months is 16; the number of workers jumps from 16 to 38 for the second 6 months of the project. Increasing the length of the time interval adds to the inaccuracy, and the model ignores fluctuation of the workload during a long period of time.

30% of the workers in the market are experienced workers and there is no inflow to the Experienced Workers stock variable in the experience chain model; new workers are assumed to be inexperienced. As such, the combination of experienced and inexperienced workers in the experience chain model is biased in favour of inexperienced workers, which returns a reduced percentage for experienced workers over time (Figure 4-5). It results in reduced performance and causes an inflated number of required workers during the course of the project (Figure 4-4). (Figure 4-4).

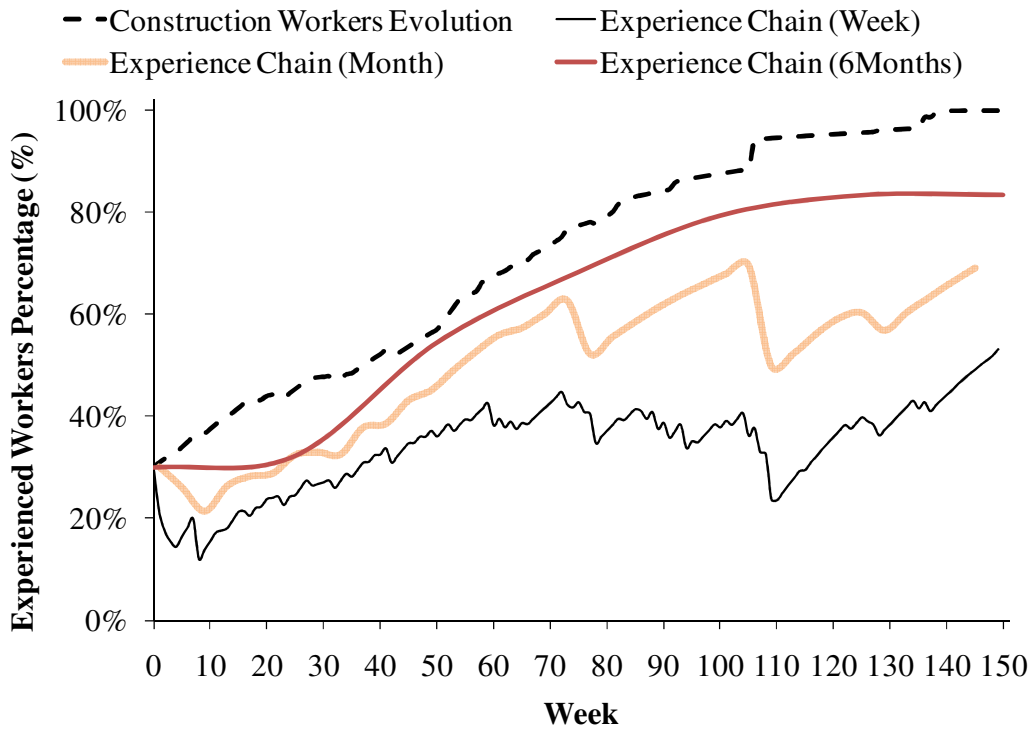


Figure 4-5. Fluctuation in percentage of experienced workers over the course of the project

2) *Skill Distinction*

In real construction work different types of skills are required at different periods of time to complete the project. To make the example more realistic, I divided the project into the four main work packages, including earthmoving, concrete work, steel work, and carpentry. In real construction work, the types of skills required for each of these packages are different. Therefore, the workers are not movable from one work package to the other. This is contrary to what is assumed in the experience chain model, where a workforce growth rate is assumed for an entire organization. Figure 4-6 presents the workloads required for different types of skills. The workload related to every skill is distinguished through a different colour.

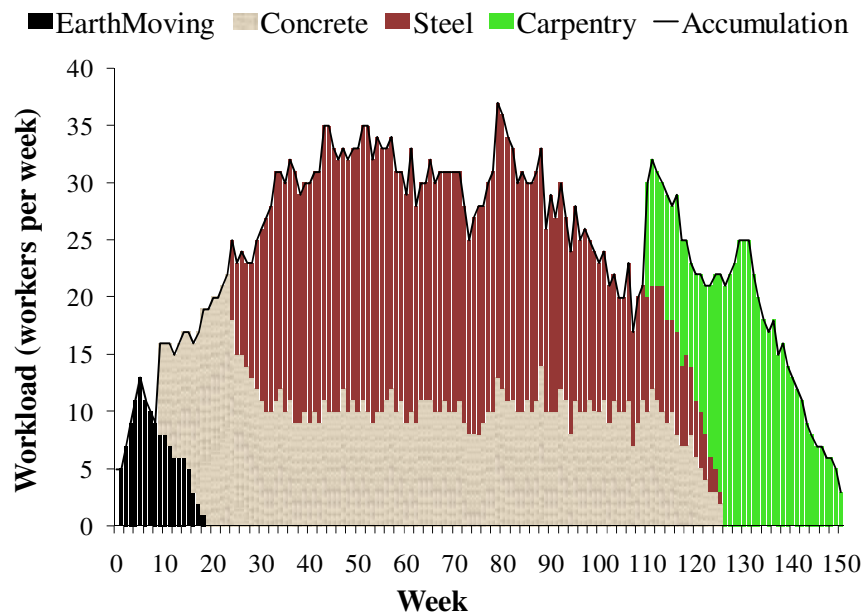


Figure 4-6. Weekly workload distribution for different work skills during the project

In this step I compare the accumulative results of the separate workforce evolution models developed for each working package and the model which just tracks the aggregated workload with no skill separation. Figure 4-7 presents the fluctuations in the numbers of workers achieved from these two models. As it was expected, making a distinction among different types of skills imposes higher level of fluctuations and boosts the number of workers required over the course of the project; it ends up with the total number of 3998 worker-weeks, which is 16% higher than the results from the aggregated workload model. This indicates a high potential discrepancy between models developed for aggregated workload and separate workload when different skills working in the project.

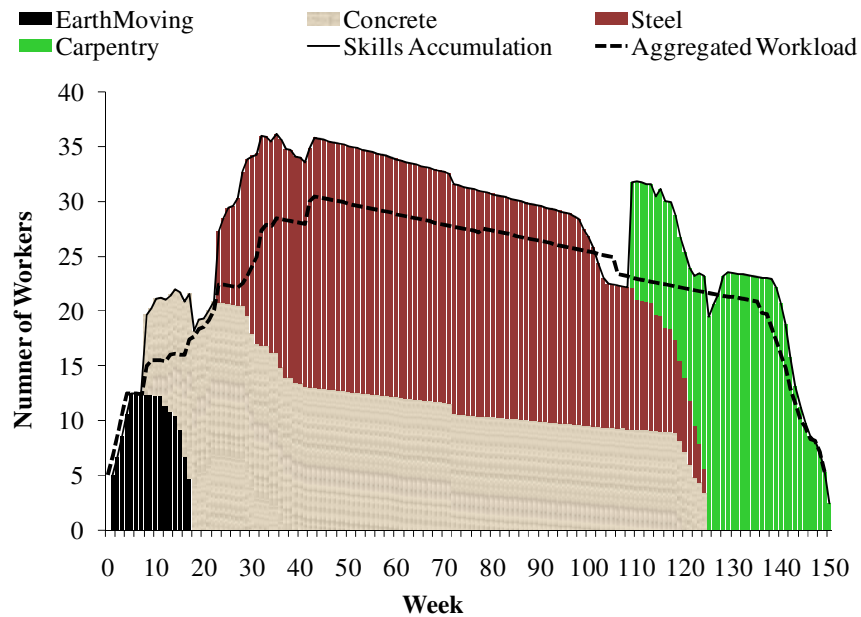


Figure 4-7. Effects of skill distinction on the number of workers

4.3. Inclusive Model of Construction Workforce Dynamics

The conceptual dynamic models discussed in the previous section are more focused on the skill evolution dynamics rather than tracing concurrent changes in skill and wage. A single focus on the skill evolution in the systems is suitable whenever only the final productivity of the workers is most important, not the total money spent for human resources. However, for an inclusive view of workforce dynamics in construction projects I have covered both productivity and monetary aspects of human resources in the model, since both are important factors to be considered concurrently in the decision making process for many construction projects.

4.3.1. Core Dynamic Model of Workforce evolution

The main focus of the customized conceptual dynamic model of workforce evolution in construction projects presented in Section 4.2 is the skill evolution dynamics. In this part of the research I have developed the core dynamic model of workforce evolution by maintaining the capabilities of the conceptual dynamic model and adding new capabilities to the model for tracing dynamic changes in the number and wage levels of the workers. Figure 4-8 presents the core workforce evolution model in the form of a stock and flow diagram for a two-level skill and wage organization.

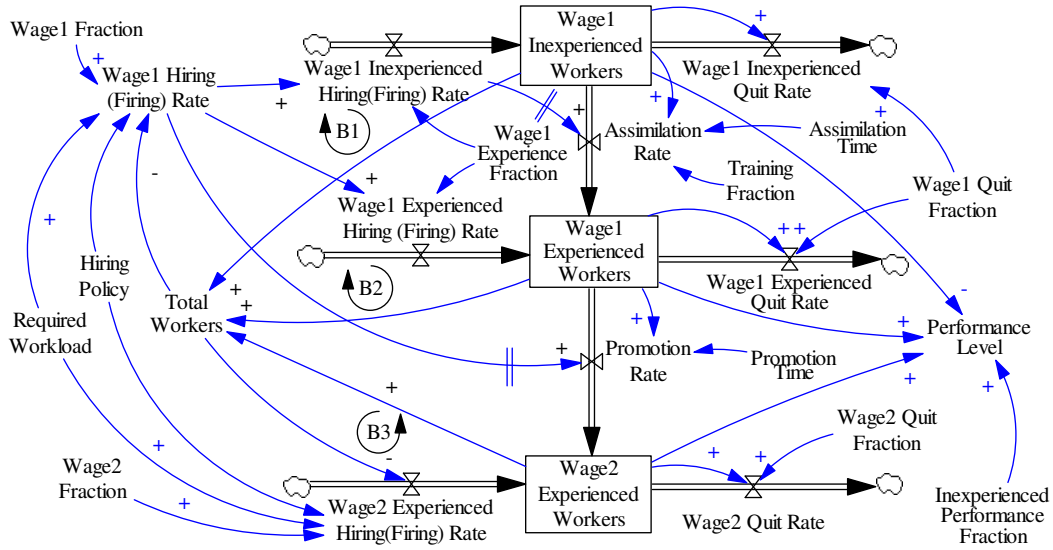


Figure 4-8. Core dynamic model of the workforce evolution in a two-level skill/wage organization

The main modification in this model, compared to the conceptual model of workforce dynamics in construction (Figure 4-2), is the interim stock variable introduced between inexperienced and experienced workers, called Wage1 Experienced Workers. This stock variable accumulates the experienced portion of the workers who still receive low level wages, i.e., the same as inexperienced workers. By adding this middle stock variable, the meaning and the name of two other stock variables also is changed; Wage1 Inexperienced Workers (previously called Inexperienced Workers) refers to the inexperienced portion of workers who receive low level wages, and Wage2 Experienced Workers (previously called Experienced Workers) present the skilled workers who receive higher level wages. The Promotion Rate is also a new flow variable introduced to the core model (Figure 4-8) and represents the rate of workers promoted from the low wage level (or Wage1 Experienced Workers) to the high wage level (or Wage1

Experienced Workers). These modifications in the model enable tracking both the level of skill (by comparing the summation of Wage1 Experienced Workers and Wage2 Experienced Workers with the value of Wage1 Inexperienced Workers) and the number of workers at different levels of wages (summation of Wage1 Inexperienced Workers and Wage1 Experienced Workers represents the number of workers with lower wage levels and the value of Wage2 Experienced Workers represents the number of workers with higher wage levels). Each of the stock variables mentioned in the core model forms a balancing causal feedback loop; i.e., an increase in the number of any group of workers causes an increase to the Total (number of) Workers and reduces the Hiring (Firing) Rate which, later on, slows down the increase in that group of workers.

As a result of the changes made to the construction workforce evolution model, the set of equations describing the workforce evolution model, discussed in Section 4.2, is subject to some changes. Four equations which describe the major changes in the model developed are presented here:

- $$\begin{aligned} & \text{Wage1ExperiencedWorkers}_t(\text{Worker}) = \\ & \text{Wage1ExperiencedWorkers}_{t-\Delta t}(\text{Worker}) + \text{AssimilationRate}_{t-\Delta t} \frac{\text{Worker}}{\text{TimeUnit}} + \text{Wage1ExperiencedHiringRate}_{t-\Delta t} \frac{\text{Worker}}{\text{TimeUnit}} - \text{PromotionRate}_{t-\Delta t} \frac{\text{Worker}}{\text{TimeUnit}} - \text{Wage1ExperiencedQuitRate}_{t-\Delta t} \frac{\text{Worker}}{\text{TimeUnit}} \times \Delta t(\text{TimeUnit}) \end{aligned} \quad (4-5)$$

- $$\begin{aligned} & \text{Wage2ExperiencedWorkers}_t(\text{Worker}) = \\ & \text{Wage2ExperiencedWorkers}_{t-\Delta t}(\text{Worker}) + \left(\text{PromotionRate}_{t-\Delta t} \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) + \text{ExperiencedHiringRate}_{t-\Delta t} \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) - \text{Wage2QuitRate}_{t-\Delta t} \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) \right) \times \Delta t(\text{TimeUnit}) \end{aligned} \quad (4-6)$$

- $$\text{PromotionRate}_t \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) = \frac{\text{Min}(\text{InexperiencedHiringRate}_{t-\text{PromotionTime}} \left(\frac{\text{Worker}}{\text{TimeUnit}} \right), \text{Wage1ExperiencedWorkers}_t(\text{Worker}))}{\Delta t(\text{TimeUnit})}$$
(4-7)

- $$\text{Wage1ExperiencedHiringRate} \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) = \text{Wage1HiringRate} \left(\frac{\text{Worker}}{\text{TimeUnit}} \right) \times \text{Wage1ExperienceFraction}(\)$$
(4-8)

t: Current time (e.g., in week)

Δt: Length of time interval (e.g., one week)

Equations 4-5 and 4-6 present the stock and flow relations. Equation 4-7 refers to the prevalent method of wage increase or promotion which is enforced after working for a specific period of the Promotion Time (e.g., one year). The minimum function here is for accounting for the number of the workers who have been fired or have quit the job before reaching their Promotion Time, which may cause the number of Wage1 Experienced Workers accumulated to become less than the original number of workers hired; the minimum function prevents the number of Wage1 Experienced Workers from going below zero. Equation 4-8 calculates the experienced portion of the new workers hired at the low wage level by multiplying the Experience Fraction of the new workers by the Wage1 Hiring Rate. The model presented in Figure 4-8 is for tracking workforce evolution in two-level (skill and wage) jobs where Assimilation Time is less than or equal to the Promotion Time. If more than two levels of skill or wage are distinguishable or the length of Assimilation Time is longer than the Promotion Time, the model should be extended to a multi-level workforce evolution model which is formed by repeating a similar structure of two-level models on every other level.

The organizational policy for hiring/firing workers has been considered in the model through Wage1Fraction, Wage2Fraction and Hiring Policy parameters. Wage1Fraction and Wage2Fraction show the percentage of new workers hired, respectively considered as level 1 and level 2 of the wage. These parameters may be set either as a result of number of available workers in the job market or the company's policy to only hire workers at specific wage levels. The Hiring Policy parameter reflects the margin at which the project manager either decides to hire required or fire redundant workers. Hiring Margin (HM) is the point where, if the Required Workload goes beyond it, the project manager starts hiring new workers for the project. When the Required Workload goes below Firing Margin (FM), the project manager starts firing redundant workers. These two parameters are determined as the percentage of the total work capacity of current workers. For example, an HM of 150% indicates that if the workload goes beyond 150% of the current workers' capacity, the manager will begin hiring new workers to fill this gap. As such, the value of FM can have a range between 0 to 100%, and HM can have values above 100%.

4.3.2. Workload Dynamics

Figure 4-9 presents the supporting dynamics of the Required Workload which is an exogenous variable to the workforce evolution core dynamic model (Figure 4-8). For the sake of simplification, I avoided repeating non-contributing elements of the core dynamic in the new causal relations in Figure 4-9. Six main causal

feedback loops are distinguishable in this figure as the causal effects of the Required Workload on the number of workers with different levels of skill and wage. The first group of feedback loops (includes three feedback loops) form as a result of the changes in worker distribution and its consequent effects on the level of performance, and the second group of feedback loops (includes three feedback loops) initiate because of changes made in the number of workers. Both the performance level and the number of workers affect the Work Progress Rate and the total Work Done, which consequently adjust the Required Workload and finally Hiring Rates of different types of workers. All recognized feedback loops in the model are balancing, except one feedback loop. The only reinforcing feedback loop — where an increase in a variable on the loop initiates an increase in that variable in the future — out of the six feedback loops of the model is formed as a result of the negative relation between the Wage1 Inexperienced Workers and the Performance Level.

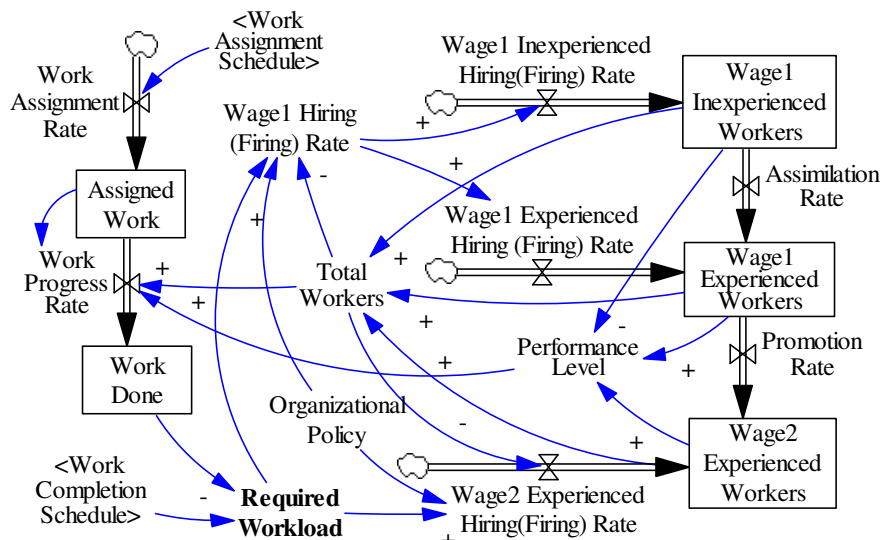


Figure 4-9. Supporting dynamics on the required workload

The value of exogenous variables inside angle brackets (< >) in the model (Figure 4-9), i.e., Work Assignment Schedule and Work Completion Schedule, are read from data tables calculated from the project schedule. The equations describing the new introduced causal relations are formulated as shown below. Because of their similar structure, equations of the new stocks and flows are not presented here:

$$\text{RequiredWorkload}_{t+\Delta t}(\text{Worker} \cdot \text{Hour}) = \text{WorkCompletionSchedule}_{t+\Delta t}(\text{Worker} \cdot \text{Hour}) - \text{WorkDone}_t(\text{Worker} \cdot \text{Hour}) \quad (4-9)$$

$$\text{WorkProgressRate}_t \left(\frac{\text{Worker} \cdot \text{Hour}}{\text{TimeUnit}} \right) = \text{Min} \left(\frac{\text{TotalWorkers}_t(\text{Worker}) \times \text{PerformanceLevel}_t(\%)}{\Delta t(\text{TimeUnit})}, \frac{\text{AssignedWork}_t(\text{Worker} \cdot \text{Hour})}{\Delta t(\text{TimeUnit})} \right) \quad (4-10)$$

Equation 4-9 calculates the current Required Workload as a balance between actual Work Done and the expected Work Completion Schedule. In Equation 4-10, Current Work Progress Rate is calculated by multiplying the Total (number of) Workers by the Performance Level, divided by one time unit, subject to the Assigned Work left.

4.3.3. Overtime Dynamics

Setting overtime is a common practice in construction projects as a response to working capacity shortfall or schedule delays. It is a rival solution to hiring new workers when the project is behind schedule. Use of overtime is usually considered for schedule delays when the capacity shortage is expected to be temporary or insignificant; hiring new workers for the project happens when a

considerable or a permanent capacity shortage occurs in the project. However, depending on organizational hiring policies, decision rules set for shifting from setting overtime to hiring new workers may vary from one company to another, or even in different projects within a company.

The dynamic models of project working hours and the effect of overtime on the project performance level have been previously discussed in the literature (e.g., Homer 1985; Sterman 2000, pp. 577-583). In this part of the research the dynamic model of overtime is built on the previously developed overtime model customized for construction (as discussed in Section 3.2.3 of Chapter 3). Figure 4-10 presents the stock and flow diagram of the adapted overtime model. According to the diagram, the Required Overtime is calculated based on the Required Workload, Total (number of) Workers, and Organizational (Overtime) Policy. The control parameter for overtime policy (OP) here is either using (OP = 1) or not using (OP = 0) overtime as a response to the over-capacity workload. Setting up overtime and hiring new workers (Section 4.3.1) are rival policies for responding to over-capacity workloads. In fact, the hiring margin (HM) determines the scope of the overtime and hiring new workers actions. For example, HM = 150% indicates that the organization's policy on responding to over-capacity workloads up to 150% is setting overtime, and above this margin is hiring new workers. In this research I assume that every company has at least one active policy to respond to over-capacity workloads (i.e., overtime or hiring policy). This assumption puts a restriction on the maximum value that HM can get

based on the overtime maximum set in the organization. For example, if the standard working hours are 40 hours a week and the maximum overtime is 20 hours a week (50% of the standard time), the maximum possible value for HM will be 150%, since overtime cannot go beyond this point and hiring new workers is the only possible action. Because of the complications involved in hiring new and firing redundant workers, normally project managers prefer to set overtime as much as they can and to postpone hiring new workers.

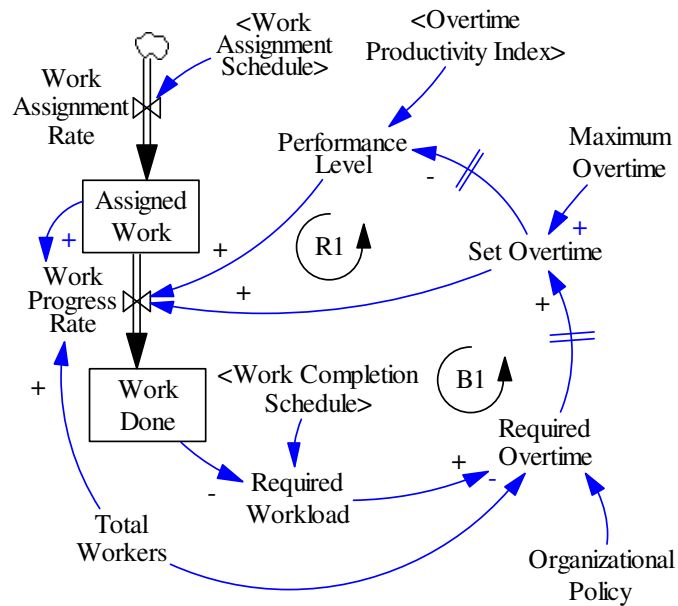


Figure 4-10. Overtime dynamics in construction projects (modified from Figure 3.4 in Chapter 3)

The direct effect of overtime is increased working hours and consequently a higher Work Progress Rate. However, it also has an indirect adverse effect on the Work Progress Rate by reducing the Performance Level. As the result of causal relations in the overtime dynamic model, one balancing loop and one reinforcing

loop are formed. More explanation of the model and the model's describing equations are presented in Section 3.2.3 of Chapter 3.

4.3.4. Dynamic Data Collecting

One reason for developing an SD model of the workforce evolution is tracing the effects of different organizational policies on the completion cost of construction projects. This provides a quantitative tool for evaluating potential organizational policies prior to their implementation. To provide such capability in the model, a dynamic data collecting mechanism has been foreseen in the model. The total money paid to the workers, extra costs for covering workload gaps, training expenses and hiring (firing) expenses are the cost items that have been considered in the dynamic model (Figure 4-11). Again, here, only contributing variables from other parts of the model have been presented in the model diagram.

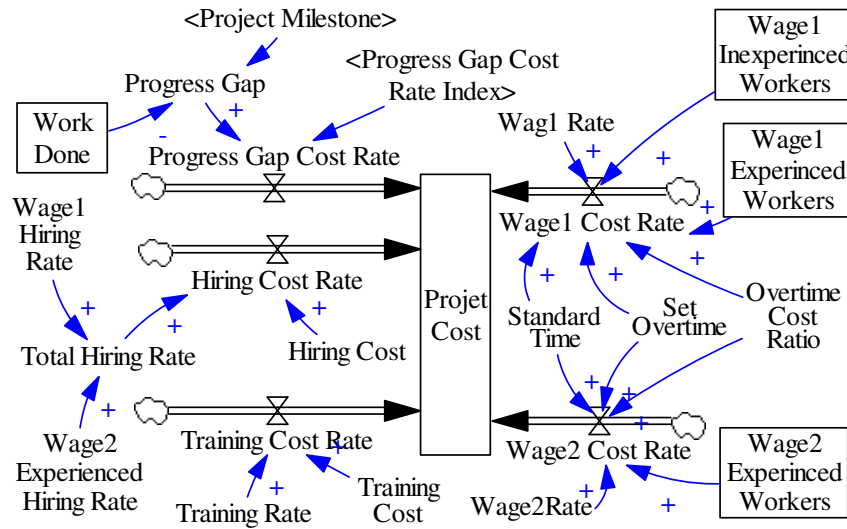


Figure 4-11. Built in mechanism for collecting the cost information

There is no causal feedback loop formed in this part of the model, since this model only collects data received from other parts of the model and presents them in an aggregated manner. The equations describing this part of the model are as following (some apparent equations have not been presented.)

- $$\text{ProgressGap}_t(\text{Boolean}) = \begin{cases} 1; & \text{if } \text{ProjectMilestone}_t(\text{Worker.Hour}) > \text{WorkDone}_t(\text{Worker.Hour}) \\ 0; & \text{else} \end{cases} \quad (4-11)$$

- $$\text{ProgressGapCostRate}_t\left(\frac{\text{Dollar}}{\text{TimeUnit}}\right) = \text{ProgressGap}_t(\text{Boolean}) \times \text{ProgressGapCostRateIndex}_t\left(\frac{\text{Dollar}}{\text{TimeUnit}}\right) \quad (4-12)$$

- $$\text{HiringCostRate}_t\left(\frac{\text{Dollar}}{\text{TimeUnit}}\right) = \text{TotalHiringRate}_t\left(\frac{\text{Worker}}{\text{TimeUnit}}\right) \times \text{HiringCost}\left(\frac{\text{Dollar}}{\text{Worker}}\right) \quad (4-13)$$

- $$\text{TrainingCostRate}_t\left(\frac{\text{Dollar}}{\text{TimeUnit}}\right) = \text{TotalTrainingRate}_t\left(\frac{\text{Worker}}{\text{TimeUnit}}\right) \times \text{TrainingCost}\left(\frac{\text{Dollar}}{\text{Worker}}\right) \quad (4-14)$$

- $$\text{Wage1CostRate}_t\left(\frac{\text{Dollar}}{\text{TimeUnit}}\right) = \left(\text{Wage1InexperiencedWorkers}_t(\text{Worker}) + \text{Wage1ExperiencedWorkerstWorker} \times \left(1 + \frac{\text{SetOvertime}_t\left(\frac{\text{Hour}}{\text{Week}}\right) \times \text{OvertimeCostRatio}(\cdot)}{\text{StandardTime}\left(\frac{\text{Hour}}{\text{Week}}\right)} \right) \right) \times \text{Wage1Rate}\left(\frac{\text{Dollar}}{\text{Worker.TimeUnit}}\right) \quad (4-15)$$

- $$\text{Wage2CostRate}_t\left(\frac{\text{Dollar}}{\text{TimeUnit}}\right) = \text{Wage2ExperiencedWorkers}_t(\text{Worker}) \times \left(1 + \frac{\text{SetOvertime}_t\left(\frac{\text{Hour}}{\text{Week}}\right) \times \text{OvertimeCostRatio}(\cdot)}{\text{StandardTime}\left(\frac{\text{Hour}}{\text{Week}}\right)} \right) \times \text{Wage2Rate}\left(\frac{\text{Dollar}}{\text{Worker.TimeUnit}}\right) \quad (4-16)$$

As presented in Equation 4-11, Progress Gap is a Boolean (value 0 or 1) variable determining whether the current progress achieved in the project meets the progress expected based on the most recent Project Milestone. Progress Gap results in an increase in the Project Cost by the rate Progress Gap Cost Rate, read from the project milestone cost rate index (Equation 4-12). Hiring Cost Rate

(Equation 4-13) and Training Cost Rate (Equation 4-14) are set respectively based on the Total Hiring Rate and Total Training Rate which occurred during the time period, multiplied by their related cost rate. Finally, Wage1 Cost Rate (Equation 4-15) and Wage2 Cost Rate (Equation 4-16) are calculated based on the total number of workers in each wage category multiplied by their related Wage Rate considering their overtime pay. The overtime cost portion of the money is considered as a coefficient in the equations; it is calculated by comparing the value of the Set Overtime and the Standard (working) Time multiplied by the Overtime Cost Ratio.

4.3.5. Model Validity

A series of validity tests have been run for the model, following the methods suggested for validating dynamic models by Sterman (2000, pp. 843-858). Through these methods I have investigated the model's usefulness and revealed its capabilities, limitations, and flaws to prospective model users. Table 4-1 summarizes the results of validating tests run for the model. Most of the structure validation tests have been applied during the model development as presented in Sections 4.2 and 4.3. However, for the behavioral tests, a simple example of a fabricated construction project was modeled. Table 4-2 presents the base values assumed for the model's exogenous variables and constant parameters. Some examples of the behavioral test run for the model are discussed in the following.

Table 4-1. Summary of applied validation tests

Test	Purpose of the Test	Summary of Test Process
Boundary adequacy and Structure assessment	To ensure that the model contains the important concepts and is consistent with the descriptive knowledge of construction projects.	The dynamic model is based on established dynamic models and has been adjusted step by step in accordance with the main related concepts in construction.
Dimensional consistency and Parameter assessment	To test the meaning and dimension of the variables participated in the equations.	The variables have been adopted from real concepts in construction. The consistency of the variable units has been checked in every equation.
Extreme conditions	To investigate reasonability of the model behavior in response to extreme values of the model parameters.	Extreme values of the model variables have been tested and the adequacy of the model behavior has been tested. Some examples have been presented in this section.
Integration error	To check the effects of different time steps on the model results.	Different time steps were tested in the model. The result of this test has been presented in this section.
Behavior reproduction	To test whether the model can generate behaviors of real construction projects.	The model has been applied in a real construction case and the behavior reproduction test has been conducted for the case (Section 4.5).
Behavior anomaly	To examine the importance of the relationships created in different parts of the model by observing anomalous behavior of the system with the absence of those relationships.	The model behavior has been examined in response to removal of some relationships in the model. An example of this model test has been presented in this section.
Sensitivity analysis	To analyze the reasonability of the results achieved in the model by varying uncertain or adjustable assumptions.	Sensitivity analysis has been conducted on the model parameters. An example has been presented in this section.

Table 4-2. Base values considered for the test model's exogenous variables and constant parameters

Constant/ Exogenous Variable	Value	Description
Hiring Cost	5000 \$/Worker	
Training Cost	2000 \$/Worker	
Wage1 Rate	700 \$/Worker	
Wage2 Rate	900 \$/Worker	
Standard Time	40 h/Week	
Overtime Cost Ratio	1.5	
Max Overtime	20 h/Week	
Project Scheduled Duration	101 Week	
Total Scheduled Work	348002 Worker.Week	
Project Milestone 1	73997 Worker.Week	Week 20 Deadline
Project Milestone 2	198997 Worker.Week	Week 45 Deadline
Project Milestone 3	315999 Worker.Week	Week 70 Deadline
Project Milestone 4	348002 Worker.Week	Week 101 Deadline
Progress Gap Cost Rate Index	100000 \$/Week	
Wage1 Experience Fraction	0.3	
Assimilation Time	50 Week	
Promotion Time	50 Week	
Training Fraction	0.2	
Wage1 Quit Fraction	10% Annually	50 weeks per year
Wage2 Quit Fraction	20% Annually	50 weeks per year

The base model was tested for integration errors by applying a range of time steps starting as small as 1×10^{-4} weeks up to a time step of 0.1 weeks. The final results showed small deviations by increasing the time step from 1×10^{-4} to 0.01; for example, the total cost fluctuated from \$11.041 million to \$11.027 million (i.e., a deviation of -0.1%). However, the deviations rapidly increased after going beyond this point. A time step of 0.05 resulted in a total cost of \$10.962 million (i.e., a deviation of -0.7%), and a time step of 0.1 resulted in a total cost of \$10.599 million (i.e., a deviation of -4%). Therefore, to prevent integration errors, time steps chosen for the model simulation are kept below 0.01.

One example of a model behavior test under extreme conditions was the investigation of extreme policies regarding hiring new workers and overtime

policies. Two extreme conditions set for the hiring policy were: 1) only hiring inexperienced workers with Wage1 salary, and 2) only hiring experienced workers with Wage2 salary. The extreme condition for overtime was 1) no overtime set and, 2) workers always do the maximum overtime. The simulation results are presented in Table 4-3.

Table 4-3. Simulation results for employment and overtime extreme policies

Extreme Conditions	Cost (M\$)					Avg. Performance	
	Gap	Hiring	Training	Overtime	Standard Time	Total	Level (%)
Hire Inexperienced/ No Overtime	0.03	1.05	0.31	0.00	8.07	9.46	85%
Hire Inexperienced/ Full Overtime	0.08	0.78	0.20	5.00	6.68	12.73	68%
Hire Experienced/ No Overtime	0.00	0.88	0.00	0.00	7.83	8.71	100%
Hire Experienced/ Full Overtime	0.08	0.68	0.00	4.89	6.52	12.16	80%

Four models were created as the result of the combination of these two policies. All results achieved in the model were in accordance with the direction of the changes. The hiring costs when the policy states only inexperienced workers are to be hired are higher due to the increased number of workers required to account for the reduced performance level of inexperienced workers. There is no training cost required when the policy dictates only experienced workers are to be hired. No overtime is charged with the no-overtime policy; with full overtime, there is a decreased gap cost. The changes in the average performance level followed the direction of the changes in the policy; e.g., the minimum performance level of 68% achieved given the scenario of hiring inexperienced workers with full time

set overtime, and the 100% performance level happened when the policy was set to hiring only experienced workers and setting no overtime.

The anomaly test was another type of behavioral test run for the model, in this case by cutting the workload feedback loop (Section 4.3.2) and setting the value of the workload required to a constant value. The value of workload was first set to 3450 worker hours a week — the average of the weekly workload in the project. This anomaly in the model resulted in 40 weeks of project delay and doubled the total cost of the project compared to the base model, reaching \$22.6 million. The lowest cost achieved by adjusting the constant value of the workload in this anomaly test was \$14.7 million, when the value of the workload was set to 5500 worker hours a week. The cost achieved at this point was still 33% higher than the base model. This test affirms the importance of the workload feedback loop in the project estimation.

The model behavior was also tested via running sensitivity analysis on some uncertain model parameters. For example, in the base model there are no constraints set on the construction company's hiring capacity for new workers. However, the number of new workers is a constraint in many construction projects, since construction companies' project management groups and human resource departments have limited resources for going through the employment process, such as background checks, evaluating the certifications, and running interviews. The result of this sensitivity analysis is summarized in Figure 4-12.

The total cost of the project when the maximum possible number of new workers hired in one period varies from 1 to an infinite value ranges from \$10.0 to \$22.8 million.

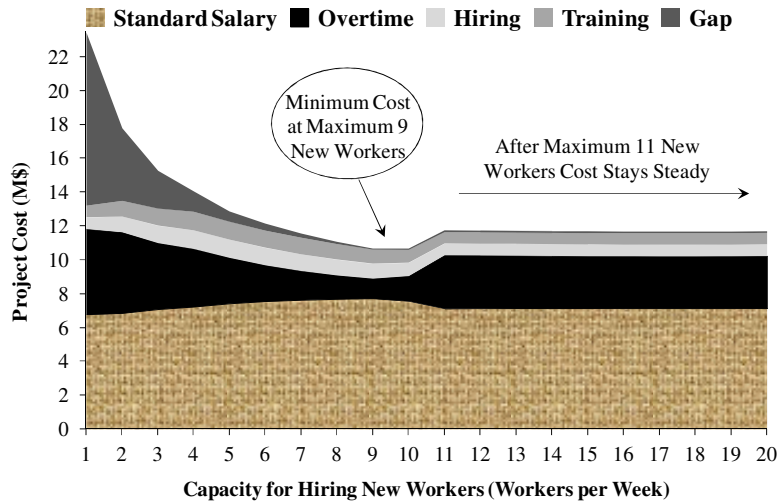


Figure 4-12. Sensitivity analysis on the maximum number of new workers

As was expected, the majority of this deviation can be attributed to the progress gap cost and overtime cost (Figure 4-11). Increased overtime cost is a direct response of recruitment constraints. However, with a severe recruitment limitation (e.g., 1 or 2 maximum new workers per week), increased overtime still cannot prevent the project from missing milestones and paying gap costs. An interesting result from the sensitivity analysis was seeing the minimum cost at the point of a maximum of 9 new workers per week, regardless of easing constraints after this point. The gap cost at this point approaches zero, which demonstrates that the constraint on the number of new workers does not cause a delay in the project anymore. Regardless of the limitation on the maximum number of new workers, the average number of workers is at its highest level at this point. This requires

less overtime for the project and minimizes the overtime cost paid. The trend of the diagram in Figure 4-12 highlights the adverse effects of overtime on the final cost of the project. It shows that at this project parameter setting, compared to setting overtime for current workers of the project, hiring more new workers is preferred. As was expected, when the hiring cost of new workers was gradually increased, the policy preference moved from hiring new workers to setting overtime.

4.4. Hybrid Model of Construction Worker Dynamics

Construction projects are complex combinations of interdependent tasks done by a range of workers with different types of skills. Estimating the workload required in the project during each time period is not something that can be done properly with only the project plan, since deviations to the scheduled progress as the result of dynamic changes in effective factors are quite common in construction projects. Rather, a tool that can dynamically trace the operational details of construction projects (e.g., the physical specification of the work environment and the flow of material, workers, and equipment) can enhance the workload estimation process. I propose the use of discrete event simulation (DES) as a tool to capture these operational details and communicate with the SD part of the model. This approach to system modeling, which accommodates both SD and DES and their in-between interactions, is called a hybrid SD-DES modeling. Detailed explanations of different aspects of the hybrid SD-DES modeling tool

are found in Chapter 2 of the thesis. When modeling workforce evolution dynamics with a hybrid approach, a DES model can be used for tracking changes in the work assigned, work progress rate, and work done at every working station (i.e., a group of workers with similar types of skills doing similar types of tasks) of the project. The DES part of the model sends the updated work done to the SD part of the model where it is used for calculating the workload required and adjusting the number of workers, overtime, and level of performance. The updated number of workers, overtime, and performance level are sent from the SD part to the DES part. One DES model can be used for capturing different aspects of the operational part of the project. However, since different types of skills are not usually exchangeable in a given project, I suggest that separate SD models should be developed for every group of exchangeable workers (i.e., workers with similar type of skills).

4.5. Case Study

To investigate the potential effects that different components of human resource policies can have on construction companies, a hybrid SD-DES model of workforce evolution dynamics has been developed for a collaborating construction company fabricating structural steel for multiple construction projects. First, a hybrid model for the current worker policies (the base model) has been developed and validated over a period of two and a half years, and then deviations to the workers policies were made and their effects on the final cost

were determined. The best alternative human resource policy has been suggested, based on the minimum cost achieved.

4.5.1. Case Specification

Steel fabrication projects are a type of off-site construction project which constitute a major component of many construction projects, to reduce the portion of on-site construction jobs and improve the final productivity of construction projects (Eastman and Sacks 2008). Shippable structural steel elements (or pieces) are the products of the fabrication shop which are mainly fabricated through four operations, including cutting, fitting, welding, and painting, done on standard structural steel materials such as beams, angles and plates. The fabrication shop in this case consists of 3 types of specialized shops; 1) the main cutting shop; 2) the fitting-welding shops which contain three main shops, shop AB for heavy and time consuming pieces, shop C for non-time consuming pieces, and shop D for light but time consuming pieces; and 3) the painting shop. The normal sequence of fabrication is cutting, fitting, welding, and painting; however, some pieces might skip some operations or have different orders.

Workers in every specialized shop are divided in 6 categories, including Apprentice 1, Apprentice 2, Apprentice 3, Apprentice 4, Journeyman 1 and Journeyman 2 with different levels of wages including 41 \$/h for Apprentice 1, 45 \$/h for Apprentice 2, 50 \$/h for Apprentice 3, 54 \$/h for Apprentice 4, 60 \$/h for Journeyman 1 and 65 \$/h for Journeyman 2. To be considered as a specific level

of apprentice in a specialized shop, the worker should have the related level of apprenticeship certificate and at least one year of experience working at every lower level of apprenticeship certificate. The apprenticeship certificate can be received by attending the related crash program and passing the certificate test. There is no certificate or special training for the Journeyman 1 and Journeyman 2 workers, just the work experience. The highest level of wage that is given to a new employed worker, even with highest level of skills and certificates, is Apprentice 4; Journeyman 1 and Journeyman 2 levels are only achievable by gaining experience in the company. According to interviews of the shop manager and foremen, the biggest difference in performance level happens during the first four levels of apprenticeship, from 70% to 100%, and after that changes in performance are small. The fabrication shop works in two shifts, a day shift and a night shift. To minimize hiring and firing fluctuations, the company's decision on hiring new or firing current workers is based on the workload average for the next 3 months, which is checked on a weekly basis. Average workload of more than 50% of the shop's capacity results in new worker employment. The decision on overtime, however, is decided based on week by week workload and can be set up to a maximum of 20 hours a week.

4.5.2. Base Model Experiment

In the DES model of the shop, steel pieces are represented by entities, each type of operation in the specialized shops (or every station) is modeled by an activity

(or task), and the workers are represented by resources assigned to activities or stations (for a detailed explanation of modeling elements in DES, see Banks, 2010). A separate SD model of workforce evolution dynamics has been assigned to every type of the operation (i.e., cutting, fitting, and welding) held in every working shift (i.e., dayshift and nightshift). The painting operation in this case requires a limited number of the workers, and since SD models work with a pool of workers, not individuals, I avoided linking SD models to this operation. A constant number of three workers (equal to the usual case in the collaborating company) and an average shop performance level have been assigned to the painting shop. Since the pace of operations done in the fabrication shop starts from several minutes and can go up to several days or even weeks of work, the time interval selected in the DES model is one minute, though the rate of changes in the workforce evolution dynamics is suited to a time interval of one week to be set for the SD models. All parameters in the model are set based on the minute. The implementation details of the model have been presented in Appendix D.

The case study is from January 1st, 2008 to July 1st, 2010 during which the fabrication shop served 350 different structural steel construction projects with nearly 550,000 structural steel pieces (and elements) and scheduled working hours of 2.5 million man-hours. As shown in Figure 4-13, the simulated completion curve almost follows the actual completion curve. Behavior reproduction test has been run on the model as mentioned by Sterman (2000, pp. 874-880). The test results showed a strong relation between actual and simulated

results, with a correlation coefficient (r) of 0.999 and a coefficient of determination (R^2) of 0.998. Theil's inequality statistics (Theil 1966) were also run to assess possible systematic errors in the results. The calculated values for Theil's indices were 0.03 for bias, 0.19 for variation and 0.78 for co-variation, which indicate unsystematic error according to the interpretation method presented by Sterman (2000, p. 876).

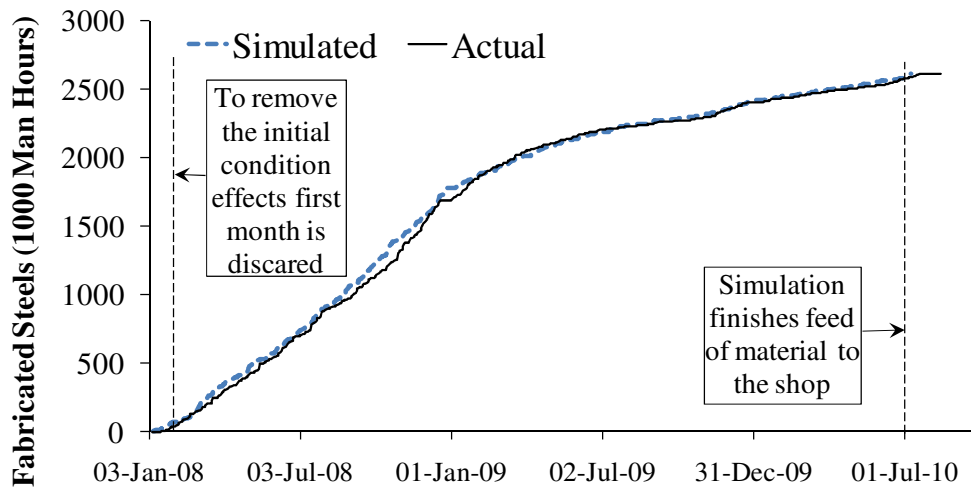


Figure 4-13. Comparison between the results from simulation model and the actual results

4.5.3. Alternative Policies Simulation

Many control factors can contribute to human resource policies (see Table 4-4) and numerous alternative policies can be created as the results of different combinations of these control factors. Analyzing all possible alternative policies is not feasible and is not suggested. However, a two level model analysis has been used here for evaluating different policies. First, simple alternatives, the results of changing just one control factor, have been created and tested. Then, based on the

results achieved in the simple alternatives, compound alternatives have been created and assessed. Table 4-4 describes the simple alternatives created and Table 4-5 presents the results.

Table 4-4. Simple alternative policies applied to the case

Alternative ControlFactor	Base model	Workload Average Changes (WA)	Hiring/ Overtime Policy Changes (HO)	New Worker Changes (NW)		
Period of Workload Calculation	Next 12 weeks	WA1) Next 9 weeks WA2) Next 6 weeks WA3) Next 3 weeks WA4) Next 1 weeks	(The same as the Base Model)	(The same as the Base Model)		
	Hiring and Overtime Policy*	FM = 80% HM=150% OP=1	(The same as the Base Model)	HO1) FM = 70%, HM=150%, OP=1 HO2) FM = 90%, HM=150%, OP=1 HO3) FM = 80%, HM=140%, OP=1 HO4) FM = 80%, HM=130%, OP=1 HO5) FM = 80%, HM=120%, OP=1 HO6) FM = 80%, HM=110%, OP=1 HO7) FM = 80%, HM=100%, OP=1 HO8) FM = 80%, HM=100%, OP=0	(The same as the Base Model)	
		New Workers Combination	Wage 1: 50% Wage 2: 20% Wage 3: 15% Wage 4: 15%	(The same as the Base Model)	(The same as the Base Model)	NW1) Wage 1: 100% NW2) Wage 2: 100% NW3) Wage 3: 100% NW4) Wage 4: 100%

* Refer to Section 4.3.1 for definition of HM (Hiring Margin) and FM (Firing Margin) and Section 4.3.3 for definition of OP (Overtime Policy)

Table 4-5. Results achieved for simple alternative policies

Simple Alternative	Total Cost M\$	Average Performance Level	Work Finish
<i>Base</i>	<i>138</i>	<i>67.5%</i>	<i>08-Jul-10</i>
WA1	142	67.0%	08-Jul-10
WA2	141	66.7%	08-Jul-10
WA3	163	65.8%	14-Jul-10
WA4	257	65.2%	08-Jul-10
HO1	141	67.5%	08-Jul-10
HO2	134	67.3%	08-Jul-10
HO3	121	69.0%	08-Jul-10
HO4	117	69.8%	08-Jul-10
HO5	116	70.4%	13-Jul-10
HO6	90	72.6%	07-Jul-10
HO7	96	72.9%	07-Jul-10
HO8	95	73.0%	07-Jul-10
NW1	147	59.8%	08-Jul-10
NW2	135	68.4%	08-Jul-10
NW3	130	76.5%	08-Jul-10
NW4	138	81.9%	08-Jul-10

The rows of the results indicating improvement in the total cost and earlier or equal project completion compared to the base scenario are highlighted in Table 4-5. The first group of simple alternatives was created by simply decreasing the workload average period, since 12 weeks is the maximum reliable duration and workload schedule is subject to changes after that. As shown in the table, none of the workload average (WA) alternatives indicate improvement. In the second group of simple alternatives, Hiring/Overtime Policy (HO), increase in the firing margin (FM), and decrease in the hiring margin (HM) have created improvement. A comparison between using overtime and a no-overtime policy with no changes to other parameters can only be made when $HM = 100\%$. This comparison indicated the no-overtime policy will improve the result. The last group of simple alternatives was created by changing the proportion of different types of workers to be hired. The result indicated that hiring wage2 and wage3 new workers

improves the final results. In the next step two compound alternatives (CA) were created and compared as the result of combination of the best results achieved in the simple alternatives as in below:

CA1: WA = Next 9 weeks, FM = 90%, HM=110%, OP=1, Wage 3: 100%: Total cost of \$82 million and work finished on July 7th 2010

CA2: WA = Next 9 weeks, FM = 90%, HM=100%, OP=0, Wage 3: 100%: Total cost of \$87 million and work finished on July 7th 2010

Combining the values of the parameters with the best results by 1) considering overtime as a response to over-capacity workloads (OP = 1) created the first compound alternative (CA1); and 2) setting no overtime (OP = 0) formed the second compound alternative (CA2). The total costs achieved in both compound alternatives were lower than every single simple alternative; however, CA1 (with total cost of \$82 million) resulted in \$5 million less total cost than CA2 (with total cost of \$87 million). This shows a potential saving of 40% in human resource related costs compared to the base model (with total cost of \$137 million). To investigate the sensitivity level of the final results of the CA1 alternative, a sensitivity analysis was run on two control parameters for this alternative. First, a sensitivity analysis was run by examining a 25% increase in hiring costs (from \$5000 per new worker hired to \$6250). This sensitivity was run on the alternative because it is created as the result of decreased hiring margin (HM) and increased firing margin (FM), which encourages project managers to hire new and fire redundant workers more easily. The second sensitivity analysis was run by decreasing the performance level of the Wage3 workers (Section 4.3.1) by 5%

from 80% to 76%. The performance levels in other wage groups were not changed. Since in this alternative the focus of new hiring is on Wage3 workers, by this change the effects of possible over-estimation for the performance level in this group of workers can be investigated. Total cost achieved in the first sensitivity analysis was \$88 million and in the second sensitivity was \$89 million, which shows a 7% and 8% increase in the total cost, respectively. However, the total cost is still more than 35% below the base model, which is a safe buffer, as long as deviations in the control parameters stay within this range.

4.5.4. Case Analysis

The result achieved in the case experiment indicated a significant potential improvement (i.e., 40% of total human resource cost reduction) in the project by merely adjusting human resource policies during the construction period. It should be noted that some values used in the models are estimated and approximate values. For example, the performance level values assigned to different levels of experience are based on interviews run with shop managers and superintendents, which are subject to human judgment and approximations. If we are to be able to consider the proposed approach as an effective and practical tool for real construction projects, this kind of approximation is inevitable. Running statistically verified data collecting methods for effective parameters in the model prior to every construction project is basically impossible. However, the use of sensitivity analysis on specific parameters with a possible range of variations is

suggested, to avoid excessive additional costs in highly sensitive situations. This experiment was not meant to search for the optimum alternative policy, but was intended to show the direction that policy control parameters can follow to reduce the final cost of projects.

4.6. Chapter Conclusion

In this part of the research I introduced and validated a new approach for improving construction projects using a hybrid SD and DES simulation model. The model targets improvement in construction projects by adjusting human resource policies. In the first part of the study I explored different aspects of workforce dynamics within the construction industry by studying the prevalent workforce dynamics models in different disciplines, and customizing the model for construction. I then ran different validity tests to validate the applicability and validity of the model for construction projects. The SD model I developed was then integrated with a DES model of a real construction case of structural steel fabrication, where it was used for simulating two and a half years of the fabrication operation, serving a variety of structural steel construction projects. In this experiment I showed a potential cost savings of up to 40% in total human resource costs by adjusting control parameters in human resource policies. The same approach introduced in this part of the research can be applied to different construction projects during the planning and implementation phase of a variety

of construction projects to improve their human resource policies and find potential room for cost savings in their human resource management.

The use of a hybrid SD-DES modeling approach for modeling and improving different aspects of construction projects is a new practice in the construction domain. In the previous part of the study (presented in Chapter 3), I introduced and developed a hybrid SD-DES model of working hours and demonstrated how we can improve construction productivity by adjusting the working hour arrangements. These two applications highlight just two out of many other aspects of the construction industry that can be studied and improved by the use of hybrid SD-DES modeling. Some other instances for possible future studies using hybrid SD-DES modeling are improving quality policies, minimizing the adverse effects of environmental changes on construction, and enhancing sustainability in the construction.

CHAPTER 5. Conclusions and Recommendations

5.1. Research Summary

The main motivation of this research was to introduce a new practical approach for improving some aspects of construction projects which have previously been ignored because of limited capabilities of currently available construction management tools. A hybrid SD-DES modeling approach is the tool introduced in this research for capturing complex interactions between the operation and context levels of construction projects and helping construction managers in their decision and policy-making tasks. The hybrid SD-DES modeling approach was first introduced about a decade ago (Rus et al. 1999; Zeigler et al. 2000; Martin and Raffo 2001; Choi et al. 2006) to improve software project development processes by analyzing hybrid interactions which affect them. Even though a decade has passed since the introduction of the hybrid SD-DES modeling approach, the extent of its usage in the construction domain was quite limited and still in the conceptual and introductory stage. I studied several hybrid SD-DES models which were developed in other disciplines and analyzed their specifications. From these studies I recognized several challenging points which either could have postponed the use of hybrid SD-DES models in the construction industry or were potential points of future problems during hybrid model development in construction projects. These challenging issues (which have been discussed in Chapter 2) are summarized below:

- 1) Lack of a modeling framework which can be used as a guide for hybrid model developers during the design phase of these model developments.
- 2) The computational complexity which might arise as a result of different time advancing methods in SD (continuous time advancing) and DES (discrete time advancing).
- 3) The complexity involved in creating communication channels required for communication among different parts of the hybrid model.

These challenging issues were addressed in the hybrid modeling framework and the architecture was proposed and validated (Chapter 2). Three main types of hybrid structures, five forms of hybrid interactions, the meaningful level of change (MLC) concept and the use of high level architecture (HLA) were the set of tools introduced in the hybrid modeling framework and architecture.

Considering the expected duration of the research, the direction of the research at the next stage could have continued in either of the two following directions: 1) looking into real applications of hybrid SD-DES models in construction and investigating their potential benefits, or 2) developing a special purpose hybrid simulation package which can facilitate implementation of hybrid SD-DES in construction following the proposed hybrid framework and architecture. To be able to demonstrate real advantages of the use of hybrid SD-DES in construction, prior to any investment in the development of a special purpose hybrid simulation

package, I chose the first direction (i.e., applying hybrid SD-DES models in construction applications). However, since there is no hybrid modeling tool which fully conforms to the proposed hybrid modeling architecture, I used the AnyLogic simulation package, which supports implementation of hybrid SD-DES models, but does not support the HLA framework or the MLC concept. Although this simplification affected the independence of the models from the experimental case and limited the expansibility of the model, it let me investigate the potential benefits that each model could bring to two decision making problem instances in construction.

Improving construction projects by adjusting working hour arrangements was the first hybrid application of the research (Chapter 3). In this application, different aspects of working hours in construction projects, including the duration of working periods and rest breaks, duration of set overtime, and work start and finish times, were analyzed, and their complex impacts on final worker performance were captured through a hybrid model. The hybrid model was developed in accordance with prevalent theories on every aspect of the working hours. The energy consumption theory presented by Oglesby et al. (1989, pp. 240-251) was used for capturing the physical effects of the working period duration, and the limited resource theory (Smit et al. 2004; and Helton and Warm 2008) was used for modeling the mental effects of the working period duration. The effects of work start and finish time were traced in the model by using the normalized function provided by Folkard and Tucker (2003) and the effects of

overtime on work productivity were added to the model by using the equations extracted from Sterman's (2000, p. 581) work.

The proposed model was first validated and then applied to a real structural steel fabrication shop serving multiple steel construction projects. The projects assigned to the fabrication shop during the course of three months were studied. Four groups of working hour alternatives were studied; finally, I was able to show that there is potential room for 6% total productivity improvement, just by adjusting the working hours. The best result achieved was when the working period duration was reduced from 1.75 hours to 1 hour.

The second hybrid application explored in the research was improving human resource policies in the construction industry (Chapter 4). There are many effective parameters in human resource policies which can affect the final output of construction projects. Some examples of these effective parameters are:

- 1) Set hiring and firing margins as the responses to the fluctuations in the workload
- 2) The Company's policy for using overtime
- 3) The Company's policy regarding the level of skill and training programs
- 4) Method for estimating the upcoming workload

A hybrid model of human resource policies was developed by capturing the effects of different aspects of human resource policies in construction projects. The developed model was first validated and then run for projects assigned to a structural steel fabrication shop during a course of two and half years. Because of the numerous possible alternatives, a two step analysis was run for analyzing different alternatives. First, simple alternatives, created as the result of changes in just one effective parameter, were evaluated. Then compound alternatives were created by combining the best simple alternatives and were themselves evaluated. The result of this analysis indicated a considerable potential improvement of 40% by adjusting human resource policies.

In sum, this research introduced a new tool to be used by construction managers to trace the effects of their decisions and set their policies based on the outcomes expected in construction projects. Two different applications of the introduced tool were explored and discussed and the capability of this hybrid tool for improving construction projects was investigated. Both applications presented considerable room for construction project improvement.

5.2. Research Contributions

The main contributions of this research are summarized below:

- 1) Introducing and validating a new hybrid SD-DES modeling framework and architecture for facilitating the hybrid model development process within the construction industry.

- 2) Demonstrating the capacity of the proposed hybrid modeling framework and architecture to integrate hybrid SD-DES models with other computer aided managerial tools used in construction, such as 3D visualization and real time control systems. The proposed hybrid modeling framework and architecture can be used for developing a construction “virtual enterprise” where different aspects of construction are modeled by different types of computer aided tools.
- 3) Demonstrating the capabilities and usefulness of the hybrid SD-DES modeling approach in the construction domain by successfully applying the hybrid model to two different construction decision making problems.
- 4) Contributing to construction project working hour management by exploring and validating the complex effects of working hour arrangements on the final output of construction projects and demonstrating the implementation method by applying it to a real construction case.
- 5) Contributing to construction project human resource management by exploring and validating the complex effects of human resource policies on construction project productivity and showing a step by step implementation method in a real construction case.
- 6) Contributing to steel construction by running two case studies on structural steel construction and proposing potential room for improvement in the structural steel fabrication process.

5.3. Lessons learned

Some lesson learned in the research are shown below:

- 1) PhD research requires continuous work for several years, and contains many different phases and aspects, including literature review, detecting potential gaps and points of improvement and potential contributions, theory statement, model development, tests and experiments, communication skills, teamwork, data gathering and preparing the equipment required. Good planning and control is the main tool that can assist PhD students during their long journey. This planning and control is not usually a day to day activity, but depending on the expected rate of progress in different periods of PhD research it can have a range of one week to several months.
- 2) Unlike my initial perception of literature review – that it mainly happens during the initial parts of the research – it was almost a permanent aspect of my research at every stage of the work. In every stage of my research I was facing new concepts or areas of research which which I was not quite familiar, and I needed to review the literature to make sure that I was not missing any major points. Therefore, at least in some types of research and even in the later stages of the work, the researcher should always be prepared to take some time to get back to literature review.
- 3) Although the nature of the construction industry is different compared to other industries, such as the software and manufacturing industries, methods and tools introduced for other disciplines can also be useful in the construction

industry. However, some adjustments and modifications might be required to fit those methods and tools to the construction industry. The hybrid SD-DES modeling approach is one of these methods which was first used in the software and then manufacturing industries, and I was able to customize it to meet construction industry requirements. Therefore, when making improvements in some aspects of the construction industry, looking into and getting ideas from successful experiences in similar aspects of other disciplines can be a valid alternative.

- 4) By breaking complex systems down into simple building blocks, these complex systems (which were thought to be completely out of reach) can be captured and their behaviour can be thoroughly studied and analyzed. In my research, one constant item of feedback I received from my research audiences when, for example, I was discussing the way that I planned to capture the effects of working hours on worker fatigue and project productivity was: “It is impossible! How are you going to measure worker fatigue? It is so complicated!” But by breaking the complex structure of hybrid interactions down into building blocks and capturing the effects of hybrid feedback loops by using the developed hybrid modeling framework and architecture I could successfully capture the ultimate effects of the hybrid interactions on construction project behaviour.
- 5) When it comes to developing models for complex systems, choosing an iterative approach helps the developer to not get lost during their model

development and to stay on the right track. Since the nature of complex systems usually do not let a person capture all aspects of the system at once, by starting the development of a simplified model and then adding more complexity to the model during the next iterations, model developers can concurrently learn and resolve development issues and improve their understanding from the system as the model development goes on from one iteration to the other.

- 6) Practical examples are an important part of the research which help the researcher present ideas in a more tangible fashion. In my research, the initial parts of the research were all about theoretical parts of the hybrid modeling framework and architecture. It was quite difficult for me to convince my research audience that this framework and architecture is meant to be used as an applicable tool in the construction industry to solve issues that construction managers deal with during their daily jobs. By applying the hybrid modeling framework and architecture to construction cases, the contributions that this research can make to the construction industry became quite clear for the research audience.
- 7) Unlike the perception that equipping construction projects with advanced construction equipment and IT technologies is the most important factor for successful construction projects, the managerial aspects and more specifically the decision making process in construction projects showed huge potential for improving projects. Compared to the monetary investment required for

equipping construction projects with new equipment, improving the decision making process in many situations requires less investment, since basically the main investment required for such improvements is the model development.

- 8) Close relationships with the industry gave me a great opportunity to experiment with my proposed ideas and implement and analyze them in real world construction conditions. Many initially missed or immature aspects of the model were discovered during the implementation of these experimental cases and were added to or fixed in the body of the research. This close relation with the industry can have benefits to many construction research efforts and should be considered during construction research development.

5.4. Recommendations for Future Research

This research sheds light on some new areas which warrant further research efforts. Some of these areas of research are highlighted below:

- 1) Modeling new complicated decision and policy making issues in the construction domain that have not been studied previously as the result of modeling tool shortfalls is a prospective area of research. The capabilities of the hybrid modeling approach suggest many new areas in construction projects that can be studied which previously were out of reach. Some examples for such prospective research topics are improving quality policies, sustainability policies, and safety policies in construction projects.

- 2) Adding agent based simulation (ABS) to hybrid SD-DES modeling as another complementary simulation modeling tool, to be used for capturing the effects of system object (agent) evolution and adaptation during system operation, is another promising area of research. This type of hybrid model can more properly capture construction projects that depend on evolving construction components. One instance of evolving components in construction projects is the variation in the properties of concrete during its production, transportation, casting and treatment. Effects of disciplinary action against negligent workers and its effects on other workers behaviour is another example of evolution and adaptation in the work environment. Very like the stream of the current research, the development of a framework and architecture which considers different aspects of the integration of SD, DES and ABS could be the first stage of this research project. The potential benefits of this hybrid modeling approach can be examined by applying it to real construction applications at the second stage of the research.
- 3) The development of a special purpose software package for implementation of a variety of hybrid SD-DES models for construction projects, following and improving the proposed hybrid architecture in this research, is another prospective research topic. By providing different capabilities required in the hybrid SD-DES models, including visual SD and DES modeling elements, handling the interactions between different parts of SD and DES models, supporting the MLC concept in the hybrid interactions, and providing HLA

services for distributed simulation as well as involving generic hybrid modeling elements for common hybrid interactions within construction projects, this simulation package could be very useful for implementing hybrid models and even virtual enterprises within the construction domain. This research project could be a joint endeavour between the construction engineering and software engineering disciplines.

References

Abdel-Hamid, T.K. and Madnick, S.E. (1989) Lessons Learned from Modeling the Dynamics of Software Development. *Communications of the ACM*, 32(2), 1426-1438.

AbouRizk, S.M. and Hague, S. (2009). An Overview of the COSYE Environment for Construction Simulation. *Proceedings of the Winter Simulation Conference*, Austin, Texas, USA, December 2009, 2624-2634.

AbouRizk, S.M. and Hajjar, D. (1998). A Framework for Applying Simulation in Construction. *Canadian Journal of Civil Engineering*, 25(3), 604-617.

Adeli, H. and Karim, A. (1997). Scheduling/Cost Optimization and Neural Dynamics Model for Construction. *Journal of Construction Management and Engineering*, ASCE, 123(4), 450-458.

Alvanchi, A., Azimi, R., Lee, S. and AbouRizk, S. (2010). Virtual Model of Structural Steel Construction using COSYE Framework. *Proceedings of the 10th International Conference on Construction Applications of Virtual Reality*, Sendai, Miyagi, Japan, Nov 2010, 283-290.

Alvanchi, A., Lee, S. and AbouRizk, S. (2009a). Modeling Architecture for Hybrid System Dynamics and Discrete Event Simulation. *Proceedings of the Construction Research Conference*, Seattle, April 2009, 1290-1299.

Alvanchi, A., Lee, S. and AbouRizk, S. (2009b). Meaningful Level of Change in Hybrid Simulation for Construction Analysis. *Proceeding of the Winter Simulation Conference*, Austin, Texas, U.S.A, December 2009, 2647-2652.

Alvanchi, A., Lee, S. and AbouRizk, S. (2011a). Modeling Framework and Architecture of Hybrid System Dynamics and Discrete Event Simulation for Construction. *Computer-Aided Civil and Infrastructure Engineering*, 26(2), 77-91.

Alvanchi, A., Lee, S. and AbouRizk, S. (2011b). Dynamics of Working Hours in Construction. *Journal of Construction Engineering and Management*, ASCE, In press (Accepted and Published online in March).

Alvanchi, A., Lee, S. and AbouRizk, S. (2011c). Modeling of Workforce evolution in Construction Projects. *Proceeding of the Annual Conference of Canadian Society for Civil Engineering*, Ottawa, Canada, June 2011.

Alvanchi, A., Lee, S. and AbouRizk, S. (2011d). Modeling Evolution of Human Resources Dynamics in the Construction Projects. *Under Preparation to be Submitted to the Journal of Automation in Construction*, September 2011.

An, L., Ren, C., Jeng, J.J. and Lee, Y. (2007). Effective Workforce Lifecycle Management via System Dynamics Modeling and Simulation. *Proceedings of the 2007 Winter Simulation Conference*, Washington, D.C., USA, 2187-2195.

Azimi, R., Lee, S., AbouRizk, S., (2009). Automated Project Control System for Steel Projects. Proceeding of the Joint International Conference on Construction Engineering and Management and Construction Project Management (ICCEM-ICCPM), Jeju, Korea, 479-486.

Azimi, R., Lee, S., AbouRizk, S., Alvanchi, A. (2011). A Framework for Automated and Integrated Project Monitoring and Control System for Steel Fabrication Projects. *Automation in Construction*, 20(1), 88-97.

Banks, J. (2010). *Discrete-Event System Simulation*. Upper Saddle River, N.J., USA: Prentice Hall.

Borshchev, A. and Filippov, A. (2004). From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools. *The 22nd International Conference of the System Dynamics Society*, July 2004, Oxford, England.

Borshchev, A., Karpov, Y., and Kharitonov, V. (2002). Distributed simulation of hybrid systems with AnyLogic and HLA. *Future Generation Computer Systems*, 18, 829-839.

Chi, S., Caldas, C.H., and Kim, D.Y. (2009). A Methodology for Object Identification and Tracking in Construction based on Spatial Modeling and Image Matching Techniques. *Computer-Aided Civil and Infrastructure Engineering*, 24(3), 199-211.

Choi, K., Bae, D., and Kim, T. (2006). An approach to a hybrid software process simulation using the DEVS formalism. *Software Process: Improvement and Practice*, 11, 373-383.

Cohen, C.J. and Muehl, G.E. (1977). Human circadian rhythms in resting and exercise pulse rates. *Ergonomics*, 20, 475-479.

Dijk, D.J., Duffy, J. and Czeisler, C.A. (1992). Circadian and sleep/wake dependent aspects of subjective alertness and cognitive performance. *Journal of Sleep Research*, 1, 112-117.

Eastman, C.M. and Sacks R. (2008). Relative Productivity in the AEC Industries in the United States for On-Site and Offsite Activities. *ASCE Journal of Construction Engineering and Management*, 134(7), 517-526.

Fehlberg, E. (1970). New high-order Runge-Kutta formulas with an arbitrary small truncation error. *Computing*, 6, 61-71.

Folkard, S. and Tucker, P. (2003). In-Depth Review: Shift work; Shift work, safety and productivity. *Occupational Medicine*, 53, 95-101.

Ford, D., Anderson, S., Damron, A., Casas, R., Gokmen, N., and Kuennen. S. (2004). Managing Constructibility Reviews to Reduce Highway Project Durations. *Journal of Construction Engineering and Management*, 130(1), 33-42.

Forrester, J.W. (1958). Industrial Dynamics: A Major Breakthrough for Decision Makers. *Harvard Business Review*, 36(4), 37-66.

Go Welding.Org. (2011). *Welding Certification*. gowelding.org/Welding_Certification.html. Accessed January 26, 2011.

Hafeez, K., Aburawi, I. and Norcliffe, A. (2004). Human resource modelling using system dynamics. *Proceeding of the 22nd International Conference of the System Dynamics Society*, Oxford, UK, July.

Halpin, D. W. (1973). *An Investigation of the Use of simulation Networks for Modeling Construction Operations*. Doctoral dissertation, Dept. of Civil Engineering, University of Illinois at Urbana-Champaign, IL.

Halpin, D.W. (2011) *Construction Management*. Hoboken, NJ:John Wiley and Sons Ltd.

Healy, A.F., Kole, J.A., Buck-Gengler, C.J. and Bourne, L.E. (2004). Effects of Prolonged Work on Data Entry Speed and Accuracy. *Journal of Experimental Psychology: Applied*. 10(3), 188-199.

Helton, W.S. and Warm, J.S. (2008). Signal Saliency and the Mindlessness Theory of Vigilance. *Acta Psychologica*, 129, 18-25.

Homer, J.B. (1985). Worker Burnout: a Dynamic Model with Implications for Prevention and Control. *System Dynamics Review*, 1(1), 42-62.

Karim, A. and Adeli, H. (1999). CONSCOM: An OO Construction Scheduling and Change Management System. *Journal of Construction Engineering and Management*, 125(5), 368-376.

Koshio, A. and Akiyama, M. (2008). Physician's Burning out and Human Resource Crisis in Japanese Hospital: Management for Sustaining Medical Services in Japan. *The 2008 International Conference of the System Dynamics Society*, Athens, Greece.

Kuhl F., Weatherly, R., and Dahmann, J. (1999). *Creating computer simulation systems: An introduction to the high level architecture*. Prentice Hall PTR, Upper Saddle River, NJ 07458, 5-6.

Lee, S., Han, S., and Peña-Mora, F. (2007). Hybrid system dynamics and discrete event simulation for construction management. *Proceeding of the ASCE International Workshop on Computing in Civil Engineering*, Pittsburgh, PA, 232-239.

Lee, S., Han, S., Peña-Mora, F. (2009). Integrating Construction Operation and Context in Large-scale Construction Using Hybrid Computer Simulation. *Journal of Computing in Civil Engineering*, ASCE, March/April 2009, 75-83

Lee, S., Peña-Mora, F., Park, M. (2005). Quality and Change Management Model For Large Scale Concurrent Design and Construction Projects. *Journal of Construction Engineering and Management*, ASCE, Reston, VA, July/August, 2005, 131(8), 890-902.

Lee, S., Peña-Mora, F., Park, M. (2006). Dynamic Planning and Control Methodology For Strategic and Operational Construction Project Management. *Automation in Construction*, Elsevier, New York, NY, 15(1), 84-97.

Lyneis, J. and Ford, D. (2007). System Dynamics Applied to Project Management: A Survey Assessment, and Directions for Future Research. *System Dynamics Review*. 23(2-3), 157-189.

Martinez, J. C., Ioannou, P. G., and Carr, R. I. (1994). State and Resource Based Construction Process Simulation. *Proceedings of the First Congress on Computing in Civil Engineering*, ASCE, Washington, D.C., US, 177-184.

Martin, R., and Raffo, D. (2001). Application of a Hybrid Process Simulation Model to a Software Development Project. *Journal of systems and software*, 59, 237-246.

Matthews, G., Davies, D.R. and Lees, J.L. (1990). Arousal, Extroversion, and Individual Differences in Resource Availability. *Journal of Personality and Social Psychology*, 59(1), 150-168.

Nuechterlein, K., Parasuraman, R. and Jiang, Q. (1983). Visual Vigilance: Image Degradation Produces Rapid Sensitivity Decrement Over Time. *Science*, 220, 327-329.

Oglesby, C.H., Parker, H.W., and Howel, G.A. (1989). *Productivity Improvement in Construction*. New York, NY: McGraw-Hill.

Oliva, R.A. (1996). *Dynamic Theory of Service Delivery Implications for Managing Service Quality*. PhD Dissertation. Sloan School of Management. MIT, MA, USA.

Packer, D.W. (1964). *Resource Acquisition in Corporate Growth*. MIT Press, Cambridge, MA, USA.

Park, M., and Peña-Mora, F. (2003). Dynamic Change Management for Construction: Introducing the Change Cycle into Model-based Project Management. *System Dynamics Review*, 19(3), 213-242.

Paulson, B.C. (1987). Interactive Graphics for simulation Construction Operations. *Journal of the Construction Division*, 104(1), 69-76.

Pena-Mora, F., Han S., Lee, S., and Park, M. (2008). Strategic-operational Construction Management: Hybrid System Dynamics and Discrete Event Approach. *Journal of Construction Engineering and Management*, ASCE, Reston, VA, 134(9), 701-710.

Pritsker, A.A., O'Reilly, J.J., and Laval, D.K. (1997). *Simulation with Visual Slam and Awesim*. John Wiley & Sons, Inc., New York.

Rabelo, L., Helal, M., Jones, A., and Min, H. (2005). Enterprise Simulation: A Hybrid System Approach. *International Journal of Computer Integrated Manufacturing*, 18(6), 498-508.

Rekapalli, P.V. and Martinez, J.C. (2007). Time Advance Synchronization in Concurrent Discrete-event Simulation and Animation of Construction Operations. *Proceeding of the Eleventh International Conference on Civil, Structural, and Environmental Engineering Computing*, Civil-Comp Ltd., Stirling, U. K.

Rekapalli, P.V., Martínez, J.C., and Kamat, V.R. (2009). Algorithms for Accurate Three-Dimensional Scene Graph Updates in High Speed Animations of Simulated Construction Operations. *Computer-Aided Civil and Infrastructure Engineering*, 24(3), 186-198.

Richardson, G. and Pugh, A.L. (1981). *Introduction to System Dynamics Modeling with Dynamo*. MIT Press, Cambridge, MA.

Ritter, F.E. and Schooler, L.J. (2002). The Learning Curve. *International Encyclopedia of the Social and Behavioral Sciences*, Amsterdam, Pergamon, 8602-8605.

Rohmert, W. (1973a). Problems of Determination of Rest allowances. Part 1: Use of Modern Methods to Evaluate Stress and Strain in Static Muscular Work. *Applied Ergonomics* 4(2), 91-95.

Rohmert, W. (1973b). Problems of Determination of Rest Allowances. Part 2: Determining rest allowances in different human tasks. *Applied Ergonomics* 4(3) 158-162.

Rosa, R.R., Bonnet, M.H. and Cole, L.L. (1998). Work Schedule and Task Factors in Upper-Extremity Fatigue. *Human Factors*, 40(1), March, 150-158.

RSMeans (2010). *Metric Construction Cost Data*. Reed Construction Data Inc., USA.

Rus, I., Collofello, J., and Lakey, P. (1999). Software Process Simulation for Reliability Management. *Journal of Systems and Software*, 46(2-3), 173-182.

Skibniewski, M.J. and Jang, W.S. (2009), Simulation of Accuracy Performance for Wireless Sensor-Based Localization in Construction Asset Tracking. *Computer-Aided Civil and Infrastructure Engineering*, 24(5), 335-345.

Smit, A.S., Eling, P.A.T.M. and Coenen, A.M.L. (2004). Mental Effort Causes Vigilance Decrease Due to Resource Depletion. *Acta Psychologica*, 115, 35-42.

Sterman J. D. (1992). *System Dynamics Modeling for Project Management*. Cambridge, MA, Sloan School of Management, MIT.

Sterman, J. D. (2000). *Reference: Business Dynamics: System Thinking and Modeling for a Complex World*. New York, NY: McGraw-Hill Higher Education.

The EI Group. (2011) *Campus Starter: Apprenticeship Programs in Canada*. apprenticeshipprogramsincanada.com. Accessed January 26, 2011.

Taylor, F.W. (1911). *The Principles of Scientific Management*. Harper and Brothers Publishers, New York. Reprinted in 1967 by Norton, New York.

Theil, H. (1966). *Applied Economic Forecasting*. Chicago, IL, USA: Rand McNally.

Venkateswaran, J., Son, Y., and Jones, A. (2004). Hierarchical production planning using a hybrid system dynamic-discrete event simulation architecture. *Proceedings of the 2004 Winter Simulation Conference*, 1094-1102.

Vidacek, S., Kaliterna, L., Radosevic-Vidacek, B. and Folkard, S. (1986). Productivity on a weekly rotating shift system: circadian adjustment and sleep deprivation effects? *Ergonomics*, 29, 1583-1590.

Wang, J. (2005). *A review of operations research applications in workforce planning and potential modeling of military training*. DSTO Systems Sciences Laboratory, Department of Defence, Australian Government, DSTO-TR-1688.

Zeigler, B.P., Praehofer, H. and Kim, T. (2000). *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. London, UK: 2nd Ed., Academic Press.

Zeigler, B.P., Cho, H.J., Kim, J.G., Sarjoughian, H.S. and Lee, J.S. (2002). Quantization-based Filtering in Distributed Discrete Event Simulation. *Journal of Parallel and Distributed Computing*, 62, 1629-1647.

Appendix A⁵

Meaningful Level of Change in Hybrid Simulation for Construction Analysis

Amin Alvanchi

SangHyun Lee

Simaan M. AbouRizk

Dept. of Civil and

Dept. of Civil and

Dept. of Civil and

5-080, NREF, University of
Edmonton, AB, T6G 2W2,

3-009, NREF, University of
Edmonton, AB, T6G 2W2,

3-014, NREF, University of
Edmonton, AB, T6G 2W2,

ABSTRACT

Hybrid models of System Dynamics (SD) and Discrete Event Simulation (DES) in the construction industry aim to provide decision makers with more accurate analysis. However, there are certain issues that can limit the applicability of SD-DES hybrid models for real construction job situations. Meaningful Level of Change (MLC) is a concept that has been proposed to prevent the time advancing issue in the hybrid models used within the construction domain. It is claimed that by utilizing the MLC, the running time of hybrid simulation models can be reduced while only slightly contributing to model inaccuracy. In this paper, we investigate the effects of utilizing the MLC for SD-DES hybrid models used for construction systems. First, the theoretical aspects of applying the MLC in hybrid models are investigated. Second, the effects of using different set values of MLC in an experimental model of a real construction system are illustrated.

⁵ This paper was published in the Proceedings of the 2009 Winter Simulation Conference, Austin, Texas, U.S.A, Dec 2009, pp. 2647-2652.

1 Introduction

Hybrid models of System Dynamics (SD) and Discrete Event Simulation (DES) attempt to capture more parts of reality by combining two different system modeling tools (i.e., SD and DES) (Lee et al. 2007). However, the difference between SD and DES modeling tools raises some challenging issues (Alvanchi et al. 2009a). One of these issues results from the fundamental difference between the ways that SD and DES advance time. While SD continuously follows the system behavior over time and selects an even step for its time progress, DES follows the uneven time steps based on the prescheduled time of system events.

The difference in the time advancement approaches of SD and DES may cause situations in which the simulation runs are computationally overloaded, which subsequently drastically increases the simulation time. The Meaningful Level of Change (MLC) (Alvanchi et al. 2009b) concept has been introduced to address the computational issue in such situations. The MLC is intended to keep the accuracy of simulation results at a reasonable level, according to the accepted level of changes, while significantly decreasing the running time of simulation. Different values can be set for MLCs of different interacting variables in an SD-DES hybrid model. However, the effects that different chosen values for the MLC, or the accepted level of inaccuracy for interacting variables, have on the final simulation results have not yet been thoroughly considered.

To address this issue, the current paper aims at investigating how selecting different values for the MLC could affect the prospective results of hybrid

simulation. The paper consists of the following 4 sections: (1) the essence of the MLC concept; (2) the theoretical acting mechanism of the MLC; (3) experiencing the effects of different values for the MLC in a simplified hybrid model of a real steel fabrication shop; and (4) brief conclusion.

2 Computational Problem of Time Progress in Hybrid Models

The fundamental difference between the SD and DES simulation time advancing methods (i.e., the continuous method of time advancing in SD and the discrete time advancing method in DES) is the source of potential deficiencies during hybrid simulation runs. SD simulation models are categorized as continuous simulation models. However, the implementation of SD computer simulation is not actually continuous, but instead follows evenly set time advancing steps. The set time advancing step should be small enough to be able to capture all significant changes within the system, but should not be so small that it only counts the time and delays the simulation runs without any expected significant changes in the system. For example, if an SD model keeps track of labor hiring/ firing, a time step of a second will waste time of the simulation runs, as it can easily be assumed that any factors that affect labor hiring/ firing will not exhibit critical changes within such a short increment of time.

The time of a DES model is changed based on the prescheduled events that usually follow uneven steps. The progress method in DES originates from the fact that we can follow system behavior over time by following the system's states over time. Every significant change in the system state is called an event. A DES

model of a system can be developed when the deterministic or probabilistic time distances between occurrences of all possible types of events can be determined by knowing the occurrence time of a set of initial events. The time advances in a DES model by changing the current time to the time of the event that has an occurrence time closest to the current time. After the occurrence of each event, all related events—which have happenings that can be determined by the occurrence of the occurred event—are scheduled and listed to occur sequentially in the future.

The potential computational problem in SD-DES hybrid models is raised when a variable in an SD modeling part interacts with a variable in the DES part that sets the duration time between event occurrences. If the time step in the SD modeling part is significantly (e.g., ten times) less than the duration time of the event occurrence in the DES part, before an event occurs in the DES part of the model, the interacting variables will interact with each other multiple times. As a result of any changes initiated by the SD part, not only are the values of the interact receiver in the DES part being changed, but the scheduled events that are related to the interact receiver variable of the DES part should also be found and withdrawn from the list of the scheduled events, calculated by a new set value of variables, and finally rescheduled and added to the future event list. This consequent series of changes will add too many calculations to the hybrid model, specifically when this process is going to happen numerous times before the occurrence of next scheduled events.

3 Meaningful Level of Change (MLC)

In the literature, two different approaches can be found that try to reduce the number of interactions and thus decrease the negative effects of the time advancing issue. One of these approaches considers adapting criteria for SD time steps, based on the fourth-order Runge-Kutta method, to adjust the length of time steps according to the chronological rates of change (Fehlberg 1970). Proposed by Venkateswaran et al. (2004), the other approach limits all required data exchanges among different SD and DES parts within the SD-DES hybrid models to the set time intervals.

The adjusted time steps in the first approach are based on the recent trend of the changes in the SD part, which will reduce the number of null interactions (i.e., the interactions that send the same value as the previously sent value). However, if the interacting variables in the SD part are changed at every short time interval, this approach cannot help in improving the simulation time. On the other hand, by adjusting the length of time steps based on the chronological rates of change, there is a possibility that the unexpected fluctuations of the SD variables over a short period of time will be neglected.

In the second approach, the SD and DES model parts work separately according to their regular solving methods, and a time step is set for the data exchanges among the different parts. The set time step in this approach should be big enough to cause no interruptions to the DES parts due to sent interactions from the SD parts. On the other hand, the time step should be small enough to be able to

capture all significant changes within the SD model parts. While the system behavior and rate of the changes in different parts of the system may vary during the system life cycle, by setting a constant time step for hybrid interactions among different model parts, this approach may not efficiently capture all hybrid interactions during the system life cycle.

To address the time advancing issue while eliminating the undesired side effects, a concept called the Meaningful Level of Change (MLC) is utilized in the hybrid models for construction analysis (Alvanchi et al. 2009b). This concept suggests setting a meaningful level of change for interacting variables in the SD parts that may cause the time advancing issue. By setting the MLC for a variable, the achieved changes for that variable that are less than the set MLC are assumed to be trivial and are consequently not reported to the interacting parts of the model. For example, when there is a set MLC of 1% for the skill level variable, as an interacting variable from an SD part, and the last reported skill level is 80%, the new skill level is reported if its value crosses 79% or 81%. However, construction related models, especially in the SD parts, usually deal with many approximately estimated variables, such as skill level, productivity level, fatigue level and safety level. Taking this into account, putting a limit on the variable updates (i.e., setting the MLC) will not have a significant influence on the quality of the provided analyses.

The MLC concept has the same role as the thresholds in quantization based filtering in distributed discrete event simulation discussed by Zeigler et al. (2002).

However, the quantization approach is used to reduce the number of interactions within a distributed DES model rather than an SD-DES hybrid model. Zetigler et al. (2002) have thoroughly discussed the benefits that can be achieved for the distributed DES models in terms of simulation time by using the thresholds and quantization based filtering. By applying the MLC concept, it is also expected that we can enhance hybrid simulation models by reducing their simulation time while accepting some inaccuracies; to visualize this, a comparison with an unlimitedly interacting hybrid model, namely the base hybrid model, is conducted.

4 ESTIMATING EFFECTS OF SET MLC on FINAL RESULTS

By setting the MLC for an interacting variable in the SD part, we are accepting a level of inaccuracy. But what are the main influencing factors that contribute to lessened accuracy in the final simulation results? The immediate and apparent influencing factor is that the higher the value set for the MLC, the more inaccuracy in the final results. This means that by increasing the set MLC value for an interacting variable, the number of crossings will be reduced and, consequently, the number of initiated interactions by that variable will be decreased. An increase of the MLC will also make the interaction receiving parts blind to the variable changes within a broader range of variable changes, which will result in the simulation results having a lower level of accuracy.

The effect of the changes at the MLC value is not the same at different interacting variables, although the significance level of the variable participation in the hybrid model also plays a main role. Within a hybrid model, the effects of

changes in the model variables are captured through different formulas that represent different aspects of the system behaviors over time. The significance level of the variable participation can be defined based on: 1) the importance of the system behaviors that the variable participates in and their related formulas; 2) the range of values that a variable can have; and 3) the places that the variable stays at in the formulas (e.g., multiplicand, addend, base or exponent at exponentiation, and numerator or denominator in a fraction). An example for estimating the effects of the MLC on the final simulation results is discussed in the next section.

5 Experimental Case

To test the effects of using the MLC concept on a real hybrid model, a simplified experimental case of an actual structural steel fabrication shop is used. Through this case study, we examine the effects of setting different values for the MLCs of variables, which can result in the time advancing issue, on the final simulation results.

5.1 Interacting Variables

Variables that come from either the SD or DES model parts may initiate or receive hybrid interactions. However, the MLC will be set only for some of the interaction initiator variables in the SD parts which usually have significantly faster updating rates than the related interaction receiver variables in the DES part. According to the comparison conducted for the rate of updates, station productivity was found to be one of the variables that may cause the time

advancing issue for the fabrication shop hybrid model. The station productivity variable is set based on the shop productivity factor and station utilization. The more utilization of a station, the more fatigue will occur for the station operators, which results in less station productivity being achieved. The utilization of a station is equal to the busy portion of the station during the working day and is actively changed at any set time step. In the model, the set time step is one second and, correspondingly, the station productivity is changed at every second.

Station productivity, which is set in the SD model part, participates in the operation duration formula, which is used in the DES model part. Usually, operation duration within the fabrication shop exceeds several minutes. This affirms that in the base hybrid model of the fabrication shop, where there is no MLC concept used for the model, the duration of each single operation will be changed hundreds of times before the operation is finished. According to the explanation of the time advancing issue, this will be a potential point for creating the time advancing issue and a value is set for the MLC of the station productivity variable.

5.2 Analyzing the Effects of Set MLC for Station Productivity

To obtain an estimation of the expected changes in the final results, according to the set values for the MLC, an assessment is conducted based on the three steps introduced in Section

1. Importance of the related system behaviors: Station productivity sets the operation duration at each station, which defines the main behavior of the fabrication shop system. No duration for the station operations means that

there is no work to be done and this station becomes practically closed. However, it should be considered that station productivity sets the productivity at each station, not the entire fabrication shop. Taking the total number of n stations within the fabrication shop into account, the importance of each single station falls to the one nth (1/n) for the fabrication shop.

2. Station productivity is measured based on a percentage. Theoretically, this variable can be any real number greater than zero. However, in real job situations, receiving productivity values less than 50% and greater than 150% is rare.
3. Role of station productivity in the operation duration formula: operation duration is related to the two main factors: (1) the volume of the job and (2) the rate of doing the job.

$$SOD = \frac{VJ}{RDJ}$$

Where:

SOD = Station Operation Duration

VJ = Volume of the Job

RDJ = Rate of Doing the Job

Station productivity contributes in the rate of doing the job. However, there are different factors that set the rate of doing the job, as presented in the following formula:

$$RDJ = NRRJN \times ENOS \times POTI \times SP$$

Where:

RDJ= Rate of Doing the Job at Each Station

NRRJN = Normal Rate Related to the Job Nature

ENOS = Effect of Number of Operators at the Station

POTI = Percentage of Over Time Increase

SP = Station Productivity

While station productivity is a multiplicand factor in the operation duration formula, the created inaccuracy is directly transferred to the

duration. For example, if the MLC is set at 2%, real station productivity is 98% and last reported productivity is 97%, and the duration will face 1% of inaccuracy. As a result, for this specific problem, the expected maximum inaccuracy for the duration of the fabrication shop jobs is equal to the set MLC value, but considering the existing randomness in the model, it is also possible that for the limited number of observations, the final results will also show higher inaccuracy. However, there are usually several other factors that urge the final inaccuracy to a lower level than the set MLC.

For example, when the MLC is set at 2%, an approximately 2% inaccuracy will be achieved only in the cases in which the current productivity always has a difference of approximately 2% with the last reported productivity. But in the real model runs, the current productivity will usually yield different values, in both negative and positive directions, within the set MLC interval, which will reduce the final inaccuracy. Additionally, as there are multiple stations in the fabrication shop, there is a slight probability that all unreported productivity variations (because of the set MLC) will have the same direction of inaccuracy. However, inaccuracies pointing in different directions push the system to a more balanced situation.

5.3 Experiencing the Effects of Set MLC for Station Productivity

To visualize the real effects of MLC changes on the final simulation results, different modeling scenarios with different MLC values of 1%, 2%, 3%, 5% and 10% were set and run in the hybrid simulation model of a fabrication shop. The achieved results from different scenarios were compared among themselves and to the base hybrid model (i.e., the hybrid model in which no MLC is set and all changes to station productivity are reported regardless of their significance)unlimitedly interacting hybrid model unlimitedly interacting hybrid

model unlimitedly interacting hybrid model unlimitedly interacting hybrid model. Three months of material feeds (from January 20, 2009 to April 20, 2009) from the fabrication shop were simulated for each model. During this period, the steel materials were fed through the steel fabrication shop at the scheduled date to pass cutting, fitting, welding and painting operations. The simulation runs were completed after all of the fed materials were fabricated and prepared to be shipped to the field.

Two types of comparison were conducted for this research. The first test was performed to compare the total duration for fabricating all the fed materials at the fabrication shop. The second test was conducted to assess how the MLC concept affects the simulation time. Figure 1 shows the comparative achieved results for the duration of the steel fabrication at the specified period of material feeds, while Figure 2 presents the simulation time of different runs. The presented results for each class are based on five runs of hybrid model simulation. While obtaining a more rigorous conclusion for the conducted experiments requires a greater number of simulation runs, during this experiment, we aim to perform a preliminary investigation in order to determine visual trends of applying the MLC to hybrid model outputs.

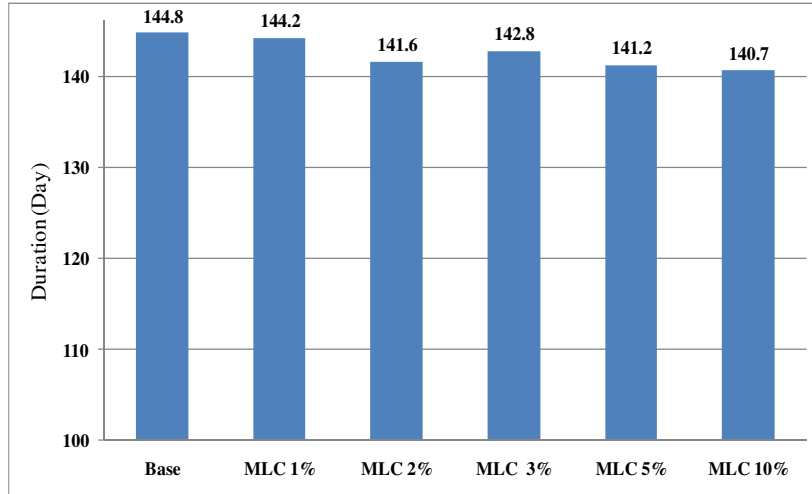


Figure 1: Comparative results for the fabrication duration of different scenarios

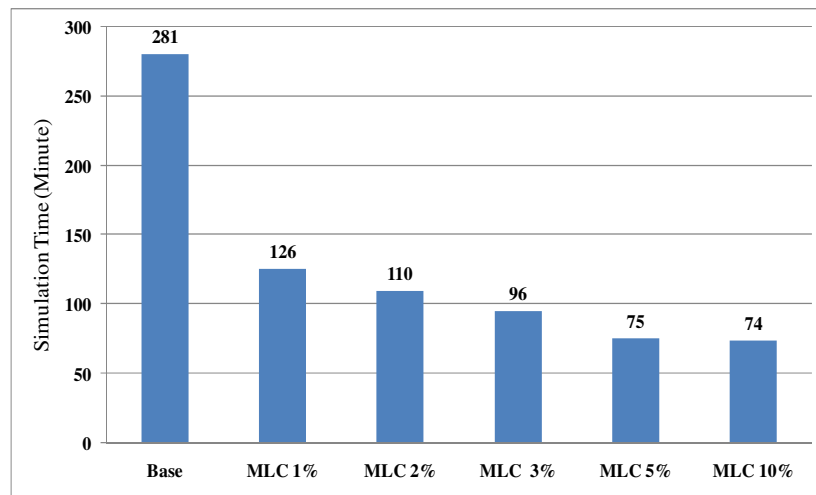


Figure 2: Simulation time of different scenarios In Figure 1, the achieved results for the duration time of steel fabrication show that the inaccuracy goes up by increasing the value of the MLC. All inaccuracies stay within the set MLC; the only exception is the MLC of 2%, as its difference with the base model goes up to 2.2%. This case can be explained based on the limited number of runs (i.e., five runs for each scenario). What is significant about the achieved results for the fabrication durations is that all of the MLC scenarios have shorter duration than

the achieved duration of the base hybrid model. This demonstrates that during the operation of the stations, the reported productivities have had a higher level than the real productivities, which has caused shorter durations for each single operation and consequently for the fabrication shop duration. In other words, the productivity changes have usually followed the declining direction. The declining direction for productivity during operation results from operators' fatigue and the station experiencing a busy time. However, growth in productivity usually occurs during the idle time of stations, which does not have any effect on the work duration (because there is no job to be done at that time).

Figure 2 presents the expected simulation time reduction by using the MLC concept. The simulation time shows a significant reduction in MLC based models compared to the base hybrid model. The trend of the results shows that by increasing the value of the MLC, a shorter simulation time is attained.

The comparisons conducted illustrate two different aspects of the set MLC values. The lesser the value selected for the MLC, the lesser accepted inaccuracy will be achieved, but with a greater simulation time. In the developed hybrid model, the MLC concept has been applied for one interacting variable (i.e., station productivity) and its inaccuracy is directly transferred to the work duration (see Section 2.5.2). Consequently, the set value for the MLC will stay at the upper limit of the expected inaccuracy. In this case, it is suggested that the MLC value be set to the maximum acceptable inaccuracy level for the output results to get the shortest possible simulation time while staying within the acceptable level of

inaccuracy. For example, if the maximum acceptable inaccuracy is 5%, the suggested MLC value for station productivity will be 5%.

6 Conclusion

The MLC concept is proposed as a solution for the existing time advancing issue in SD-DES hybrid models. It is While coming up with more rigorous conclusion for the conducted experiments requires more number of simulation runs, during this experiment we aim to visualize effects of using MLC on hybrid model outputs. It is claimed that the MLC is capable of considerably reducing the simulation time with insignificant effects on the accuracy of the final results. This paper assesses the theoretical and practical aspects of using the MLC in SD-DES hybrid models by applying the concept to a simplified hybrid model of a real steel fabrication shop. At this stage of the research, we have focused on investigating the trends of the achieved results. In future, a rigorous assessment with a greater number of simulation runs and a more detailed hybrid model will be conducted.

The trend of the results shows that using the MLC can increase the speed of SD-DES hybrid simulation. The significance of the effects of different set MLCs on the final results depends on the significance of the effects of their related variables. Thus, it is recommended that before setting values of MLCs, their effects first be theoretically estimated at the final simulation results. By considering the possible effects of any chosen value and by preventing the occurrence of a high level of inaccuracy, such theoretical investigations can build the confidence of model developers.

References

Alvanchi, A., S. Lee, and S. M. AbouRizk. 2009. Modeling Architecture for Hybrid System Dynamics and Discrete Event Simulation. Proceedings of the Construction Research Conference, Seattle, Apr. 2009

Fehlberg, E. 1970. New high-order Runge-Kutta formulas with an arbitrary small truncation error. *Computing*, 6, 61-71.

Lee, S., S. Han, and F. Peña-Mora. 2007. Hybrid system dynamics and discrete event simulation for construction management. ASCE International Workshop on Computing in Civil Engineering, Pittsburgh, PA.

Venkateswaran, J., Y. Son, and A. Jones. 2004. Hierarchical production planning using a hybrid system dynamic-discrete event simulation architecture. Proceedings of the 2004 Winter Simulation Conference, eds. R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 1094-1102. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zeigler, B. P., H. J. Cho, J. G. Kim, H. S. Sarjoughian, and J. S. Lee. 2002. Quantization-based filtering in distributed discrete event simulation. *Journal of Parallel and Distributed Computing*, 62, 1629-1647.

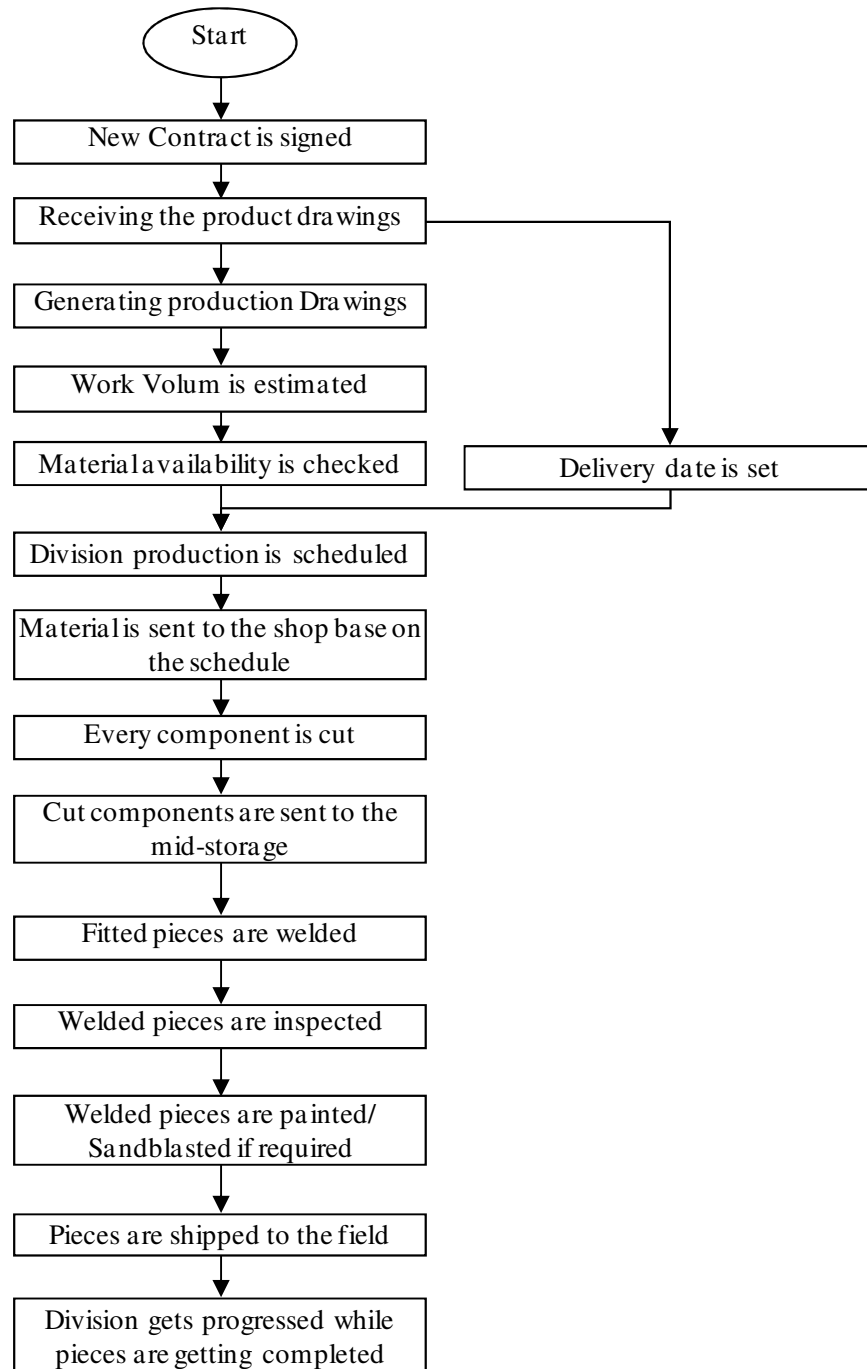
Appendix B

Programming Details of the Simulation Model Used for Hybrid Framework and Architecture Test

Model design, Visual Basic codes and data-tables used for developing the simulation model of the experimental case of structural steel construction hybrid model development in Section 2.5 of Chapter 2 are presented in this appendix.

B.1. Model Design

Development of an extensive hybrid model with a variety of building components requires a punctual design to be used in the implementation phase of the model development. As a result Unified Modeling Language (UML) (Rumbaugh et al. 1999) for object oriented design and programming was followed in the entire design and programming phases of the model development. At the first step, different structural steel construction operations and their interactions were summarized in a form of flowchart (Figure B.1). Then different objects within structural steel construction were recognized and their relationship (Figure B.2) and their hierarchical structure (Figure B.3) were extracted.



B.1. Structural steel construction operations summarized in a flowchart

Since at this stage of the model development structural steel fabrication process is modeled, a detailed object model relationship for fabrication shop was developed to be used for coding the object classes in the programming phase (Figure B.4).

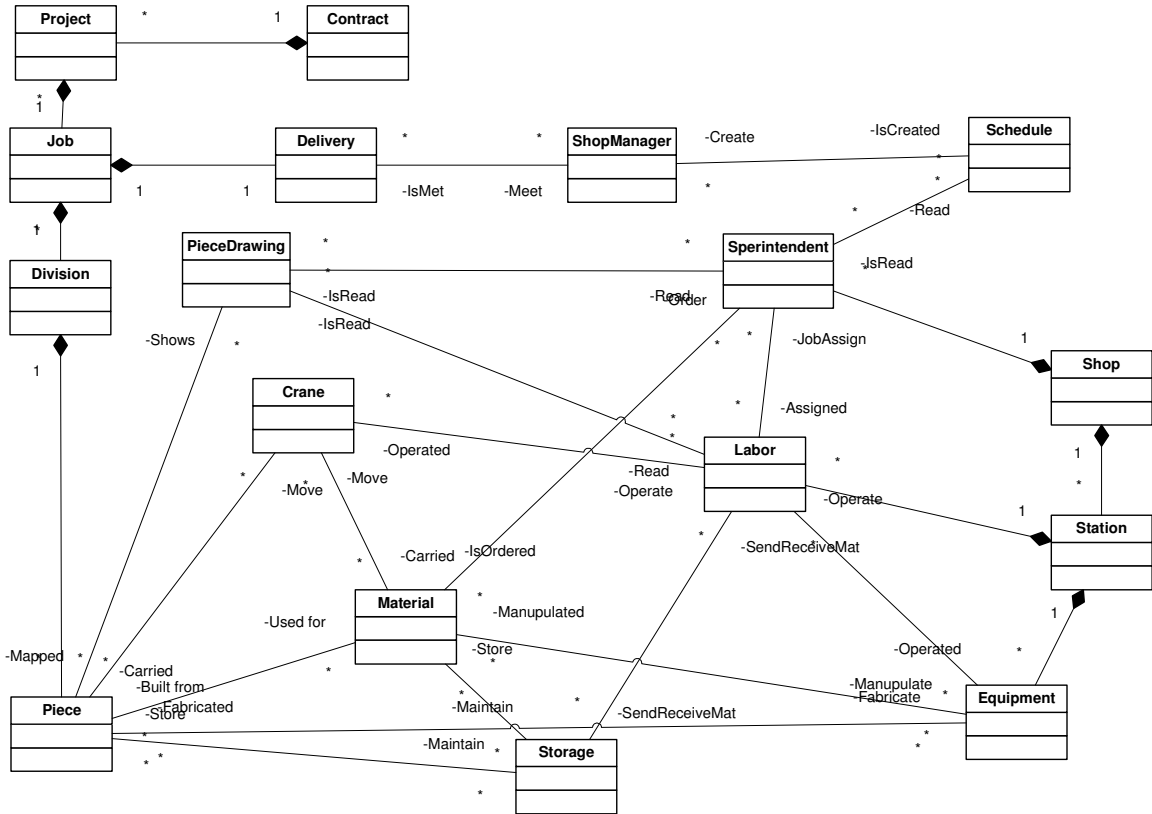


Figure B.4. Object model relationship for the structural steel fabrication shop
 Interactions among four initial federates required for developing fabrication shop federation (including DES model of the shop, SD model of the shop organization, calendar and data management) were summarized in a form of data flow diagram (Figure B.5). These interactions were used, at the next step, for drawing the Federation Object Model (FOM) (Table 2-2 in Chapter 2).

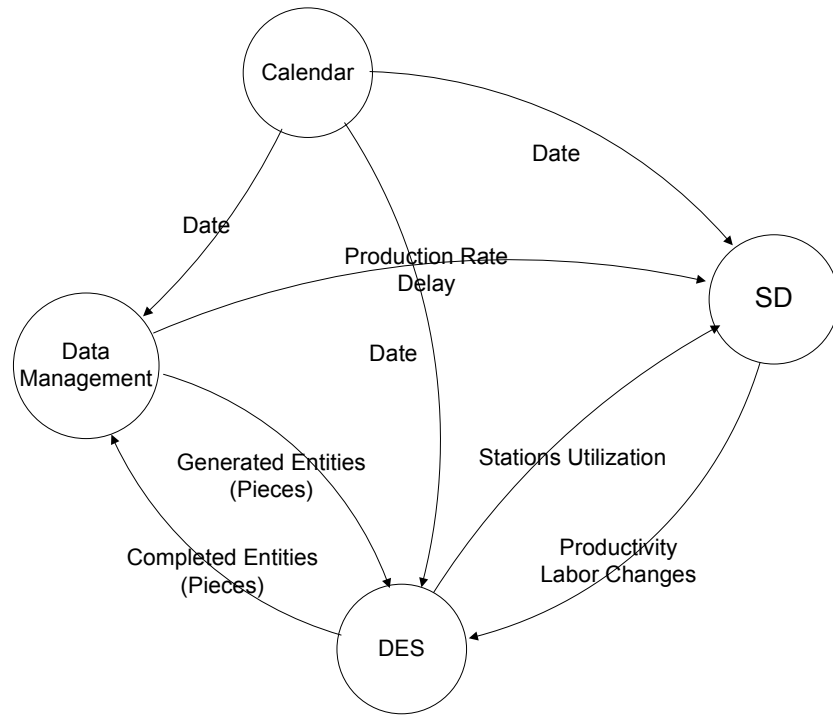


Figure B.5. Initial federates and their data flow

At the final stage of the model design and prior to the implementation phase, Simple flowcharts were developed for identifying different programming steps to be taken within every federate (Figures B.6, B.7 and B.8). The flowchart diagram of the SD model was not developed since the major portion of its programming parts include difference equations (or recursive functions) which are well established and easy to implement with general programming languages.

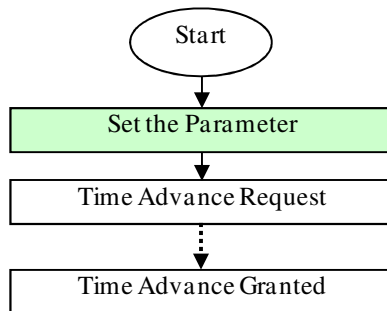


Figure B.6.Calendar federate procedure

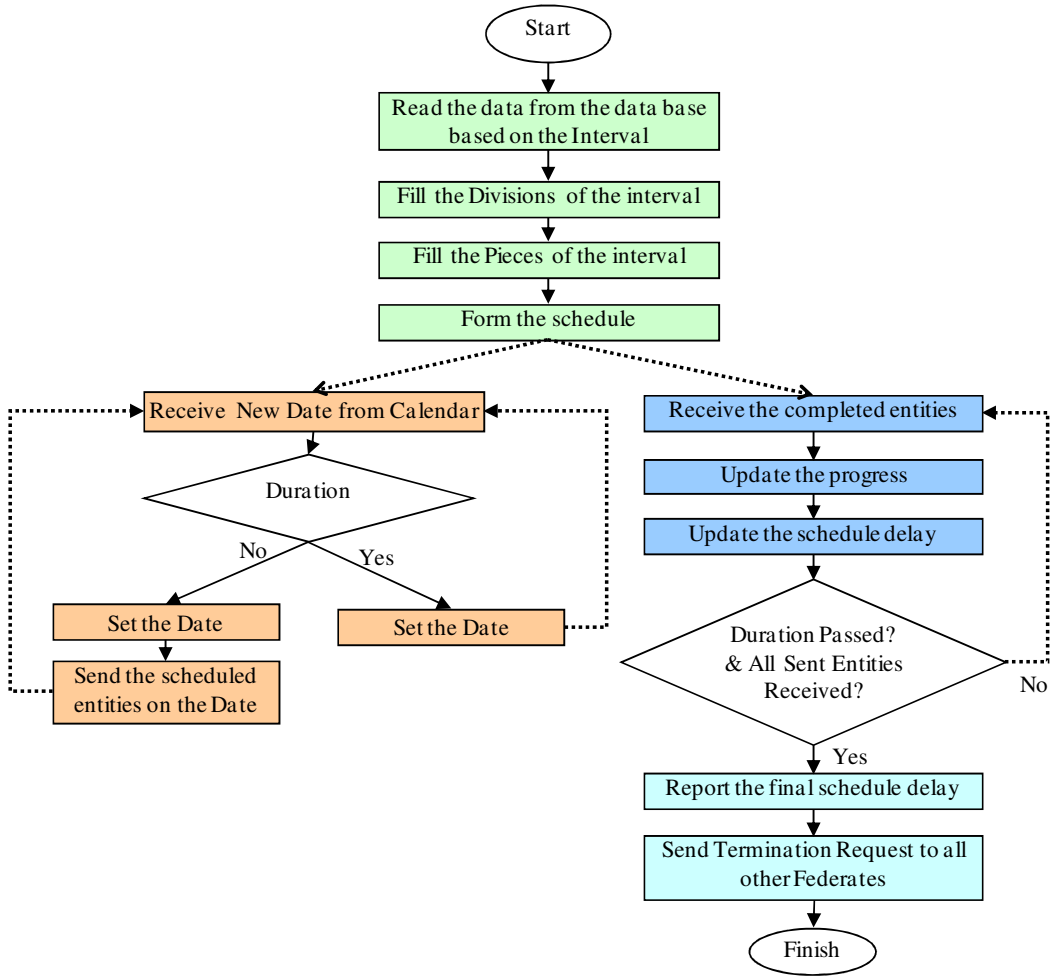


Figure B.7. Data management federate procedure

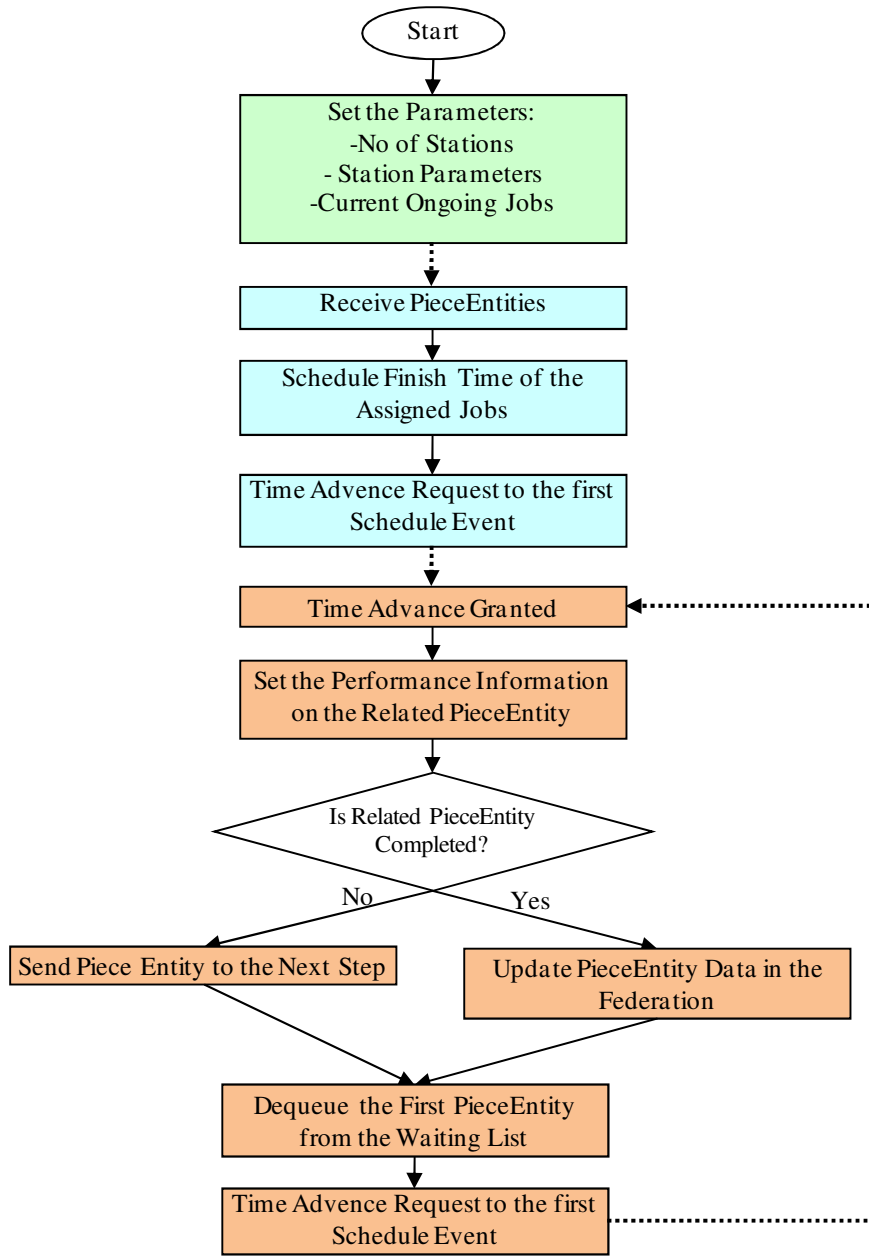


Figure B.8.DES federate procedure

Looking into the flowcharts presented in Figures B.6, B.7 and B.8, two main differences from regular flowcharts are noticeable. First, there is just one finish point in all three flowcharts which shows the termination of the entire federation happens at that point. Second, there are dashed arrows linking two flowchart task-boxes which represent broken relations between tasks. This means these tasks are

logically after each other, but time gaps might happen between them and program can get followed in other parts for a while before moving from first task to the second one.

B.2. Visual Basic Codes

The Visual Basic package of Visual Studio.NET 2008 has been used as the programming tool for developing different federates. In addition to the standard classes embedded in the Visual Basic, set of HLA related classes provided by COSYE framework (irc.construction.ualberta.ca/cosye/) also have been imported and used in the program. Set of general purpose discrete event simulation classes, provided by Symphony 3.5 (irc.construction.ualberta.ca/symphony35/) also have been imported for developing the DES federate. In following, the main body of the Visual Basic codes used for developing different federates of the structural steel federation in Section 2.5 of Chapter 2 are presented.

1) Work Calendar Federate

```
Imports Cosye.Hla.Rti
Public Class CalendarFederate
    Dim Holidays As New Collection
    Dim MyCalendar As Cosye.Steel.Steel_Calendar
    Dim WeekDayHours(6, 3) As Single
    Dim ShiftInterval(3) As ShiftType
    Dim CurrentShiftID As Integer = 1
    Dim CurrentDayLeftHours As Double = 24

    Private Sub ReadWeekDayHours()
        Try
            WeekDayHours(0, 1) = CSng(TxtSunWork.Text)
            WeekDayHours(0, 2) = CSng(TxtSunOver.Text)
            WeekDayHours(0, 3) = Math.Max(0, 24 - WeekDayHours(0, 1) - WeekDayHours(0, 2))
        Catch ex As Exception
            MessageBox.Show("Enter a valid Number for Sunday Hours!")
        End Try
        Try
            WeekDayHours(1, 1) = CSng(TxtMonWork.Text)
            WeekDayHours(1, 2) = CSng(TxtMonOver.Text)
            WeekDayHours(1, 3) = Math.Max(0, 24 - WeekDayHours(1, 1) - WeekDayHours(1, 2))
        Catch ex As Exception
```

```

    MsgBox.Show("Enter a valid Number for Monday Hours!")
End Try
Try
    WeekDayHours(2, 1) = CSng(TxtTueWork.Text)
    WeekDayHours(2, 2) = CSng(TxtTueOver.Text)
    WeekDayHours(2, 3) = Math.Max(0, 24 - WeekDayHours(2, 1) - WeekDayHours(2, 2))
Catch ex As Exception
    MsgBox.Show("Enter a valid Number for Tuesday Hours!")
End Try
Try
    WeekDayHours(3, 1) = CSng(TxtWedWork.Text)
    WeekDayHours(3, 2) = CSng(TxtWedOver.Text)
    WeekDayHours(3, 3) = Math.Max(0, 24 - WeekDayHours(3, 1) - WeekDayHours(3, 2))
Catch ex As Exception
    MsgBox.Show("Enter a valid Number for Wednesday Hours!")
End Try
Try
    WeekDayHours(4, 1) = CSng(TxtThuWork.Text)
    WeekDayHours(4, 2) = CSng(TxtThuOver.Text)
    WeekDayHours(4, 3) = Math.Max(0, 24 - WeekDayHours(4, 1) - WeekDayHours(4, 2))
Catch ex As Exception
    MsgBox.Show("Enter a valid Number for Thursday Hours!")
End Try
Try
    WeekDayHours(5, 1) = CSng(TxtFriWork.Text)
    WeekDayHours(5, 2) = CSng(TxtFriOver.Text)
    WeekDayHours(5, 3) = Math.Max(0, 24 - WeekDayHours(5, 1) - WeekDayHours(5, 2))
Catch ex As Exception
    MsgBox.Show("Enter a valid Number for Friday Hours!")
End Try
Try
    WeekDayHours(6, 1) = CSng(TxtSatWork.Text)
    WeekDayHours(6, 2) = CSng(TxtSatOver.Text)
    WeekDayHours(6, 3) = Math.Max(0, 24 - WeekDayHours(6, 1) - WeekDayHours(6, 2))
Catch ex As Exception
    MsgBox.Show("Enter a valid Number for Saturday Hours!")
End Try
End Sub

Private Sub ReadShiftInterval()
    ShifInterval(1) = ShiftType.DayShift
    Select Case CmbShift2.Text
        Case "Shift2"
            ShifInterval(2) = ShiftType.Shift2
        Case "MaxOverTime"
            ShifInterval(2) = ShiftType.OverTime
        Case "Close"
            ShifInterval(2) = ShiftType.Close
    End Select
    Select Case CmbShift3.Text
        Case "Shift3"
            ShifInterval(3) = ShiftType.Shift3
        Case "Close"
            ShifInterval(3) = ShiftType.Close
    End Select
End Sub

Private Sub BtnDelete_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnDelete.Click
    If LstHoliday.SelectedItems.Count > 0 Then
        Dim ToBeDeletedDate As New Date
        For i As Integer = LstHoliday.SelectedItems.Count - 1 To 0
            Holidays.Remove(CInt(LstHoliday.SelectedItems.Item(i)))
            LstHoliday.Items.Remove(LstHoliday.SelectedItems.Item(i))
        Next
    End If
End Sub

Private Sub BtnAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnAdd.Click
    'Check If the picked hodliday has not been added yet
    If Not Holidays.Contains(LstPickHoliday.SelectionStart.ToShortDateString) Then

```

```

Holidays.Add(LstPickHoliday.SelectionStart.Date, LstPickHoliday.SelectionStart.ToShortDateString)
If LstHoliday.Items.Count = 0 Then 'No need for sorting
    LstHoliday.Items.Add(LstPickHoliday.SelectionStart.ToShortDateString)
Else 'Item should be sorted
    Dim AddedToList As Boolean = False
    For i As Integer = 0 To LstHoliday.Items.Count - 1
        If Convert.ToDateTime(LstHoliday.Items(i)).Date > LstPickHoliday.SelectionStart.Date Then
            LstHoliday.Items.Insert(i, LstPickHoliday.SelectionStart.ToShortDateString)
            AddedToList = True
        End If
    Next
    'Add to the list as the last item
    If Not AddedToList Then
        LstHoliday.Items.Add(LstPickHoliday.SelectionStart.ToShortDateString)
    End If
End If
End If
End Sub

Dim LastWorkingDate As Date = Nothing
Private Sub ChangeTheShift()
    MyCalendar.CurrentShiftHours = 0
    While MyCalendar.CurrentShiftHours = 0
        If MyCalendar.CurrentShiftType = ShiftType.Close Or _
        MyCalendar.CurrentShiftType = ShiftType.Shift3 Then 'Next shift is in tomorrow
            MyCalendar.CurrentDate = DateAdd(DateInterval.Day, 1, MyCalendar.CurrentDate)
            'Check if tomorrow is holiday
            If Holidays.Contains(MyCalendar.CurrentDate.ToShortDateString) Then 'Set all day as Close
                MyCalendar.CurrentShiftType = ShiftType.Close
                MyCalendar.CurrentShiftHours = 24
                CurrentDayLeftHours = 0
            Else 'Start from first shift of the day
                ReadWeekDayHours()
                Dim DayInWeek As Integer = MyCalendar.CurrentDate.DayOfWeek
                CurrentShiftID = 1
                MyCalendar.CurrentShiftHours = WeekDayHours(DayInWeek, CurrentShiftID)
                MyCalendar.CurrentShiftType = ShiftInterval(CurrentShiftID)
                CurrentDayLeftHours = 24 - MyCalendar.CurrentShiftHours
            End If
        Else 'Next shift is in today
            Dim DayInWeek As Integer = MyCalendar.CurrentDate.DayOfWeek
            CurrentShiftID = CurrentShiftID + 1
            MyCalendar.CurrentShiftType = ShiftInterval(CurrentShiftID)
            'Set the over time base on the desired overtime and max overtime
            If MyCalendar.CurrentShiftType = ShiftType.OverTime Then
                Try
                    MyCalendar.CurrentShiftHours = Math.Min(MyCalendar.DesireOverTime, WeekDayHours(DayInWeek,
                    CurrentShiftID))
                Catch ex As Exception
                    MyCalendar.CurrentShiftHours = 0
                End Try
            ElseIf MyCalendar.CurrentShiftType = ShiftType.Close Then
                MyCalendar.CurrentShiftHours = Math.Max(CurrentDayLeftHours, WeekDayHours(DayInWeek,
                CurrentShiftID))
            Else
                MyCalendar.CurrentShiftHours = WeekDayHours(DayInWeek, CurrentShiftID)
            End If
            CurrentDayLeftHours = CurrentDayLeftHours - MyCalendar.CurrentShiftHours
        End If
    End While
    'Check if the working days have been counted
    If Not (MyCalendar.CurrentShiftType = ShiftType.Close Or LastWorkingDate = MyCalendar.CurrentDate) Then
        MyCalendar.DayNo = MyCalendar.DayNo + 1
    End If
End Sub

Private Sub MyCalendarFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyCalendarFactory.ReflectAttributeValues

```

```

'Sets the Parameters
ReadWeekDayHours()
ReadShiftInterval()
'Determine the Max possible OverTime
Dim MaxOvertime As Double = 0
For i As Integer = 1 To 7
    If WeekDayHours(i - 1, 2) > MaxOvertime Then
        MaxOvertime = WeekDayHours(i, 2)
    End If
Next
'Handle the update attributes
MyCalendar = MyCalendarFactory(e.theObject)
'Get the CurrentDate attribute handle
Dim CurDateHandle As AttributeHandle = MyCalendarFactory.GetAttributeHandle("CurrentDate")
'Just do it for the first time when Current time is defined
If e.theValues.Contains(CurDateHandle) Then
    MyCalendar.AttributeOwnershipAcquisition("CurrentDate", "CurrentShiftHours", "CurrentShiftType", "DayNo",
"MaxOverTime", "SetOverTimeForDay")
    MyCalendar.DayNo = 0
    CurrentShiftID = 0
    MyCalendar.MaxOverTime = MaxOvertime
    If Holidays.Contains(MyCalendar.CurrentDate.ToShortDateString) Then
        MyCalendar.CurrentShiftType = ShiftType.Close
        MyCalendar.CurrentShiftHours = 24
        CurrentDayLeftHours = 0
    Else
        Dim DayInWeek As Integer = MyCalendar.CurrentDate.DayOfWeek
        MyCalendar.CurrentShiftHours = 0
        'Change the shift until ShiftHours becomes > 0
        CurrentDayLeftHours = 24
        While MyCalendar.CurrentShiftHours = 0
            CurrentShiftID = CurrentShiftID + 1
            MyCalendar.CurrentShiftType = ShiftInterval(CurrentShiftID)
            If MyCalendar.CurrentShiftType = ShiftType.OverTime Then
                Try
                    MyCalendar.CurrentShiftHours = Math.Min(MyCalendar.DesireOverTime, WeekDayHours(DayInWeek,
CurrentShiftID))
                Catch ex As Exception
                    MyCalendar.CurrentShiftHours = 0
                End Try
            ElseIf MyCalendar.CurrentShiftType = ShiftType.Close Then
                MyCalendar.CurrentShiftHours = Math.Max(CurrentDayLeftHours, WeekDayHours(DayInWeek,
CurrentShiftID))
            Else
                MyCalendar.CurrentShiftHours = WeekDayHours(DayInWeek, CurrentShiftID)
            End If
        End While
        CurrentDayLeftHours = CurrentDayLeftHours - MyCalendar.CurrentShiftHours
    End If
    MyCalendar.UpdateAttributeValues()
End If
'Handle the update attributes
Dim DesireOverTimeHandle As AttributeHandle = MyCalendarFactory.GetAttributeHandle("DesireOverTime")
'Get the DesireOverTime attribute handle
If e.theValues.Contains(DesireOverTimeHandle) Then
    If MyCalendar.DesireOverTime > 0 Then
        MyCalendar.SetOverTimeForDay = Math.Min(MyCalendar.DesireOverTime,
WeekDayHours(MyCalendar.CurrentDate.DayOfWeek, 2))
    Else
        MyCalendar.SetOverTimeForDay = 0
    End If
    'Just for resolving possible the ownership
    Dim GainedOwnership As Boolean = False
    'Loop until ownership is granted
    While Not GainedOwnership
        Try
            MyCalendar.UpdateAttributeValues()
            GainedOwnership = True
        Catch ex As Exception
    
```

```

        End Try
    End While
End If
'Update the Interface
Try
    TxtCurDate.Text = MyCalendar.CurrentDate.ToShortDateString
    TxtShift.Text = [Enum].GetName(GetType(Cosye.Steel.ShiftType), MyCalendar.CurrentShiftType)
Catch ex As Exception
    'No need for interface update at this time
End Try
End Sub

Private Sub fedAmb_TimeAdvanceGrant(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.TimeAdvanceGrantEventArgs) Handles fedAmb.TimeAdvanceGrant
    'Update the Interface
    TxtCurDate.Text = MyCalendar.CurrentDate.ToShortDateString
    TxtShift.Text = [Enum].GetName(GetType(Cosye.Steel.ShiftType), MyCalendar.CurrentShiftType)
    'Store the current shift's Hours
    Dim CurrentShiftHours As Double = MyCalendar.CurrentShiftHours
    'Prepare the parameters of the next shift
    ChangeTheShift()
    'Update the next shift when current shift gets over
    MyCalendar.UpdateAttributeValues(e.theTime + CurrentShiftHours * 3600)
    'Time Advance Request when current shift gets over
    rtiAmb.TimeAdvanceRequest(e.theTime + CurrentShiftHours * 3600)
End Sub
End Class

```

2) Data Management Federate

```

Imports Symphony.Mathematics
Imports Cosye.Hla.Rti

Public Class DataManagement
    'Dim PieceEntityItem As PieceEntity
    Dim Cutting, Fitting, FitInspection, Welding, WeldInspection, Painting As Activity
    Dim NoPieceReceived As Integer = 0
    Dim MyShopProductivity As Steel_ShopProductivity
    Dim SimulationStartTime As New Date
    Dim MyDataEnvironment As New DataEnvironment
    Dim DivisionsProgressList As New Dictionary(Of String, String)
    Dim InShopPieceSent As Boolean = False

    Public Sub SetParameters()
        ""Date Related Parameters
        Try
            MyDataEnvironment.DataImportDuration = Val(TxtDuration.Text)
        Catch ex As Exception
            MessageBox.Show("Enter a valid Number for Simulation Duration!")
        End Try
        ""Activity Related Parameters
        Try
            Cutting.ActivityMod = Convert.ToDouble(TxtCutMod.Text)
            Cutting.ActivityMin = Convert.ToDouble(TxtCutMin.Text)
            Cutting.ActivityMax = Convert.ToDouble(TxtCutMax.Text)
        Catch ex As Exception
            MessageBox.Show("Enter a valid Number For Cutting!")
        End Try
        Try
            Fitting.ActivityMod = Convert.ToDouble(TxtFitMod.Text)
            Fitting.ActivityMin = Convert.ToDouble(TxtFitMin.Text)
            Fitting.ActivityMax = Convert.ToDouble(TxtFitMax.Text)
        Catch ex As Exception
            MessageBox.Show("Enter a valid Number For Fitting!")
        End Try
        Try
            Welding.ActivityMod = Convert.ToDouble(TxtWeldMod.Text)

```



```

        Welding.ActivityMin = Convert.ToDouble(TxtWeldMin.Text)
        Welding.ActivityMax = Convert.ToDouble(TxtWeldMax.Text)
    Catch ex As Exception
        MessageBox.Show("Enter a valid Number For Welding!")
    End Try
Try
    FitInspection.ActivityMod = Val(TxtInspectMod.Text) / 2
    FitInspection.ActivityMin = Val(TxtInspectMin.Text) / 2
    FitInspection.ActivityMax = Val(TxtInspectMax.Text) / 2
Catch ex As Exception
    MessageBox.Show("Enter a valid Number For Inspection!")
End Try
Try
    WeldInspection.ActivityMod = Val(TxtInspectMod.Text) / 2
    WeldInspection.ActivityMin = Val(TxtInspectMin.Text) / 2
    WeldInspection.ActivityMax = Val(TxtInspectMax.Text) / 2
Catch ex As Exception
    MessageBox.Show("Enter a valid Number For Inspection!")
End Try
Try
    Painting.ActivityMod = Convert.ToDouble(TxtPaintMod.Text)
    Painting.ActivityMin = Convert.ToDouble(TxtPaintMin.Text)
    Painting.ActivityMax = Convert.ToDouble(TxtPaintMax.Text)
Catch ex As Exception
    MessageBox.Show("Enter a valid Number For Painting!")
End Try
'Set the Simulation Start Time
LblSimStart.Text = Date.Now.TimeOfDay.ToString
SimulationStartTime = Date.Now
End Sub
" """"""""Sends Entities
Private Sub SendEntities(ByVal SendDate As Date, ByVal SendingTime As Double)
    'Import the Data for the Date
    MyDataEnvironment.ImportData(SendDate)
    Dim DivisionItem As Division
    'Send the imported pieces for today
    For Each PieceItem As Piece In MyDataEnvironment.TodayPieceList.Values
        DivisionItem = New Division
        DivisionItem = MyDataEnvironment.DivisionList(PieceItem.DivisionID)
        Dim PaintingRequirePortion As Double = 1
        Dim PiecePaintingActivityDurationPortion As Double = 1 'Is set to calculate Painting Portion
        If Not DivisionItem.PaintRequire Then
            PaintingRequirePortion = 100 / (100 - Painting.ActivityMod)
            PiecePaintingActivityDurationPortion = 0
        End If
        Dim PieceActivityDuration As Double = Math.Min(Math.Max(DivisionItem.FabManHour * PieceItem.Weight,
200), 5 * 3600)
        PiecePaintingActivityDurationPortion = PiecePaintingActivityDurationPortion * PieceActivityDuration
        Dim RFIDTagCount As Integer = 0
        If MyDataEnvironment.InShopPieceList.Keys.Contains(PieceItem.PieceID) Then
            RFIDTagCount = PieceItem.RFIDList.Count
        End If
        For j As Integer = 1 To PieceItem.Quantity - RFIDTagCount
            Dim NewPieceEntity As Steel_PieceEntity = MyPieceEntityFactory.RegisterObjectInstance()
            NewPieceEntity.PieceID = PieceItem.PieceID
            NewPieceEntity.StartDate = SendDate.Date
            NewPieceEntity.Weight = PieceItem.Weight
            NewPieceEntity.DimentionLevel = Convert.ToInt32(PieceItem.Weight)
            NewPieceEntity.CuttingFinish = Nothing
            NewPieceEntity.CuttingManHour = PieceActivityDuration * Cutting.DurationPortion * PaintingRequirePortion
            NewPieceEntity.WeldingManHour = PieceActivityDuration * Welding.DurationPortion *
PaintingRequirePortion
            NewPieceEntity.FittingManHour = PieceActivityDuration * Fitting.DurationPortion * PaintingRequirePortion
            NewPieceEntity.FitInspectionManHour = PieceActivityDuration * FitInspection.DurationPortion *
PaintingRequirePortion
            NewPieceEntity.WeldInspectionManHour = PieceActivityDuration * WeldInspection.DurationPortion *
PaintingRequirePortion
            NewPieceEntity.PaintingManHour = PiecePaintingActivityDurationPortion * Painting.DurationPortion *
PaintingRequirePortion
            If SendingTime > 0 Then

```

```

        NewPieceEntity.UpdateAttributeValues(SendingTime)
        NewPieceEntity.UnconditionalAttributeOwnershipDivestiture("CuttingManHour", "FittingManHour",
"WeldingManHour", "FitInspectionManHour", "WeldInspectionManHour", "PaintingManHour", _
        "CuttingFinish", "FittingFinish", "WeldingFinish", "FitInspectionFinish", "WeldInspectionFinish",
"PaintingFinish", _
        "CuttingStart", "FittingStart", "WeldingStart", "FitInspectionStart", "WeldInspectionStart", "PaintingStart")
    Else
        NewPieceEntity.UpdateAttributeValues()
        NewPieceEntity.UnconditionalAttributeOwnershipDivestiture("CuttingManHour", "FittingManHour",
"WeldingManHour", "FitInspectionManHour", "WeldInspectionManHour", "PaintingManHour", _
        "CuttingFinish", "FittingFinish", "WeldingFinish", "FitInspectionFinish", "WeldInspectionFinish",
"PaintingFinish", _
        "CuttingStart", "FittingStart", "WeldingStart", "FitInspectionStart", "WeldInspectionStart", "PaintingStart")
    End If
Next
Next
End Sub

" """"""""Sends In-Shop Entities
Private Sub SendInShopEntities(ByVal SendDate As Date)
'Import the Data for the Date
MyDataEnvironment.ImportInShopPieces(SendDate)
Dim DivisionItem As Division
Dim MaxNumberOfRFIDUpdate As Integer = 3
'Send the imported pieces for today
For Each PieceItem As Piece In MyDataEnvironment.InShopPieceList.Values
    DivisionItem = New Division
    DivisionItem = MyDataEnvironment.DivisionList(PieceItem.DivisionID)
    Dim PaintingRequirePortion As Double = 1
    Dim PiecePaintingActivityDurationPortion As Double = 1 'Is set to calculate Painting Portion
    If Not DivisionItem.PaintRequire Then
        PaintingRequirePortion = 100 / (100 - Painting.ActivityMod)
        PiecePaintingActivityDurationPortion = 0
    Else
        MaxNumberOfRFIDUpdate = 1 + MaxNumberOfRFIDUpdate
    End If
    Dim PieceActivityDuration As Double = DivisionItem.FabManHour * PieceItem.Portion
    PiecePaintingActivityDurationPortion = PiecePaintingActivityDurationPortion * PieceActivityDuration
    For j As Integer = 0 To PieceItem.RFIDList.Count - 1
        Dim RFIDItem As New RFID
        RFIDItem = PieceItem.RFIDList(PieceItem.RFIDList.Keys(j))
        If MaxNumberOfRFIDUpdate > RFIDItem.RFIDCount Then
            Dim NewPieceEntity As Steel_PieceEntity = MyPieceEntityFactory.RegisterObjectInstance()
            NewPieceEntity.PieceID = PieceItem.PieceID
            NewPieceEntity.StartDate = SendDate.Date
            NewPieceEntity.Weight = PieceItem.Weight
            NewPieceEntity.DimentionLevel = Convert.ToInt32(PieceItem.Weight)
            NewPieceEntity.CuttingManHour = PieceActivityDuration * Cutting.DurationPortion *
PaintingRequirePortion
            NewPieceEntity.WeldingManHour = PieceActivityDuration * Welding.DurationPortion *
PaintingRequirePortion
            NewPieceEntity.FittingManHour = PieceActivityDuration * Fitting.DurationPortion * PaintingRequirePortion
            NewPieceEntity.FitInspectionManHour = PieceActivityDuration * FitInspection.DurationPortion *
PaintingRequirePortion
            NewPieceEntity.WeldInspectionManHour = PieceActivityDuration * WeldInspection.DurationPortion *
PaintingRequirePortion
            NewPieceEntity.PaintingManHour = PiecePaintingActivityDurationPortion * Painting.DurationPortion *
PaintingRequirePortion
            'Reset the values of the activity start and finish
            NewPieceEntity.CuttingFinish = Nothing
            NewPieceEntity.FittingFinish = Nothing
            NewPieceEntity.FitInspectionFinish = Nothing
            NewPieceEntity.WeldingFinish = Nothing
            NewPieceEntity.WeldInspectionFinish = Nothing
            NewPieceEntity.PaintingFinish = Nothing
            NewPieceEntity.CuttingStart = Nothing
            NewPieceEntity.FittingStart = Nothing
            NewPieceEntity.FitInspectionStart = Nothing
            NewPieceEntity.WeldingStart = Nothing
            NewPieceEntity.WeldInspectionStart = Nothing

```

```

NewPieceEntity.PaintingStart = Nothing

Select Case RFIDItem.RFIDCount
Case 1
    NewPieceEntity.CuttingFinish = RFIDItem.MinDate
Case 2
    NewPieceEntity.CuttingFinish = RFIDItem.MinDate
    NewPieceEntity.FittingFinish = RFIDItem.MaxDate
    NewPieceEntity.FitInspectionFinish = RFIDItem.MaxDate
Case 3
    NewPieceEntity.CuttingFinish = RFIDItem.MinDate
    NewPieceEntity.FittingFinish = RFIDItem.MinDate
    NewPieceEntity.FitInspectionFinish = RFIDItem.MinDate
    NewPieceEntity.WeldingStart = RFIDItem.MaxDate
Case 4
    NewPieceEntity.CuttingFinish = RFIDItem.MinDate
    NewPieceEntity.FittingFinish = RFIDItem.MinDate
    NewPieceEntity.FitInspectionFinish = RFIDItem.MinDate
    NewPieceEntity.WeldingFinish = RFIDItem.MaxDate
    NewPieceEntity.WeldInspectionFinish = RFIDItem.MaxDate
Case 5
    NewPieceEntity.CuttingFinish = RFIDItem.MinDate
    NewPieceEntity.FittingFinish = RFIDItem.MinDate
    NewPieceEntity.FitInspectionFinish = RFIDItem.MinDate
    NewPieceEntity.WeldingFinish = RFIDItem.MaxDate
    NewPieceEntity.WeldInspectionFinish = RFIDItem.MaxDate
    NewPieceEntity.PaintingStart = RFIDItem.MaxDate
End Select
NewPieceEntity.UpdateAttributeValues()
NewPieceEntity.UnconditionalAttributeOwnershipDivestiture("CuttingManHour", "FittingManHour",
"WeldingManHour", "FitInspectionManHour", "WeldInspectionManHour", "PaintingManHour", _
    "CuttingFinish", "FittingFinish", "WeldingFinish", "FitInspectionFinish", "WeldInspectionFinish",
"PaintingFinish", _
    "CuttingStart", "FittingStart", "WeldingStart", "FitInspectionStart", "WeldInspectionStart", "PaintingStart")
Else 'MaxNumberOfRFIDUpdate <= RFIDItem.RFIDCount
    "Current piece progress"
    PieceItem.CompletedPieces = 1 + PieceItem.CompletedPieces
End If
Next
Next
End Sub

Private Sub MyPieceEntityFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyPieceEntityFactory.ReflectAttributeValues
    Dim CompletedPieceEntity As Steel_PieceEntity = MyPieceEntityFactory(e.theObject)
    'Check if the Piece Exists'"Sometimes the piece will be ignored if its weight is Zero
    If MyDataEnvironment.PieceList.Keys.Contains(CompletedPieceEntity.PieceID) Then
        Dim RelatedPiece As Piece = MyDataEnvironment.PieceList(CompletedPieceEntity.PieceID)
        Dim RelatedDivision As Division = MyDataEnvironment.DivisionList(RelatedPiece.DivisionID)
        'Calculate and send piece CPI at piece level
        "Current piece progress"
        RelatedPiece.CompletedPieces = 1 + RelatedPiece.CompletedPieces
        Dim PieceProgress As Double = RelatedPiece.CompletedPieces / RelatedPiece.Quantity
        "Piece CPI"
        Dim PieceSpentHour As Double = (CompletedPieceEntity.CuttingManHour + _
            CompletedPieceEntity.FittingManHour + _
            CompletedPieceEntity.WeldingManHour + _
            CompletedPieceEntity.FitInspectionManHour + _
            CompletedPieceEntity.WeldInspectionManHour + _
            CompletedPieceEntity.PaintingManHour) ' In Second
        RelatedPiece.TotalSpentHours = PieceSpentHour + RelatedPiece.TotalSpentHours
        RelatedDivision.TotalSpentHours = PieceSpentHour + RelatedDivision.TotalSpentHours
        Dim PieceEarnedHour As Double = RelatedPiece.CompletedPieces * RelatedDivision.FabManHour _
            * RelatedPiece.Portion 'In second
        Dim PieceCPI = PieceEarnedHour / RelatedPiece.TotalSpentHours
        'Check the Division CPI and Both piece and division SPI
        "Check the start date"
        If RelatedDivision.Progress = 0 Then
            If Not (CompletedPieceEntity.CuttingStart = Nothing) Then
                RelatedDivision.Start = CompletedPieceEntity.CuttingStart
            End If
        End If
    End If
End Sub

```

```

Else
    RelatedDivision.Start = MyDataEnvironment.CurrentDate
End If
End If
'Update No of Pieces
NoPieceReceived = NoPieceReceived + 1
RelatedDivision.CompletedPieces = RelatedDivision.CompletedPieces + 1
'Update the Piece ListBox
LblCompPiece.Text = NoPieceReceived.ToString
'Update the current earned progress at the division level
RelatedDivision.Progress = Math.Min(1, RelatedDivision.Progress + RelatedPiece.Portion)
" Calculate Division CPI
Dim DivisionCPI As Double = 1
If RelatedDivision.TotalSpentHours > 0 Then
    Dim DivisionEarnedManhour As Double = RelatedDivision.Progress * RelatedDivision.FabManHour
    DivisionCPI = DivisionEarnedManhour / RelatedDivision.TotalSpentHours
End If
'Calculate Scheduled Progress
Dim SchPassedDays As Double
If RelatedDivision.Start < MyDataEnvironment.CurrentDate Then
    SchPassedDays = Math.Max(0, DateDiff(DateInterval.Day, RelatedDivision.Start,
MyDataEnvironment.CurrentDate))
Else
    SchPassedDays = 0
End If
Dim SchTotalDays As Double = Math.Max(1, DateDiff(DateInterval.Day, RelatedDivision.Start,
RelatedDivision.Required))
Dim SchProgress As Double = Math.Min(1, SchPassedDays / SchTotalDays)
'Calculate SPI
Dim DivisionPieceSPI As Double
If SchProgress = 0 Then
    DivisionPieceSPI = 1
Else
    DivisionPieceSPI = RelatedDivision.Progress / SchProgress
End If
'Reflec latest CPI and SPI to the related Division
RelatedDivision.CPI = DivisionCPI
RelatedDivision.SPI = DivisionPieceSPI
'Update VPiece
Dim MyVpiece As Steel_VPiece = MyVPieceFactory.RegisterObjectInstance()
MyVpiece.DivisionID = RelatedDivision.DivisionID
MyVpiece.PieceKey = RelatedPiece.PieceKey
MyVpiece.CPI = PieceCPI
MyVpiece.SPI = DivisionPieceSPI
MyVpiece.Progress = PieceProgress
MyVpiece.UpdateAttributeValues()
'Report the latest achieved progress
Dim DivPreviousProgress As String
If DivisionsProgressList.Keys.Contains(RelatedDivision.DivisionID) Then
    'Remove the previous progress and add the new one
    DivPreviousProgress = DivisionsProgressList(RelatedDivision.DivisionID)
    DivisionsProgressList.Remove(RelatedDivision.DivisionID)
    LstCompDivision.Items.Remove(DivPreviousProgress)
    DivisionsProgressList.Add(RelatedDivision.DivisionID, RelatedDivision.ToString)
    LstCompDivision.Items.Add(RelatedDivision.ToString)
Else 'Just add the new progress
    DivisionsProgressList.Add(RelatedDivision.DivisionID, RelatedDivision.ToString)
    LstCompDivision.Items.Add(RelatedDivision.ToString)
End If
Me.Refresh()
" Add number of completed divisions if division completed
If Math.Round(RelatedDivision.Progress, 3) >= 1 And Not
MyDataEnvironment.CompletedDivisions.Contains(RelatedDivision.DivisionID) Then
    MyDataEnvironment.CompletedDivisions.Add(RelatedDivision.DivisionID)
End If
End If 'Piece Exists
>Delete the Piece from the RTI
MyPieceEntityFactory.DeleteObjectInstance(CompletedPieceEntity)
End Sub

```

```

Private Sub MyCalendarFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyCalendarFactory.ReflectAttributeValues
Dim DateHandle As Cosye.Hla.Rti.AttributeHandle = MyCalendarFactory.GetAttributeHandle("CurrentDate")
Dim NewDay As Steel_Calendar = MyCalendarFactory(e.theObject)
'Move the day ahead and check the scheduled pieces
MyDataEnvironment.CurrentDate = NewDay.CurrentDate
MyDataEnvironment.WorkingDays = NewDay.DayNo
'Check if the day no has been changed/ New day has been started
If e.theValues.Contains(DateHandle) And NewDay.CurrentDate <= MyDataEnvironment.FinishDate Then
'Send the Scheduled pieces for this day if any existed
SendEntities(MyDataEnvironment.CurrentDate, e.theTime + 1)
'Calculate the Delay
CalculateCurrentDelay()
'Update the Delay
MyShopProductivity.TotalDelay = MyDataEnvironment.MySchedule.TotalDelay
MyShopProductivity.DelayRate = MyDataEnvironment.DelayRate
MyShopProductivity.UpdateAttributeValues()
'Set the Values on screen
SetInterface()
End If
End Sub
'Calculate and set delays
Private Sub CalculateCurrentDelay()
Dim CurrentTotalDelay As New Delay
'Calculate current total delay
For Each DivisionItem As Division In MyDataEnvironment.DivisionList.Values
If DivisionItem.Progress < 1 And DivisionItem.Required < MyDataEnvironment.CurrentDate And Not
DivisionItem.Required = Nothing Then
DivisionItem.Delay = Math.Min(0, DateDiff(DateInterval.Day, MyDataEnvironment.CurrentDate,
DivisionItem.Required))
End If
CurrentTotalDelay.Delays = CurrentTotalDelay.Delays + DivisionItem.Delay
CurrentTotalDelay.WeightedDelays = CurrentTotalDelay.WeightedDelays + DivisionItem.WeightedDelay
Next
'Assign the calculated delay
MyDataEnvironment.SetDelay(CurrentTotalDelay)
End Sub

Private Sub MyCalendarFactory_InitializeInitialInstances(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyCalendarFactory.InitializeInitialInstances
'Sets the Entered Paramers
SetParameters()
'Reads the start date and sets it for My Data Environment
MyDataEnvironment.StartDate = (CmbStart.Value.Date)
Dim DataImportDuration As Double = Val(TxtDuration.Text)
MyDataEnvironment.FinishDate = DateAdd(DateInterval.Month, DataImportDuration,
MyDataEnvironment.StartDate)
MyDataEnvironment.FinishDate = DateAdd(DateInterval.Day, (DataImportDuration Mod 1) * 30,
MyDataEnvironment.FinishDate)
MyDataEnvironment.CurrentDate = MyDataEnvironment.StartDate
'Set the entered date as the first day of Federation
Dim StartDay As Steel_Calendar = MyCalendarFactory.RegisterObjectInstance()
StartDay.StartDate = MyDataEnvironment.StartDate
StartDay.CurrentDate = MyDataEnvironment.CurrentDate
StartDay.SetOverTimeForDay = 0
StartDay.UpdateAttributeValues()
'Divest the ownership of the calendar attribute
StartDay.UnconditionalAttributeOwnershipDivestiture("CurrentShiftType", "CurrentShiftHours", "CurrentDate",
"SetOverTimeForDay")
'Send in shop entities
If Not InShopPieceSent Then
SendInShopEntities(MyDataEnvironment.CurrentDate)
InShopPieceSent = True
End If
'Send initial piece entities
SendEntities(MyDataEnvironment.CurrentDate, 0)
'Set Interface
SetInterface()
End Sub

```



```

CmbPaint.Text = MyDataShop.FillMainFormControl("CmbPaint")
CmbShop.Text = MyDataShop.FillMainFormControl("CmbShop")
TxtMidBufferNum.Text = MyDataShop.FillMainFormControl("TxtMidBufferNum")
TxtMoverNum.Text = MyDataShop.FillMainFormControl("TxtMoverNum")

```

```

" """"""""Initialize the Buttons

```

```

'Form Cutting Collection

```

```

FabShopControl(1) = New List(Of Control)
FabShopControl(1).Add(Me.BtnCut1)
FabShopControl(1).Add(Me.BtnCut2)
FabShopControl(1).Add(Me.BtnCut3)
FabShopControl(1).Add(Me.BtnCut4)
FabShopControl(1).Add(Me.BtnCut5)
FabShopControl(1).Add(Me.BtnCut6)
FabShopControl(1).Add(Me.BtnCut7)
FabShopControl(1).Add(Me.BtnCut8)
SetVisibilityControl(Convert.ToInt32(CmbCut.Text), FabShopControl(1))

```

```

'Form Fitting Collection

```

```

FabShopControl(2) = New List(Of Control)
FabShopControl(2).Add(Me.BtnFit1)
FabShopControl(2).Add(Me.BtnFit2)
FabShopControl(2).Add(Me.BtnFit3)
FabShopControl(2).Add(Me.BtnFit4)
FabShopControl(2).Add(Me.BtnFit5)
FabShopControl(2).Add(Me.BtnFit6)
FabShopControl(2).Add(Me.BtnFit7)
FabShopControl(2).Add(Me.BtnFit8)
SetVisibilityControl(Convert.ToInt32(CmbShop.Text), FabShopControl(2))

```

```

'Form Welding Collection

```

```

FabShopControl(3) = New List(Of Control)
FabShopControl(3).Add(Me.BtnWeld1)
FabShopControl(3).Add(Me.BtnWeld2)
FabShopControl(3).Add(Me.BtnWeld3)
FabShopControl(3).Add(Me.BtnWeld4)
FabShopControl(3).Add(Me.BtnWeld5)
FabShopControl(3).Add(Me.BtnWeld6)
FabShopControl(3).Add(Me.BtnWeld7)
FabShopControl(3).Add(Me.BtnWeld8)
SetVisibilityControl(Convert.ToInt32(CmbShop.Text), FabShopControl(3))

```

```

'Form Inspection Collection

```

```

FabShopControl(4) = New List(Of Control)
FabShopControl(4).Add(Me.BtnInspect1)
FabShopControl(4).Add(Me.BtnInspect2)
FabShopControl(4).Add(Me.BtnInspect3)
FabShopControl(4).Add(Me.BtnInspect4)
FabShopControl(4).Add(Me.BtnInspect5)
FabShopControl(4).Add(Me.BtnInspect6)
FabShopControl(4).Add(Me.BtnInspect7)
FabShopControl(4).Add(Me.BtnInspect8)
SetVisibilityControl(Convert.ToInt32(CmbInspect.Text), FabShopControl(4))

```

```

'Form Painting Collection

```

```

FabShopControl(5) = New List(Of Control)
FabShopControl(5).Add(Me.BtnPaint1)
FabShopControl(5).Add(Me.BtnPaint2)
FabShopControl(5).Add(Me.BtnPaint3)
FabShopControl(5).Add(Me.BtnPaint4)
FabShopControl(5).Add(Me.BtnPaint5)
FabShopControl(5).Add(Me.BtnPaint6)
FabShopControl(5).Add(Me.BtnPaint7)
FabShopControl(5).Add(Me.BtnPaint8)
SetVisibilityControl(Convert.ToInt32(CmbPaint.Text), FabShopControl(5))

```

```

'Form Mover Collection

```

```

FabShopControl(6) = New List(Of Control)
FabShopControl(6).Add(Me.BtnMover50)
FabShopControl(6).Add(Me.BtnMover51)
FabShopControl(6).Add(Me.BtnMover52)
FabShopControl(6).Add(Me.BtnMover53)
FabShopControl(6).Add(Me.BtnMover54)
FabShopControl(6).Add(Me.BtnMover55)
FabShopControl(6).Add(Me.BtnMover56)

```

```

'FabShopControl(6).Add(Me.BtnMover57)
Dim MoverNum As Integer = Convert.ToInt32((Math.Min(Val(TxtMoverNum.Text), 8)))
'SetVisibilityControl(MoverNum, FabShopControl(6))
Dim MidBufNum As Integer = Convert.ToInt32(Math.Min(Val(TxtMidBufferNum.Text) + 1, 12))
'Form MidBuffer Lable Collection
FabShopControl(7) = New List(Of Control)
FabShopControl(7).Add(Me.LblMidBuf70)
FabShopControl(7).Add(Me.LblMidBuf71)
FabShopControl(7).Add(Me.LblMidBuf72)
FabShopControl(7).Add(Me.LblMidBuf73)
FabShopControl(7).Add(Me.LblMidBuf74)
FabShopControl(7).Add(Me.LblMidBuf75)
FabShopControl(7).Add(Me.LblMidBuf76)
FabShopControl(7).Add(Me.LblMidBuf77)
FabShopControl(7).Add(Me.LblMidBuf78)
FabShopControl(7).Add(Me.LblMidBuf79)
FabShopControl(7).Add(Me.LblMidBuf80)
FabShopControl(7).Add(Me.LblMidBuf81)
SetVisibilityControl(MidBufNum, FabShopControl(7))
'Form MidBuffer Collection
FabShopListBox = New List(Of ListBox)
FabShopListBox.Add(Me.LstMidbuf70)
FabShopListBox.Add(Me.LstMidbuf71)
FabShopListBox.Add(Me.LstMidbuf72)
FabShopListBox.Add(Me.LstMidbuf73)
FabShopListBox.Add(Me.LstMidbuf74)
FabShopListBox.Add(Me.LstMidbuf75)
FabShopListBox.Add(Me.LstMidbuf76)
FabShopListBox.Add(Me.LstMidbuf77)
FabShopListBox.Add(Me.LstMidbuf78)
FabShopListBox.Add(Me.LstMidbuf79)
FabShopListBox.Add(Me.LstMidbuf80)
FabShopListBox.Add(Me.LstMidbuf81)
SetVisibilityListBox(MidBufNum, FabShopListBox)
"*****Initialize values of the Stations
Dim StationItem As Station
For i As Integer = 0 To 5
    For j As Integer = 1 To 8
        StationItem = New Station
        StationItem.ID = i * 10 + j
        Stations.Add(StationItem.ID.ToString, StationItem)
    Next
Next
"*****Initialize values of the MidBuffers
Dim MidBufferItem As MidBuffer
For i As Integer = 70 To 89
    MidBufferItem = New MidBuffer
    MidBufferItem.ID = i
    Midbuffers.Add(MidBufferItem.ID.ToString, MidBufferItem)
Next
"*****Initialize values of the Movers
Dim MoverItem As Mover
For i As Integer = 50 To 69
    MoverItem = New Mover
    MoverItem.ID = i
    Movers.Add(MoverItem.ID.ToString, MoverItem)
Next
End Sub

'Show selected number of FabShop Control
Private Sub SetVisibilityControl(ByVal VisNo As Integer, ByRef ShopControl As List(Of Control))
    Dim ShopCntrl As New Control
    For i As Integer = 0 To VisNo - 1
        ShopCntrl = ShopControl.Item(i)
        ShopCntrl.Visible = True
    Next
    For i As Integer = VisNo To ShopControl.Count - 1
        ShopCntrl = ShopControl.Item(i)
        ShopCntrl.Visible = False
    Next
Next

```



```

End Sub

'Show selected number of FabShop ListBox
Private Sub SetVisibilityListBox(ByVal VisNo As Integer, ByRef ShopControl As List(Of ListBox))
    Dim ShopCntrl As New Control
    For i As Integer = 0 To VisNo - 1
        ShopCntrl = ShopControl.Item(i)
        ShopCntrl.Visible = True
    Next
    For i As Integer = VisNo To ShopControl.Count - 1
        ShopCntrl = ShopControl.Item(i)
        ShopCntrl.Visible = False
    Next
End Sub

'Set text on the FabShop Controls
Private Sub SetControlsText()
    Dim ShopCntrl As Control
    Dim PieceID As String = ""
    'Set the Station related controls
    Dim StationItem As Station
    Dim StationID As Integer
    For i As Integer = 0 To 4
        For j As Integer = 0 To 7
            'Check if station ID exists
            StationID = i * 10 + j + 1
            If Stations.Keys.Contains(StationID.ToString) Then
                'Read the control
                ShopCntrl = New Control
                ShopCntrl = FabShopControl(i + 1)(j)
                'Read the station
                StationItem = New Station
                StationItem = Stations(StationID.ToString)
                PieceID = Shop.PieceIDofHandle(StationItem.CurrentPieceHandle)
                'Check if any Piece is on the station
                If PieceID.Length > 0 Then
                    ShopCntrl.Text = Microsoft.VisualBasic.Right(PieceID, 7)
                    ShopCntrl.BackColor = Color.LightPink
                ElseIf StationItem.State = ToolState.Suspend Then 'Suspended
                    ShopCntrl.Text = ""
                    ShopCntrl.BackColor = Color.LightBlue
                Else 'Idle
                    ShopCntrl.Text = ""
                    ShopCntrl.BackColor = Color.LightGreen
                End If
            End If
        Next
    Next
    'Set the Controls related to the Mover
    Dim MoverItem As Mover
    Dim MoverID As Integer
    Dim BusyMover As Integer = 0
    Dim TotalMover As Integer = CType(TxtMoverNum.Text, Integer)
    For j As Integer = 0 To TotalMover - 1
        'Check if station ID exists
        MoverID = 50 + j
        If Movers.Keys.Contains(MoverID.ToString) Then
            'Read the control
            ShopCntrl = New Control
            ShopCntrl = FabShopControl(6)(j)
            'Read the station
            MoverItem = New Mover
            MoverItem = Movers(MoverID.ToString)
            PieceID = Shop.PieceIDofHandle(MoverItem.PieceOnMover.PieceHandle)
            'Check if any Piece is on the Mover
            If PieceID.Length > 0 Then
                BusyMover = BusyMover + 1
                ShopCntrl.Text = MoverItem.NoOfAssignedJob.ToString & "_" & Microsoft.VisualBasic.Right(PieceID, 7)
                ShopCntrl.BackColor = Color.LightPink
            Else 'No Piece

```

```

        'ShopCntrl.Text = ""
        'ShopCntrl.BackColor = Color.LightGreen
    End If
End If
Next
LblCraneBusy.Text = BusyMover.ToString
LblCraneIdle.Text = (TotalMover - BusyMover).ToString
'Set the Controls related to the Midbuffer
Dim MidBufferItem As MidBuffer
Dim MidBufferID As Integer
Dim MidBufferList As ListBox
Dim Count As Integer
For j As Integer = 0 To 11
    'Check if station ID exists
    MidBufferID = 70 + j
    If Midbuffers.Keys.Contains(MidBufferID.ToString) Then
        'Read the control
        MidBufferList = New ListBox
        MidBufferList = FabShopListBox(j)
        'Read the midbuffer
        MidBufferItem = New MidBuffer
        MidBufferItem = Midbuffers(MidBufferID.ToString)
        'Read count from mid buffer
        For Each Count In MidBufferList.Items
            Next
            If MidBufferItem.PieceHandleList.Count > 0 Then
                If Not Count = MidBufferItem.PieceHandleList.Count Then
                    MidBufferList.BackColor = Color.LightPink
                    MidBufferList.Items.Clear()
                    MidBufferList.Items.Add(MidBufferItem.PieceHandleList.Count)
                End If
            Else
                MidBufferList.Items.Clear()
                MidBufferList.BackColor = Color.LightGreen
            End If
        End If
    Next
    'Set the Hour Label
    LblHour.Text = (Convert.ToInt64(Shop.CurTime / 36) / 100).ToString
    'Set Piece Completed Lable
    LblPieceCompleted.Text = NoPieceCompleted.ToString
    'Referesh
    Me.Refresh()
End Sub

'Set the controls Unable
Private Sub SetTheControlsUnable()
    'Text boxes
    TxtMidBufferNum.Enabled = False
    TxtMoverNum.Enabled = False
    'Combo boxes
    CmbCut.Enabled = False
    CmbInspect.Enabled = False
    CmbPaint.Enabled = False
    CmbShop.Enabled = False
End Sub

'Set the final entered Values for the shop Object
Private Sub SetFinalValuesForTheShopObject()
    'Set Stations
    Dim StationItemID As Integer
    Dim FabShopControlItm As Control
    Dim i As Integer
    For i = 0 To 4
        For j As Integer = 0 To 7
            FabShopControlItm = New Control
            'Retrive the related button to the station
            FabShopControlItm = FabShopControl(i + 1).Item(j)
            If FabShopControlItm.Visible = False Then
                StationItemID = i * 10 + j + 1
            End If
        Next
    Next
End Sub

```

```

        Stations.Remove(StationItemID.ToString)
    End If
Next
Next
'Set Movers
For i = (50 + Convert.ToInt32((TxtMoverNum.Text))) To 69
    Movers.Remove(i.ToString)
Next
'Set MidBuffers
For i = (71 + Convert.ToInt32((TxtMidBufferNum.Text))) To 89
    Midbuffers.Remove(i.ToString)
Next
End Sub
Private Sub DES_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    'Set interface
    SetInterface()
    Dim MyDataShop As New DataShop
    'Fill Stations
    'Dim StationForm As New StationInfo
    MyDataShop.FillStationList(Stations)
    'Fill MidBuffers
    'Dim MidBufForm As New MidBufferInfo
    MyDataShop.FillMidBuferList(Midbuffers)
    'Fill Mover
    'Dim MoverForm As New MoverInfo
    MyDataShop.FillmovererList(Movers)
End Sub
Private Sub CmbCut_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CmbCut.TextChanged
    'Check if all controls have been loaded
    If (Not FabShopControl(1) Is Nothing) Then
        If FabShopControl(1).Count = 8 Then
            SetVisibilityControl(Convert.ToInt32(CmbCut.Text), FabShopControl(1))
        End If
    End If
End Sub

Private Sub CmbShop_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CmbShop.TextChanged
    'Check if all controls have been loaded
    If (Not FabShopControl(2) Is Nothing) And (Not FabShopControl(3) Is Nothing) Then
        If FabShopControl(2).Count = 8 And FabShopControl(3).Count = 8 Then
            SetVisibilityControl(Convert.ToInt32(CmbShop.Text), FabShopControl(2))
            SetVisibilityControl(Convert.ToInt32(CmbShop.Text), FabShopControl(3))
        End If
    End If
End Sub

Private Sub CmbInspect_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CmbInspect.SelectedIndexChanged
    'Check if all controls have been loaded
    If (Not FabShopControl(4) Is Nothing) Then
        If FabShopControl(4).Count = 8 Then
            SetVisibilityControl(Convert.ToInt32(CmbInspect.Text), FabShopControl(4))
        End If
    End If
End Sub

Private Sub CmbPaint_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CmbPaint.SelectedIndexChanged
    'Check if all controls have been loaded
    If (Not FabShopControl(5) Is Nothing) Then
        If FabShopControl(5).Count = 8 Then
            SetVisibilityControl(Convert.ToInt32(CmbPaint.Text), FabShopControl(5))
        End If
    End If
End Sub

Private Sub TxtMoverNum_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
TxtMoverNum.TextChanged

```

```

End Sub

Private Sub TxtMidBufferNum_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
TxtMidBufferNum.TextChanged
    'Check if all controls have been loaded
    If (Not FabShopListBox Is Nothing) Then
        If FabShopListBox.Count = 12 Then
            Dim MidBufNum As Integer = Math.Min(Convert.ToInt32(TxtMidBufferNum.Text) + 1, 5)
            SetVisibilityListBox(MidBufNum, FabShopListBox)
            SetVisibilityControl(MidBufNum, FabShopControl(7))
        End If
    End If
End Sub

'Load the station form
Private Sub BtnStationInfo_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnStationInfo.Click
    Dim StationForm As New StationInfo
    StationForm.LinkedListStations = Stations
    StationForm.FillListsCombos(Convert.ToInt32(CmbCut.Text), Convert.ToInt32(CmbShop.Text),
Convert.ToInt32(CmbInspect.Text), Convert.ToInt32(CmbPaint.Text))
    If Shop.CurTime > 0 Then
        StationForm.ShouldSaveTheChanges = False
    End If
    StationForm.Initialize()
    StationForm.ShowDialog()
End Sub

'Load the Midbuffer form
Private Sub BtnMidBuf_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnMidBuf.Click
    Dim MidBufForm As New MidBufferInfo
    MidBufForm.LinkedListMidbuffers = Midbuffers
    Dim MidBufNum As Integer = Math.Min(Convert.ToInt32(TxtMidBufferNum.Text) + 1, 12)
    SetVisibilityListBox(MidBufNum, FabShopListBox)
    SetVisibilityControl(MidBufNum, FabShopControl(7))
    MidBufForm.FillListsCombos(MidBufNum - 1, Convert.ToInt32(CmbCut.Text), Convert.ToInt32(CmbShop.Text),
Convert.ToInt32(CmbInspect.Text), Convert.ToInt32(CmbPaint.Text))
    If Shop.CurTime > 0 Then
        MidBufForm.ShouldSaveTheChanges = False
    End If
    MidBufForm.Initialize()
    MidBufForm.ShowDialog()
End Sub

'Load the Mover form
Private Sub BtnMover_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnMover.Click
    Dim MoverForm As New MoverInfo
    MoverForm.LinkedListMovers = Movers
    MoverForm.LinkedListMidbuffers = Midbuffers
    MoverForm.LinkedListStations = Stations
    Dim MoverNum As Integer = Math.Min(Convert.ToInt32(TxtMoverNum.Text), 20)
    MoverForm.FillListsCombos(MoverNum)
    If Shop.CurTime > 0 Then
        MoverForm.ShouldSaveTheChanges = False
    End If
    MoverForm.Initialize()
    MoverForm.ShowDialog()
End Sub

Private Sub MyCalendarFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyCalendarFactory.ReflectAttributeValues
    Dim ShiftTypeHanle As Cosye.Hla.Rti.AttributeHandle =
MyCalendarFactory.GetAttributeHandle("CurrentShiftType")
    'Check if the Current ShiftType has been changed
    If e.theValues.Contains(ShiftTypeHanle) Then
        MyCalendar = MyCalendarFactory(e.theObject)
        Shop.CurDate = MyCalendar.CurrentDate
        If Shop.CurrentShiftType = ShiftType.Close And e.theTime > 0 Then 'Set the total close time
            Shop.TotalCloseTime = Shop.TotalCloseTime + e.theTime - Shop.LastCloseTimeStarted

```

```

    Shop.LastCloseTimeStarted = 0
End If
If MyCalendar.CurrentShiftType = ShiftType.Close Then 'Set the start of close time
    If e.theTime < Double.MaxValue Then
        Shop.LastCloseTimeStarted = e.theTime
        Shop.CloseTimeWillBeFinished = e.theTime + MyCalendar.CurrentShiftHours * 3600
    Else
        Shop.LastCloseTimeStarted = 0
        Shop.CloseTimeWillBeFinished = MyCalendar.CurrentShiftHours * 3600
    End If
End If
Shop.CurrentShiftType = MyCalendar.CurrentShiftType
LblCurDate.Text = MyCalendar.CurrentDate.ToString
Try
    LblWorkDay.Text = MyCalendar.DayNo.ToString
Catch ex As Exception
    'DayNo Has not been published yet
End Try
End If
End Sub
'Register Station Productivities
Private Sub RegisterStationsProductivities()
    Dim MyStationProductivity As Steel_StationProductivity
    For Each stationItem As Station In Stations.Values
        MyStationProductivity = MyStationProductivityFactory.RegisterObjectInstance
        MyStationProductivity.ID = stationItem.ID
        MyStationProductivity.MaxOperator = stationItem.MaxReqOperators
        MyStationProductivity.MinOperator = stationItem.MinReqOperators
        MyStationProductivity.CurOperator = stationItem.AssignedOperatorNo
        MyStationProductivity.SFunction = stationItem.SFunction
        MyStationProductivity.UpdateAttributeValues()
        TotalHybridInteractionsFromDES = TotalHybridInteractionsFromDES + 1
        TxtHybridFromDES.Text = TotalHybridInteractionsFromDES.ToString
        MyStationProductivity.UnconditionalAttributeOwnershipDivestiture("CurOperator")
    Next
End Sub

Dim PreviouShiftOfUpdateStationState As ShiftType = ShiftType.Close
'Update Station State
Public Sub UpdateStationsStates(ByVal MyShift As ShiftType)
    Dim StationItem As Station
    If MyShift = ShiftType.Close Then
        If PreviouShiftOfUpdateStationState <> ShiftType.Close Then 'Report all stations as idle
            PreviouShiftOfUpdateStationState = ShiftType.Close
            For Each MyStationProductivity As Steel_StationProductivity In MyStationProductivityFactory
                StationItem = New Station
                StationItem = Stations(MyStationProductivity.ID.ToString)
                If StationItem.State <> ToolState.Idle Then
                    MyStationProductivity.State = ToolState.Idle
                    MyStationProductivity.UpdateAttributeValues()
                    StationItem.ReportedState = ToolState.Idle
                    TotalHybridInteractionsFromDES = TotalHybridInteractionsFromDES + 1
                    TxtHybridFromDES.Text = TotalHybridInteractionsFromDES.ToString
                End If
            Next
        End If
    ElseIf PreviouShiftOfUpdateStationState <> ShiftType.Close Then 'Report busy station by end of close shift
        PreviouShiftOfUpdateStationState = ShiftType.DayShift
        For Each MyStationProductivity As Steel_StationProductivity In MyStationProductivityFactory
            StationItem = New Station
            StationItem = Stations(MyStationProductivity.ID.ToString)
            If StationItem.State = ToolState.Busy Then
                MyStationProductivity.State = ToolState.Busy
                MyStationProductivity.UpdateAttributeValues()
                StationItem.ReportedState = ToolState.Busy
                TotalHybridInteractionsFromDES = TotalHybridInteractionsFromDES + 1
                TxtHybridFromDES.Text = TotalHybridInteractionsFromDES.ToString
            End If
        Next
    End If
End Sub

```

```

For Each MyStationProductivity As Steel_StationProductivity In MyStationProductivityFactory
    StationItem = New Station
    StationItem = Stations(MyStationProductivity.ID.ToString)
    If StationItem.State <> StationItem.ReportedState Then
        MyStationProductivity.State = StationItem.State
        MyStationProductivity.UpdateAttributeValues()
        StationItem.ReportedState = StationItem.State
        TotalHybridInteractionsFromDES = TotalHybridInteractionsFromDES + 1
        TxtHybridFromDES.Text = TotalHybridInteractionsFromDES.ToString
    End If
Next
End If
End Sub

Dim ItIsFirstReceivedPiece As Boolean = True
Private Sub MyPieceEntityFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyPieceEntityFactory.ReflectAttributeValues
    Dim NewPieceEntity As New PieceEntityHandle
    NewPieceEntity.Piece = MyPieceEntityFactory(e.theObject)
    NewPieceEntity.PieceHandle = e.theObject
    NewPieceEntity.Piece.AttributeOwnershipAcquisition("CuttingStart", "CuttingManHour", "CuttingFinish",
"FittingStart", "FittingManHour", "FittingFinish", _
    "WeldingStart", "WeldingManHour", "WeldingFinish", "FitInspectionStart", "FitInspectionManHour",
"FitInspectionFinish", "WeldInspectionStart", "WeldInspectionManHour", "WeldInspectionFinish", "PaintingStart",
"PaintingManHour", "PaintingFinish")
    "should be done only on first time received updated
    If ItIsFirstReceivedPiece Then
        ItIsFirstReceivedPiece = False
        'Set the controls unEnable
        SetTheControlsUnable()
        'Set the final entered Values for the shop Object
        SetFinalValuesForTheShopObject()
        'Register and update the StationProductivity objects
        RegisterStationsProductivities()
    End If
    If NewPieceEntity.Piece.CuttingFinish = Nothing Then 'Piece is not in shop
        'Handle New arrived piece
        Shop.HandleNewPieceArrived(NewPieceEntity)
    Else 'Piece is in shop (Initial Condition of the piece)
        Dim LocationID As Integer
        Dim LocationType As AssociatedTool
        Dim NumOperationDone As Integer = 0
        Shop.HandlePieceInShopArrived(NewPieceEntity, LocationID, LocationType, NumOperationDone)
        If NumOperationDone = 1 Then
            LstCutFinish.Items.Add(NewPieceEntity.Piece.PieceID)
        ElseIf NumOperationDone = 2 Or NumOperationDone = 3 Then
            LstFitFinish.Items.Add(NewPieceEntity.Piece.PieceID)
        ElseIf NumOperationDone = 4 Or NumOperationDone = 5 Then
            LstWeldFinish.Items.Add(NewPieceEntity.Piece.PieceID)
        End If
    End If
    'Update the controls
    " ....."
End Sub

Dim NoEventScheduledTimeStep As Integer = 1
Private Sub fedAmb_TimeAdvanceGrant(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.TimeAdvanceGrantEventArgs) Handles fedAmb.TimeAdvanceGrant
    'Check if shop is close or open
    If Shop.CurrentShiftType = ShiftType.Close Then
        rtiAmb.NextMessageRequest(Shop.CloseTimeWillBeFinished)
    Else
        'Set the current time of the shop
        Shop.CurTime = e.theTime - Shop.TotalCloseTime
        'Process any internal events that should occur at the current time.
        Shop.MyEngine.Simulate(Shop.CurTime)
        If Shop.MyEngine.ScheduledEventCount = 0 Then 'Create a fake event
            'Schedule an Empty Event
            Dim EmptyEventParameters As New EventParameters

```

```

Shop.MyEngine.ScheduleEvent(EmptyEventParameters, Shop.EmptyEvent, NoEventScheduledTimeStep)
'Set the time advancement step
NoEventScheduledTimeStep = 1
Dim InitialMidBuffer As MidBuffer = Shop.MidBufferList("70")
If InitialMidBuffer.PieceHandleList.Count = 0 Then
    NoEventScheduledTimeStep = 1000
End If
End If
'Update the controls on the form
SetControlsText()
'Update all completed pieces
If Not Shop.CompletedPieceList Is Nothing Then
    Dim CompletedPiece As PieceEntityHandle
    Dim CompletedPieceEntity As Steel_PieceEntity
    While Shop.CompletedPieceList.Count > 0
        NoPieceCompleted = NoPieceCompleted + 1
        CompletedPiece = New PieceEntityHandle
        CompletedPiece = Shop.CompletedPieceList(0)
        Shop.CompletedPieceList.RemoveAt(0)
        'Updated the attributes
        CompletedPieceEntity = MyPieceEntityFactory(CompletedPiece.PieceHandle)
        CompletedPieceEntity.CuttingStart = CompletedPiece.Piece.CuttingStart
        CompletedPieceEntity.CuttingManHour = CompletedPiece.Piece.CuttingManHour
        CompletedPieceEntity.CuttingFinish = CompletedPiece.Piece.CuttingFinish
        CompletedPieceEntity.FittingStart = CompletedPiece.Piece.FittingStart
        CompletedPieceEntity.FittingManHour = CompletedPiece.Piece.FittingManHour
        CompletedPieceEntity.FittingFinish = CompletedPiece.Piece.FittingFinish
        CompletedPieceEntity.WeldingStart = CompletedPiece.Piece.WeldingStart
        CompletedPieceEntity.WeldingManHour = CompletedPiece.Piece.WeldingManHour
        CompletedPieceEntity.WeldingFinish = CompletedPiece.Piece.WeldingFinish
        CompletedPieceEntity.FitInspectionStart = CompletedPiece.Piece.FitInspectionStart
        CompletedPieceEntity.FitInspectionManHour = CompletedPiece.Piece.FitInspectionManHour
        CompletedPieceEntity.FitInspectionFinish = CompletedPiece.Piece.FitInspectionFinish
        CompletedPieceEntity.WeldInspectionStart = CompletedPiece.Piece.WeldInspectionStart
        CompletedPieceEntity.WeldInspectionManHour = CompletedPiece.Piece.WeldInspectionManHour
        CompletedPieceEntity.WeldInspectionFinish = CompletedPiece.Piece.WeldInspectionFinish
        Try
            CompletedPieceEntity.PaintingStart = CompletedPiece.Piece.PaintingStart
            CompletedPieceEntity.PaintingManHour = CompletedPiece.Piece.PaintingManHour
            CompletedPieceEntity.PaintingFinish = CompletedPiece.Piece.PaintingFinish
        Catch ex As Exception
            'No Painting is required
        End Try
        CompletedPieceEntity.UpdateAttributeValues()
    End While
End If
'Request Next Message consider the Engine Accuracy
Dim NextMessageTime As Double
If (Shop.MyEngine.TimeNext - Shop.CurTime) >= fedAmb.Lookahead Then
    NextMessageTime = Shop.MyEngine.TimeNext + Shop.TotalCloseTime
Else
    NextMessageTime = (Shop.CurTime + fedAmb.Lookahead + Shop.TotalCloseTime)
End If
rtiAmb.NextMessageRequest(NextMessageTime)
End If
'Update the station state if it has changed
UpdateStationsStates(Shop.CurrentShiftType)
End Sub

Private Sub MyStationProductivityFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyStationProductivityFactory.ReflectAttributeValues
    Dim MyStationProd As Steel_StationProductivity = MyStationProductivityFactory(e.theObject)
    Dim MyProductivityHandle As AttributeHandle = MyStationProductivityFactory.GetAttributeHandle("Productivity")
    Dim MyCurOperatorHandle As AttributeHandle = MyStationProductivityFactory.GetAttributeHandle("CurOperator")
    Dim MyStation As Station = Stations(MyStationProd.ID.ToString)
    Dim EventKey As String = MyStation.ID.ToString & "_" & MyStation.CurrentPieceHandle.ToString
    'Count the interactions
    TotalHybridInteractionsFromSD = TotalHybridInteractionsFromSD + 1
    TxtHybridFromSD.Text = TotalHybridInteractionsFromSD.ToString
    'Remove Extra Engine Supplementaries

```

```

Shop.RemoveExtraEngineSupplementaries()
'Check if Productivity has been updated
If e.theValues.Contains(MyProductivityHandle) Then
    'Check if rescheduling is required
    If Shop.EngineEntities.Keys.Contains(EventKey) Then
        Dim ScheduledTime As Double = Shop.EngineTimes(EventKey)
        If ScheduledTime > Shop.CurTime Then
            'Count num of reschedules
            ReScheduleNum = ReScheduleNum + 1
            TxtReschedules.Text = ReScheduleNum.ToString
            'Calculate New Time
            Dim NewTime As Double = (ScheduledTime - Shop.CurTime) * _
                MyStation.Productivity / MyStationProd.Productivity
            'Retrieve the Event information
            Dim ChangedEventParameters As New EventParameters
            ChangedEventParameters = Shop.EngineEntities(EventKey)
            'Cancel Currently Scheduled Information
            Shop.MyEngine.CancelEvent(ChangedEventParameters)
            Shop.EngineEntities.Remove(EventKey)
            Shop.EngineTimes.Remove(EventKey)
            'Schedule New Event
            Shop.MyEngine.ScheduleEvent(ChangedEventParameters, Shop.StationSrvceFinished, NewTime)
            Shop.EngineTimes.Add(EventKey, NewTime + Shop.CurTime)
            Shop.EngineEntities.Add(EventKey, ChangedEventParameters)
        End If
    End If
    'Set the new Productivity
    MyStation.Productivity = MyStationProd.Productivity
End If
'Check if Current Operator has been updated
If e.theValues.Contains(MyCurOperatorHandle) Then
    If Shop.EngineEntities.Keys.Contains(EventKey) Then
        Dim ScheduledTime As Double = Shop.EngineTimes(EventKey)
        If ScheduledTime > Shop.CurTime Then
            'Retrieve the Event information
            Dim ChangedEventParameters As New EventParameters
            ChangedEventParameters = Shop.EngineEntities(EventKey)
            'Sample a duration for new and old operator #
            Dim NewOprNo As Integer = MyStationProd.CurOperator
            Dim OldOprNo As Integer = MyStation.AssignedOperatorNo
            Dim NewOprDistribution As Distribution = MyStation.DurationStructure(NewOprNo.ToString)
            Dim OldOprDistribution As Distribution = MyStation.DurationStructure(OldOprNo.ToString)
            Dim NewOprDuration As Double = NewOprDistribution.DurationFactor
            Dim OldOprDuration As Double = OldOprDistribution.DurationFactor
            'Count num of reschedules
            ReScheduleNum = ReScheduleNum + 1
            TxtReschedules.Text = ReScheduleNum.ToString
            'Calculate New Time
            Dim NewTime As Double = (ScheduledTime - Shop.CurTime) * NewOprDuration / OldOprDuration
            'Cancel Currently Scheduled Information
            Shop.MyEngine.CancelEvent(ChangedEventParameters)
            Shop.EngineEntities.Remove(EventKey)
            Shop.EngineTimes.Remove(EventKey)
            'Schedule New Event
            Shop.MyEngine.ScheduleEvent(ChangedEventParameters, Shop.StationSrvceFinished, NewTime)
            Shop.EngineTimes.Add(EventKey, NewTime + Shop.CurTime)
            Shop.EngineEntities.Add(EventKey, ChangedEventParameters)
        End If
    End If
    'Set the Current Operator #
    MyStation.AssignedOperatorNo = MyStationProd.CurOperator
End If
End Sub

Private Sub fedAmb_EndExecution(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
fedAmb.EndExecution
    'Define the Data shop and its related methods
    Dim MyDataShop As New DataShop
    'Update the statios tables mainly for the utilization purposes
    MyDataShop.UpdateStationTables(Stations, "StationsReport")

```



```

'Update Mainform Controls
MyDataShop.UpdateMainFormTables("CmbCut", CmbCut.Text)
MyDataShop.UpdateMainFormTables("CmbInspect", CmbInspect.Text)
MyDataShop.UpdateMainFormTables("CmbPaint", CmbPaint.Text)
MyDataShop.UpdateMainFormTables("CmbShop", CmbShop.Text)
MyDataShop.UpdateMainFormTables("TxtMidBufferNum", TxtMidBufferNum.Text)
MyDataShop.UpdateMainFormTables("TxtMoverNum", TxtMoverNum.Text)
'Update MLC Table
MyDataShop.ReportMLCResult(MLC, MyCalendar.StartDate, MyCalendar.CurrentDate, NoPieceCompleted,
SimulationStartTime, TotalHybridInteractionsFromSD, TotalHybridInteractionsFromDES, ReScheduleNum)
End Sub

Private Sub fedAmb_BeginExecution(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
fedAmb.BeginExecution
SimulationStartTime = DateTime.Now
End Sub

Private Sub BtnSaveLayout_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnSaveLayout.Click
"Update the set value in data base
Dim MyDataShop As New DataShop
MyDataShop.UpdateMainFormTables("CmbShop", CmbShop.Text)
MyDataShop.UpdateMainFormTables("CmbCut", CmbCut.Text)
MyDataShop.UpdateMainFormTables("CmbInspect", CmbInspect.Text)
MyDataShop.UpdateMainFormTables("CmbPaint", CmbPaint.Text)
MyDataShop.UpdateMainFormTables("TxtMoverNum", TxtMoverNum.Text)
MyDataShop.UpdateMainFormTables("TxtMidBufferNum", TxtMidBufferNum.Text)
End Sub

Private Sub MyShopProductivityFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyShopProductivityFactory.ReflectAttributeValues
Dim MyShopProductivity As Steel_ShopProductivity = MyShopProductivityFactory(e.theObject)
MLC = MyShopProductivity.MLC
End Sub

Dim SimulationStartTime As New DateTime

Private Sub fedAmb_AnnounceSynchronizationPoint(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.AnnounceSynchronizationPointEventArgs) Handles fedAmb.AnnounceSynchronizationPoint
'Initialize the simulation engine
Shop.MyEngine.InitializeEngine()
Shop.MyEngine.InitializeScenario()
End Sub

End Class

```

4) System Dynamics Federate

```

Imports Cosye.Hla.Rti
Imports Simphony.Mathematics

Public Class FrmFabSD
Dim MyShopProductivity As Steel_ShopProductivity
Dim MySDShop As New SDShop
Dim StationsProductivities As Dictionary(Of String, SDStation) = MySDShop.StationsProductivity
Dim MyCalendar As Steel_Calendar
Dim TotalHybridInteractions As Integer = 0

Private Sub Initialize()
'Initialize Shop Parameters
" Over Time Effect Delay
MySDShop.SetOverTime.Duration = 7
MySDShop.SetOverTime.Interval = Intervals.D
" New operator herring Delay
MySDShop.Requestedoperator.Duration = 3
MySDShop.Requestedoperator.Interval = Intervals.D
" Operator under training Delay

```

```

MySDShop.OperatorUnderTraining.Duration = 60
MySDShop.OperatorUnderTraining.Interval = Intervals.D
" SD Loop Constant Parameters
MySDShop.ChanceOfLeaving = Val(TxtLeavingChance.Text)
MySDShop.ChanceOfUnskilledOperator = Val(TxtUnSkillChance.Text)
MySDShop.DesireUtilizationLevel = Val(TxtDesireUtil.Text)
" Update the Top Level SD Loop For the first time
UpdateSDTopLevel()
End Sub

'Updating Top Level SD Loop Interface
Public Sub UpdateSDTopLevel()
    TxtDesireOverTime.Text = MyCalendar.DesireOverTime.ToString
    TxtDesireUtil.Text = MySDShop.DesireUtilizationLevel.ToString
    TxtMaxUtil.Text = MySDShop.MaxUtilization.ToString
    TxtDelay.Text = MyShopProductivity.DelayRate.ToString
    TxtSetOverTime.Text = MySDShop.SetOverTime.Value(True).ToString
    TxtShopProd.Text = MySDShop.ShopProductivity.ToString
    TxtSkilled.Text = MySDShop.CurSkilledOperators.ToString
    TxtUnSkilled.Text = MySDShop.CurUnSkilledOperators.ToString
    TxtMaxOverTime.Text = MySDShop.MaxOverTime.ToString
    TxtStationShopProd.Text = TxtShopProd.Text
End Sub

Private Sub MyShopProductivityFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyShopProductivityFactory.ReflectAttributeValues
    MyShopProductivity = MyShopProductivityFactory(e.theObject)
    TotalHybridInteractions = TotalHybridInteractions + 1
    TxtHybrid.Text = TotalHybridInteractions.ToString
End Sub

Private Sub UpdateTopLevelSDLoop(ByRef TotalNewOperator As Integer, ByRef TotalLeavingOperators As Integer)
    "Test
    If (MySDShop.CurTime > 1030000) And (Not MySDShop.SetOverTime.CurDate > #1/1/2000# Or _
        Not MySDShop.Requestedoperator.CurDate > #1/1/2000# Or _
        Not MySDShop.OperatorUnderTraining.CurDate > #1/1/2000#) Then
        Dim test As Integer = 1
    End If
    "Test

    'Reset All local interface Variables in the SD top level loop
    TxtHiringReq.Text = "0"
    TxtOverTimeReq.Text = "0"
    'Set the first attributes of the loops which get effect from Fabshop
    'Check schedule Delay
    If MyShopProductivity.DelayRate < 0 And MySDShop.MaxUtilization > MySDShop.DesireUtilizationLevel Then
        'Check the Current OverTime Status for choosing between change in over Time or Operator
        If MySDShop.SetOverTime.Value(True) > 20 And MyCalendar.DesireOverTime = MySDShop.MaxOverTime
Then
            'Check if operator employment is not currently under process
            If MySDShop.Requestedoperator.Value(True) = 0 Then
                'Determine number of operator that can be added
                MySDShop.Requestedoperator.AddValue(MySDShop.RequiredOperators)
                TxtHiringReq.Text = MySDShop.RequiredOperators.ToString
            End If
            ElseIf MyCalendar.DesireOverTime < MySDShop.MaxOverTime Then
                MyCalendar.DesireOverTime = MyCalendar.DesireOverTime + 1
                TxtOverTimeReq.Text = "1"
            End If
            'Check the if Utilization is not greater that not desire
            ElseIf MySDShop.MaxUtilization < MySDShop.DesireUtilizationLevel And MyCalendar.DesireOverTime > 0 Then
                'Check if there is still room for reducinig overtime
                If MyCalendar.DesireOverTime > 0 Then
                    MyCalendar.DesireOverTime = MyCalendar.DesireOverTime - 1
                    TxtOverTimeReq.Text = "-1"
                End If
            End If
            'Calculate the Operator loop variables
            " New Operators
            TotalNewOperator = Convert.ToInt32(MySDShop.Requestedoperator.Value(False))

```

```

Dim NewUnSkilledOperator As Integer = Convert.ToInt32(TotalNewOperator * _
    Triangular.Sample(0, 1, MySDShop.ChanceOfUnskilledOperator))
Dim NewSkilledOperator As Integer = TotalNewOperator - NewUnSkilledOperator
'Leaving Operators
Dim UnSkilledOperatorToLeave As Integer = 0
If Uniform.Sample(0, 1) > _
((1 - MySDShop.ChanceOfLeaving) ^ MySDShop.CurUnSkilledOperators) Then
    UnSkilledOperatorToLeave = 1
End If
Dim SkilledOperatorToLeave As Integer = 0
If Uniform.Sample(0, 1) > _
((1 - MySDShop.ChanceOfLeaving) ^ MySDShop.CurSkilledOperators) And _
MySDShop.CurSkilledOperators > 50 Then
    SkilledOperatorToLeave = 1
End If
TotalLeavingOperators = SkilledOperatorToLeave + UnSkilledOperatorToLeave
Dim TrainedOperators As Integer = Convert.ToInt32(MySDShop.OperatorUnderTraining.Value(False))
'Stock Variables ---- Skilled and unskilled Operators
MySDShop.CurSkilledOperators = MySDShop.CurSkilledOperators + NewSkilledOperator _
- SkilledOperatorToLeave + TrainedOperators
MySDShop.CurUnSkilledOperators = MySDShop.CurUnSkilledOperators + NewUnSkilledOperator _
- UnSkilledOperatorToLeave - TrainedOperators
Dim OperatorsSkillLevel As Double = (MySDShop.CurSkilledOperators + MySDShop.CurUnSkilledOperators / 2) _
/ (MySDShop.CurSkilledOperators + MySDShop.CurUnSkilledOperators)
'Calculate the OverTime loop variables
Dim FatigueLevel As Double = 0
Select Case MySDShop.SetOverTime.Value(True)
    Case Is < 20
        FatigueLevel = 0.8 + (20 - MySDShop.SetOverTime.Value(True)) / 20 * 0.2
    Case 20 To 40
        FatigueLevel = 0.4 + (40 - MySDShop.SetOverTime.Value(True)) / 20 * 0.4
    Case Else
        FatigueLevel = Math.Max(0.01, (140 - MySDShop.SetOverTime.Value(True)) / 100 * 0.4)
End Select
'Calculate Shop Productivity
MySDShop.ShopProductivity = FatigueLevel * OperatorsSkillLevel
'Update Local defined Variables for the interface
End Sub

Private Sub MyStationProductivityFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyStationProductivityFactory.ReflectAttributeValues
    Dim MyStationProductivity As Cosye.Steel.Steel_StationProductivity = MyStationProductivityFactory(e.theObject)
    Dim MyIDHandle As AttributeHandle = MyStationProductivityFactory.GetAttributeHandle("ID")
    Dim MyUtilizationHandle As AttributeHandle = MyStationProductivityFactory.GetAttributeHandle("Utilization")
    Dim MyStateHandle As AttributeHandle = MyStationProductivityFactory.GetAttributeHandle("State")
    'Calculate the interactions
    TotalHybridInteractions = TotalHybridInteractions + 1
    TxtHybrid.Text = TotalHybridInteractions.ToString

    'If the ID has been Updated, i.e. for the first time for each station when the station object just has been created
    If e.theValues.Contains(MyIDHandle) Then
        MyStationProductivity.AttributeOwnershipAcquisition("Productivity", "CurOperator")
        'Update the Station in the SD shop
        Dim StationProdItem As New SDStation
        StationProdItem.ID = MyStationProductivity.ID
        StationProdItem.Handle = e.theObject
        StationProdItem.CurOperator = MyStationProductivity.CurOperator
        StationProdItem.MaxOperator = MyStationProductivity.MaxOperator
        StationProdItem.MinOperator = MyStationProductivity.MinOperator
        StationProdItem.SFunction = MyStationProductivity.SFunction
        StationsProductivities.Add(StationProdItem.ID.ToString, StationProdItem)
        'Add the station's current operators to the shop's operaorts
        MySDShop.CurSkilledOperators = MySDShop.CurSkilledOperators + MyStationProductivity.CurOperator
        'Add the Stations' ID to the Combo Item
        CmbStationID.Items.Add(StationProdItem.ID)
    End If
    'If the State has been updated
    If e.theValues.Contains(MyStateHandle) Then
        Dim StationProdItem As New SDStation

```

```

        StationProdItem = StationsProductivities(MyStationProductivity.ID.ToString)
        StationProdItem.State = MyStationProductivity.State
    End If
End Sub

Private Sub fedAmb_TimeAdvanceGrant(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.TimeAdvanceGrantEventArgs) Handles fedAmb.TimeAdvanceGrant
    'Set the Current Time for the SDDelay Type Variables
    MySDSShop.UpdateShopTime(e.theTime)
    MySDSShop.SetOverTime.CurDate = MyCalendar.CurrentDate
    MySDSShop.Requestedoperator.CurDate = MyCalendar.CurrentDate
    MySDSShop.OperatorUnderTraining.CurDate = MyCalendar.CurrentDate

    "Test
    If (MySDSShop.CurTime > 1030000) And (Not MySDSShop.SetOverTime.CurDate > #1/1/2000# Or _
        Not MySDSShop.Requestedoperator.CurDate > #1/1/2000# Or _
        Not MySDSShop.OperatorUnderTraining.CurDate > #1/1/2000#) Then
        Dim test As Integer = 1
    End If
    "Test

    If e.theTime > 0 Then
        MySDSShop.SetOverTime.AddValue(MyCalendar.SetOverTimeForDay)
    Else
        MySDSShop.SetOverTime.AddValue(0)
    End If
    'Set the Hour Label
    LblHour.Text = (Convert.ToInt64(e.theTime / 36) / 100).ToString
    'Check updates for the Top Level SD Loops
    If MySDSShop.TopLevelSDNextTime <= e.theTime Then
        ' Update the loop
        Dim TotalNewOperators, TotalLeavingOperators As Integer
        UpdateTopLevelSDLoop(TotalNewOperators, TotalLeavingOperators)
        'Update the Required Values for the federation
        Dim OperatorChangedStations As New Dictionary(Of String, Integer)
        Dim StationItem As SDStation
        Dim MyStationProductivity As Steel_StationProductivity
        MySDSShop.AccomodateOperators(TotalNewOperators, TotalLeavingOperators, OperatorChangedStations)
        'Update the Required Values for the federation
        For Each StationID As Integer In OperatorChangedStations.Values
            StationItem = New SDStation
            StationItem = StationsProductivities(StationID.ToString)
            MyStationProductivity = MyStationProductivityFactory(StationItem.Handle)
            MyStationProductivity.CurOperator = StationItem.CurOperator
            MyStationProductivity.UpdateAttributeValues()
        Next
        'Update Desire Overtime for the Calendar
        MyCalendar.UpdateAttributeValues()
        'Set the next time of update
        MySDSShop.TopLevelSDNextTime = e.theTime + MySDSShop.TopLevelSDStep
        'Update the Top Level SD Loop Interface
        UpdateSDTopLevel()
    End If 'Top Level
    'Check updates for the Operator Exchange Loops
    If MySDSShop.OperatorExchangeLoopNextTime <= e.theTime Then
        ' Update the loop
        Dim OperatorChangedStations As New Collection
        Dim StationItem As SDStation
        Dim MyStationProductivity As Steel_StationProductivity
        MySDSShop.OperatorBalance(OperatorChangedStations)
        'Update the Required Values for the federation
        For Each StationID As Integer In OperatorChangedStations
            StationItem = New SDStation
            StationItem = StationsProductivities(StationID.ToString)
            MyStationProductivity = MyStationProductivityFactory(StationItem.Handle)
            MyStationProductivity.CurOperator = StationItem.CurOperator
            MyStationProductivity.UpdateAttributeValues()
        Next
        'Set the interface
        SetCurrentStationFunction()

```

```

        'Set the next time of update
        MySDShop.OperatorExchangeLoopNextTime = e.theTime + MySDShop.OperatorExchangeLoopStep
    End If 'Operator Exchange

    'Check updates for the Continuous Work Loops
    If MySDShop.OperatorContinuouWorkNextTime <= e.theTime Then
        ' Update the loop
        For Each StationProdItem As SDStation In StationsProductivities.Values
            ' Calculate the current Utilization of the station
            StationProdItem.CurUtilization = StationProdItem.Utilization(ToolState.Busy, 0)

            Dim MyStationProductivity As Steel_StationProductivity
            'Check if Productivity should be updated
            StationProdItem.CurProductivity = MySDShop.ShopProductivity / _
            (Math.Max(MySDShop.EasyUtilizationLevel, StationProdItem.CurUtilization) /
            MySDShop.EasyUtilizationLevel)
            'Check if productivity update is required
            If StationProdItem.ProductivityLag > MySDShop.MLC Then
                StationProdItem.SetProductivity = StationProdItem.CurProductivity
                MyStationProductivity = MyStationProductivityFactory(StationProdItem.Handle)
                MyStationProductivity.Productivity = StationProdItem.SetProductivity
                MyStationProductivity.UpdateAttributeValues()
                'Update the Utilization loop Interface/replied update
                LblStationUtilRepliedUpdate.Text = (Val(LblStationUtilRepliedUpdate.Text) + 1).ToString
            End If
            'Update the Utilization loop Interface
            SetCurrentStationIDInterface()
            LblStationUtilTotalUpdate.Text = (Val(LblStationUtilTotalUpdate.Text) + 1).ToString
        Next
        'Set the next time
        MySDShop.OperatorContinuouWorkNextTime = e.theTime + MySDShop.OperatorContinuouWorkLoopStep
    End If 'Continuous work
    'Calculate next time
    Dim NextTime As Double = Math.Min(Math.Max(e.theTime + 1, MySDShop.OperatorExchangeLoopNextTime), _
        Math.Min(Math.Max(e.theTime + 1, MySDShop.TopLevelSDNextTime),
        Math.Max(e.theTime + 1, MySDShop.OperatorContinuouWorkNextTime)))
    rtiAmb.TimeAdvanceRequest(NextTime)
End Sub

Private Sub MyCalendarFactory_DiscoverObjectInstance(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.DiscoverObjectInstanceEventArgs) Handles MyCalendarFactory.DiscoverObjectInstance
    MyCalendar = MyCalendarFactory(e.theObject)
    MyCalendar.AttributeOwnershipAcquisition("DesireOverTime")
    MyCalendar.DesireOverTime = 0
    MyCalendar.UpdateAttributeValues()
End Sub

Private Sub MyCalendarFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyCalendarFactory.ReflectAttributeValues
    MyCalendar = MyCalendarFactory(e.theObject)
    LblCurDate.Text = MyCalendar.CurrentDate.ToString
    'Check if MaxOverTime has been Updated
    Dim MaxOverTimeHandle As AttributeHandle = MyCalendarFactory.GetAttributeHandle("MaxOverTime")
    If e.theValues.Contains(MaxOverTimeHandle) Then
        MySDShop.CurDate = MyCalendar.CurrentDate
    End If
End Sub

'Initialize
Private Sub fedAmb_BeginExecution(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
fedAmb.BeginExecution
    Initialize()
    UpdateSDTopLevel()
End Sub

'Changing the Operator Balance Combo current staion function
Private Sub CmbSFunction_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CmbSFunction.TextChanged
    If CmbSFunction.Items.Count >= 5 Then
        SetCurrentStationFunction()
    End If

```

```

End Sub

'Setting the current staion function for the function Combo Box
Private Sub SetCurrentStationFunction()
    Dim SFunctionKey As Integer
    Select Case CmbSFunction.Text
        Case "Cutting"
            SFunctionKey = StationFunction.Cutting
        Case "Fitting"
            SFunctionKey = StationFunction.Fitting
        Case "Welding"
            SFunctionKey = StationFunction.Welding
        Case "FitInspection"
            SFunctionKey = StationFunction.FitInspection
        Case "WeldInspection"
            SFunctionKey = StationFunction.WeldInspection
        Case "Painting"
            SFunctionKey = StationFunction.Painting
    End Select
    'Set the interface based on the achieved result
    Dim BalanceResult As New OperatorBalanceResult
    If MySDShop.OperatorBalanceResultList.Keys.Contains(SFunctionKey.ToString) Then
        BalanceResult = MySDShop.OperatorBalanceResultList(SFunctionKey.ToString)
    End If
    UpdateOperatorBalnceInterce(BalanceResult)
End Sub

'Update operator balance result in interface
Private Sub UpdateOperatorBalnceInterce(ByVal Result As OperatorBalanceResult)
    TxtDesireStationUtil.Text = TxtDesireUtil.Text
    TxtMaxID.Text = Result.MaxID.ToString
    TxtMinID.Text = Result.MinID.ToString
    TxtMaxStationUtil.Text = Result.MaxUtil.ToString
    TxtMinStationUtil.Text = Result.MinUtil.ToString
    TxtOprMaxExchange.Text = Result.MaxOperatorChange.ToString
    TxtOprMinExchange.Text = Result.MinOperatorChange.ToString
End Sub

'Action based on the change in the station ID combo in the utilization loop
Private Sub CmbStationID_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CmbStationID.TextChanged
    If CmbStationID.Items.Count > 1 Then
        SetCurrentStationIDInterface()
    End If
End Sub

'Set the utilization loop interface
Private Sub SetCurrentStationIDInterface()
    Dim StationUtil As SDStation = StationsProductivities(CmbStationID.Text)
    TxtStationCurProd.Text = StationUtil.CurProductivity.ToString
    TxtStationSetProd.Text = StationUtil.SetProductivity.ToString
    TxtStationUtil.Text = StationUtil.CurUtilization.ToString
End Sub

Private Sub MyShopProductivityFactory_DiscoverObjectInstance(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.DiscoverObjectInstanceEventArgs) Handles MyShopProductivityFactory.DiscoverObjectInstance
    MyShopProductivity = MyShopProductivityFactory(e.theObject)
    MyShopProductivity.AttributeOwnershipAcquisition("MLC")
    " Set MLC
    MySDShop.MLC = Val(TxtProdMargine.Text)
    MyShopProductivity.MLC = MySDShop.MLC
    MyShopProductivity.UpdateAttributeValues()
End Sub
End Class

```

5) Visualization Federate

Imports Cosye.Hla.Rti

Public Class VisualizationFederate

Dim MyTekla As New Tekla
Dim MyVisualData As New DataVisual
Dim MyCalendar As Steel_Calendar
Dim NewDivisionRequest As Boolean = False
Dim RequestedDivision As New DivisionItem
Dim PieceProgressList As New Dictionary(Of String, VisualPiece)
Dim PauseTheFederation As Boolean = False
Dim ShowTekla As Boolean = False
Dim ActiveShowTeklaMode As Boolean = False

Private Sub MyVPieceFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyVPieceFactory.ReflectAttributeValues

Dim NewVPiece As Steel_VPiece
Dim NewVisualPiece As New VisualPiece
NewVPiece = MyVPieceFactory(e.theObject)
NewVisualPiece.PieceKey = NewVPiece.PieceKey
NewVisualPiece.DivisionID = NewVPiece.DivisionID
NewVisualPiece.CPI = NewVPiece.CPI
NewVisualPiece.SPI = NewVPiece.SPI
NewVisualPiece.Progress = NewVPiece.Progress
'Update the model color if current show tekla mode is true
If ActiveShowTeklaMode Then
MyTekla.UpdateColors(NewVisualPiece)
End If
'Update the data base and return true if related division is new
If MyVisualData.UpdateCompletedPiecesTable(NewVisualPiece) Then
Dim MyDivisionItem As New DivisionItem
MyDivisionItem.DivisionID = NewVPiece.DivisionID
MyVisualData.ReadDivisionFile(MyDivisionItem.DivisionID, MyDivisionItem.DivisionFile)
LstDivision.Items.Add(MyDivisionItem)
End If
'Updating the piece progress list
If MyTekla.CurrentDivision = NewVisualPiece.DivisionID Then
If PieceProgressList.Keys.Contains(NewVisualPiece.PieceKey) Then
Dim OldVisualPiece As VisualPiece = PieceProgressList(NewVisualPiece.PieceKey)
LstPieceProgress.Items.Remove(OldVisualPiece.ToString)
LstPieceProgress.Items.Add(NewVisualPiece.ToString)
PieceProgressList.Remove(NewVisualPiece.PieceKey)
PieceProgressList.Add(NewVisualPiece.PieceKey, NewVisualPiece)
Else
LstPieceProgress.Items.Add(NewVisualPiece.ToString)
PieceProgressList.Add(NewVisualPiece.PieceKey, NewVisualPiece)
End If
End If
'Go to the last item in the list
If LstPieceProgress.Items.Count > 0 Then
LstPieceProgress.SetSelected((LstPieceProgress.Items.Count - 1), True)
End If

Me.Refresh()

End Sub

Private Sub BtnChangeDivision_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnChangeDivision.Click

RequestedDivision = CType(LstDivision.SelectedItem, DivisionItem)
'Change the current division
If Not RequestedDivision.DivisionID = MyTekla.CurrentDivision _
Or (MyTekla.RelatedTeklaFileIsOpen And Not ShowTekla And ActiveShowTeklaMode) Then
LblRequestedDivision.Text = RequestedDivision.ToString
MyTekla.RelatedTeklaFileIsOpen = False
NewDivisionRequest = True
ElseIf (Not MyTekla.RelatedTeklaFileIsOpen And ShowTekla) Or _
(Not ActiveShowTeklaMode = ShowTekla) Then
LblRequestedDivision.Text = RequestedDivision.ToString
NewDivisionRequest = True
Else
RequestedDivision = New DivisionItem

```

    End If
End Sub
Private Sub MyCalendarFactory_ReflectAttributeValues(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.ReflectAttributeValuesEventArgs) Handles MyCalendarFactory.ReflectAttributeValues
    MyCalendar = MyCalendarFactory(e.theObject)
    'Check if DayNo has been Updated to commit the changes
    If MyTekla.CurrentDivision.Length > 0 Then
        Dim DayNoHandle As AttributeHandle = MyCalendarFactory.GetAttributeHandle("DayNo")
        If e.theValues.Contains(DayNoHandle) Then
            MyTekla.CommitChanges()
        End If
    End If
    LblDate.Text = MyCalendar.CurrentDate.ToString
    Me.Refresh()
End Sub

Private Sub fedAmb_BeginExecution(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
fedAmb.BeginExecution
    MyVisualData.InitializeTables()
    If RdBShowTekla.Checked Then
        MyTekla.OpenApplicaion()
    End If
End Sub

Private Sub fedAmb_EndExecution(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
fedAmb.EndExecution
    MyTekla.CloseApplicaion()
End Sub

Private Sub fedAmb_TimeAdvanceGrant(ByVal sender As System.Object, ByVal e As
Cosye.Hla.Rti.TimeAdvanceGrantEventArgs) Handles fedAmb.TimeAdvanceGrant
    If NewDivisionRequest Then
        LblRelatedTeklaFileOpen.ForeColor = Color.Red
        LblRelatedTeklaFileOpen.Text = "No 3D Model Linked!"
        ActiveShowTeklaMode = False
        If MyTekla.SetNewDivision(RequestedDivision.DivisionID, PieceProgressList, ShowTekla) Then ' returns true if
the file successfully gets open
            LblRelatedTeklaFileOpen.ForeColor = Color.Green
            LblRelatedTeklaFileOpen.Text = "See the Linked 3D Model!"
            ActiveShowTeklaMode = True
        End If
        LblCurrentDivision.Text = RequestedDivision.ToString
        LblRequestedDivision.Text = ""
        ' Set the Piece Progress List Box
        LstPieceProgress.Items.Clear()
        For Each VisualPieceItem As VisualPiece In PieceProgressList.Values
            LstPieceProgress.Items.Add(VisualPieceItem.ToString)
        Next
        If LstPieceProgress.Items.Count > 0 Then
            LstPieceProgress.SetSelected((LstPieceProgress.Items.Count - 1), True)
        End If
        MyTekla.CurrentDivision = RequestedDivision.DivisionID
        NewDivisionRequest = False
    ElseIf MyTekla.RelatedTeklaFilesOpen Then
        MyTekla.CommitChanges()
    End If
    'Check if user wants to pause the Simulation
    Dim Result As DialogResult
    While PauseTheFederation
        Result = MessageBox.Show("Do you want to continue?", "Continue", MessageBoxButtons.YesNo)
        If Result = Windows.Forms.DialogResult.Yes Then
            PauseTheFederation = False
            LblFedPause.Text = ""
        End If
    End While
    Dim NextDay As Double = (e.theTime + 60 * 60)
    rtiAmb.TimeAdvanceRequest(NextDay)
End Sub

```



```

Private Sub BtnPauseFederation_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BtnPauseFederation.Click
    PauseTheFederation = True
    LblFedPause.Text = "Federation Pause Request"
End Sub

Private Sub RdBDontShowTekla_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
RdBDontShowTekla.Click
    RdBShowTekla.Checked = False
    ShowTekla = False
    MyTekla.RelatedTeklaFileIsOpen = False
End Sub

Private Sub RdBShowTekla_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles RdBShowTekla.CheckedChanged
    RdBDontShowTekla.Checked = False
    ShowTekla = True
End Sub
End Class

```

Visualization Classes

```

Imports System.Data.OleDb
Imports System.Diagnostics
Imports Tekla.Structures.Model
Imports Tekla.Structures
Imports System.Threading

Public Class PieceIdentifiers 'For relating pieces to the current division
    Public ListofIdentifiers As New Dictionary(Of String, Identifier)
    Public PieceKey As String = ""
    Public CPI As Double = 0
    Public SPI As Double = 0
End Class

Public Class DivisionItem
    Public DivisionID As String = ""
    Public DivisionFile As String = ""
    Public Overrides Function ToString() As String
        Return DivisionID & " | " & DivisionFile
    End Function
End Class

Public Class VisualPiece
    Public PieceKey As String
    Public DivisionID As String
    Public CPI As Double
    Public SPI As Double
    Public Progress As Double
    Public Overrides Function ToString() As String
        Return PieceKey & " (CPI=" & Math.Round(CPI, 2).ToString & ") (SPI=" & Math.Round(SPI, 2).ToString & ") (%"
& Math.Round(Progress * 100, 1).ToString & "%)"
    End Function
End Class

Public Class Tekla
    Public CurrentDivision As String = "" 'Shows the current loaded DivisionID in the Tekla
    Public ListofPieceIdentifiers As New Dictionary(Of String, PieceIdentifiers)
    Public MyModel As Model
    Public RelatedTeklaFileIsOpen As Boolean = False

    'Open Tekla Software if is closed
    Public Function OpenApplicaion() As Boolean
        'Check if Tekla is already open
        Dim P_Check As Process() = Process.GetProcessesByName("TeklaStructures")
        Dim TeklaIsOpen As Boolean = True
        'Try to open Tekla if it was not open
        If P_Check.Count = 0 Then

```

```

    Dim P_Open As New Process
    P_Open.StartInfo.FileName = "C:\Documents and Settings\All Users\Start Menu\Programs\Tekla Structures
14.0\Tekla Structures 14.0 US Metric"
    TeklaIsOpen = P_Open.Start()
End If
'Act base on that Tekla is Open or Close
If TeklaIsOpen Then
    Return True
Else
    Return False
End If
End Function

'Close Tekla Software if is Open
Public Sub CloseApplicaion()
    'Retrieve Tekla process if Tekla is open
    Dim P As Process() = Process.GetProcessesByName("TeklaStructures")
    'Close Tekla process if Tekla is open
    If P.Count > 0 Then
        P(0).Kill()
        P(0).Close()
    End If
End Sub

'Load new division in the Tekla window
Public Function SetNewDivision(ByVal MyDivision As String, ByRef PieceProgressList As Dictionary(Of String,
VisualPiece), ByVal ShowTekla As Boolean) As Boolean
    Dim MyDataVisual As New DataVisual ' For reading the data
    Dim DivisionFileName As String = ""
    Dim MyVPieceList As New Dictionary(Of String, VisualPiece)
    'Find the file name of the Division from local data base
    MyDataVisual.ReadDivisionFile(MyDivision, DivisionFileName)
    'Read completed pieces from the data base
    MyDataVisual.ReadVPieces(MyDivision, MyVPieceList)
    If Not DivisionFileName = "No Tekla File" And ShowTekla Then 'No Tekla File or no request for tekla
        'Do the entire procedure if Tekla is open or could be open and division is new
        If OpenApplicaion() And ((Not CurrentDivision = MyDivision) _
            Or (Not RelatedTeklaFileIsOpen)) Then
            Dim MyVPiece As New VisualPiece
            Dim ClassColor As String = ""
            Dim MyPieceKey As String = ""
            Dim MyPieceIdentifier As PieceIdentifiers
            Try 'If there is any prolem with opening the file
                'Save latest changes to the current open file
                If CurrentDivision.Length > 0 Then
                    CommitChanges()
                End If
                'Open the division file in Tekla
                MyModel = New Model
                MyModel.Open("C:\TeklaVisualization\" & DivisionFileName)
                Dim Result As DialogResult
                'First the model should get completely open
                Result = MessageBox.Show("Push Yes button when the model is completely open!", "File Opening
Completed", MessageBoxButtons.YesNo)
                If Result = DialogResult.Yes Then
                    'Read all pieces (BEAM type) from the model and check their color
                    Dim MyBeamEnum As ModelObjectEnumerator =
MyModel.GetModelObjectSelector.GetAllObjectsWithType(ModelObject.ModelObjectEnum.BEAM)
                    Dim MyBeam As Beam
                    'Reset the ListofPieceIdentifiers
                    ListofPieceIdentifiers.Clear()
                    While MyBeamEnum.MoveNext
                        'Retrieve the model's Piece (Beams) in order
                        MyBeam = CType(MyModel.SelectModelObject(MyBeamEnum.Current.Identifier), Beam)
                        'Read the piece key (ASSEMBLY_POSITION) of the current Piece (Beam)
                        MyBeam.GetReportProperty("ASSEMBLY_POS", MyPieceKey)
                        'Eliminate extra characters from read piece key (ASSEMBLY_POSITION)
                        MyPieceKey = Replace(MyPieceKey, "(?)0", "")
                        MyPieceKey = Replace(MyPieceKey, "(?)", "")
                        ClassColor = "1" 'Gray color which shows incompleted pieces

```

```

'Check if current Piece (Beam) already has been completed and retrived its related color
Dim CPI As Double = 0
Dim SPI As Double = 0
If MyVPieceList.Keys.Contains(MyPieceKey) Then
    MyVPiece = New VisualPiece
    MyVPiece = MyVPieceList(MyPieceKey)
    ClassColor = RelatedColor(MyVPiece.CPI, MyVPiece.SPI)
    CPI = MyVPiece.CPI
    SPI = MyVPiece.SPI
End If
'Set the related color of the piece
If Not (MyBeam.Class = ClassColor) Then
    MyBeam.Class = ClassColor
    MyBeam.Modify()
End If
'Retrive related PieceIdentifier
If ListofPieceIdentifiers.Keys.Contains(MyPieceKey) Then
    'Retrieve the stored PieceIdentifier
    MyPieceIdentifier = ListofPieceIdentifiers(MyPieceKey)
    'Add current identifier to the retrieved PieceIdentifier
    MyPieceIdentifier.ListofIdentifiers.Add(MyBeam.Identifier.ToString, MyBeam.Identifier)
Else 'The piece has not been stored yet; create a new one
    MyPieceIdentifier = New PieceIdentifiers
    MyPieceIdentifier.PieceKey = MyPieceKey
    MyPieceIdentifier.CPI = CPI
    MyPieceIdentifier.SPI = SPI
    'Add current identifier to the retrieved PieceIdentifier
    MyPieceIdentifier.ListofIdentifiers.Add(MyBeam.Identifier.ToString, MyBeam.Identifier)
    'Add Piece identifier to the retrieved List of PieceIdentifiers
    ListofPieceIdentifiers.Add(MyPieceKey, MyPieceIdentifier)
End If
End While

'Read all pieces (Plate type) from the model and check their color
Dim MyPlateEnum As ModelObjectEnumerator =
MyModel.GetModelObjectSelector.GetAllObjectsWithType(ModelObject.ModelObjectEnum.CONTOURPLATE)
Dim MyPlate As ContourPlate
While MyPlateEnum.MoveNext
    'Retrieve the model's Piece (Plates) in order
    MyPlate = CType(MyModel.SelectModelObject(MyPlateEnum.Current.Identifier), ContourPlate)
    'Read the piece key (ASSEMBLY_POSITION) of the current Piece (Plate)
    MyPlate.GetReportProperty("ASSEMBLY_POS", MyPieceKey)
    'Eliminate extra characters from read piece key (ASSEMBLY_POSITION)
    MyPieceKey = Replace(MyPieceKey, "(?0)", "")
    MyPieceKey = Replace(MyPieceKey, "(?)", "")
    ClassColor = "1" 'Gray color which shows incompleted pieces
    'Check if current Piece (Plate) already has been completed and retrived its related color
    Dim CPI As Double = 0
    Dim SPI As Double = 0
    If MyVPieceList.Keys.Contains(MyPieceKey) Then
        MyVPiece = New VisualPiece
        MyVPiece = MyVPieceList(MyPieceKey)
        ClassColor = RelatedColor(MyVPiece.CPI, MyVPiece.SPI)
        CPI = MyVPiece.CPI
        SPI = MyVPiece.SPI
    End If
    'Set the related color of the piece
    If Not (MyPlate.Class = ClassColor) Then
        MyPlate.Class = ClassColor
        MyPlate.Modify()
    End If
    'Retrive related PieceIdentifier
    If ListofPieceIdentifiers.Keys.Contains(MyPieceKey) Then
        'Retrieve the stored PieceIdentifier
        MyPieceIdentifier = ListofPieceIdentifiers(MyPieceKey)
        'Add current identifier to the retrieved PieceIdentifier
        MyPieceIdentifier.ListofIdentifiers.Add(MyPlate.Identifier.ToString, MyPlate.Identifier)
    Else 'The piece has not been stored yet; create a new one
        MyPieceIdentifier = New PieceIdentifiers
        MyPieceIdentifier.PieceKey = MyPieceKey

```

```

        MyPieceIdentifier.CPI = CPI
        MyPieceIdentifier.SPI = SPI
        'Add current identifier to the retrieved PieceIdentifier
        MyPieceIdentifier.ListofIdentifiers.Add(MyPlate.Identifier.ToString, MyPlate.Identifier)
        'Add Piece identifier to the retrieved List of PieceIdentifiers
        ListofPieceIdentifiers.Add(MyPieceKey, MyPieceIdentifier)
    End If
End While
'Redraw the drawing
CommitChanges()
RelatedTeklaFileIsOpen = True
Else
    MessageBox.Show("First finish the model opening process and then continue!")
End If 'Result Yes
Catch ex As Exception
    MessageBox.Show(ex.Message)
End Try
ElseIf Not OpenApplicaion() Then
    MessageBox.Show("First finish the model opening process and then continue!")
End If 'No Tekla Program or currently working
Else ' No Tekla File or no tekla show requested
    RelatedTeklaFileIsOpen = False
End If
'Set New Piece Progress List
PieceProgressList = MyVPieceList
Return RelatedTeklaFileIsOpen
End Function

```

'Updates new received pieces at the Tekla model

```

Public Sub UpdateColors(ByVal MyVPiece As VisualPiece)
    'Do the entire procedure if Tekla is open or can be open
    If OpenApplicaion() And (CurrentDivision = MyVPiece.DivisionID) Then
        'Check if piece key exists
        If ListofPieceIdentifiers.Keys.Contains(MyVPiece.PieceKey) Then
            'Find the Piece from the piece list
            Dim MyPieceIdentifier As PieceIdentifiers = ListofPieceIdentifiers(MyVPiece.PieceKey)
            'Set the current SPI and CPI
            MyPieceIdentifier.CPI = MyVPiece.CPI
            MyPieceIdentifier.SPI = MyVPiece.SPI
            'Change the color of all related objects to the piece key
            Dim MyObject As ModelObject
            Dim ClassColor As String = RelatedColor(MyVPiece.CPI, MyVPiece.SPI)
            For Each IdentifierItem As Identifier In MyPieceIdentifier.ListofIdentifiers.Values
                Try
                    MyObject = MyModel.SelectModelObject(IdentifierItem)
                    'Set the related color of the piece
                    "Check the object Type
                    Dim ObjType As String = ""
                    MyObject.GetReportProperty("OBJECT_TYPE", ObjType)
                    "Cast the onject in current type and change the color
                    If ObjType = "PLATE" Then
                        Dim MyPlate As ContourPlate = CType(MyObject, ContourPlate)
                        If Not (MyPlate.Class = ClassColor) Then
                            MyPlate.Class = ClassColor
                            MyPlate.Modify()
                        End If
                    ElseIf ObjType = "BEAM" Then
                        Dim MyBeam As Beam = CType(MyObject, Beam)
                        If Not (MyBeam.Class = ClassColor) Then
                            MyBeam.Class = ClassColor
                            MyBeam.Modify()
                        End If
                    ElseIf ObjType = "PART" Then
                        Dim MyPart As Part = CType(MyObject, Part)
                        If Not (MyPart.Class = ClassColor) Then
                            MyPart.Class = ClassColor
                            MyPart.Modify()
                        End If
                    End If
                End If
            Next
            CommitChanges()
        End If
    End Sub

```

```

        Catch ex As Exception
            Ignore the error
        End Try
    Next
End If 'piece key exists
End If 'open application
End Sub

'Commit the changes and redraw previously updated colors
Public Sub CommitChanges()
    Try
        Commit the changes
        MyModel.CommitChanges()
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    End Try
End Sub

'Returns the current color based on the SPI and CPI of the piece
Public Function RelatedColor(ByVal CPI As Double, ByVal SPI As Double) As String
    Dim ColorClass As Integer = 1 'Gray
    Select Case SPI
        Case 0 To 0.9 'SPI
            Select Case CPI
                Case 0 To 0.9
                    ColorClass = 9 'Dark Red
                Case 0.9 To 0.9999999999
                    ColorClass = 9 'Dark Red
                Case Is >= 1
                    ColorClass = 6 'Yellow
            End Select
        Case 0.9 To 0.9999999999 'SPI
            Select Case CPI
                Case 0 To 0.9
                    ColorClass = 9 'Dark Red
                Case 0.9 To 0.9999999999
                    ColorClass = 4 'Light Blue
                Case Is >= 1
                    ColorClass = 6 'Yellow
            End Select
        Case Is >= 1 'SPI
            Select Case CPI
                Case 0 To 0.9
                    ColorClass = 13 'Orange
                Case 0.9 To 0.9999999999
                    ColorClass = 13 'Orange
                Case Is >= 1
                    ColorClass = 10 'Dark Green ""3 is light green
            End Select
    End Select
    Return ColorClass.ToString

'Color Codes:
'1: Gray()
'2: Light(Red)
'3: Light(Green)
'4: Light(Blue)
'5: Cyan()
'6: Yellow()
'7: Purple()
'8: Dark(Gray)
'9: Dark(Red)
'10: Dark(Green)
'11: Dark(Cyan)
'12: Dark(Purple)
'13: Orange()
'14: Dark(Blue)
'15: Gray()

End Function

```

End Class

Public Class DataVisual

```
Public DivisionList As New List(Of String) 'List of divisions on hand or completed

" ***** Initialize the VisualData Tables
Public Sub InitializeTables() 'Deletes the previously saved data in CompletedPieces table
    Dim MyDataName As String = "CompletedPieces"
    'Create a connection
    Dim MyAccessConn As OleDbConnection = New OleDbConnection("provider = Microsoft.jet.OLEDB.4.0;" &
"Data Source= ..\..\..\VisualData.mdb")
    'Create the Select Command
    Dim CommandText As String = "Select * from " & MyDataName & " ;"
    'Create a data adapter
    Dim DA As OleDbDataAdapter = New OleDbDataAdapter(CommandText, MyAccessConn)
    'Create the Delete Command
    CommandText = "Delete from " & MyDataName & " ;"
    'Create a data adapter delete command
    DA.DeleteCommand = New OleDbCommand(CommandText, MyAccessConn)
    'Use it when want to apply update command
    Dim builder As OleDbCommandBuilder = New OleDbCommandBuilder(DA)
    'Create a data Set
    Dim DS As DataSet = New DataSet
    Try
        MyAccessConn.Open()
        DA.Fill(DS, MyDataName)
        DA.DeleteCommand.ExecuteNonQuery()
        DS.Tables(MyDataName).Rows.Clear()
    Catch ex As Exception
        'Error
        MessageBox.Show(ex.Message)
    End Try
    MyAccessConn.Close()
End Sub

" ***** Update the VisualData Tables
Public Function UpdateCompletedPiecesTable(ByVal MyVPiece As VisualPiece) As Boolean
    " *****Update Completed Pieces Table and returns True if division previously has been added to the list
    Dim MyDataName As String = "CompletedPieces"
    'Create a connection
    Dim MyAccessConn As OleDbConnection = New OleDbConnection("provider = Microsoft.jet.OLEDB.4.0;" &
"Data Source= ..\..\..\VisualData.mdb")
    'Create the Select Command
    Dim CommandText As String = "Select * from " & MyDataName & " Where ((DivisionID = " & Chr(34) &
MyVPiece.DivisionID & Chr(34) & ") AND (PieceKey = " & Chr(34) & MyVPiece.PieceKey & Chr(34) & ")) ;"
    'Create a data adapter
    Dim DA As New OleDbDataAdapter(CommandText, MyAccessConn)
    "" Delete the old data of piece
    'Create the Delete Command
    CommandText = "Delete * from " & MyDataName & " Where ((DivisionID = " & Chr(34) & MyVPiece.DivisionID
& Chr(34) & ") AND (PieceKey = " & Chr(34) & MyVPiece.PieceKey & Chr(34) & ")) ;"
    'Create a data adapter delete command
    DA.DeleteCommand = New OleDbCommand(CommandText, MyAccessConn)
    'Use it when want to apply update command
    Dim Builder As OleDbCommandBuilder = New OleDbCommandBuilder(DA)
    'Create a data Set
    Dim DS As New DataSet
    Try
        'Open the connection
        MyAccessConn.Open()
        'Fill the Data Set
        DA.Fill(DS, MyDataName)
        'Delete the old data related the piece
        DA.DeleteCommand.ExecuteNonQuery()
        DS.Tables(MyDataName).Rows.Clear()
    Catch ex As Exception
        ' Error
        MessageBox.Show(ex.Message)
    End Try
End Function
```

```

End Try

'''    Add updated data of piece
'Define a row in the table
Dim MyRow As DataRow = DS.Tables(MyDataName).NewRow
MyRow("PieceKey") = MyVPiece.PieceKey
MyRow("CPI") = MyVPiece.CPI
MyRow("SPI") = MyVPiece.SPI
MyRow("Progress") = MyVPiece.Progress
MyRow("DivisionID") = MyVPiece.DivisionID
DS.Tables(MyDataName).Rows.Add(MyRow)
'Update the Data
DA.Update(DS, MyDataName)
MyAccessConn.Close()
'Add the division to the DivisionList if it has not been added
If DivisionList.Contains(MyVPiece.DivisionID) Then
    Return False
Else
    DivisionList.Add(MyVPiece.DivisionID)
    Return True
End If
End Function

" """"""""""""""""""""Reads Data from VisualData
Public Sub ReadVPieces(ByVal MyDivision As String, ByRef MyVPieceList As Dictionary(Of String, VisualPiece))
    'Read Completed Pieces data from the Data Base
    " """"""""""""""""""""Set CompletedPieces as the table name
    Dim MyDataName As String = "CompletedPieces"
    'Create a connection
    Dim MyAccessConn As New OleDbConnection("provider = Microsoft.jet.OLEDB.4.0;" & "Data Source=
..A.A.A.\VisualData.mdb")
    'Create the Select Command
    Dim CommandText As String = "Select * from " & MyDataName & " Where DivisionID = " & Chr(34) &
MyDivision & Chr(34) & ";"
    'Create a data adapter
    Dim DA As New OleDbDataAdapter(CommandText, MyAccessConn)
    'Create a DataSet
    Dim DS As New DataSet
    'Create a DataSet
    Dim MyData As New DataTable
    Try
        MyAccessConn.Open()
        'Fill Data to Data set
        DA.Fill(DS, MyDataName)
        'Fill Data to DataTable
        MyData = DS.Tables(MyDataName)
        MyAccessConn.Close()
    Catch ex As Exception
        ' There is an error
        MessageBox.Show(ex.Message)
    End Try
    Dim MyVPiece As VisualPiece
    'Add the read Pieces to the MyVPieceList
    For Each MyVPieceRow As DataRow In MyData.Rows
        MyVPiece = New VisualPiece
        MyVPiece.PieceKey = MyVPieceRow("PieceKey").ToString
        MyVPiece.CPI = Convert.ToDouble(MyVPieceRow("CPI"))
        MyVPiece.SPI = Convert.ToDouble(MyVPieceRow("SPI"))
        MyVPiece.Progress = Convert.ToDouble(MyVPieceRow("Progress"))
        MyVPiece.DivisionID = MyVPieceRow("DivisionID").ToString
        MyVPieceList.Add(MyVPiece.PieceKey, MyVPiece)
    Next
End Sub

Public Sub ReadDivisionFile(ByVal MyDivision As String, ByRef DivisionFile As String)
    'Read DivisionFiles data from the Data Base
    " """"""""""""""""""""Set CompletedPieces as the table name
    Dim MyDataName As String = "DivisionFile"
    'Create a connection

```

```

    Dim MyAccessConn As New OleDbConnection("provider = Microsoft.jet.OLEDB.4.0;" & "Data Source=
..A.A.A.\VisualData.mdb")
    'Create the Select Command
    Dim CommandText As String = "Select * from " & MyDataName & " Where DivisionID = " & Chr(34) &
MyDivision & Chr(34) & ";"
    'Create a data adapter
    Dim DA As New OleDbDataAdapter(CommandText, MyAccessConn)
    'Create a DataSet
    Dim DS As New DataSet
    'Create a DataSet
    Dim MyData As New DataTable
    Try
        MyAccessConn.Open()
        'Fill Data to Data set
        DA.Fill(DS, MyDataName)
        'Fill Data to DataTable
        MyData = DS.Tables(MyDataName)
        MyAccessConn.Close()
    Catch ex As Exception
        ' There is an error
        MessageBox.Show(ex.Message)
    End Try
    'Check see if there is any file available
    If MyData.Rows.Count > 0 Then
        'Pass the read Division information to the parameters
        For Each MyDivisionFile As DataRow In MyData.Rows
            DivisionFile = CStr(MyDivisionFile("FileName"))
        Next
    Else 'No file is stored
        DivisionFile = "No Tekla File"
    End If
End Sub
End Class

```


B.3. Data-tables

B.3.1. Piece Table

Table B.1 presents Piece data-table used in the developed model as the main source of data input. The column headings in Table B.1 are the data-fields in the data-table developed in MS Access. The presented values in Table B.1 show some sample data used for the model.

Table B.1. Piece table structure with sample data

div_id	piece_id	quantity	weight	fab_mhrs	fabdwg_no	assembly_pos
13028	386536	1	635	15.41	24	16R24
13028	386524	1	2170	15.41	12	16R12
13028	386588	1	109	15.41	80	16R80
13028	386577	5	876	15.41	68	16R68
13028	386574	2	361	15.41	65	16R65
9772	160533	4	3	20.00	8600	1A38600
9772	160553	1	0	20.00	1	1A31
9772	160568	1	0	20.00	16	1A316
9772	161920	9	6	20.00	8600	1A38600
11808	222038	7	1	60.00	8600	71A8600
11808	222038	7	1	96.25	8600	71A8600
12855	227213	4	48	38.81	90003	3B90003
12855	227213	4	48	60.00	90003	3B90003
12855	227214	1	154	38.81	9008	3B9008
12855	227214	1	154	60.00	9008	3B9008
12857	227245	8	96	38.99	90004	4A90004
12857	227245	8	96	60.00	90004	4A90004
12857	227246	1	154	38.99	9009	4A9009
12857	227246	1	154	60.00	9009	4A9009
11057	227323	1	74	60.00	9039	59B9039
11808	230173	1	65	60.00	5001	71A5001
11808	230173	1	65	96.25	5001	71A5001
11808	230179	1	55	96.25	5000	71A5000
11898	230697	1	33	39.03	1	13C1
11898	230697	1	33	60.00	1	13C1

Brief explanations on the meaning of each data-field are presented in the following:

div_id: The id of the division which piece is belonged to.

piece_id: The piece id.

quantity: The number of identical pieces, with the same piece id, which should be fabricated.

weight: The weight of the piece

fab_mhrs: The average man-hour required for fabricating every kilogram of the piece.

fabdwg_no: The number of the drawing which contains the piece's drawing. This is used for locating Tekla (Tekla Corporation, Finland, <http://www.tekla.com>) drawing file for visualization purposes.

assembly_pos: Determines the position of piece drawing inside the Tekla file. This is used for visualization purposes.

B.3.2. Division Table

Table B.2 presents Division data-table used in the program as the supplementary data source to the Piece data-table. The column headings of Table B.2 represent the data-fields in the data-table implemented in MS Access. The presented values in Table B.2 show some sample data used in the model.

Table B.2. Division table structure with sample data

div_id	fab_start_date	required_date	Weight_sum	requires_painting	NoOfPieces
15017	02-Jan-09	27-Jan-09	55304	FALSE	13062
15012	05-Jan-09	06-Jul-09	44720	FALSE	6776
15638	05-Jan-09	18-Dec-08	6003	FALSE	1270
17897	06-Jan-09	08-Jan-09	724	TRUE	2
16092	06-Jan-09	13-Oct-08	17806	FALSE	1009
15616	06-Jan-09	10-Oct-08	144504	TRUE	40572
17019	07-Jan-09	26-Jan-09	6910	FALSE	138
17018	07-Jan-09	26-Jan-09	52971	FALSE	6711
13970	07-Jan-09	17-Jan-09	32961	FALSE	15330
16532	07-Jan-09	01-Jan-09	94549	FALSE	26730
16527	07-Jan-09	13-Jan-09	66636	FALSE	8754
16521	07-Jan-09	09-Jan-09	27524	FALSE	3062
16652	07-Jan-09	11-Feb-09	19003	FALSE	6513
16519	08-Jan-09	14-Nov-08	37701	FALSE	5942
13769	08-Jan-09	08-Aug-08	52098	FALSE	13368
17710	08-Jan-09	28-Jan-09	5203	TRUE	182
16518	12-Jan-09	08-Jan-09	24270	FALSE	5019
14395	12-Jan-09	30-Jan-09	28645	FALSE	8312
17653	12-Jan-09	26-Jan-09	15878	FALSE	978
17017	12-Jan-09	26-Jan-09	15111	TRUE	3354
17016	12-Jan-09	26-Jan-09	42775	FALSE	5388
16687	12-Jan-09	26-Jan-09	68661	FALSE	17162
16340	12-Jan-09	09-Feb-09	4285	FALSE	552
14394	12-Jan-09	30-Jan-09	28645	FALSE	8312
13974	12-Jan-09	18-Jan-09	45166	FALSE	16500

Brief explanations on the meaning of each data-field are presented in the following:

div_id: The id of the division.

fab_start_date: The scheduled date in which division (i.e., all pieces within the division) is sent to the shop.

required_date: The date in which fabrication of the division (i.e., fabrication of all pieces within the division) is required to be complete.

weight_sum: Total weight of the pieces within a division.

requires_painting: Determines where pieces within division require painting (True) or not (False).

NoOfPieces: Total number of pieces within the division.

B.3.3. ReportDivision Table

Table B.3 presents ReportDivision data-table used in the program for collecting the simulation results during the model runs. The collected data in the data-table were used for testing the model performance as presented in Section 2.5.3 of Chapter 2.

Table B.3. Sample output reports to the ReportDivision data-table

div_id	ReportTime	Start	Finish	Delay	WeightedDelay
15452	10-Apr-09 12:09:10AM	20-Jan-09	20-Jan-09	-68	-64464
15767	10-Apr-09 12:12:55AM	21-Jan-09	26-Jan-09	-103	-2265897
15492	10-Apr-09 12:13:06AM	26-Jan-09	27-Jan-09	-50	-84150
15490	10-Apr-09 12:13:08AM	26-Jan-09	27-Jan-09	-50	-584750
15759	10-Apr-09 12:16:27AM	27-Jan-09	04-Feb-09	-128	-5863552
15763	10-Apr-09 12:21:41AM	03-Feb-09	12-Feb-09	-122	-5034574
16400	10-Apr-09 12:24:34AM	12-Feb-09	18-Feb-09	0	0
15391	10-Apr-09 12:27:55AM	17-Feb-09	23-Feb-09	0	0
14914	10-Apr-09 12:30:22AM	23-Feb-09	02-Mar-09	-348	-9961152
17431	10-Apr-09 12:31:09AM	04-Mar-09	04-Mar-09	-107	0
15564	10-Apr-09 12:31:18AM	28-Feb-09	05-Mar-09	-108	-4641732
16401	10-Apr-09 12:31:19AM	04-Mar-09	05-Mar-09	0	0
16409	10-Apr-09 12:33:36AM	04-Mar-09	10-Mar-09	0	0
16165	10-Apr-09 12:34:10AM	09-Mar-09	11-Mar-09	-1	-7034
16878	10-Apr-09 1:01:21 AM	10-Mar-09	08-Apr-09	-12	-1694892
15080	10-Apr-09 1:07:21 AM	07-Apr-09	16-Apr-09	0	0
15081	10-Apr-09 1:28:20 AM	23-Apr-09	05-May-09	0	0
16402	10-Apr-09 1:32:45 AM	15-Apr-09	08-May-09	0	0
15082	10-Apr-09 1:54:16 AM	13-May-09	25-May-09	0	0
16403	10-Apr-09 1:54:23 AM	05-May-09	25-May-09	0	0
16404	10-Apr-09 1:59:42 AM	23-May-09	02-Jun-09	0	0
16002	10-Apr-09 1:59:47 AM	01-Jun-09	02-Jun-09	-15	0
15996	10-Apr-09 2:08:01 AM	01-Jun-09	12-Jun-09	-36	-1692108
18029	10-Apr-09 2:08:03 AM	10-Jun-09	12-Jun-09	-64	-268800

Brief explanations on the meaning of each data-field are presented in the following:

div_id: The id of the division.

ReportTime: The time and date that the division report is stored in the database.

Start: The simulated start date of the division.

Finish: The simulated finish date of the division.

Delay: The achieved delay for completing the division in days.

WeightedDelay: The achieved delay for completing the division in day-kilogram.

B.3.4. ReportSchedule Table

Table B.4 presents ReportSchedule data-table used in the model for collecting the duration of the simulation runs. The collected data in the data-table were used for testing the simulation model calculation time as presented in Section 2.5.3 of Chapter 2.

Table B.4. ReportSchdeule data table with Sample output data from the model

ReportTime	SimDuration
13-Feb-09	277.52
20-Feb-09	31.13
21-Feb-09	47.33
03-Apr-09	89.00
07-Apr-09	157.95
08-Apr-09	88.12
08-Apr-09	66.88
08-Apr-09	148.30
08-Apr-09	93.02
09-Apr-09	92.47
09-Apr-09	90.78
10-Apr-09	119.00
10-Apr-09	82.57
10-Apr-09	79.72
11-Apr-09	68.33
12-Apr-09	72.52
12-Apr-09	75.60
12-Apr-09	80.93
12-Apr-09	72.18
12-Apr-09	78.73
13-Apr-09	75.37
13-Apr-09	77.28
13-Apr-09	72.53
13-Apr-09	143.57
13-Apr-09	68.32

Brief explanations on the meaning of each data-field are presented in the following:

ReportTime: The time and date that the division report is stored in the database.

SimDuration: The duration of the run of simulation in minute.

B.3.5. RFIDPieceRelation Table

Table B.5 presents RFIDPieceRelation data-table used in the model for relating RFID tags to pieces which they are attached to. This data-table added to the model for testing the expandability of the model, as presented in Section 2.5.4 in Chapter 2.

Table B.5. RFIDPieceRelation table with sample data

RFID	div_id	piece_ID
1_2137_16782_19741_394483	16782	394483
1_2137_16782_19741_395342	16782	395342
53_2111_13563_19859_253092	13563	253092
53_2111_13563_19859_280228	13563	280228
53_2396_17875_19656_467591	17875	467591
53_2397_17876_19657_464192	17876	464192
53_2397_17876_19657_467592	17876	467592
130_2352_17144_18642_405453	17144	405453
130_2383_17753_19429_433492	17753	433492
130_2383_17753_19429_433493	17753	433493
131_2360_17439_18994_396803	17439	396803
131_2360_17440_18995_396786	17440	396786
131_2360_17441_18996_396839	17441	396839
151_1898_17581_19176_477740	17581	477740
151_1948_16653_19765_458096	16653	458096
151_1948_16653_19765_458787	16653	458787
151_1948_17394_18942_443855	17394	443855
155_2004_16731_19654_410775	16731	410775
155_2004_16731_19654_410776	16731	410776
160_1911_17870_19643_469920	17870	469920
160_1911_17870_19643_469924	17870	469924
191_2124_13028_13934_386544	13028	386544
191_2124_13028_13934_386545	13028	386545
191_2124_13028_13934_386546	13028	386546
191_2124_13028_13934_386547	13028	386547

Brief explanations on the meaning of each data-field are presented in the following:

RFID: The id assigned to the RFID tag.

div_id: The id of the division which piece is belonged to.

piece_id: The piece id.

B.3.6. RFIDRead Table

Table B.6 presents RFIDRead data-table which contains the date and the time of every RFID tag read in the shop. This table is updated based on the data sent by RFID reader. Since each tag record is read at the completion of a fabrication operation, the model uses these records for locating the location and progress of the pieces within the fabrication shop.

Table B.6. RFIDRead table with sample data

RFID	ReadTime
1_2137_16782_19741_394483	08-Feb-09 7:44:21 PM
191_2124_13028_13934_386513	08-Feb-09 7:55:35 PM
191_2124_13028_13934_386513	08-Feb-09 7:59:02 PM
151_1898_17581_19176_477740	08-Feb-09 7:59:11 PM
130_2352_17144_18642_405453	08-Feb-09 7:59:48 PM
130_2383_17753_19429_433492	08-Feb-09 8:08:47 PM
130_2383_17753_19429_433493	08-Feb-09 8:10:30 PM
131_2360_17439_18994_396803	08-Feb-09 8:53:19 PM
1_2137_16782_19741_394483	08-Feb-09 9:11:11 PM
131_2360_17441_18996_396839	08-Feb-09 10:15:27 PM
151_1898_17581_19176_477740	08-Feb-09 10:23:48 PM
1_2137_16782_19741_394483	08-Feb-09 11:13:03 PM
151_1948_16653_19765_458787	08-Feb-09 11:15:32 PM
151_1948_17394_18942_443855	08-Feb-09 11:43:50 PM
155_2004_16731_19654_410775	08-Feb-09 11:43:58 PM
155_2004_16731_19654_410776	09-Feb-09 12:06:35 AM
160_1911_17870_19643_469920	09-Feb-09 12:06:39 AM
160_1911_17870_19643_469924	13-Feb-09 12:01:22 PM
191_2124_13028_13934_386544	13-Feb-09 12:13:14 PM
151_1898_17581_19176_477740	13-Feb-09 12:14:41 PM
131_2360_17440_18995_396786	13-Feb-09 12:14:58 PM
131_2360_17441_18996_396839	13-Feb-09 12:30:43 PM
191_2124_13028_13934_386513	13-Feb-09 12:50:24 PM
151_1948_16653_19765_458787	13-Feb-09 12:57:59 PM
151_1948_17394_18942_443855	13-Feb-09 1:05:55 PM

Brief explanations on the meaning of each data-field are presented in the following:

RFID: The id assigned to the RFID tag.

ReadTime: The date and the time that RFID has been read in the shop.

Reference

Rumbaugh, J., Jacobson, I. and Booch, G. (1999) The unified modeling language reference manual. Addison Wesley Longman Inc. One Jacob Way, Reading, Massachusetts, 01867, USA.

Appendix C

Programming Details of the Simulation Model Used for Working Hours Dynamics

AnyLogic 6.4 was used for implementing all SD and hybrid SD-DES models developed for testing and analyzing working hour dynamics in Chapter 3. Data-tables in MS Access database were used for providing the required data-tables and the links to the collaborative company's database for the experimental case.

C.1. AnyLogic Model

The developed AnyLogic model consists of four main parts; SD model, DES model, supporting classes and simulation setting (Figure C.1).

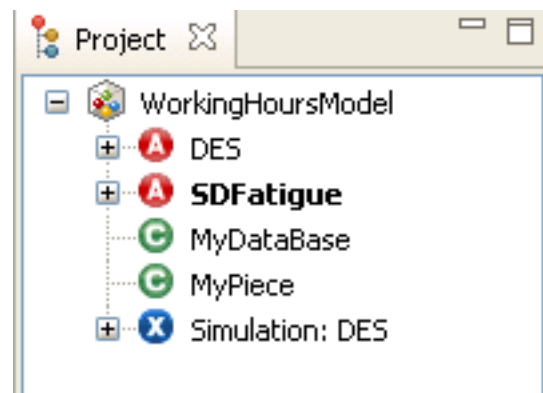


Figure C.1. Structure of the AnyLogic

Brief explanation on first three parts comes in following. Simulation setting part of the model is a standard part of every AnyLogic model and is explained in the AnyLogic user manuals accessible at: www.xjtek.com.

C.1. 1. SD Model

The developed SD model in AnyLogic consists of four main sub-models, as explained in Section 3.2 of Chapter 3, including Physical Energy Dynamics, Mental Resource Dynamics, Hour in Day Dynamics and Overtime Fatigue Dynamics (Figure C.2).

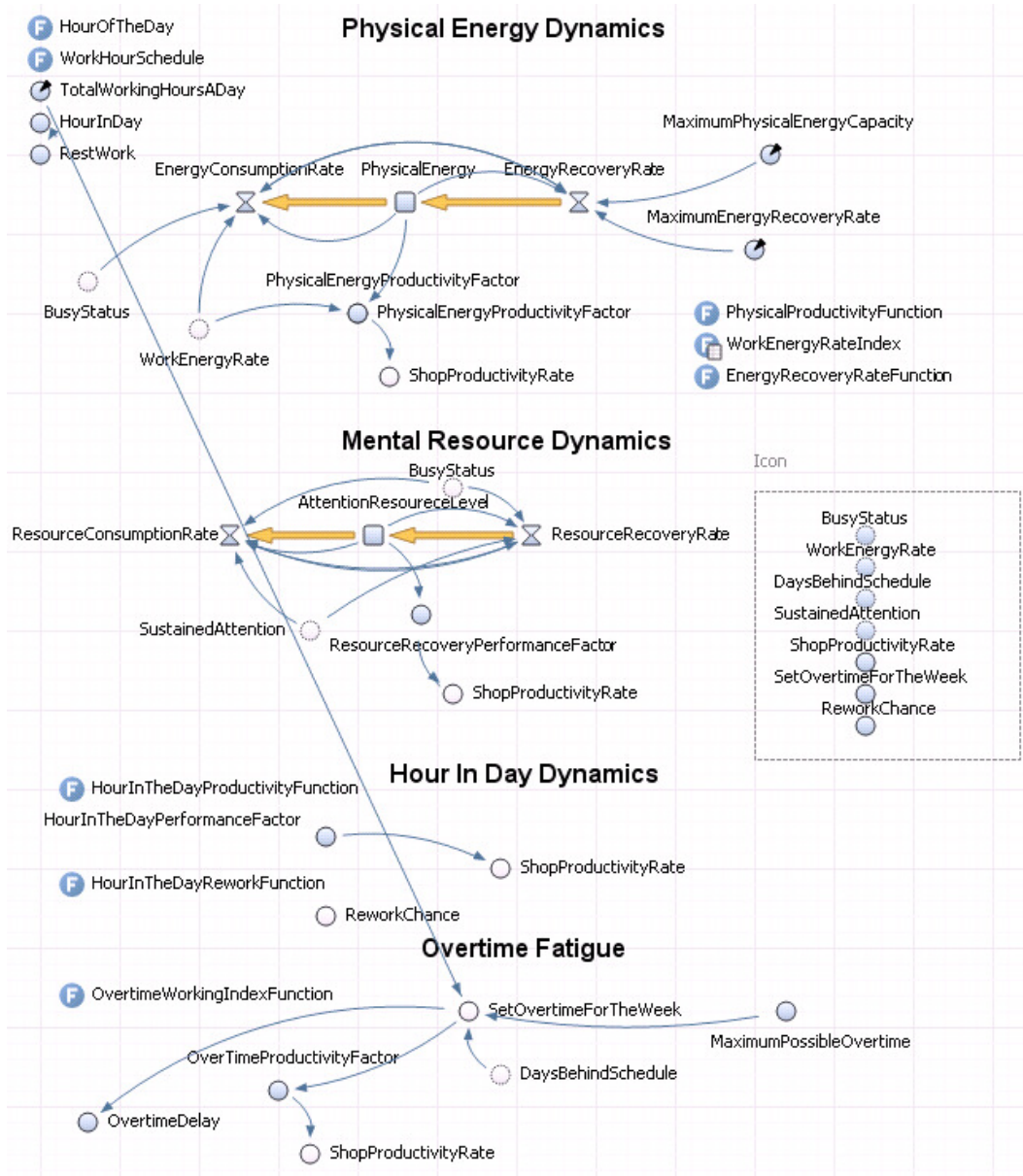







Figure C.2. SD model of working hours dynamics

Equations, explained in Chapter 3, are added to the model using different model elements. In AnyLogic stock variables are shown in square (), flows are shown in valve shape (), auxiliary variables are shown in circles () and model parameters are represented by circle with a black triangle on its top-right side (). The related variables are linked by arrows. Function elements () were used in the model when some equation parameters are read from tables or there are multi-conditional equations linking the model variables (e.g., OvertimeWorkingIndexFunction calculates overtime productivity ratio as explained in Section 3.2.3 of Chapter 3, HourInTheDayReworkFunction captures reliability changes based on changes in the hour of the day as explained in Section 3.2.3 of Chapter 3, and HourInTheDayProductivityFunction which captures productivity ratio changes based on the changes in the hour of the day as explained in Section 3.2.3 of Chapter 3). In addition to that, functions were used for adding more capabilities to the model. HourOfTheDay and WorkHourSchedule are two functions which are used respectively for translating the simulation logical hours to the actual daily hours and determining the status of the working hour (i.e., working or non-working hours). The codes used inside these functions (in Java) are as in below:

HourOfTheDay Function:

```

double HourValue = ((TimeNow/60)%TotalWorkingHoursADay);
double HourInDay=0;
if (HourValue>=0 & HourValue<=2) { HourInDay = HourValue+2.5;}
else if (HourValue>2 & HourValue<=4) { HourInDay = HourValue+2.75;}
else if (HourValue>4 & HourValue<=6) { HourInDay = HourValue+3.25;}
else if (HourValue>6 & HourValue<=8) { HourInDay = HourValue+3.5;}
else if (HourValue>8 & HourValue<=9.25) { HourInDay = HourValue+3.75;}
else if (HourValue>9.25 & HourValue<=11.25) { HourInDay = HourValue+4.25;}
else if (HourValue>11.25 & HourValue<=13.25) { HourInDay = HourValue+4.5;}
else if (HourValue>13.25 & HourValue<=15.25) { HourInDay = HourValue+5;}

```

```

else if (HourValue>15.25 & HourValue<=17.25) { HourInDay = HourValue+5.25;}
else if (HourValue>17.25 & HourValue<=18) { HourInDay = HourValue+5.5;}
return (HourInDay%24);

```

WorkHourSchedule Function:

```

double HourValue = ((TimeNow/60)%TotalWorkingHoursADay);
double HourInDay=0;
if (HourValue>=0 & HourValue<=2) { HourInDay = HourValue+2.5;}
else if (HourValue>2 & HourValue<=4) { HourInDay = HourValue+2.75;}
else if (HourValue>4 & HourValue<=6) { HourInDay = HourValue+3.25;}
else if (HourValue>6 & HourValue<=8) { HourInDay = HourValue+3.5;}
else if (HourValue>8 & HourValue<=9.25) { HourInDay = HourValue+3.75;}
else if (HourValue>9.25 & HourValue<=11.25) { HourInDay = HourValue+4.25;}
else if (HourValue>11.25 & HourValue<=13.25) { HourInDay = HourValue+4.5;}
else if (HourValue>13.25 & HourValue<=15.25) { HourInDay = HourValue+5;}
else if (HourValue>15.25 & HourValue<=17.25) { HourInDay = HourValue+5.25;}
else if (HourValue>17.25 & HourValue<=18) { HourInDay = HourValue+5.5;}

return (HourInDay%24);

```

Finally, variables inside the dashed rectangle, staying at the right side of Figure C.2, are interface variables which handle hybrid interactions between SD and DES models (as explained in Section 3.5 and Figure 3-10 in Chapter 3).

C.1. 2. DES Model

Figure C.3 presents a screen shot of the DES model developed for the structural steel fabrication shop case study explained in Section 3.6 of Chapter 3. Unlike the SD model structure, the DES model has a specific structure to the project.

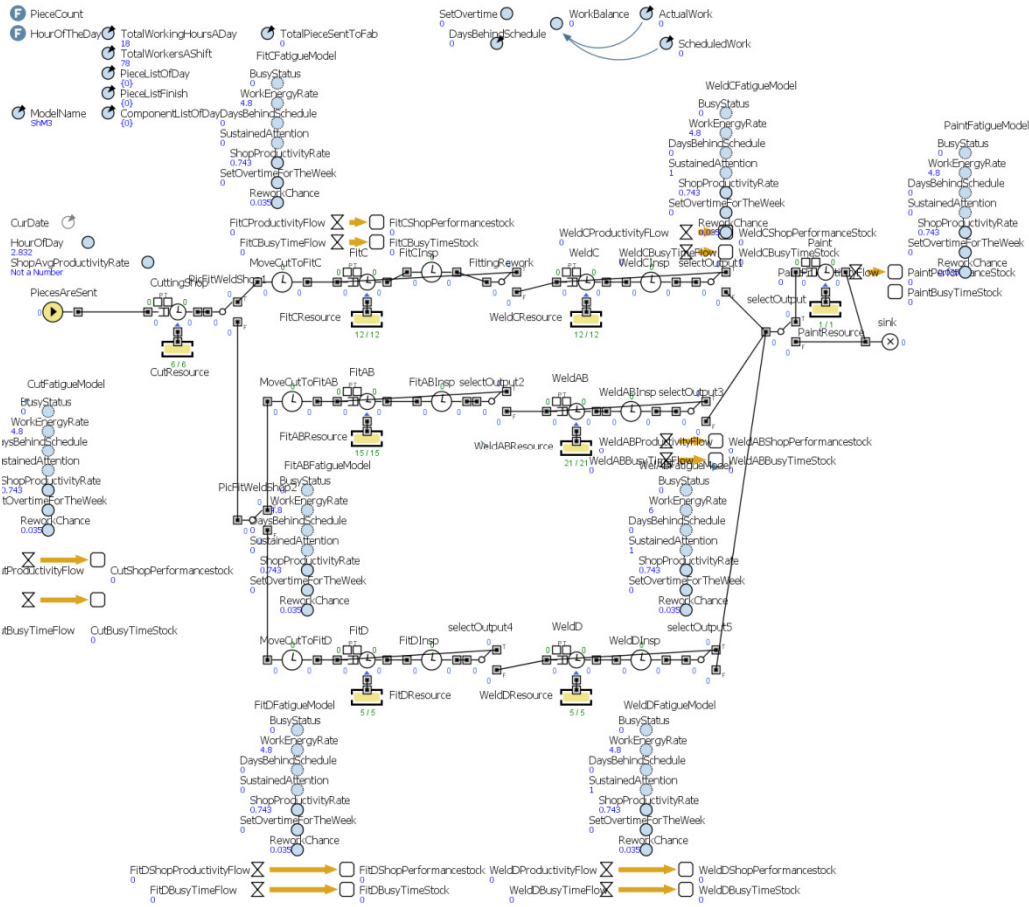


Figure C.3. DES model of the structural steel fabrication shop

The main structure of the DES model has been developed by relating different DES model elements provided by AnyLogic, including entity creator for generating the pieces (🔵➡️), services which represent working stations (🏠), delays which represent inspection stations (🕒), resources which represent crews (👷) and conditional branches for direction the flow of pieces base of different conditions (🔍). There are two sets of interface variables linked to each other by arrows beside every service element (eight sets of interface variables in total). One set of variables come from SD model and the other sets is linked to different parameters of the working station (represented by service element). The direction of the arrows shows the direction of data flow (i.e., the variable at the tail updates the variable at the head of the arrow).

A stock and flow mechanism, placed beside every station, has been used for continuously tracking the average level of the productivity in the shop. Finally, two auxiliary functions have been used in the model; HourOfTheDay and PieceCount. HourOfTheDay is a duplicate function from SD model. PieceCount function uses the database functions (presented in MyDataBase supporting class in Section C.1.3) to get linked to the database and sends the pieces to the shop on daily basis. The codes used inside this function (in Java) are as in below:

PieceCount:

```

CountedPiecesOfTheDay=0;
if (!SetCalendar){
    CurrentDate.set(2009,0,5);
    StartDate.set(2009,0,5);
    FinishDate.set(2009,3,5);
    SetCalendar=true;
}
MyDataBase myDB = new
MyDataBase(this,CurrentDate.getTime(),StartDate.getTime(),FinishDate.getTime());
if (CurrentDate.before(FinishDate))
{
    myDB.PieceOfTheCurDay(PieceListOfDay,ComponentListOfDay);
}
System.out.println(CurrentDate.getTime() + "          " + ActualWork + "          " + AvgProductivity+"
                    " + TotalPieceSentToFab+"          " + TotalPieceFabricated);

//Increase the the day
int DayNumToIncrease=1;
if (CurrentDate.getTime().getDay()==4)
{
    if (SetOvertime ==0)
    {
        DayNumToIncrease=4;
    } else {
        DayNumToIncrease=1;
    }
} else if (CurrentDate.getTime().getDay()==5){
    if (SetOvertime <=1)
    {
        DayNumToIncrease=3;
    } else {
        DayNumToIncrease=1;
    }
} else if (CurrentDate.getTime().getDay()==6){
    DayNumToIncrease=2;
} else {
    DayNumToIncrease=1;
}
//Increase the date and receive any scheduled pieces
CurrentDate.add(Calendar.DAY_OF_MONTH,1);
for (int i=1; i<DayNumToIncrease; i++)
{
    if (CurrentDate.before(FinishDate))
    {
        MyDataBase DayOffDB = new
MyDataBase(this,CurrentDate.getTime(),StartDate.getTime(),FinishDate.getTime());

```

```

        DayOffDB.PieceOfTheCurDay(PieceListOfDay,ComponentListOfDay);
        CountedPiecesOfTheDay = PieceListOfDay.size();
    }
    CurrentDate.add(Calendar.DAY_OF_MONTH,1);
}
//Calculate Required Overtime Based on the progress just on weekly basis
if (CurrentDate.getTime().getDay()==1){
    ScheduledWork=myDB.ScheduledWork();
    DaysBehindSchedule=(int)max(((ScheduledWork-
    ActualWork)/TotalWorkersAShift/TotalWorkingHoursADay),0);
}
CurDate=format(CurrentDate.getTime());
//Report the finished pieces
//myDB.ReportCompletedPieces(PieceListFinish, ModelName);
//myDB.ReportCompletedPieces(ComponentListOfDay, ModelName);

CountedPiecesOfTheDay = PieceListOfDay.size();
PieceListFinish.clear();
ComponentListOfDay.clear();
return CountedPiecesOfTheDay;

```

C.1.3. Supporting Classes

Two supporting classes (in Java) are added to the model; MyPiece and MyDataBase. MyPiece class provides the set of attributes required for the entities (pieces) and MyDataBase class provides the set of attributes and functions required for communicating with the collaborative company's database. Codes used in every class are as in below:

MyPiece Class:

```

* MyPiece
*/
public class MyPiece extends Entity{

    /**
    * Default constructor
    */
    public MyPiece(){

        // Project ID
        public String ProjID;

        // Job ID
        public String JobID;

        // Division ID
        public String DivID;

        // Sub Division ID

```

```

    public String SubID;

    // Piece ID
    public String PieceID;

    // Piece Count
    public int Count;

    // Piece Quantity
    public int Quantity;

    // Piece Weight
    public double PieceWeight;

    // Piece Man Hour per Ton
    public double PieceMhTon;

    // Piece Man Hour per Ton
    public Boolean PaintIsRequired;

    // Piece Cut Duration
    public double PieceCutDur;

    // Piece Fit Duration
    public double PieceFitDur;

    // Piece Fit Inspection Duration
    public double PieceFitInspDur;

    // Piece Welding Duration
    public double PieceWeldDur;

    // Piece Welding Inspection Duration
    public double PieceWeldInspDur;

    // Piece Painting Duration
    public double PiecePaintDur;

    // Division Weight
    public double DivWeight;

    // Fab Finish Date
    public Date FabFinishDate ;

    @Override
    public String toString() {
        return super.toString();
    }
}

```

MyDataBase Class:

```

import com.xj.anylogic.engine.connectivity.*;
/**
 * MyDataBase
 */
public class MyDataBase {

    Date CurrentDate ;
    Date StartDate ;
    Date FinishDate ;
    int ComponentsCount=0;
    int PiecesCount=0;
    Presentable Owner;

```



```

//      MyPiece readPiece;
/**
 * Default constructor
 */
public MyDataBase(Presentable myOwner, Date CurDate, Date StrDate, Date FnshDate)
{
    Owner=myOwner;
    CurrentDate=CurDate;
    StartDate=StrDate;
    FinishDate=FnshDate;
}

public void PieceOfTheCurDay(List<MyPiece> PiecesOfTheDay, List<MyPiece> ComponentsOfTheDay)
{
    // Databases
    Database PieceDataBase = new Database( Owner, "PieceDataBase",
"D:\User\Amin\PhdCourse\03_Project\RA_DrLeeGroup\DrLee_Modeling\17_PilotModel_FatigueInFabS
hop\Pilot_FatigueInShop.mdb");
    String SelectStatement = "SELECT Piece.proj_id, Piece.job_id, Piece.div_id, Piece.Sub_ID,
Piece.piece_id, Piece.quantity, Piece.WeightofPiece, Piece.fab_mhrs,
[Piece].[fab_mhrs]*[WeightofPiece]/1000*60 AS Piece_Fab_Minute, Division.requires_painting,
Division.WeightSum From Piece INNER JOIN Division ON (Piece.proj_id = Division.proj_id) AND
(Piece.job_id = Division.job_id) AND (Piece.div_id = Division.div_id) AND (Piece.Sub_ID = Division.sub_id)
Where (Piece.weight>0 AND Piece.fab_start_date =#" + (CurrentDate.getMonth()+1)
+"/" +CurrentDate.getDate() +"/" +(CurrentDate.getYear()+1900)+"#)";
    ResultSet rs= PieceDataBase.getResultSet(SelectStatement);
    int PieceOrComponent=1;
    MyPiece CurPiece ;
    int Count ;
    double PaintDurFactor =1;//it is used in other durations than paint as a factor
    int PaintRequired=1;//it is used just in Paint durations as a factor
    while (rs.next()) // this will step through our data row-by-row
    {
        /* the next line will get the first column in our current row's ResultSet
as a String ( getString( columnNumber ) ) and output it to the screen */

        Count = 0;//Set the count as 0 for the current read piece type
            // Project ID
            String ProjID= rs.getString(1);

            // Job ID
            String JobID= rs.getString(2);

            // Division ID
            String DivID= rs.getString(3);

            // Sub Division ID
            String SubID= rs.getString(4);

            // Piece ID
            String PieceID=rs.getString(5);

            // Piece Quantity
            int Quantity=rs.getInt(6);

            // Piece Weight
            double PieceWeight=rs.getDouble(7);

            // Piece Man Hour per Ton
            double PieceMhTon=rs.getDouble(8);

            // Piece Total Duration in Minutes
            double PieceDur=Math.min(rs.getDouble(9)/2,1000);

            //Paint Required
            Boolean PaintsRequired = rs.getBoolean(10);

            //Division Weight
            double DivWeight = rs.getDouble(11);
    }
}

```

```

if (PaintIsRequired)//If paint is required: paint duration factor is 1
{
    PaintDurFactor =1    ;
    PaintRequired=1;
}
else
{
    PaintDurFactor =1.1;
    PaintRequired=0;
}
while (Quantity>Count)//Quantity Loop: Loop as long as all piece instances of the same type are
created
{
    CurPiece = new MyPiece();
    Count++;

    // Project ID
    CurPiece.ProjID= ProjID;

    // Job ID
    CurPiece.JobID= JobID;

    // Division ID
    CurPiece.DivID= DivID;

    // Sub Division ID
    CurPiece.SubID= SubID;

    // Piece ID
    CurPiece.PieceID=PieceID;

    // Piece Count
    CurPiece.Count=Count;

    // Piece Quantity
    CurPiece.Quantity=Quantity;

    // Piece Weight
    CurPiece.PieceWeight=PieceWeight;

    // Piece Man Hour per Ton
    CurPiece.PieceMhTon=PieceMhTon;

    // Piece Man Hour per Ton
    CurPiece.PaintIsRequired=PaintIsRequired;

if (PieceWeight<=5)//Piece just will be drilled, no fit no weld is required
{
    PieceOrComponent=2;
    ComponentsCount++;
    CurPiece.PieceCutDur=PieceDur* PaintDurFactor;
    CurPiece.FabFinishDate=CurrentDate;
}
else
{
    PieceOrComponent=1;
    PiecesCount++;

    // Piece Cut Duration
    CurPiece.PieceCutDur=PieceDur* 0.01 * PaintDurFactor;

    // Piece Fit Duration
    CurPiece.PieceFitDur=PieceDur* 0.4 * PaintDurFactor;

    // Piece Fit Inspection Duration
    CurPiece.PieceFitInspDur=PieceDur* 0.045 * PaintDurFactor;

    // Piece Welding Duration
    CurPiece.PieceWeldDur=PieceDur* 0.4 * PaintDurFactor;

    // Piece Welding Inspection Duration

```

```

        CurPiece.PieceWeldInspDur=PieceDur* 0.045 * PaintDurFactor;
    }
    // Piece Painting Duration
    CurPiece.PiecePaintDur=PieceDur* 0.1 * PaintRequired;

    //Division Weight
    CurPiece.DivWeight=DivWeight;

    if (PieceOrComponent==1)
    {
        PiecesOfTheDay.add(CurPiece);//Add the read piece to the list
    } else
    {
        ComponentsOfTheDay.add(CurPiece);//Add the read piece to the list
    }

    } //Quantity Loop
}

PieceDataBase.destroy();
}

public double ScheduledWork()
{
    // From Databases

    Database PieceDataBase = new Database( Owner, "PieceDataBase",
"D:\User\Amin\PhdCourse\03_Project\RA_DrLeeGroup\DrLee_Modeling\17_PilotModel_FatigueInFabS
hop\Pilot_FatigueInShop.mdb");
    String SelectStatement = "SELECT
Sum([Quantity]*[WeightofPiece]/1000*[Piece].[fab_mhrs]) AS TotalScheduledWork FROM Piece INNER
JOIN Division ON (Piece.Sub_ID = Division.sub_id) AND (Piece.div_id = Division.div_id) AND
(Piece.job_id = Division.job_id) AND (Piece.proj_id = Division.proj_id) Where
(((Division.required_date)>=#" + (StartDate.getMonth()+1) + "/" + StartDate.getDate()
+ "/" + (StartDate.getYear()+1900)+ "#)And ((Division.required_date)<=#" + (CurrentDate.getMonth()+1)
+ "/" + CurrentDate.getDate() + "/" + (CurrentDate.getYear()+1900)+ "#)AND ((Piece.fab_start_date)>=#" +
(StartDate.getMonth()+1) + "/" + StartDate.getDate() + "/" + (StartDate.getYear()+1900)+ "#) And
((Piece.fab_start_date)<=#" + (FinishDate.getMonth()+1) + "/" + FinishDate.getDate()
+ "/" + (FinishDate.getYear()+1900)+ "#)AND Piece.WeightofPiece>1)";
    String SScheduledWorkManHours= PieceDataBase.getValue(SelectStatement);
    double ScheduledWorkManHours= 0;

    if
(SScheduledWorkManHours!=null){ScheduledWorkManHours=Double.parseDouble(SScheduledWorkManHo
urs);}

    PieceDataBase.destroy();
    return ScheduledWorkManHours;
}

public void ReportCompletedPieces(List<MyPiece> PiecesFinished, String MdlType)
{
    // To Databases
    if (!PiecesFinished.isEmpty())
    {
        Database PieceDataBase = new Database( Owner, "PieceDataBase",
"D:\User\Amin\PhdCourse\03_Project\RA_DrLeeGroup\DrLee_Modeling\17_PilotModel_FatigueInFabS
hop\Pilot_FatigueInShop.mdb");
        MyPiece FinPiece;
        Calendar Cal= Calendar.getInstance();
        for (int i=0; i<PiecesFinished.size(); i++)
        {
            FinPiece=new MyPiece();
            FinPiece=PiecesFinished.get(i);
            String InsertStatement = "INSERT INTO PieceFinishReport
(ModelType, RunDate, proj_id, job_id, div_id, Sub_ID, piece_id,
fab_Finsh_date,PieceWeight,ManHourPerTon ) SELECT ""+ MdlType + "" AS Model, #" +
(Cal.getTime().getMonth()+1) + "/" + Cal.getTime().getDate() + "/" + (Cal.getTime().getYear()+1900)+ "# AS
RunDate, " + FinPiece.ProjID + " AS Proj, " + FinPiece.JobID + " AS Job, " + FinPiece.DivID + " AS Div, " +
FinPiece.SubID + " AS Sub, " + FinPiece.PieceID + " AS Piece, #" + (FinPiece.FabFinishDate.getMonth()+1)
+ "/" + FinPiece.FabFinishDate.getDate() + "/" + (FinPiece.FabFinishDate.getYear()+1900)+ "# AS Finish, " +
FinPiece.PieceWeight + " As Weight, " + FinPiece.PieceMhTon + " As ManHour";

```

```

        PieceDataBase.modify(InsertStatement);
    }
    PieceDataBase.destroy();
}

}

public void ReportResult(String MdlType, String RunTime, double ProductivityRate, String Result)
{
    // To Databases
    Database PieceDataBase = new Database( Owner, "PieceDataBase",
"D:\User\Amin\PhdCourse\03_Project\RA_DrLeeGroup\DrLee_Modeling\17_PilotModel_FatigueInFabS
hop\Pilot_FatigueInShop.mdb");
    String InsertStatement = "INSERT INTO ProductivityResult (ModelType,
RunTime, ProductivityRate, Result) SELECT '"+ MdlType + "' AS Model, '" + RunTime+ "' As RunTime, '" +
ProductivityRate + "' As ProductivityRate, '" + Result + "' AS Result";
    PieceDataBase.modify(InsertStatement);
    PieceDataBase.destroy();
}

@Override
public String toString() {
    return super.toString();
}
}

```

C.2. Data-tables

Two input data-tables in the model are Piece and Division data-tables. These two tables have similar structure to the Piece and Division tables explained in Appendix B and we prevent the duplication here. In addition to the input data-tables, two output data-tables were also used for collecting the model information during the simulation runs; PieceFinishReport and ProductivityResult (Some other outputs of the model are arranged to be read from AnyLogic console as well).

C.2.1. PieceFinishReport Table

Table C.1 presents PieceFinishReport data-table which collects the simulated fabrication finish time for pieces. Some sample data collected from the model runs are shown in the table as well.

Table C.1. PieceFinishReport table structure with sample data

ModelType	RunDate	div_id	piece_id	fab_Finsh_date
~				
FatigueBase	23/06/2010	11627	218662	10/02/2009
FatigueBase	23/06/2010	11627	218662	10/02/2009
FatigueBase	23/06/2010	11627	218663	10/02/2009
FatigueBase	23/06/2010	11627	218769	10/02/2009
FatigueBase	23/06/2010	11783	219603	27/01/2009
FatigueBase	23/06/2010	11783	219604	27/01/2009
FatigueBase	23/06/2010	11783	219604	27/01/2009
FatigueBase	23/06/2010	11783	219604	27/01/2009
FatigueBase	23/06/2010	11783	219604	27/01/2009
FatigueBase	23/06/2010	11783	219604	27/01/2009
FatigueBase	23/06/2010	11783	219604	27/01/2009
FatigueBase	23/06/2010	11783	219604	27/01/2009
FatigueBase	23/06/2010	11783	219607	27/01/2009
FatigueBase	23/06/2010	11783	219607	27/01/2009
FatigueBase	23/06/2010	11783	219607	27/01/2009
FatigueBase	23/06/2010	11783	219607	29/01/2009
FatigueBase	23/06/2010	11783	219608	27/01/2009
FatigueBase	23/06/2010	11627	230058	10/02/2009
FatigueBase	23/06/2010	11627	230059	10/02/2009
FatigueBase	23/06/2010	11627	230060	10/02/2009
FatigueBase	23/06/2010	11627	230061	10/02/2009
FatigueBase	23/06/2010	11627	230062	10/02/2009
FatigueBase	23/06/2010	11627	230063	10/02/2009
FatigueBase	23/06/2010	11627	230064	10/02/2009

Brief explanations on the meaning of each data-field are presented in the following:

ModelType: The name and the type of working hour alternative captured in the model.

RunDate: The date in which the model was run.

div_id: The id of the division which piece is belonged to.

piece_id: The piece id.

fab_Finsh_date: The simulated finish date of the piece

C.2.2. ProductivityResult Table

Table C.2 presents ProductivityResult data-table which collects the final results achieved from different models runs.

Table C.2. ProductivityResult table structure with sample data

ModelType	Result
1h1	RunTime:Wed Jun 30 08:03:27 MDT 2010 Fabrication Finished on:Tue Apr 07 07:39:07 MDT 2009Tue Apr 07 07:39:07 MDT 2009Current Time is: 58215.56312391517Average Productivity is: 0.8983742190452829
1h2	RunTime:Wed Jun 30 08:36:53 MDT 2010 Fabrication Finished on:Wed Apr 08 08:11:46 MDT 2009Wed Apr 08 08:11:46 MDT 2009Current Time is: 58351.40962998669Average Productivity is: 0.8984339262707333
1h3	RunTime:Wed Jun 30 09:14:09 MDT 2010 Fabrication Finished on:Tue Apr 07 08:49:35 MDT 2009Tue Apr 07 08:49:35 MDT 2009Current Time is: 58310.53125846892Average Productivity is: 0.8944369838542933
1hM1	RunTime:Wed Jun 30 10:19:55 MDT 2010 Fabrication Finished on:Tue Apr 07 09:55:38 MDT 2009Tue Apr 07 09:55:38 MDT 2009Current Time is: 58058.80074378561Average Productivity is: 0.9148512185669766
1hM2	RunTime:Wed Jun 30 11:01:18 MDT 2010 Fabrication Finished on:Tue Apr 07 10:37:02 MDT 2009Tue Apr 07 10:37:02 MDT 2009Current Time is: 58226.14056019378Average Productivity is: 0.9136835914200641
1hM3	RunTime:Wed Jun 30 14:40:48 MDT 2010 Fabrication Finished on:Wed Apr 08 14:15:11 MDT 2009Wed Apr 08 14:15:11 MDT 2009Current Time is: 58480.50315175021Average Productivity is: 0.8938040808996763
1h1	RunTime:Wed Jun 30 08:03:27 MDT 2010 Fabrication Finished on:Tue Apr 07 07:39:07 MDT 2009Tue Apr 07 07:39:07 MDT 2009Current Time is: 58215.56312391517Average Productivity is: 0.8983742190452829

Brief explanations on the meaning of each data-field are presented in the following:

ModelType: The name and the type of working hour alternative captured in the model.

Result: Provides aggregative information from the model run including run time, total fabrication finish time, total logical time spent and average productivity achieved in the model.

Appendix D

Programming Details of the Simulation Model Used for Construction Workers Evolution

Anylogic 6.4 was used for implementing all SD and hybrid SD-DES models developed for testing and analyzing Simulation model of construction workers evolution dynamics in Chapter 4. Data-tables in MS Access database were used for providing the links to the collaborative company's database for the experimental case. The developed AnyLogic model consists of four main parts; SD models, DES model, supporting classes and simulation setting (Figure D.1).

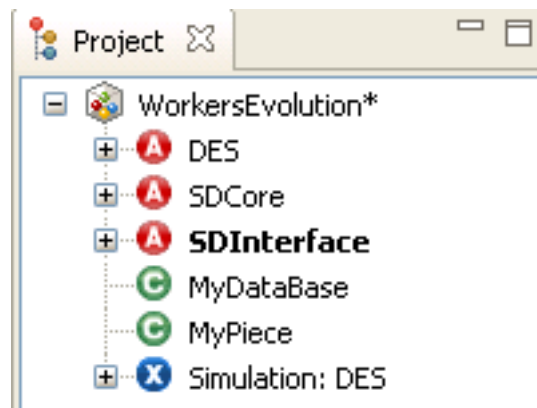


Figure D.1. Structure of the AnyLogic

Since the experiments run in Chapter 3 and Chapter 4 have been done on the same structural steel fabrication shop, the physical details of the fabrication shop captured in the DES model and the supporting classes developed for facilitating the link to the company's database and customizing the entity attributes in the

model are quite similar to the ones used in working hours dynamic model, presented in Appendix C. Therefore I avoid the duplication here. As well, detailed specifications of the simulation setting part is referred to the AnyLogic user manual (accessible at: www.xjtek.com). In addition to that the input data in the model was received from the same database and data-tables and the output data was stored in very similar data-tables to what presented in Appendixes B and C. Therefore in this appendix just two developed SD sub-models are explained.

D.1. SDCore Model

The SDCore sub-model represents the core dynamic model of the workers evolution presented in Figure 4-8 in Chapter 4 including the workforce skill evolution and promotion dynamics with workers in six different levels of experience (Figure D.2).

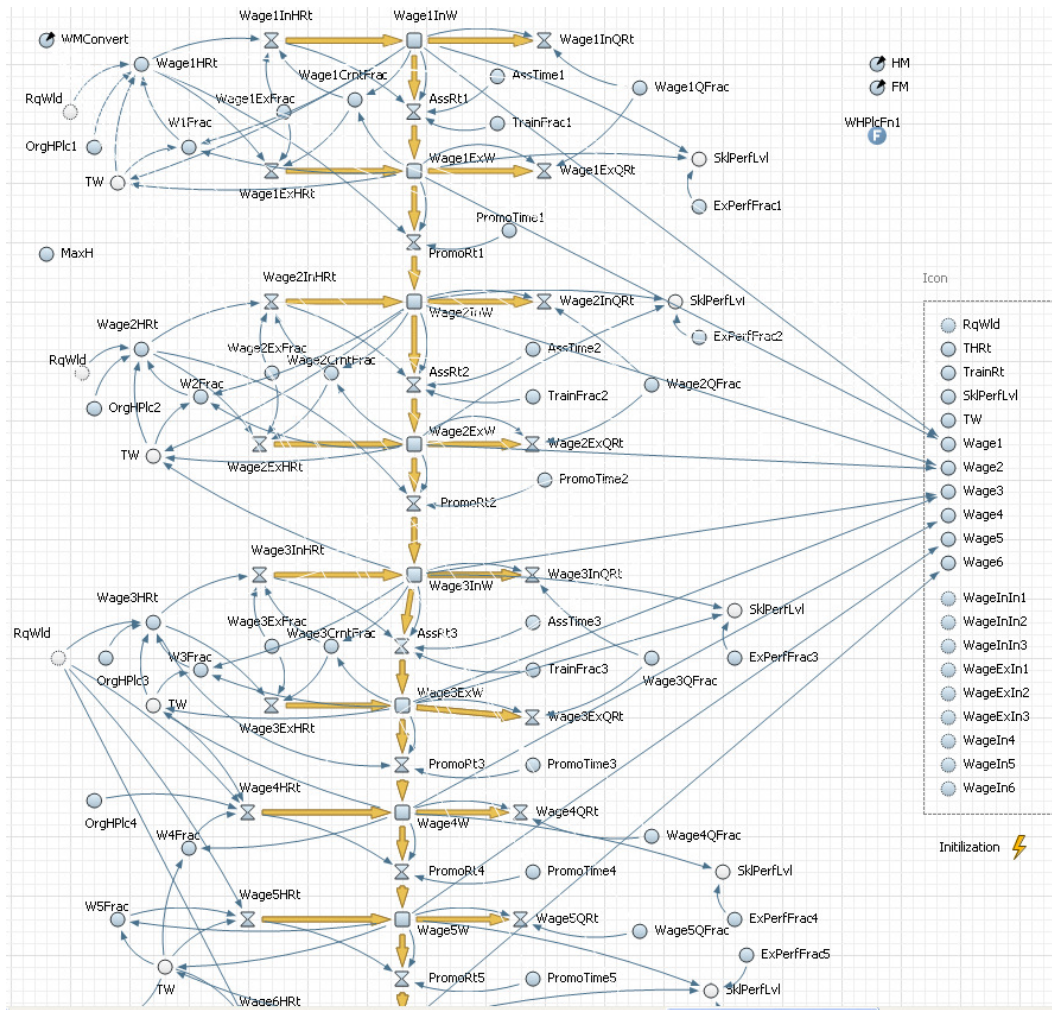


Figure D.2. SDCore model of workforce evolution dynamics

The related equations to the model are mainly captured in the variable relations (including stock variables represented by rectangle \square , flow variables represented by valve shape Σ , auxiliary variables show by circles \bigcirc and model parameters shown with a circle with a small black triangle on the top-right side of the circle \bigcirc) and the link between two variables is shown by arrows. However, more complex equations are placed in the function elements (F). For example the workforce hiring policy function has been added to WHPlcFn function as in below:

```

//Applying Company's hiring/firing policies
if ((RqWld/40)/TW > HM)
{
    if (time()<1)//First week has no limitation for hiring because of befor project preparations
    {
        return ((RqWld/40)-TW)*(OrgHPlc) ;
    }else
    {
        return min(((RqWld/40)-TW),MaxH)*(OrgHPlc) ;
    }
} else if ((RqWld/40)/TW < FM & TW>3)
{
return ((RqWld/40)-TW)*(WFrac);
}
return 0;

```

The interface variables elements, surrounded by a dashed rectangle at the right side of the model, send updates to/ receive update from model variables in the SDInterface sub-model (Section D.2.). The main input interface variables are the initial number of workers in every wage and experience level (WageInIn1, WageInIn2, WageInIn3, WageExIn1, WageExIn2, WageExIn3, WageIn4, WageIn5, WageIn6) and required workload (RqWld), and the main output variables are total number of workers (TW), number of workers in different level of wages (Wage1, Wage2, Wage3, Wage4, Wage5, Wage6), total hiring rate (THRr), total training rate (TrainRt) and the skill performance level (SkIPerfLvl).

Finally, since the SDCore model has been implemented in a generic manner to be able to be used for the workers evolution dynamics in different work stations with different number of workers and level of skills, there is an initialization event (presented by a lightning sign ⚡ in the model standing below the interface variables) has been added to the model to initialize the number of workers in different levels of skill by reading their related values from interface variables.

D.2. SDInterface Model

The SDInterface sub-model integrates the core SD model (Section D.1) with overtime policy and dynamic cost data collecting mechanism; it has the set of interface variables which communicate with the DES part of the model (Figure D.3).

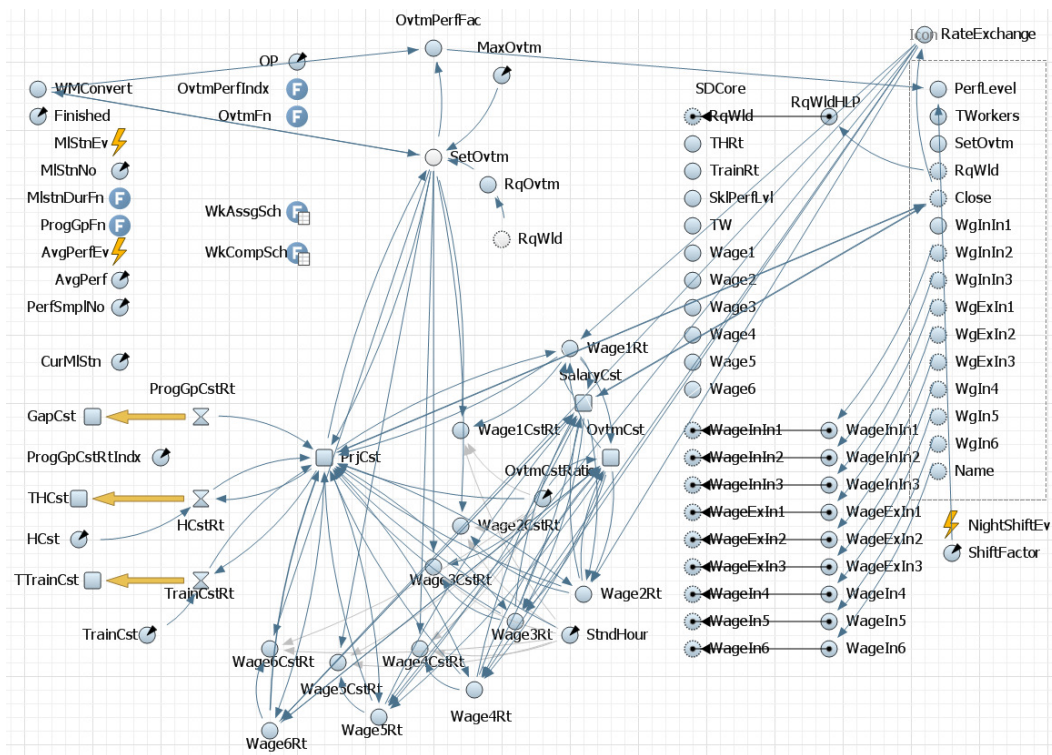



Figure D.3. Screen shot of SDInterface SD model

The dynamic cost data collecting mechanism is on the bottom-left and the overtime dynamics is on the top-left of the model. The SDCore model is placed next to these two modeling parts and is treated as a modeling component with contact points limited to its interface variables. The relations between different model variables are created in the model the same as what has been presented in the dynamic model equations in Chapter 4; the arrows between different model

variables represent these relations. However, AnyLogic does not show the relations between updating interface variables from a model component (i.e., SDCore model in this case) and the model variables in an arrow format. For example although total number of workers (TW interface variable) is updated in SDCore model component and is used in different parts of the model, these relations are not shown by arrows.

Table-function elements ()WkAssgSch and WkCompSch in the model contain the work assignment schedule and work completion schedule. These table functions can provide the estimated workloads assignments and the milestones set for the project over the time. I used these table-functions for analyzing the capabilities of the developed SD models with no input from/ output to the DES model of the project (Section 4.2.3 Chapter 4). So, basically in cases that a project does not contain high operational complexity and the estimated workload and milestones have an acceptable level of accuracy, the project behaviour can be modeled by using these function-tables and no DES model is required.

Interface variables in this part of the model are placed at the right side of the model to communicate with the DES part of the model. The main input interface variables from the DES part are the initial number of workers in every wage and experience level (WgInIn1, WgInIn2, WgInIn3, WgExIn1, WgExIn2, WgExIn3, WgIn4, WgIn5, WgIn6), required workload (RqWld), working hour alternative name (Name) and the working condition of the shop (Close), and the main output

variables to DES part are total number of workers (TWorkers), set overtime for the week (SetOvtm), and performance level (PerfLevel).