

**Dynamic Planning and Control Methodology:
Understanding and Managing Iterative Error and Change Cycles in
Large-Scale Concurrent Design and Construction Projects**

by

SangHyun Lee

B.E., Architectural Engineering, Dong-A University, 2000
M.S., Civil and Environmental Engineering, MIT, 2003

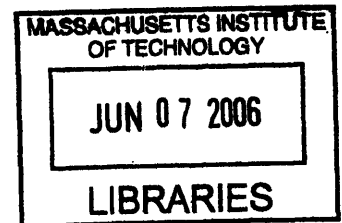
Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in the Field of Construction Management and Information Technology

at the

Massachusetts Institute of Technology

June 2006

© 2006 Massachusetts Institute of Technology
All rights reserved



Signature of Author

A handwritten signature in black ink, appearing to be "SangHyun Lee", written over a horizontal dotted line.

Department of Civil and Environmental Engineering
May 12, 2006

Certified by

A handwritten signature in black ink, appearing to be "Fred Moavenzadeh", written over a horizontal dotted line.

Fred Moavenzadeh
Professor of Civil and Environmental Engineering
Chairman, Doctoral Thesis Committee

Certified by

A handwritten signature in black ink, appearing to be "Feniosky Peña-Mora", written over a horizontal dotted line.

Feniosky Peña-Mora
Associate Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by

A handwritten signature in black ink, appearing to be "Andrew Whittle", written over a horizontal dotted line.

Andrew Whittle
Chairman, Departmental Committee for Graduate Students

ARCHIVES

V.1

**Dynamic Planning and Control Methodology:
Understanding and Managing Iterative Error and Change Cycles
in Large-Scale Concurrent Design and Construction Projects**

by

SangHyun Lee

Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in the Field of Construction Management and Information Technology

ABSTRACT

Construction projects are uncertain and complex in nature. One of the major driving forces that may account for these characteristics is iterative cycles caused by errors and changes. Errors and changes worsen project performance and consequently, cause schedule and cost overruns to be prevalent. In particular, these iterative cycles are more detrimental when large-scale concurrent design and construction is applied.

In an effort to address these issues, this research proposes Dynamic Planning and control Methodology (DPM) as a robust design and construction planning methodology for large-scale concurrent design and construction. The proposed DPM is composed of: 1) an error and change management framework that enables understanding of the construction processes associated with errors and changes and how they affect construction performance; 2) a proactive buffering strategy for reducing sensitivity to iterative error and changes cycles; 3) a System Dynamics-based construction project model which provides policy guidelines for the planning and control of projects; and 4) a web-based error and change management system, which supports coordination of errors and changes among contractors and design professionals without hardware and software compatibility issues.

Applying all research components into a couple of real world case projects, this research concludes that a concurrently developed project can benefit by: 1) adding realism to planning taking into account iterative error and change cycles; 2) implementing a proactive mechanism to look and act ahead against uncertainties; 3) making appropriate policies with the help of the system dynamics-based simulation model; and 4) facilitating coordination from the IT-supported management system; even if the time frame of a project is shortened. Also, future research opportunities are discussed extending the findings from this research.

Chairman, Doctoral Thesis Committee: Fred Moavenzadeh
Title: Professor of Civil and Environmental Engineering

Thesis Supervisor: Feniosky Peña-Mora
Title: Associate Professor of Civil and Environmental Engineering

ACKNOWLEDGEMENTS

This thesis is dedicated to my family. Without their care, my studies at MIT would not have been a reality. To my father and mother, their unconditional love has been a source of inspiration. If this thesis can make them smile, my efforts and patience will be fully paid off. In particular, I sincerely wish my completion of this thesis will keep my lovely nephew and niece 'dreaming'.

I would like to thank my mentor, Professor Feniosky Peña-Mora, for his inexhaustible support. Actually, it is not just 'support', rather, 'shadow'. He has been behind me for my research and my family. Working with him is the most valuable and exciting experience in my MIT life. I am very proud that I am his student and friend and hope I can do the same kinds of things for him throughout our lives.

I would like to express my gratitude to Professor Fred Moavenzadeh who has given me sincere guidance on this research. His sharp comments have always inspired me to work hard. I would also like to acknowledge Professor Jerome Connor's and Professor Massood Samii's contributions to this research. They have been always kind to me and further, encouraged me to get through it. Also, I would like to thank Professor MoonSeo Park at Seoul National University and ManHyoung Lee at ChungBuk National University for their endless care. Spending the time with them and their family is one of the most pleasant memories during this study.

In addition, I would like to acknowledge the contribution to this thesis by Professor Chimay Anumba at Loughborough University, Professor Indra Gunawan and Mr. Tai Soon Chi at the Malaysia University of Science and Technology, Philip Helmes and Margaret Fulenwider, Vice President and Associate Principal, respectively, at CRA International Inc., Professor Lucio Soibelman at Carnegie Mellon University, Professor Khaled El-Rayes at the University of Illinois at Urbana-Champaign, Professor NamSeek Hong at Dong-A University, JongMin Shim, Graduate Student at MIT, SangWon Han, JeongWook Son, and DoHeon Jeon, Graduate Students at UIUC, and the financial support for the thesis research received from CRA International Inc. (previously, InteCap Inc.), Kajima Corporation, the National Science Foundation CAREER and PECASE Award CMS-9875557, the National Science Foundation Award CMS-0324501, and MIT Berger Fellowship.

Finally, I wish to thank my wife, JungSook. Her love and support has been essential. I have never been afraid of any challenge and hardship because she has been and will continue to be with me. Moreover, her passion and endeavor for her dream has driven me to never give up. This thesis is all about her smiles and tears.

The completion of this thesis means not 'mastering' knowledge but 'recognizing' ignorance. It just opened the door to the truth, enabling to see the road toward it. At this moment, I am the happiest man in the world, realizing the joy of a lifetime of learning, standing with my eternal learning companion, JungSook.

TABLE of CONTENTS

CHAPTER 1 PREFACE	10
CHAPTER 2 RESEARCH APPROACH	15
2.1 ANALYSIS	17
2.2 DEVELOPMENT	17
2.3 VALIDATION	18
CHAPTER 3 ERROR AND CHANGE MANAGEMENT FRAMEWORK	19
3.1 LITERATURE REVIEW	20
3.2 FEEDBACK PROCESSES CAUSED BY ERRORS AND CHANGES	21
3.3 ADDITIONAL WORK SCOPE GENERATED BY ERRORS AND CHANGES	22
3.4 ERRPR AND CHANGE MANAGEMENT FRAMEWORK	23
3.5 PLANNED, PERCEIVED, AND REAL PERFORMANCE	37
CHAPTER 4 RELIABILITY AND STABILITY SCHEDULE BUFFERING	44
4.1 JUST-IN-TIME VS. JUST-IN-CASE	45
4.2 LITERATURE REVIEW	47
4.3 RELIABILITY AND STABILITY BUFEIRNG APPROACH	49
4.4 IMPLEMENTATION OF RELIABILITY AND STABILITY BUFEIRNG	61
CHAPTER 5 SYSTEM DYANMCIS-BASED SIMULATION MODEL.....	63
5.1 LITERATURE REVIEW	65
5.2 WHY SYSTEM DYNAMICS?	66
5.3 DYNAMIC DESIGN AND CONSTRUCTION PROJECT MODEL	70
5.4 VALIDATION	101
CHAPTER 6 WEB-BASED ERROR AND CHANGE MANAGEMENT SYSTEM.....	104
6.1 LITERATURE REVIEW	104
6.2 SYSTEM OVERVIEW	106
6.3 COLLABORATIVE ENVIRONMENT	109
6.4 IMPLEMENTATION OF THE DPM SYSTEM	112
6.5 FUNTIONALITIES OF THE DPM SYSTEM.....	116
CHAPTER 7 CASE STUDY.....	126
7.1 INPUT ELICITATION PROCESS.....	126
7.2 LABORATORY BUIDLING PROJECT IN MALAYSIA	129
7.3 HIGHWAY BRIDGE INFRASTRUCTURE PROJECT IN MASSACHUSETTS	137
CHAPTER 8 CONCLUSIONS.....	152
8.1 POTENTIAL IMPACT.....	157
CHAPTER 9 FUTURE RESEARCH	159
9.1 COMPREHENSIVE SIMULATION FRAMEWORK	160
9.2 APPLICATION OF LATENCY AND PROACTIVE BUFFERING.....	163
9.3 VISUALIZATION BASED ON THE USER'S CAPACITY AND NEED	165

REFERENCES	174
APPENDIX I SYSTEM DYANMICS MODEL STRUCTURES	181
APPENDIX II SYSTEM DYNAMICS MODEL EQUATIONS	188
APPENDIX III JAVA CODES FOR THE WEB-BASED DPM SYSTEM	211
APPENDIX IV BIOGRAPHICAL NOTE	412

LIST of FIGURES

FIGURE 1. DERIVATIVE ACTIVITY	12
FIGURE 2. RESEARCH APPROACH.....	16
FIGURE 3. SIMULTANEITY CAUSED BY ERRORS AND CHANGES.....	22
FIGURE 4. INTERNAL ERROR AND CHANGE MANAGEMENT FRAMEWORK.....	25
FIGURE 5. EXAMPLE OF LATENCY.....	27
FIGURE 6. EXTERNAL ERROR & CHANGE MANAGEMENT FRAMEWORK	36
FIGURE 7. ERROR AND CHANGE MANAGEMENT PROPAGATION EFFECT ON THE WHOLE NETWORK	37
FIGURE 8. SCHEMA OF DYNAMIC DESIGN AND CONSTRUCTION PROJECT MODEL	39
[ADAPTED FROM FORD AND STERMAN, 1998].....	39
FIGURE 9. PLANNED, PERCEIVED, AND REAL PERFORMANCE AT PILE INSTALLATION EXAMPLE	42
FIGURE 10. APPLICATION OF THE D²CPM SCHEMA.....	43
FIGURE 11. INEFFICIENCY OF CONTINGENCY BUFFER.....	48
FIGURE 12. RELIABILITY AND STABILITY BUFFER SPLIT	52
FIGURE 13. YERKES-DODSON LAW [ADOPTED FROM YERKES-DODSON, 1908 AS REFERRED ON STERMAN, 2000]	53
FIGURE 14. EXAMPLE OF PRODUCTION TYPE [ADAPTED FROM EPPINGER, 1997 AS REFERRED BY PEÑA-MORA AND LI, 2001] AND THEIR IMPACT ON BUFFER SIZE.....	56
FIGURE 15. SCHEDULE UPDATE BY DYNAMIC BUFFERING.....	60
FIGURE 16. RELIABILITY AND STABILITY BUFFERING IMPLEMENTATION STEPS IN SCHEDULE [ADAPTED FROM GOLDRATT, 1997; PARK AND PEÑA-MORA, 2004]	62
FIGURE 17. INCONSISTENT TIME STEP SIZE IN DES [ADAPTED FROM HAN ET AL., 2006].	70
FIGURE 18. FEEDBACK PROCESSES ON ERROR AND CHANGE MANAGEMENT PROCESS.....	74
FIGURE 19. MODEL BOUNDARIES	76
FIGURE 20. BASIC WORK EXECUTION STRUCTURE IN SD	77
FIGURE 21. MODELING WORK CONSTRAINTS	79
FIGURE 22. ACTIVITIES FOR SIMULATION EXPERIMENT.....	79
FIGURE 23. CPM CASE – BEHAVIOR OF WORKToDo AND WORKCOMPLETED STOCK.....	80
FIGURE 24. DYNAMIC PRODUCTION RATE.....	81
FIGURE 25. CASE 1 – BEHAVIOR OF WORKToDo AND WORKCOMPLETED STOCK.....	81
FIGURE 26. CASE 1 – EXCESS OF QM CAPACITY	82
FIGURE 27. WORK EXECUTION WITH ERROR MANAGEMENT [ADAPTED FROM PUGH-ROBERTS ASSOCIATES, 1980; FORD AND STERMAN, 1998; PARK AND PEÑA-MORA, 2003].....	84
FIGURE 28. CO-FLOW STRUCTURES FOR ERRORS.....	85
FIGURE 29. WORK EXECUTION WITH CHANGE MANAGEMENT	86
FIGURE 30. ILLUSTRATION OF TWO DIFFERENT EXCAVATION METHOD.....	88
FIGURE 31. CASE 2 – ADDITIONAL WORK SCOPE CAUSED BY ERRORS AND CHANGES	89
FIGURE 32. WORK EXECUTION WITH THE RFI PROCESS [ADAPTED FROM PUGH-ROBERTS ASSOCIATES, 1980; FORD AND STERMAN, 1998; PARK AND PEÑA-MORA, 2003].....	93

FIGURE 33. EXAMPLE - LATE DISCOVERY OF HIDDEN ERROR AND CORRECTION REQUEST FROM THE SUCCESSOR ACTIVITY	94
FIGURE 34. CASE 3 – PROPAGATION IMPACT	95
FIGURE 35. CASE 4 – IMPACT OF LATENCY	96
FIGURE 36. CASE 4 – LATE DISCOVERY OF ADDITIONAL WORK SCOPE DUE TO LATENCY ...	97
FIGURE 37. SUPPORTING MODEL STRUCTURES	98
FIGURE 38. EXAMPLE OF SUPPORTING MODEL STRUCTURE	101
FIGURE 39. DPM SYSTEM OVERVIEW	107
FIGURE 40. UML ACTIVITY DIAGRAM WITH SWIMLANES –	109
 COMPARING SIMULATION RESULT WITH ORIGINAL PLAN	109
FIGURE 41. DPM SYSTEM ARCHITECTURE FOR COLLABORATIVE ENVIRONMENT	111
FIGURE 42. SAMPLE DATA MODEL IN DPM.....	113
FIGURE 43. UML DEPLOYMENT DIAGRAM	114
FIGURE 44. INSIDE APPLICATION LOGIC TIER	115
FIGURE 45. DPM MAIN WINDOW.....	116
FIGURE 46. OVERVIEW OF INPUT AND OUTPUT OF DPM	117
FIGURE 47. BASIC DATA - DEPENDENCY STRUCTURE MATRIX WITH SMART CELLS	119
FIGURE 48. SIMULATION DATA: PROJECT POLICY-RELATED INFORMATION.....	120
FIGURE 49. PROJECT AND ACTIVITY LEVEL OUTPUT	122
FIGURE 50. TRACKING SCHEDULE CHANGES	123
FIGURE 51. GAME (INTERACTIVE SIMULATION)	124
FIGURE 52. INFORMATION EXCHANGE WITH PRIMAVERA DATABASE	125
FIGURE 53. INPUT ELICITATION PROCESS	128
FIGURE 54. EXAMPLE OF THE PRE-DETERMINED QUESTIONNAIRE FOR THE INPUT ELICITATION PROCESS	128
FIGURE 55. PROJECT LOCATION AND SITE PHOTO	130
FIGURE 56. PLANNED, ACTUAL, AND SIMULATION PERCENTAGE OF WORK COMPLETE.....	133
FIGURE 57. TOTAL ADDITIONAL WORK SCOPE CAUSED BY HIDDEN ERROR AND LATENCY CHANGES.....	135
FIGURE 58. COORDINATION WORK TO RESOLVE ERRORS AND CHANGES	136
FIGURE 59. PROJECT LOCATION AND SITE PHOTO	137
FIGURE 60. SIMULATION RESULTS OF THE CASE PROJECT BY WEB-BASED DPM	141
FIGURE 61. IMPACTS OF DESIGN ERRORS AND CHANGES	142
FIGURE 62. INCREASED WORK SCOPE IN SUCCEEDING ACTIVITIES.....	143
FIGURE 63. SIMULATION SETTING AND RESULTS	146
FIGURE 64. REDUCTION OF HIDDEN ERRORS AND LATENT CHANGES DURING THE BUFFER PERIOD	147
FIGURE 65. IMPROVEMENT OF RELIABILITY AND STABILITY DURING THE BUFFER PERIOD .	148
FIGURE 66. EARLY ADOPTION OF ERRORS AND CHANGES.....	149
FIGURE 67. EFFECTIVENESS OF RELIABILITY AND STABILITY BUFFER.....	151
FIGURE 68. COMPREHENSIVE SIMULATION FRAMEWORK.....	160
FIGURE 69. DPM’S EXTENSION TO STOCHASTIC SIMULATION.....	162
FIGURE 70. EXAMPLES OF STOCHASTIC SIMULATION RESULTS	163
FIGURE 71. VISUAL MOTION METAPHOR [EXTENDED FROM SONG ET AL., 2005]	168
FIGURE 72. AUGMENTED REALITY-BASED PROGRESS MONITORING	169

FIGURE 73. VISUALIZED PROGRESS MONITORING HISTORY 170
FIGURE 74. USE OF COLOR AND COLOR GRADIENT 172

LIST of TABLES

TABLE 1. APPLIED TEST TECHNIQUES	102
TABLE 2. SIMULATION INPUT FOR SIRIM PROJECT	132
TABLE 3. SIMULATION INPUT FOR TREBLE COVE PROJECT	139
TABLE 4. DIVERSE PROJECT DURATIONS OF THE CASE PROJECT	140
TABLE 5. EFFECTS OF DIVERSE POLICY OPTIONS	145

CHAPTER 1

PREFACE

Errors and changes are very common and are one of the major driving factors for uncertainty in construction. Errors, changes, and consequent conflicts lead to significant schedule and cost overrun if not properly managed. For example, during the execution of a project, the time taken to rectify errors is estimated to be 11% of the total working hours allocated for a project and the cost to correct these errors is approximately 6% of production costs [Josephson and Hammarlund, 1999]. The cost to implement changes is estimated to be 5.1% – 7.6% of the total project cost [Cox et al, 1999]. This translates into \$50 billion being spent annually on new change orders by the construction industry in the U.S. alone [Ibbs et al., 1998].

This is partly because of the nature of design and construction processes, which inherently involves complex and dynamic interactions among diverse variables, such as participant experience, physical attributes, resource procurement, strategies, time and cost constraints, and management techniques. Thus, an unanticipated error or change could propagate to other activities; for example, through their physical and procedural relationships. As a result, the monitored performance may not follow the planned performance and is often ambiguous in identifying the root causes for this gap. Consequent symptoms in construction projects include chronic schedule and cost overrun, despite advancements in construction equipment and

management techniques [Park and Peña-Mora, 2003]. The static approach of traditional construction planning and control tools is not very efficient in dealing with these problems [Lyneis et al, 2001], and situations become worse when concurrent design and construction is applied, a process that has been widely adopted in the A/E/C industry due to its promise for shortening project duration. This is because in concurrent design and construction: 1) succeeding activities often have to start work without complete information from preceding activities, which in turn, increases the number of assumptions to be made [Tighe, 1991] and the frequency and the number of information transfers [Ford and Sterman, 2003]; 2) overloaded workers sometimes fail to respond to communications, thereby compounding the information supply problem and compromising others' performance [Chachere et al, 2004]; and 3) pressures of a shortened project delivery time can accelerate the decision making process, which could introduce additional errors and changes [Lee et al., 2005]. Thus, combined with high procedural and physical constraints, errors and changes in concurrent design and construction may lead to a significant chain of wrong decisions in other related activities, which consequently, resulted in performance degradation and schedule and cost overruns.

However, traditional network-based tools, which have been widely used in the A/E/C industry, lack the capability to manage such dynamic and complex feedback caused by iterative error and change cycles, which are very common in construction industry [Lyneis et al, 2001]. Through some methods, such as Graphical Evaluation and Review Technique (GERT, 1966), feedback can be managed, but their usage is limited to a static work scope (e.g., rework iterations). As an example, suppose overtime is applied to rework unanticipated errors and changes. GERT can capture these rework iterations through loop relationships with a given

probability. However, when the completed predecessor activity's errors and changes are discovered at the successor activity, as illustrated in Figure 1, workers and equipment from completed activities may have to be called back to the site, in particular if the only way to make adjustments is to go back and re-execute the work on the predecessor activity. GERT does not provide an effective way to deal with those types of dynamics like supplemental activities or derivative activities, which are common in design and construction projects. In addition, feedback in construction can have intangible and dynamic effects on the construction system as well (e.g., varying workers' morale during construction). In the previous example, applying overtime could deteriorate workers' morale and consequently, decrease workers' productivity. As a result, the applied overtime can have a negative effect on the schedule performance. Traditional network-based tools may not be efficient in capturing this dynamic feedback.

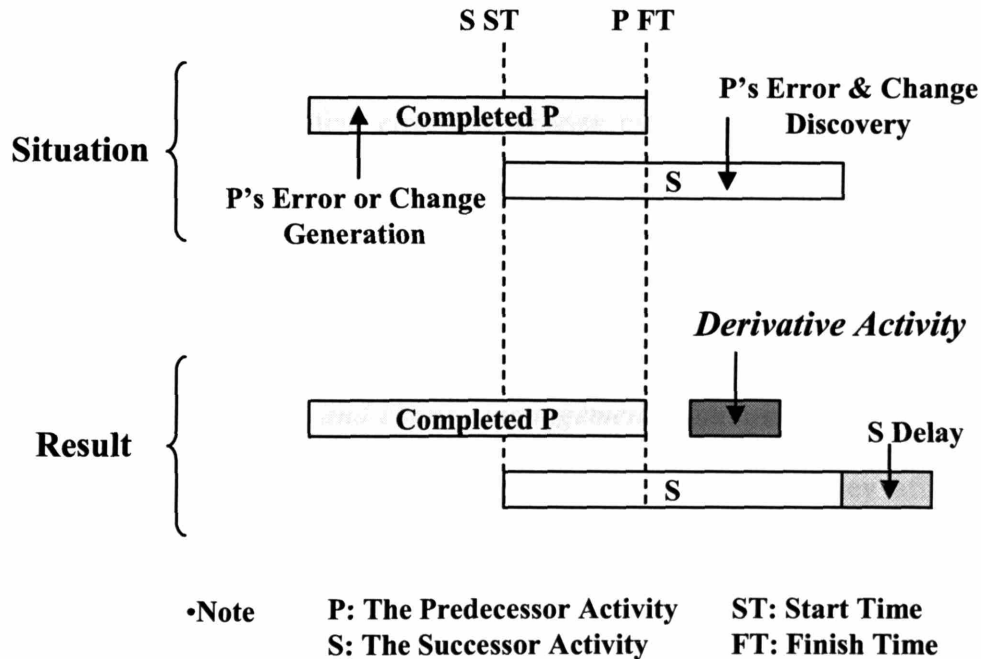


Figure 1. Derivative Activity

One notable research effort to deal with these problems is the *Dynamic Planning and control Methodology (DPM)*, which has been developed at the Intelligent Engineering Systems Laboratory (IESL), MIT [Peña-Mora and Li, 2001; Park and Peña-Mora, 2003]. DPM aims to provide policy guidelines for unexpected events by supplementing network-based tools with mechanisms to represent the dynamics of a project. However, DPM, in its original version, focused only on the quality aspect of construction performance. The original DPM did not contain mechanisms to explicitly address uncertainties resulting from changes, which is another important source of iterative cycles during the construction process. In addition, the original DPM may lack the capability of managing the impact of errors and changes, particularly in concurrent design and construction. In such an aggressive development environment, the impacts of errors and changes could be more iterative and complex so that they often substantially increase the project duration and total cost [Ibbs, 1997].

To effectively address iterative error and change cycles and their impacts in concurrent design and construction, an extension of the original DPM is presented in this research. The enhanced DPM identifies construction changes as well as quality problems in a timely manner so that possible conflicts resulting from iterative cycles can be minimized. Specifically, DPM constitutes: 1) an *analytic error and change management framework* to understand how errors and changes are associated with the construction process and how they affect construction performance; 2) a *proactive schedule buffering approach* to absorb the detrimental impacts of errors and changes; 3) a *System Dynamics-based construction project model* to analyze the impacts of errors and changes on construction performance; and 4) a *web-based error and*

change management system to support the coordination of errors and changes among geographically distributed parties involved in the construction project.

The objective of this research is to develop a DPM that helps *prepare a robust design and construction plan* against uncertainties and complexities caused by errors and changes and *provides policy guidelines for the planning and control of large-scale concurrent design and construction projects*. Thus, the proposed DPM is expected to benefit the entire life cycle of design and construction projects by reducing costs, avoiding delays, increasing quality, eliminating counterproductive disputes, and improving project management by reducing the impact of errors and changes.

This dissertation continues with a brief introduction of research methodology adopted in this research (Chapter 2). Then, four major components of this research: an analytic framework, a proactive buffering, a System Dynamics-based simulation model, and a web-based system, are discussed in subsequent chapters (Chapter 3-6). In particular, each chapter is composed of a respective problem statement, literature review, and this research's accomplishment. In the subsequent chapters, a couple of real world case projects, a highway bridge infrastructure project in Massachusetts and a laboratory building project in Malaysia, are discussed applying all research components (Chapter 7), and conclusions are drawn (Chapter 8). Lastly, future research opportunities will be discussed based on this research (Chapter 9).

CHAPTER 2

RESEARCH APPROACH

To accomplish the objectives stated in Chapter 1, this research is conducted based on three major steps: *analysis, development, and validation*, as seen in Figure 2. The analysis step sets research objectives and determines the focus of this research through diverse research methods. Based on this analysis, four major research components were developed (development). Then, these are validated through a couple of real-world case projects. One thing to note is that these steps are not sequential but spiral. For example, more analysis could be done during a development and validation phase if necessary. Because research products (e.g., model and system) evolve as research progresses, we continue to develop a better understanding of problems and challenges.

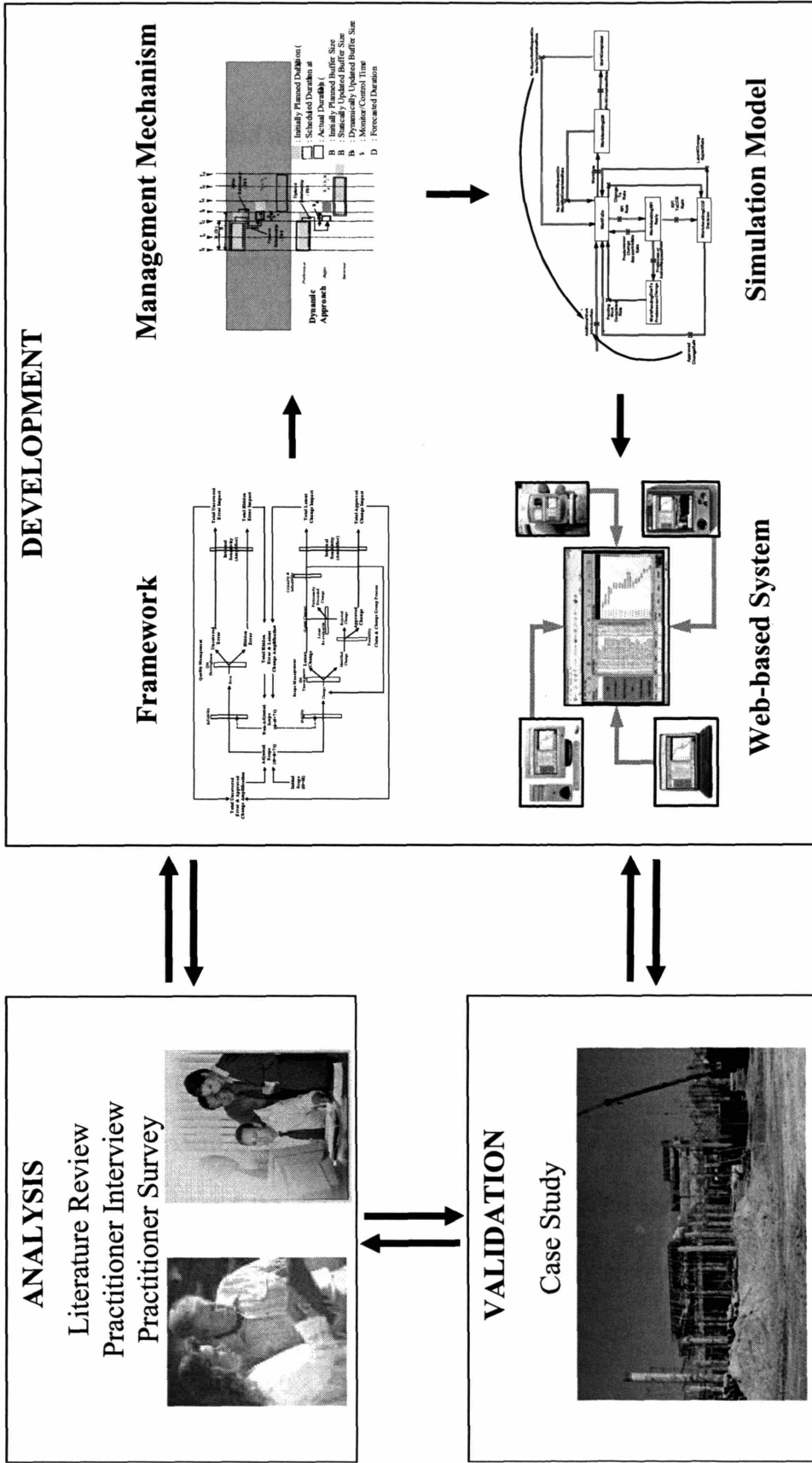


Figure 2. Research Approach

2.1 ANALYSIS

To identify what to solve and how to solve in detail, diverse research methods were conducted in this analysis phase, such as a literature review, practitioner interview, and practitioner survey. For example, an extensive literature review is conducted to study related works and their major accomplishments. In addition, diverse interviews and surveys are directly conducted with industrial partners such as Charles River Associates and Barletta Co. in the US and indirectly with Topjaya Engineering Sendirian Berhad in Malaysia through the Malaysia University of Science and Technology. Based on these research methods, four major research developments were determined. All issues raised in this analysis phase will be included in each chapter for each development.

2.2 DEVELOPMENT

There are four major developments in this research. The first one is an analytic framework that aims to understand how errors and changes are associated with the construction process and how they ultimately affect construction performance. Based on this understanding, the means to manage the detrimental impacts of errors and changes is presented focusing on construction schedule. Then, a System Dynamics-based simulation model is developed to analyze the impact of errors and changes and the benefit from the proposed management mechanism. Finally, a web-based error and change management system is implemented to support coordination of errors and changes among geographically distributed parties. In summary, a comprehensive

research development is designed from initiating a theory to implementing a working model and system in order to understand and manage the impact of errors and changes.

2.3 VALIDATION

All developments are applied to a couple of real-world case projects: a laboratory building project in Malaysia and a highway bridge infrastructure project in Massachusetts. The former was in collaboration with the Malaysia University of Science and Technology, and the latter was with Charles River Associates. Through these case studies, consequent conclusions and implications of this research are drawn.

CHAPTER 3

ERROR AND CHANGE MANAGEMENT FRAMEWORK

Most iterative cycles in design and construction are triggered by errors and changes. Error is defined as defective work of poor quality, such as the placement of piling in the wrong location or poor concrete performance; change is defined as any work required of the contractor or subcontractor that was not specified in the original contract document [Trauner, 1992]. An example of this would be the owner's request to change the building's purpose from a usual office space to a library facility.

In this chapter, *the association of errors and changes with the construction process and how they dynamically affect construction performance* are discussed. In particular, the understanding of the process with respect to errors and changes is the author's main interest. This is because if we have a good understanding of how it works, we will have a better chance to manage errors and changes successfully.

3.1 LITERATURE REVIEW

There has been a lot of research done in understanding of errors and changes. One of the main streams is the identification of their impact on productivity mostly with regression or neural network. For instance, Hester et al. (1991) studied construction change order impacts on labor productivity at the craft level, and Ibbs (1997) identified the effect of the size of change and its impact on productivity and cost. Hanna et al. (1999) researched on the impact of change orders on labor productivity using a linear regression model, and Moselhi et al. (2005) performed similar investigations using a neural network. In addition, Williams (2000) studied the risk of changes to safety regulations and their effect on a project and Lee et al. (2004) developed decision tree models to classify and quantify productivity losses caused by change order impacts.

All of these research efforts have contributed to the understanding of the impact of errors and changes on a project. However, another significant issue in understanding of errors and changes is the identification of impact mechanism caused by errors and changes. In addition to the identification of the degree of relationships between errors and changes and their impact on a project, how the construction process behaves as errors and changes are introduced is a key to make an appropriate policy to manage them. In this context, this research focuses on how errors and changes are associated with the construction process and how they affect construction performance. In addition, a timing issue in errors and changes has been rarely discussed in previous research. Recently, the author raised the detrimental impact of latency¹ (i.e., late

¹ a situation where errors and changes are not identified immediately and thus, become hidden and have a high possibility of re-appeared in a later stage of a project. It will be heavily discussed in this dissertation.

discovery of errors and changes) on performance [Lee et al., 2005], and Ibbs (2005) discussed the timing issue of changes concluding that late changes are more disruptive of project productivity than early changes. Extending these discussions, this chapter also explores latency in managing errors and changes at the process level.

3.2 FEEDBACK PROCESSES CAUSED BY ERRORS AND CHANGES

Construction projects are inherently complex and dynamic, involving *multiple feedback processes* [Sterman, 1992]. The uncertainty and complexity of design and construction projects are usually driven by these feedback processes. There are only two types of feedback processes; reinforcing and balancing [Sterman, 2000]. Dynamics in a system arise from the interaction of these two types of feedback processes among the components of the system, not from the complexity of the components themselves [Sterman, 2000]. Figure 3 shows the basic simultaneity of the reinforcing and balancing feedback in design and construction projects, which is caused by errors and changes. The control actions to address errors and changes can have the intended effect of resolving the issues that initiate the control actions, if the decision is correct and well implemented. At the same time, they can produce a side effect that may create some unintended problems, if the decision is incorrect, not well implemented, exceeds the time frame of its effectiveness or if a project manager does not realize the impact of the control actions on other related activities. As discussed earlier, overtime applied to deal with a schedule slippage could slow the project progress down by deteriorating workers' productivity.

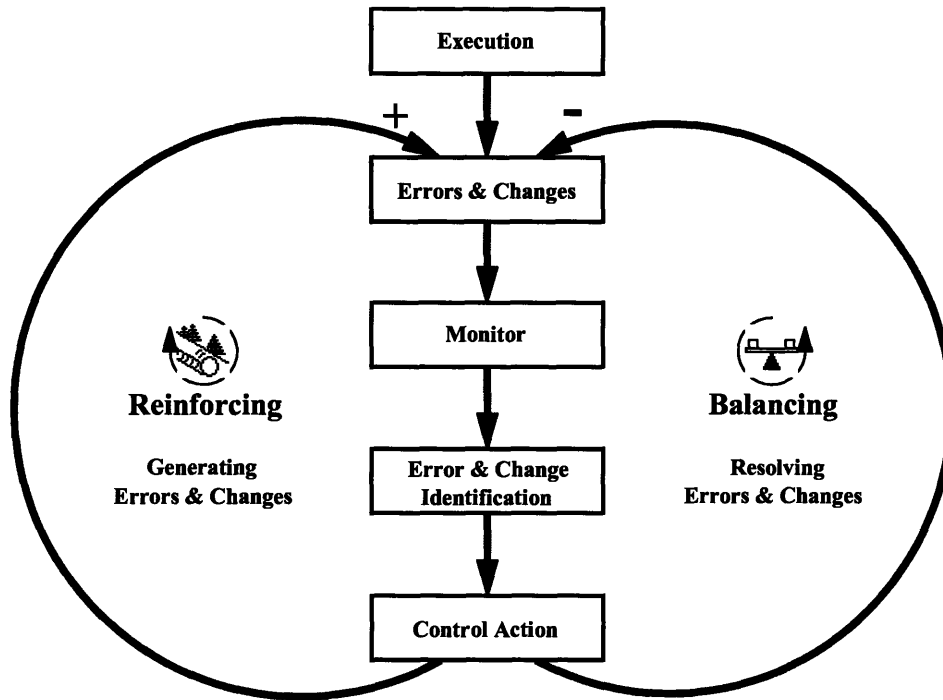


Figure 3. Simultaneity caused by Errors and Changes

3.3 ADDITIONAL WORK SCOPE GENERATED BY ERRORS AND CHANGES

Errors and changes usually cause non-value-adding iterations that deteriorate productivity and quality. These iterations are originally set according to an initial work scope. Thus, when an ***additional work amount*** is introduced by errors and changes, the designed productivity and quality may no longer be maintainable. The deterioration of productivity and quality are one of the main sources of multiple feedback processes and the corresponding actions necessary to recoup the deficit often generate unanticipated side effects.

Suppose an excavation activity can be completed by 10 backhoe loaders within 10 working days. If an additional 20% of work is added to the initial work amount to rectify errors and implement changes, a schedule delay of two days will occur (i.e., initial productivity will then require 12 days with 10 backhoe loaders). Managers will then typically utilize the tool of overtime to keep the initial 10 days. As expected, prolonging work hours will take care of the additional work, but the extended work hours simultaneously deteriorate the workforce's productivity due to increasing fatigue. Therefore, unexpected effects caused by various compensatory mechanisms need to be anticipated.

In this context, identifying how errors and changes are introduced, how they produce additional work scope and ultimately, how they affect performance, would be the key to the success of managing errors and changes in design and construction projects. Concentrating on the additional work scope as the source that affects design and construction performance, the following section will illustrate a framework that has been designed to show the impact of errors and changes in the design and construction process.

3.4 ERRPR AND CHANGE MANAGEMENT FRAMEWORK

The existence of multiple feedback processes, caused by errors and changes in complex inter-relationships of activities leads us to believe that errors and changes cannot be treated as discrete and constant events because they usually occur as iterative cycles. In other words, one response to an error and change could accompany another response if its unanticipated side

effects are not accounted for and avoided. To address this, this paper presents an *analytic framework for error and change management*, which can be used to identify the generation and management of iterative cycles caused by error and change. The proposed framework takes a holistic view of the design and construction process, considering all relevant elements continuously. This is distinct from the discrete view of traditional network planning tools that regard all the factors separately. The proposed framework would take a holistic approach, making it possible to draw the big picture of a project so that the interconnection of the components can be identified.

Internal Error Management Framework

Before exploring the project network as a whole, it is important to focus on how error behaves internally within a single activity. As seen in Figure 4, work is performed based on a given work scope during the actual execution of a design and construction project. However, the fact that all work has been performed does not guarantee that the work has been done correctly, and this will be addressed at the *quality management process*. The term, *reliability* (A in Figure 4) is used in this framework to indicate the degree to which the performed task has been done correctly during actual execution. For example, if Activity A has 90% reliability, the amount of error in Activity A is expected to be as much as 10% of the total work scope of the activity. However, this estimated reliability can vary due to the impact of a diverse set of variables during the work process. If Activity A is composed of repetitive tasks, the possibility to generate errors could decrease at the later stages as the workers become more familiar with those tasks.

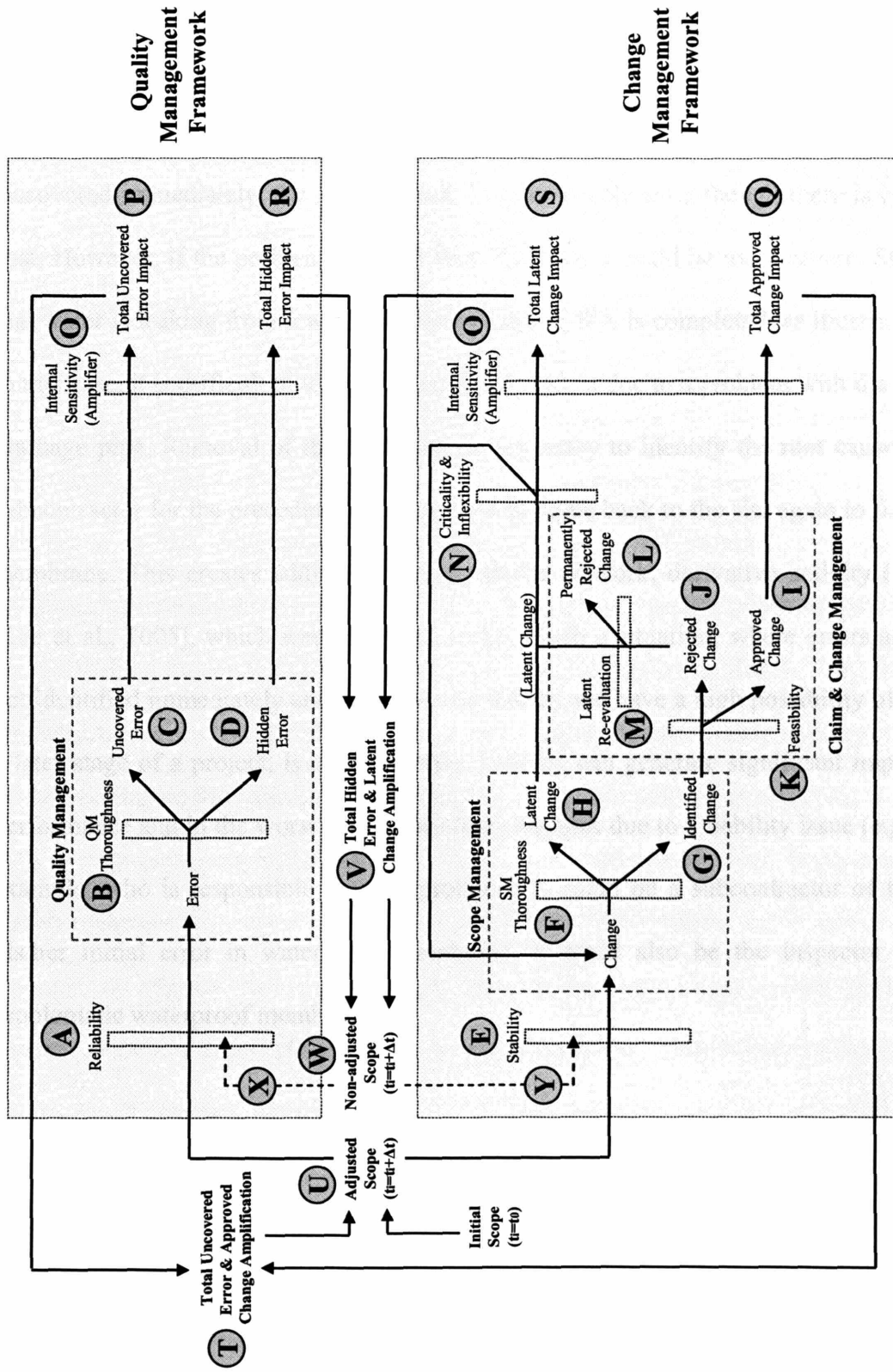


Figure 4. Internal Error and Change Management Framework

Another crucial point is that some portion of the errors generated in the execution of an activity may not be uncovered during the quality management process. Suppose a Waterstop Activity (WA) and a Substructure Activity (SA) are developed concurrently, and waterproof membrane in the WA is not properly installed, as seen in Figure 5. If the problem were discovered immediately, the impact could be manageable even though there is a corresponding cost. However, if the problem is found later, the impact could be more severe. Suppose we find that water is leaking from a wall after the preceding WA is completed, as illustrated in Figure 5. In this case, it is difficult to identify whether the leak is due to a problem with the membrane or a drainage pipe. Removal of the wall may be necessary to identify the root cause, and then, the subcontractor for the preceding WA may need to come back to the site again to fix or replace the membrane. This creates additional activity in the network, derivative activity (A in Figure 5) [Lee et al., 2005], which was discussed earlier. Such a situation, where errors and changes are not identified immediately and thus, become hidden and have a high possibility of re-appeared in a later stage of a project, is called *latency*. Latency can generate significant impacts on project performance and in the worst case, cause legal disputes due to a liability issue (e.g., in the above example, who is responsible for this problem? It could be a subcontractor of the WA due to his/her initial error in waterproof membrane. It could also be the inspector who approved problematic waterproof membrane).

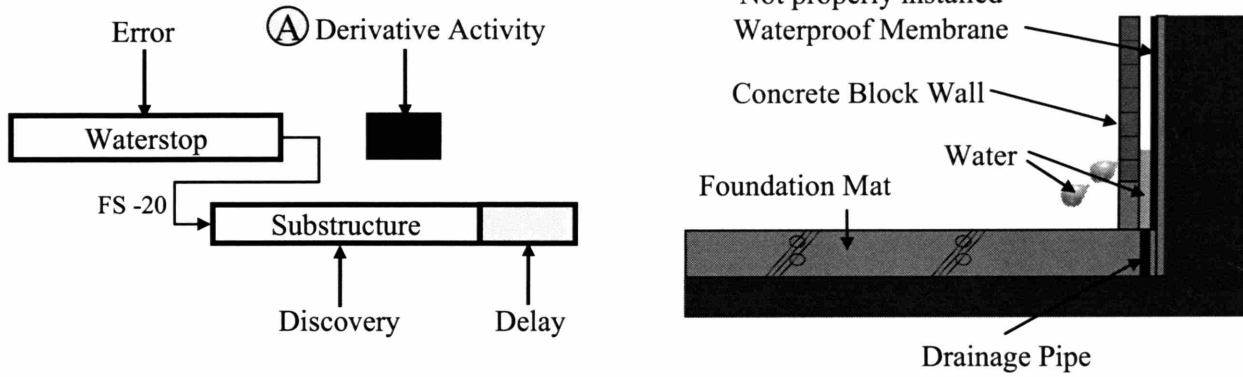


Figure 5. Example of Latency

In order to explain this latency, *quality management thoroughness* (B in Figure 4) is introduced in this framework. Quality management thoroughness is defined as a dynamic variable that represents the degree to which the existing quality problems of an activity have been identified during the quality management process applied to that activity. This is based on the fact that the applied quality management techniques or system may not be executed perfectly or the techniques themselves may not be perfect to start with. For example, if Activity A has 80% quality management thoroughness, 80% of the total errors would be discovered during the quality management process, and the remaining 20% would be elusive to the quality management process. Errors that are not uncovered during the quality management process are denoted as *hidden errors* (D in Figure 4) in the framework, while *uncovered errors* (C in Figure 4) are those that are discovered. These relationships can be mathematically formulated as follows:

$$E_t = 1 - R(t) \tag{1}$$

$$E_u = E_t \times \prod_{k=1}^n (QT_k) \quad (2)$$

$$E_h = E_t \times \prod_{k=1}^n (1-QT_k) \quad (3)$$

where, E_t is total error generation rate, E_u is uncovered error generation rate, E_h is hidden error generation rate, $R(t)$ is reliability in a given activity, QT_k is the quality management thoroughness of a particular applied technique k , and n is total number of techniques in the quality management process. The right-side product found in both Eq. (2) and (3) explains the effect of quality management thoroughness on the uncovered and hidden error generation rate. Here, the product is used in order to combine impacts of different techniques because the failure of one of the applied techniques could mean a system failure.

Diverse techniques for quality management can be applied to a particular activity at a given time. They have different abilities to discover errors, depending on various system conditions, which include the users' familiarity with applied QM techniques (QM familiarity), how well the QM techniques are implemented or how adequate they are to be applied to a particular activity (QM implementation), and the impact of the schedule pressure on the implementation of the different QM techniques (schedule pressure on QM). For example, suppose welding is performed in a pipe installation activity, and ultrasonic testing is used as a quality management technique to inspect the welding quality. Though this inspection technique may be accurate and well-implemented (i.e., high QM implementation), an inspector may not have sufficient experience and knowledge to analyze the results from this testing (i.e., low QM

familiarity). Therefore, this low QM familiarity negatively affects the quality management thoroughness. Some portion of the quality problems with the welding may not be uncovered. In addition, if inspectors have more work to inspect than expected and feel the pressure of meeting the schedule, they may make an effort to speed up the progress of the testing activity. Thus, there would be more of a chance that errors could not be uncovered (i.e., high schedule pressure on QM).

Internal Change Management Framework

In the case of the change management framework, there are two major components as presented in Figure 4. One is the *scope management process*, which can be explained as the review process necessary to make sure that the given scope of work is the same as the specified one in the drawings and specifications. The other is the *claim and change management*, which plays a major role in deciding whether a claimed change order should be accepted or rejected.

Change depends on the *stability* (E in Figure 4) of the given initial scope of an activity. Stability is used in this framework to indicate the degree to which the given work scope would be performed without a request for change. High stability means that only a small number of changes would be expected during the execution of a particular activity, while low stability represents the possibility that a great number of changes would be requested.

However, some potential changes may not be identified during the scope management process (i.e., latency). Changes that are not identified are denoted as *latent changes* (H in Figure

4) in this framework, while *identified changes* (G in Figure 4) denote those that are recognized by the project management team. To represent this situation, the concept of thoroughness is applied to the scope management process in a similar way that it was applied earlier to the quality management process. These relationships can be also formulated as follows:

$$C_t = 1 - S(t) \quad (4)$$

$$C_i = C_t \times ST \quad (5)$$

$$C_l = C_t \times (1 - ST) \quad (6)$$

where, C_t is total change generation rate, C_i is identified change generation rate, C_l is latent change generation rate, $S(t)$ is stability in a given activity, and ST is the scope management thoroughness. Estimating those values follows similar procedures as in the quality management process situation.

Scope management thoroughness (F in Figure 4) is identified by three factors, which are Scope Management (SM) familiarity, completeness of sources, and schedule pressure on SM, respectively. SM familiarity represents users' experience on or familiarity with the change review process, and completeness of sources explains the degree to which sources supporting a change, such as drawings and specifications, are complete and ready for analysis. The more complete the sources, the more opportunity there is to identify changes. Lastly, schedule pressure on SM shows the effect of schedule pressure on the scope management process.

When contrasted with the error management, the change management framework has one more process after the scope management process: the approval of an identified change request, which is determined by a special group in an organizational level (i.e., *claim and change management group*). Identified changes are usually approved (*approved change*, I in Figure 4) on the basis of their feasibility (K in Figure 4) to the project through the *claim and change management process*. For instance, if the identified change is perceived to require a significant investment or there are other options to replace it, it can be rejected (*rejected change*, J in Figure 4). If a change is approved, the corresponding additional work scope is introduced to that particular activity. A rejected change may either become a *permanently rejected change* (L in Figure 4) or it can be designated as a latent change in terms of its potential for reconsideration later in the project (i.e., *latent re-evaluation*, M in Figure 4).

Suppose that an unexpected site condition triggers a change in the construction method. This change (i.e., identified change) may have been the best way to deal with the different site condition. However, this change had not actually been approved (i.e., rejected change) at that time because it would have increased the budget of the project significantly, and there was a possibility that the original method could handle the differing site condition. Because this rejected change may be brought back to the table, it would not be permanently rejected. But it would instead be reclassified as a latent change. If there is no possibility of reconsideration, the potential change would be permanently removed. This reconsideration process for a rejected change is denoted as latent re-evaluation, and it determines if a rejected change would be adopted as a latent change or a permanently rejected change.

Lastly, a latent change may be re-introduced as a change. Continuing with the above example, suppose the originally planned method didn't work well, so the suggested change in the construction method (i.e., latent change) is brought back to the table. This re-introduction is based on two main factors, *criticality* and *inflexibility* (N in Figure 4). Criticality represents the importance of an activity in the network, and inflexibility represents the lack of other alternatives to address this change. Therefore, if the activity is on the critical path and there are no other options to substitute it, this latent change would be re-introduced as a change to be re-evaluated by the claim and change management group.

Overall Internal Error and Change Management Framework

The error and change management framework presented thus far can be integrated into one cohesive framework, since they share an important common feature; the generation of errors and changes means the potential increase of the initial scope. An important point in this scope increase is that the inter-relationships within an activity can amplify the total scope increased by errors and changes. If tasks within an activity (this research assumes that an activity can be divisible into many work units and hereafter, task is used to denote the work unit within an activity) are highly coupled due to physical and procedural constraints, the impact of errors or changes can be much more than the case in which the tasks are independent from each other. For example, if a project has an activity called column foundation that encompasses all the foundations for the columns in a given area, a problem with one of the column foundations does not affect the other column foundations. In such a case, the tasks within the column foundation activity are not highly coupled. However, if a project has an activity called pile foundation

alignment, then an error on the alignment of the first pile may affect the alignment of the remaining piles. In this case, the tasks within the pile foundation alignment are highly coupled. In this context, *internal sensitivity* (O in Figure 4) is defined as an amplifier variable to capture the degree to which the corresponding tasks are inter-related within an activity [Eppinger, 1997].

Applying internal sensitivity into the overall framework, an uncovered error becomes a *total uncovered error impact* (P in Figure 4). This is because total uncovered error impact is the actual impact on the construction process after combining uncovered error and its impact on other tasks in an activity. This can be extended to *total approved change impact* (Q in Figure 4) from approved change. On the other hand, internal sensitivity also governs hidden errors and latent changes, creating *total hidden error impact* (R in Figure 4) from hidden error as well as *total latent change impact* (S in Figure 4) from latent change. Though they are not uncovered or approved, they can have an impact on succeeding tasks, by creating quality problems. Continuing with the example of the piling activity, if misplaced string lines are not discovered immediately, this hidden error may also generate subsequent errors because workers may install piles according to the misplaced string lines.

As denoted by T in Figure 4, the total uncovered error impact and total approved change impact can be combined as the *total uncovered error & approved change amplification* because their existence has been detected by managers through the normal monitoring process. This total uncovered error & approved change amplification refers to the additional work scope that is ultimately brought about by uncovered errors and approved changes. To address this change of the work scope during actual execution, *adjusted scope* (U in Figure 4) is denoted as the newly

introduced work scope combined with the initial scope and may consistently change as the activity progresses.

Meanwhile, as denoted by V in Figure 4, hidden errors and latent changes can also be combined to produce the *total hidden error & latent change amplification* because their existence would not have been detected by managers at this point. This total hidden error & latent change amplification is the unknown work scope increased by hidden errors and latent changes and should be dealt with because a project cannot be completed without correcting hidden errors and addressing latent changes. *Non-adjusted scope* (W in Figure 4) is used here to address this unknown impact on the work scope and can explain what is called the ‘90% syndrome’, which is often encountered on design and construction projects [Ford and Sterman, 2003]. In other words, late discovery of the non-adjusted scope causes an overflow of work and consequently, could cause a project to suffer from slow progress at the later stages. Moreover, this slow progress at the later stages may increase the pressure on the project manager and influence him/her to accelerate the planned quality and scope management process in order to speed up the delay. However, this rush could generate other errors and changes. X and Y in Figure 4 illustrate how the non-adjusted scope could deteriorate the reliability and stability of an activity.

External Error and Change Management Framework

The internal error and change management framework can be extended to capture the quality and change management process impacts from one activity to other inter-related activities.

This is enabled by the extension of the concept of sensitivity to the inter-relationship between activities. *External sensitivity*, as denoted by A in Figure 6, is defined as the degree to which activities are externally inter-dependent on other activities. Therefore, the total uncovered error impact (B in Figure 6) and the total approved change impact (C in Figure 6) in the activity under study, become, respectively, the total uncovered error amplification (D in Figure 6) and the total approved change amplification (E in Figure 6) of the predecessor and the successor activities by course of the external sensitivity among those activities.

These are combined as the total uncovered error & approved change amplification (F in Figure 6), which is perceived by managers (similar to T in Figure 4). Consequently, these errors and changes would be absorbed in the adjusted scope of each corresponding activity. Similar logic also applies to the total hidden error & latent change amplification (G in Figure 6) in the predecessor and the successor activities with one major difference; this is not easily perceived by managers, which will constitute the problematic non-adjusted scope. Therefore, adjusted scope and non-adjusted scope are defined by these internal and external impacts together. On the other hand, the total uncovered error & approved change amplification (H in Figure 6) and the total hidden error & latent change amplification (I in Figure 6), in the activity under study, are also affected by error and change impact in the predecessor and the successor activities through the corresponding external sensitivity (J in Figure 6).

Ultimately, this external quality and change management framework between activities can be extended to the whole network of a project, as seen in Figure 7. Based on the concepts

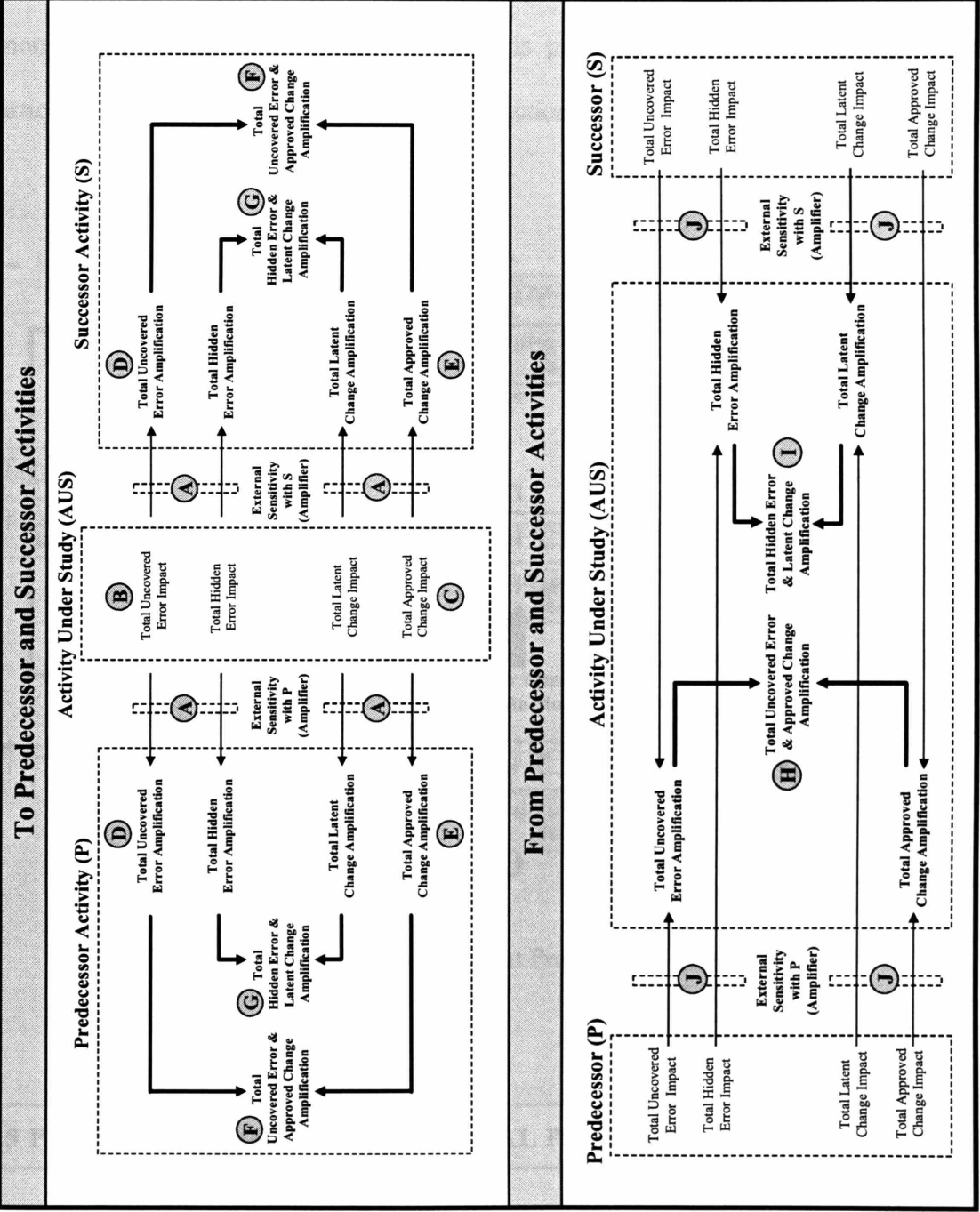


Figure 6. External Error & Change Management Framework

previously presented, error or change introduction at a certain activity may affect the activity itself as well as adjacent activities through a corresponding sensitivity. In other words, this process could iterate, in theory, until the end of the project, if external sensitivity impacts exist among all the activities. Figure 7 shows this propagation effect on the whole network, in particular, when concurrent design and construction is used in a project.

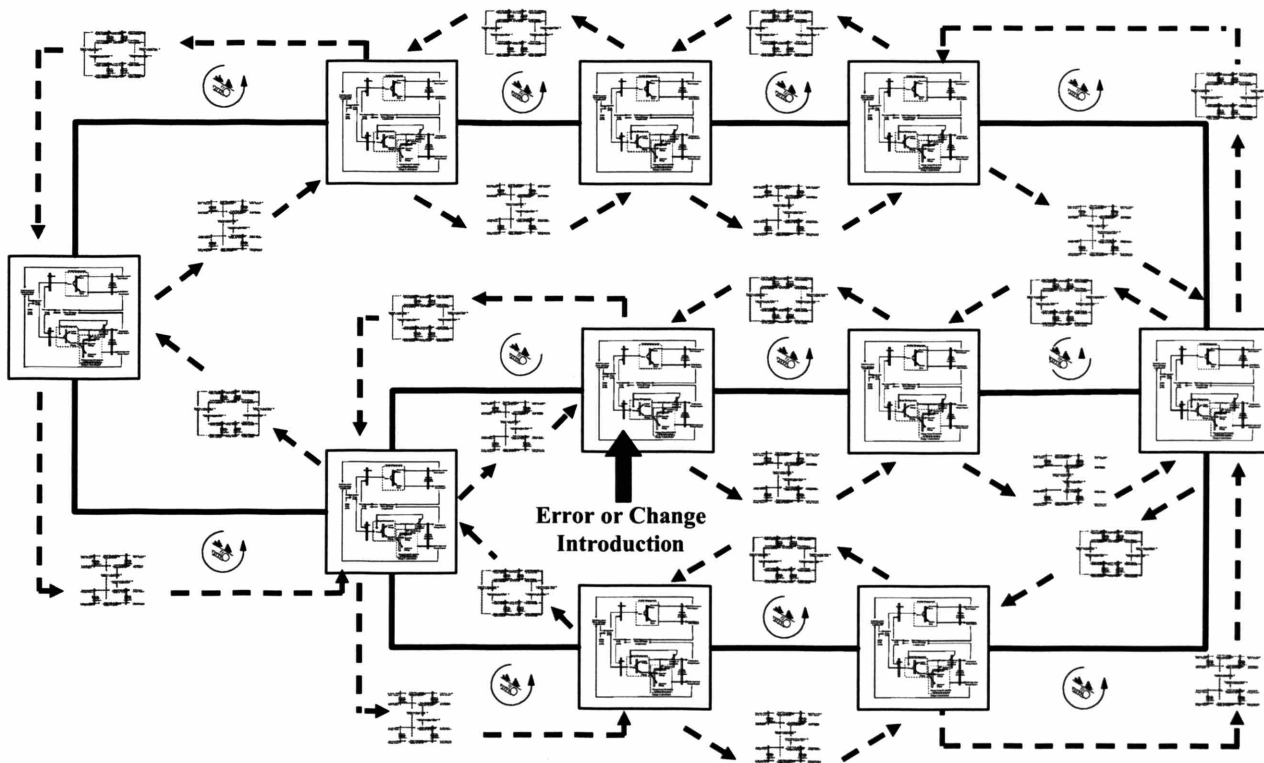


Figure 7. Error and Change Management Propagation Effect on the Whole Network

3.5 PLANNED, PERCEIVED, AND REAL PERFORMANCE

By understanding the process associated with errors and changes, and in particular, by understanding the difference between adjusted scope and non-adjusted scope caused by errors

and changes, it becomes possible to understand how errors and changes affect construction performance.

For that purpose, the schema of *Dynamic Design and Construction Project Model* (D²CPM), the basic framework used to build the system dynamics simulation model, is proposed to illustrate how the project can be understood at systems-level. The D²CPM schema consists of five sub components: the project scope, the project target (e.g., schedule, cost, and quality), the resource acquisition and allocation profile (e.g., labor, equipment, and material), the design and construction process, and the design and construction performance profile, as illustrated in Figure 8 [modified after Ford and Sterman, 1998]. The main idea of this schema is to highlight the role of the design and construction process that bridges the project scope, target, and resource profile with the design and construction performance profile. Envisioning the process as a bridge enables a clear identification of the interaction and mechanics of these five sub components in an uncertain and dynamic environment.

For example, prior to its execution, a design and construction project usually pre-sets its initial scope, resources (e.g., cost, worker, material, and equipment) and target (e.g., deadline and budget limit that should be kept). During actual execution, the actual design and construction process is determined by the interaction of these three inputs. Consequently, this process will generate a corresponding performance profile that reflects the status of design and construction. For instance, if it is not feasible that the planned resource allocation can be accomplished within the scope of the target schedule, the resulting performance would not match the expected inputs. This identified performance during actual execution is called *perceived (or monitored)*

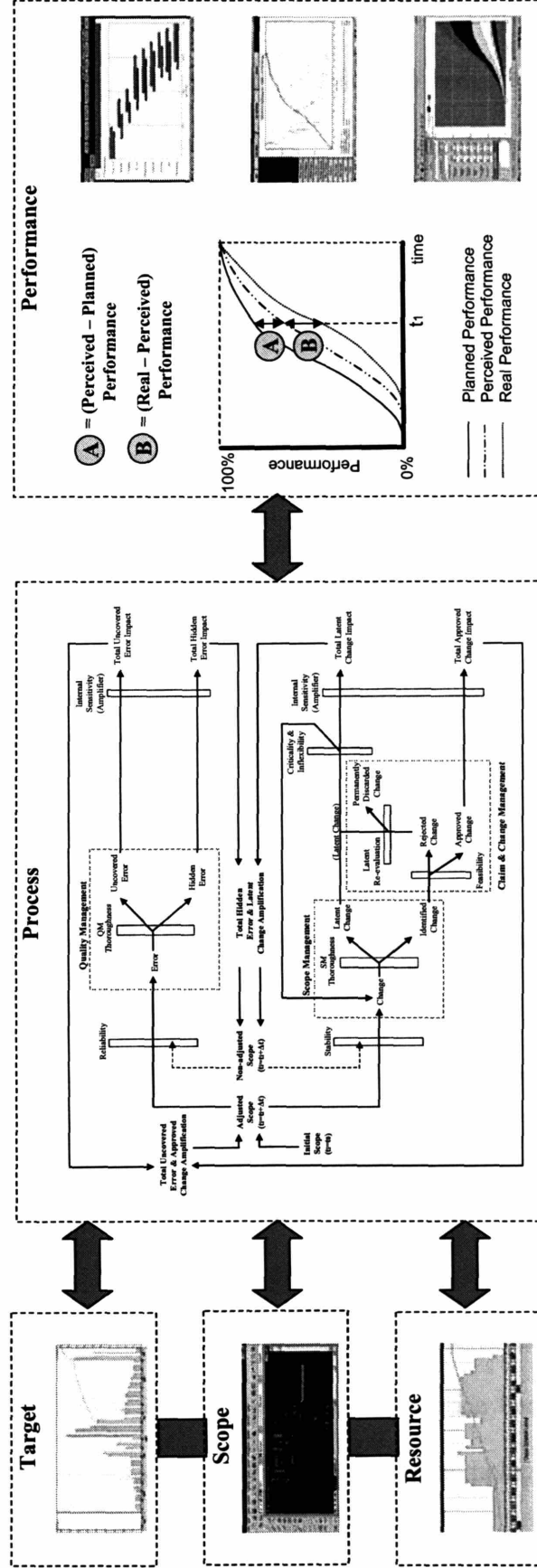


Figure 8. Schema of Dynamic Design and Construction Project Model
 [adapted from Ford and Sterman, 1998]

performance [adapted from Abdel-Hamid, 1984] and can be different from *planned performance* (A in Figure 8). If a gap between planned and perceived performance exists, the process will reflect the impact of this performance gap on these three inputs (scope, resource, and target). In other words, the three inputs will be adjusted throughout the process because managers would take control actions to meet the planned performance. Suppose the perceived performance is far behind the planned performance due to a delay caused from encountering more solid rock than expected during an excavation activity. In order to overcome this performance gap, several control actions can be made, such as putting more backhoe loaders to work (i.e., changing resource), or deferring the completion of the excavation (i.e., changing the target schedule). If more backhoe loaders are working, the construction process would be executed with a different resource (e.g., increased backhoe loaders). Accordingly, it would generate another performance profile and these interactions might be iterated until the project is completed.

Actually, this iteration would be continued until the *real performance* [adapted from Abdel-Hamid, 1984] is identified. Real performance represents what truly happens in a project. This is because perceived performance may not reflect 100% of actual performance. For example, due to the characteristics of construction (e.g., open environment and involvement of many temporary subcontractors), it is difficult to receive accurate performance information. Moreover, even if it is received, an information delay could exist (e.g., information acquired today actually reflects previous week's information). In reality, this difference between perceived and real performance (B in Figure 8) is often disregarded in practice because it has been difficult to measure. But, it has a significant impact on a project. For example, the discussed 90% syndrome,

the sudden overflow of work at a late stage of a project, exemplifies the detrimental impact of the gap between perceived and real performance.

A simple illustration of this would be to suppose we are installing 9 piles, as seen in Figure 9. If performance is measured by the number of completed piles, completing 9 piles would be expected as the output of the process, and this is planned (or expected) performance. However, errors can be generated and uncovered (i.e., uncovered error) and changes can be approved (i.e., approved change) during the process. In this pile installation example, suppose the belief that one of the piles is complete turns out to be erroneous due to a strength failure during the quality management process. In this case, 8 piles were actually completed rather than 9, and this is perceived (or monitored) performance.

On the other hand, errors and changes are not often identified immediately, giving rise to the situation called latency (i.e., hidden errors and latent changes). Continuing with the same pile installation example, suppose one of the piles is erroneous and at this time, it has not identified yet (i.e., hidden error). In this case, even though we perceived that we accomplished 8 piles, real (or actual) performance is the completion of 7 piles.

The gap between perceived performance and real performance has a crucial meaning in managing errors and changes. If there is the gap between planned and perceived performance, which is caused by uncovered errors and approved changes, we will make a decision and take control actions to eliminate this gap. However, the fact that perceived performance may not be same as real performance, which is caused by hidden errors and latent changes, informs us that

this decision making and controlling process may start with incorrect information. Thus, there is strong possibility that decisions intended to solve errors and changes rather would generate unexpected side effects, thereby disrupting construction severely. This is due to latency.

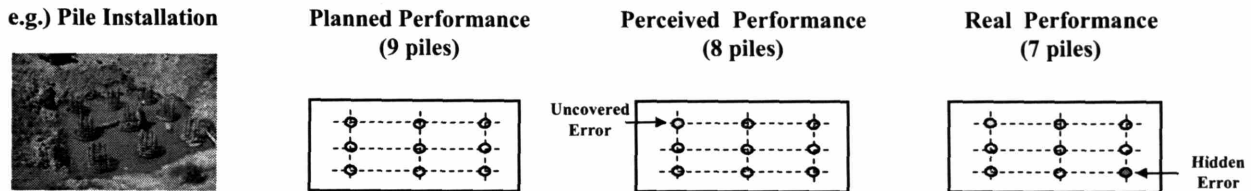


Figure 9. Planned, Perceived, and Real Performance at Pile Installation Example

This detrimental impact of latency becomes more severe when sensitivity is considered. Figure 10 illustrates a modified and simplified D²CPM schema to explain the impact of errors and changes on performance and scope. If there is a gap between planned and perceived performance, managers take some control actions in an attempt to reduce this gap, which usually are accompanied by an increase of scope. Continuing with the pile installation example, we may need to remove the existing erroneous pile and to install a new one. Thus, removing and installing a pile contribute to additional scope.

On the other hand, in the latency case, succeeding tasks, such as installing a column, could be already completed. If the hidden error is discovered after installing the column, the column may need to be removed before the erroneous pile and followed by installation of a new pile and column. This creates increased additional work compared to the first case. The increase of scope during actual execution is one of the main drivers of project disruption because most plans are

based on the estimated scope. For example, assume we are going to prepare workers and resources based on the scope of this pile installation, installing 9 piles. In order to deal with additional scope, we need to procure more workers and resources. Furthermore, the schedule extension may not be allowed for this corrective work because keeping the schedule is a primary objective in most projects. Thus, more resources tend to be assigned at a late stage of the project, such as adopting overtime and hiring new workers. However, it is well known that this late assignment does not increase productivity [Stermann, 1992]. In summary, when latency and sensitivity are taken into account together, construction becomes seriously disrupted due to unexpected increases in the scope during execution and this understanding is enabled by the D²CPM schema.

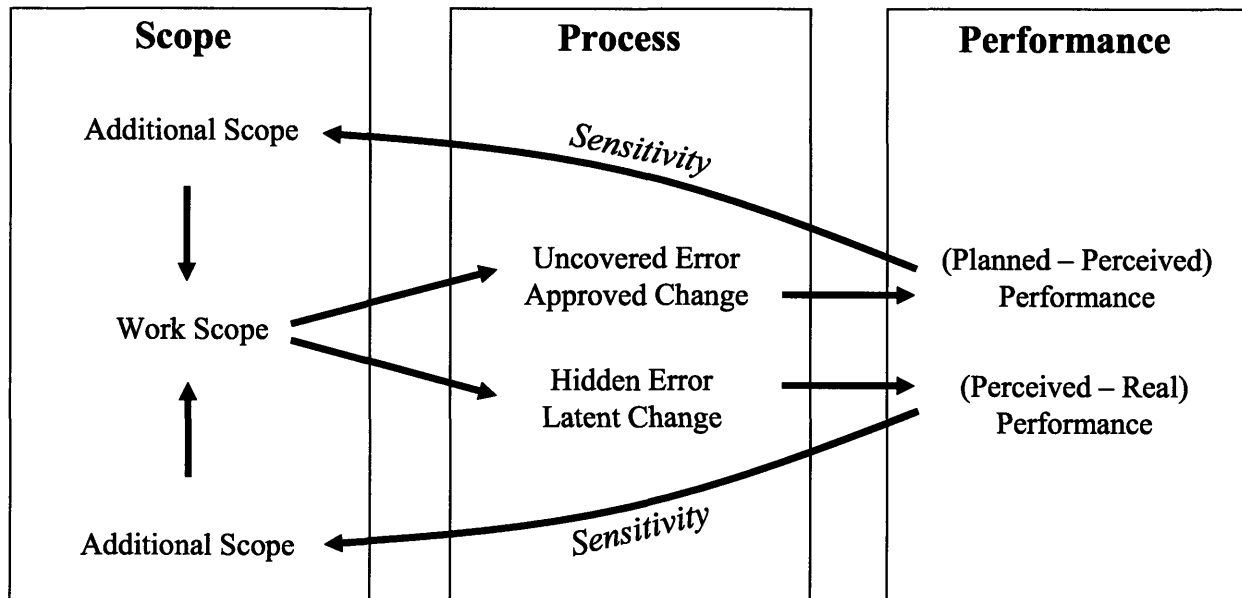


Figure 10. Application of the D²CPM Schema

CHAPTER 4

RELIABILITY AND STABILITY SCHEDULE BUFFERING

Based on the understanding of errors and changes attained from the framework, this chapter discusses the impact of errors and changes can be mitigated or absorbed. In the previous chapter, the author explained that errors and changes can have a harmful effect on project performance, most often producing a ripple effect among the different activities needed to accomplish more additional work than expected. Furthermore, when concurrent design and construction is applied, often succeeding activities have to proceed without complete information from preceding activities and this can lead to a chain of suboptimal or erroneous decisions affecting other related activities.

One method that is able to control these unanticipated effects of errors and changes is the deliberate utilization of buffers in the design and construction schedule (i.e., schedule buffer). Buffers provide a method to accommodate uncertain and variable conditions by absorbing perturbations and problems [Sakamoto et al, 2002]. However, buffers in design and construction have been mainly used as a contingency, such as adding certain percentage of the activity durations at the end of an activity without an appropriate analysis of the individual characteristics

of the activity. As a result, buffers often fail to protect schedule performance despite their potential to overcome uncertainty.

Taking these issues into account, this dissertation presents a *reliability and stability buffering approach* that extends Park and Peña-Mora's reliability buffering (2004). It further extends to account not only with error issues (i.e., reliability) but also change issues (i.e., stability), especially in concurrent design and construction. In addition, dealing with latency is seriously considered. This reliability and stability buffering approach aims to build a robust design and construction plan against uncertainties, focusing on the detrimental impacts of errors and changes, when concurrent design and construction is used. The presented buffering approach adopts a proactive and flexible buffer location: a systemic buffer size based on activity characteristics; and a dynamic update mechanism, to vary location and size, in order to maximize its benefit. Prior to describing the proposed buffering approach, how buffers have been utilized in design and construction are discussed.

4.1 JUST-IN-TIME VS. JUST-IN-CASE

Planners in construction companies do not typically discuss the deliberate insertion of buffers in their schedules to manage their programs [Horman et al, 2003]. One of the reasons is that buffers do not directly add value and are considered a waste. This viewpoint has been strongly supported in Just-In-Time (JIT) literature [Womack and Jones, 1996]. JIT emphasizes the importance of a smooth workflow of execution, such as minimizing inventory and synchronizing the production

rate. Concurrent with this viewpoint, it is believed that buffers could impede a smooth workflow of execution by maintaining a redundant capacity without value adding.

However, there has been recent research in the area of supply management which emphasizes the role of a redundant capacity to reduce significant risks, which is known as Just-In-Case (JIC) [Brown, 2003]. JIC highlights the need for preparedness against uncertain business environments and thus, buffers are strongly advocated. In other words, if uncertainty is inevitable in business, preparing a redundant capacity could avoid a worse situation; even though, it could be more costly. Ultimately, it is believed that the cost to keep the redundancy would be less than the disastrous case that could result from no preparedness.

If all activities in a project are predictable and follow a planned performance, JIT without buffers would be the best solution in terms of a speedy and efficient execution of design and construction projects. However, if uncertainty is inevitable, JIC with buffers is the best alternative for a JIT delivery by providing the capacity needed to maintain a smooth workflow of execution. Therefore, JIC with buffers should be viewed as an approach that is taken to incorporate the main idea of JIT, not to replace it, by pursuing a speedy and efficient execution. Even in JIT literature, there has been arguments that advocate the potential benefit of a redundant capacity, and particularly; the usage of buffers, in order to absorb perturbations caused by uncertainty in construction [Howell et al, 1993; Ballard and Howell, 1995; Alarcón and Ashley; 1999, Tommelein and Weissenberger, 1999; Sakamoto et al, 2002]. Thus, buffers could play a

key role in reducing possible disruptions in uncertain design and construction projects, if they can be effectively utilized (i.e., their negative impacts on the smooth workflow are minimized).

4.2 LITERATURE REVIEW

However, despite the fact that buffers' have a great potential to reduce uncertainty, they have not been effectively used in design and construction. The majority of ways that buffers have been utilized in construction project planning, is as a time contingency that aims to ensure a scheduled completion time of an activity as well as of a project. However, the current application of contingency buffers has not been well suited for uncertain design and construction in most situations. For example, the contingency buffer in construction planning is normally positioned at the end of the activity duration with a uniform rate (e.g., 10% of the activity duration). The positioning of the buffer at the end of an activity does not provide for any prevention of a possible schedule disruption; rather, it just gives time to recover from a disruption.

Suppose both a final design and a permit acquisition activity have a Finish-to-Start (FS) relationship with an excavation activity, and contingency buffers (e.g., 10% of each activity's duration) are given, as seen in Figure 11. During execution, it turns out that the actual final design activity's duration is longer than the planned, and the actual permit acquisition is the same as the planned. In this situation, the excavation activity would be delayed ($t_7 - t_6$) as much as the final design is delayed ($t_4 - t_3$). Although the contingency buffer for Just-In-Case (JIC) provides time to absorb this delay, it may not be efficient to absorb all the delay caused from the

predecessor and to protect a planned schedule. This is mainly due to the location of the contingency buffer (i.e., the end of the activity), which implies that the only way to absorb delays is by catching up with the schedule or accelerating the progress, after something has already happened, not by looking ahead. Therefore, the delay which occurred at the predecessor activity could pass to the successor activity without much absorption and prevention. In addition, the uniformly assigned buffer size, without considering any characteristics of activities, may not be effective in protecting the planned schedule. In this example, if the possibility that the final design activity could be delayed had been perceived early, a greater buffer size should have been assigned to protect the whole schedule.

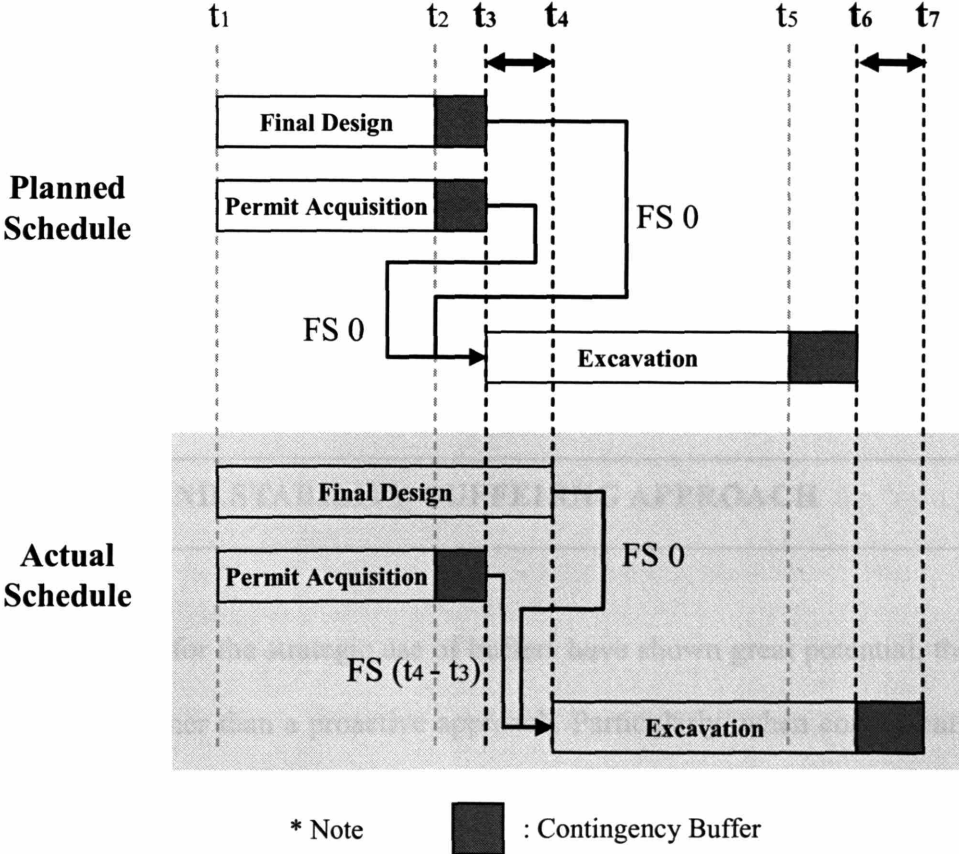


Figure 11. Inefficiency of Contingency Buffer

As an effort to overcome inefficiencies of contingency buffers, several approaches in both academic and industrial settings have been pursued [Howell et al., 1993; Ballard and Howell, 1995; Goldratt, 1997; Chua and Shen, 2001; Horman et al, 2003]. For example, some design and construction companies have implemented buffers that are both flexibly located and sized, using a diverse risk analysis techniques (e.g., multiple regression) based on historical data. Howell et al. (1993) used a buffer in decoupling tightly-linked activities in order to eliminate performance-reducing interactions between sub-cycles. Ballard and Howell (1995) argued that determining the size of buffers should be based on the degree of associated uncertainties. In addition, Goldratt (1997) has proposed a critical chain schedule, which emphasizes the role of buffers to protect the project due date. A critical chain schedule takes contingency buffers that were scattered among the activities and concentrates them where they do the most good: such as contingency buffers at the end of the critical path and where others paths feed that critical path [Goldratt, 1997]. These approaches have demonstrated that the strategic use of buffers (e.g., flexible size and location) could be more effective in protecting the planned performance than those that do not consider buffers or only apply buffers in terms of contingency for each activity.

4.3 RELIABILITY AND STABILITY BUFFEIRNG APPROACH

Although recent efforts for the strategic use of buffers have shown great potential, they still take a reactive approach, rather than a proactive approach. Particularly, when concurrent design and construction is applied, a reactive approach may not be effective since it only gives a time to recover after something happened. In addition, they are usually based on discrete and isolated

historical data only, without the identification of the cause and effect between feedback processes and buffers. In this sense, they may miss the dynamic and non-linear interactions within the activities in a design and construction project, which have more detrimental impacts on performance. In order to address these issues, reliability and stability buffering is presented as a mechanism to protect the planned performance against uncertainty caused by errors and changes.

Buffer Location – Proactive Approach

Reliability and stability buffering places buffers at the beginning of an activity, rather than at the end of an activity as the contingency buffer does to absorb delays. This different positioning allows the buffer to handle ill-defined tasks by introducing a pre-checking process that can capture and correct predecessors' hidden errors and latent changes before tasks are being performed. It reinforces the notion that one has to plan for eventualities and analyze decisions in the context of the whole project and not only from the perspective of the immediate activities being impacted. Therefore, it can be considered as a *proactive mechanism* against delay instead of a reactive mechanism, which the contingency buffer usually stands for. In addition, this different positioning initiates a proactive and collaborative communication process among the parties associated with the concurrent activities as well as a process to ramp up the resources for the remaining activity. Therefore, the related parties are able to discuss and coordinate potential issues through this collaborative process, instead of being reactive to the directives of the parties involved with the predecessor activities or upper management team. Lastly, the proactive

mechanism can avoid the waste of multiple buffers in each predecessor activity in the case of a merging point (i.e., multiple convergences) in the network.

However, putting just a single buffer at the beginning of an activity may not provide the most effective protection against iterative cycles caused by errors and changes in concurrent design and construction. For instance, suppose a final design activity and an excavation activity are conducted concurrently, and errors and changes are generated more frequently at Part F than Part S of the final design activity, as illustrated in Figure 12. A single continuous reliability and stability buffer may not be able to handle all of these errors and changes effectively, even if the buffer is sized to be the maximum possible. In this case, the excavation activity has significant overlap with Part S of the final design activity, making it extremely difficult for the single continuous reliability and stability buffer to be able to handle effectively the errors and changes introduced in Part S of the final design activity. In terms of time sequence, the single continuous reliability and stability buffer in the excavation activity is the most effective in only handling errors and changes at Part F and not Part S.

To avoid this situation, the reliability and stability *split buffer* (dual buffer) is introduced and is only activated in the case where two related activities considerably overlap and their error and change generation rate is estimated high in the latter part of a predecessor activity, as seen in Figure 12. In other words, it can be interpreted as a split on the successor activity duration to accommodate the predecessor's errors and changes. If activities have a serial precedence relationship, such as a Finish-to-Start (FS) relationship with positive lag, the reliability and stability buffer without a split (single buffer) can handle the uncertainty of a predecessor activity.

Otherwise, the split buffer may be activated to handle excessive concurrent development with highly variable error and change generation rate in the latter part of a predecessor activity.

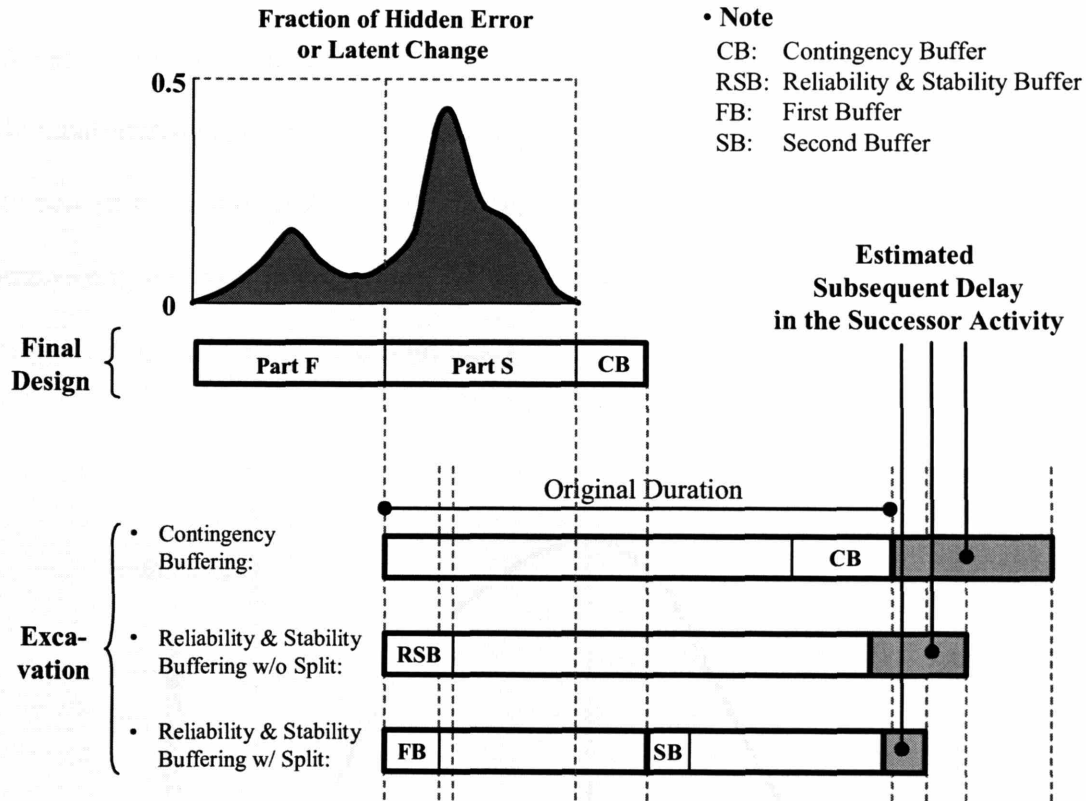


Figure 12. Reliability and Stability Buffer Split

Buffer Allocation – Role of Schedule Pressure

In general, the traditional contingency buffer in construction planning tends to be used as part of an activity without clear distinction from the original duration [Horman and Kenley, 1998]. As a result, time added to the original duration may not effectively protect the planned schedule because when people realize that they have more time to complete a task than the time actually

specified, their work productivity usually goes down [Sterman, 2000]; often with the task being deferred to the last minute, creating a 'rubber band' duration. This phenomenon can also be explained by Parkinson's law, which states the work expands to fill the time available for its completion. In this context, reliability and stability buffering takes off contingency buffers from all individual activities, if the activities have contingency buffers, and consequently makes each activity benefit from *appropriate schedule pressure* (i.e., the stress in the workers' environment by compressing the time to complete). Subsequently, the appropriate buffer size is determined and assigned (this step will be discussed later).

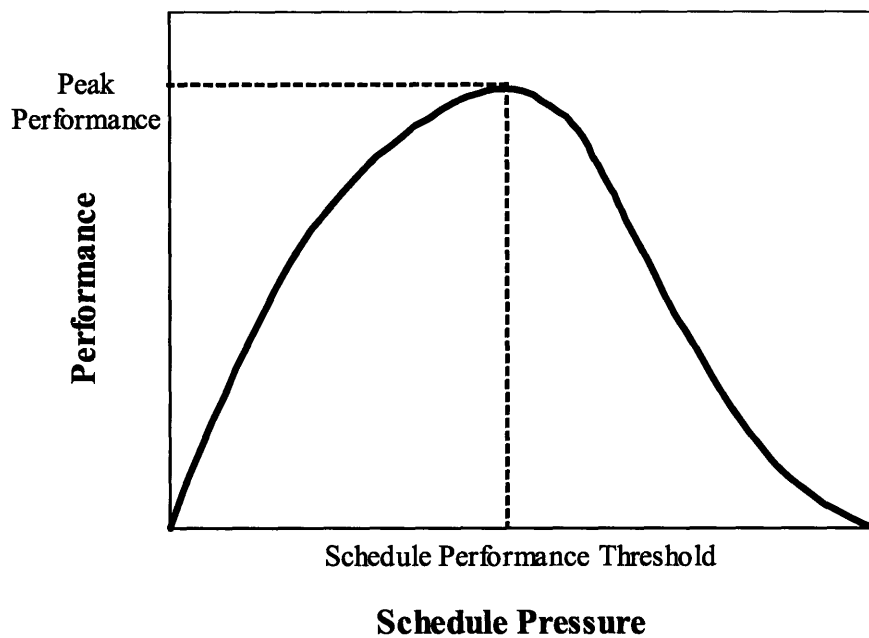


Figure 13. Yerkes-Dodson Law [Adopted from Yerkes-Dodson, 1908 as referred on Sterman, 2000]

Excessive schedule pressure may deteriorate workers' productivity, but appropriate and well managed schedule pressure can increase their productivity up to a particular level [Sterman,

2000]. This argument is rooted in the well-known Yerkes-Dodson law [1908] from psychology, which explored how performance in various tasks depends on the level of arousal or stress imposed [Serman, 2000]. Figure 13 illustrates that as schedule pressures increases, performance rises. However, if the schedule pressure surpasses the threshold at which productivity will deteriorate, performance will decrease accordingly. In this context, when this idea is applied to real-world projects, it should be assured that the schedule pressure does not surpass the threshold at which productivity will deteriorate.

Buffer Size – Characteristics of Activities

As discussed earlier, the size of the contingency buffer has been usually assigned uniformly, relying on an individual's intuitive experience. This uniform assignment may not be effective enough to protect the planned performance. For example, a lesser buffer size than what is needed may not be enough to protect uncertainties, while a greater buffer size than what is needed can hinder a smooth workflow between phases. In this context, the size of buffers should be determined by a careful approach to reduce uncertainty while maintaining a smooth workflow.

For this purpose, Peña-Mora and Li's overlapping framework for construction (2001) is utilized as a base framework to determine the size of buffers, in particular, in concurrent design and construction. The following sections introduce key determinants of the buffer size in reliability and stability buffering, applying and extending Peña-Mora and Li's overlapping framework.

Production Rate

Production rate describes the pattern of an activity work progress such as fast and slow production¹, as seen in Figure 14. For example, *fast production rate* means that there is no need for significant preparation for the activity to achieve a normal production rate. In other words, a greater amount of work is performed at early stages and then it slows down in the later stages with a non-linear shape [Eppinger and Krishnan, 1992]. On the other hand, *slow production rate* denotes the reversed; at the start of the activity, significant preparation time is required for the activity to achieve a stable and normal production rate, as illustrated in Figure 14.

Production rate plays an important role in determining the size of buffers. Suppose a final design activity (60 days) and a shop drawing activity (45 days) are performed concurrently, such as Start-to-Start (SS) relationship with a lag of 30. Usually, a final design is of a slow production rate since it takes time to concretize the schematic design at early stage while a shop drawing is of a fast production rate since it can be launched without much preparation. In this case, the size of buffers in the shop drawing activity needs to be large because there are more possibilities that problematic tasks in the final design can affect the shop drawing activity in progress, as illustrated in Figure 14. On the other hand, if the predecessor (e.g., scraping) and the successor (e.g., installing temporary structure) are of a fast and a slow production rate, respectively, the size of buffers does not need to be maximum because the chance that problematic tasks in the predecessor can affect those in the successor in progress is minimized. Thus, it is concluded that

¹ There are also other types of production type such as a linear and step shape.

only when a predecessor is of a slow production rate or a successor is of a fast production rate, the size of buffers would need to be increased.

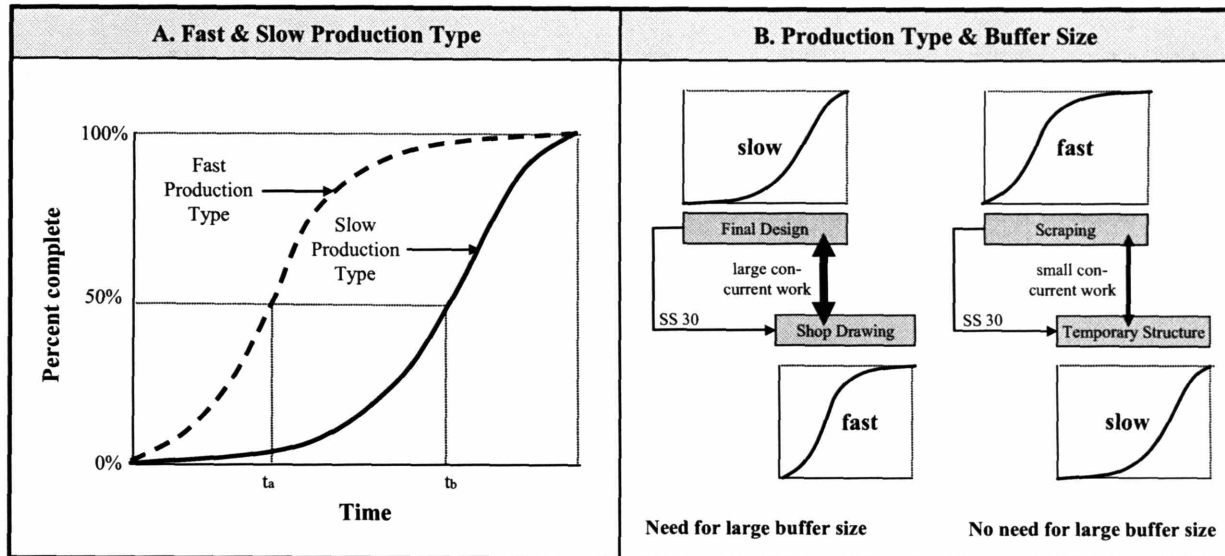


Figure 14. Example of Production Type [Adapted from Eppinger, 1997 as referred by Peña-Mora and Li, 2001] and Their Impact on Buffer Size

Reliability and Stability

Reliability and stability are defined to represent an activity's robustness against uncertainties. As discussed earlier, reliability reflects an activity's robustness against error, and stability reflects the robustness against change. Regarding the issue of the size of buffers; low reliability and stability in the predecessor activity would need a maximum reliability and stability buffer size in the successor activity because more errors and changes can lead to problems in the successor activity. On the other hand, high reliability and stability would permit a minimum reliability and stability buffer size in the successor activity.

Sensitivity

The other determinant on sizing buffers is *sensitivity*, which is an amplifier variable to capture the degree of inter-dependency [Eppinger, 1997]. Thus, a sensitive activity is more vulnerable to errors and changes than an insensitive activity. In general, if a successor activity is sensitive to the predecessor activity, the maximum reliability and stability buffer to protect them is needed. For example, a shop drawing activity can be assumed to be externally, very sensitive to the final design activity (most work in the former is specified from work in the latter). Thus, changes in a final design tend to change the shop drawings as well. In this case, the reliability and stability buffer can loosen the linkage between two activities, and large reliability and stability buffer size is needed due to their high inter-relationship.

Latency

In practice, a crucial problem is that the remaining work often needs to proceed without realizing the existence of the predecessor's hidden errors and latent changes (i.e., *latency*). This is due to the fact that the applied quality or scope management techniques or system may not be executed perfectly. In addition, tight schedules place pressure on quality and scope management, which may hinder the prompt identification of errors and changes. In such a case, a large reliability buffer size is needed, in order to coordinate with predecessor activities in identifying and solving hidden errors as early as possible. Thus, if there is a high possibility that errors and changes can be hidden or latent (i.e., low quality and scope management thoroughness), a large buffer size needs to be assigned comparing with high quality and scope management thoroughness.

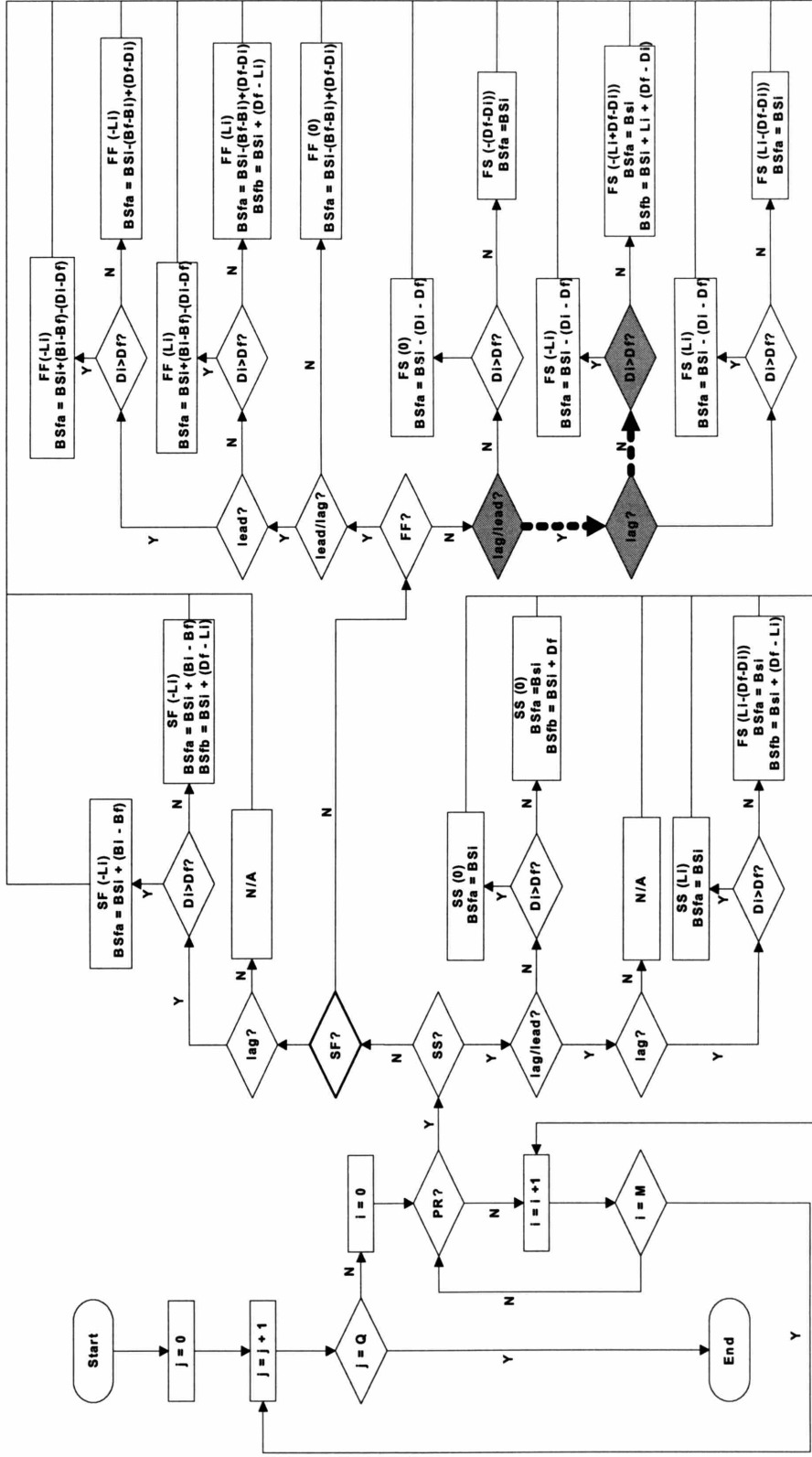
Simulation Approach for Sizing Buffers

So far, core determinants for the size of reliability and stability buffers have been identified by focusing on the characteristics of activities. In order to get the actual reliability and stability buffer size, a strategic *simulation approach* is adopted in this research. This is because effective reliability and stability buffer sizing needs to consider other design and construction conditions, such as the precedence relationships and resource constraints, as well as the changing behavior of involved determinants over time (e.g., reliability, one of the core determinants for the size of buffers, varies over time depending on many other factors, such as workers' learning curve effect, which shows reliability could increase over time as workers get accustomed to perform a given task). In this context, closed form solutions for sizing reliability and stability buffers are difficult to achieve and even unknown, and thus, a simulation approach is used, which provides a systematic way in determining the size of buffers by considering characteristics of activities and their dynamic nature.

Continuous Buffer Update – Dynamic Nature of Construction

Another important point in reliability and stability buffering is its support for the dynamic updates of buffers in order to reduce the impact of schedule deviations from the initial plan. As a project is being executed, the actual schedule performance usually differs from the planned schedule. As a mechanism to minimize the impact of this schedule deviation on the remaining design and construction performance, reliability and stability buffering location and size are *dynamically updated*, based on information obtained from monitoring the actual performance.

By updating the location and the size of buffers, a project schedule is also adjusted accordingly, for example, by changing the activity duration and the precedence relationships. The flow chart in Figure 15 illustrates how the schedule is updated. For example, the updating procedure begins with becoming aware of the existence and type of precedence relationships such as Start-to-Start (SS), Start-to-Finish (SF), Finish-to-Start (FS), and Finish-to-Finish (FF), if any. Then, the existence of lag or lead is investigated and lastly, new precedence relationships, with lead and lag information, is provided based on the updated buffer location and size.



*** Note**

- Q:** Number of Activities in the Project
- Di:** Initially Planned Duration
- Df:** Forecasted Duration
- Li:** Initially Planned Lead or Lag

- M:** Number of Predecessors of an Activity
- Bi:** Initially Planned Buffer Size
- Bfa:** Forecasted First Buffer Size
- Bfb:** Forecasted Second Buffer Size

- PR:** Precedence Relationship
- BSfa:** Forecasted First Buffer Start Time
- BSfb:** Forecasted Second Buffer Start Time
- BSi:** Initially Planned Buffer Start Time

Figure 15. Schedule Update by Dynamic Buffering

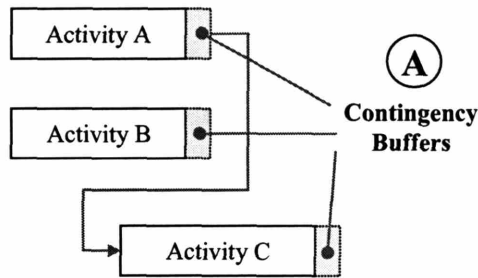
4.4 IMPLEMENTATION OF RELIABILITY AND STABILITY BUFFERING

So far, the fundamental idea of reliability and stability buffering has been discussed. In this section, how this buffering can be actually implemented in the construction schedule is described.

Figure 16 shows comprehensive steps for implementing reliability and stability buffering in the schedule. Specifically, contingency buffers (A in Figure 16) would occur at the end of each activity with a uniformly assigned size in traditional buffering practice. Then, these buffers are *taken off and combined* as the sum of buffers (B in Figure 16) creating a pool of buffers (B in Figure 16). Finally, they will be *re-located* (i.e., proactive or split), *re-sized* (i.e., activity characteristics-based), and *re-characterized* (i.e., supportive to coordination and pre-planning) as a reliability and stability buffer (C in Figure 16).

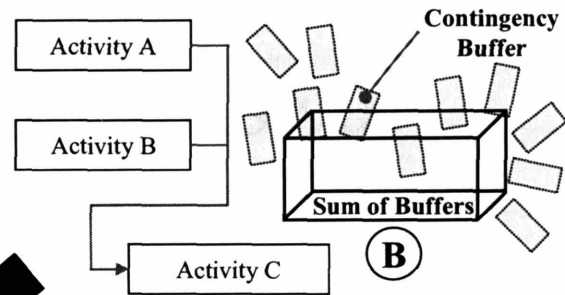
In addition, there can be some situations that a proactive reliability and stability buffer can not handle all the issues thereby delaying the target date. In this case, if there are any remaining buffers in the sum of buffers, they are used as a *path protection buffer* (D in Figure 16), which is located at the end of construction path, such as critical path or near critical path, in order to absorb a chain of delay. In most cases, the size of a reliability and stability buffer is less than traditional contingency buffers due to the consideration of schedule pressure. Thus, it can be summarized that reliability and stability buffering flexibly uses buffer location to protect the schedule.

1. Traditional Buffering Practice



2. Taking off & Combining

[Adapted from Park and Peña-Mora, 2004]



3. Re-Adjusting

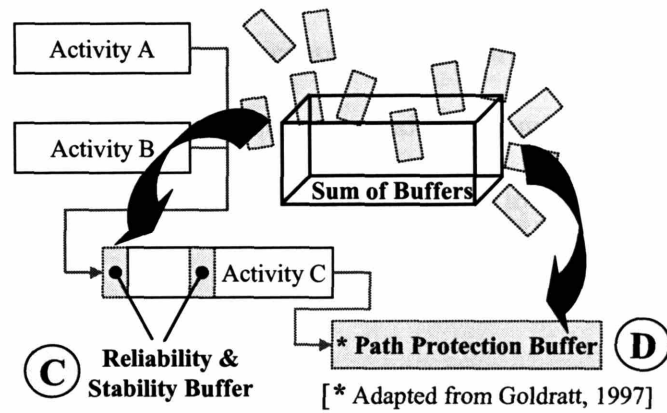


Figure 16. Reliability and Stability Buffering Implementation Steps in Schedule [Adapted from Goldratt, 1997; Park and Peña-Mora, 2004]

CHAPTER 5

SYSTEM DYNAMICIS-BASED SIMULATION MODEL

In the previous chapters, how errors and changes dynamically affect construction performance and how they can be managed have been discussed, particularly in concurrent design and construction. Based on these concepts, this chapter introduces how we can model and simulate this dynamic construction process and management mechanism to analyze the impact of errors and changes. .

Network-based tools, such as the CPM/PERT/PDM network (Critical Path Method, DuPont Inc. and Remington Rand, 1958; Program Evaluation and Review Technique, US Navy, Booz-Allen Hamilton and Lockheed Co., 1958; Precedence Diagramming Method, IBM Co., 1964), have been widely used in the A/E/C industry. However, they do not explicitly take into account errors and changes and do not work well when construction is heavily constrained by either time or resources under a dynamic environment [Hegazy, 1999]. Also, they inherently utilize a static approach that may provide users with unrealistic estimations. Thus, they may have difficulty capturing dynamic feedback processes caused by errors and changes.

As an alternative to static network-based tools, simulations have been developed to address uncertain characteristics of construction. Simulations have significantly contributed to the construction decision making process by providing ‘what-if’ scenarios. Discrete Event Simulation (DES) has been one of the primary means of simulation, focusing on construction operational details. Considering the similarity between construction operations and queuing theory, DES, which is good at representing queuing theory, would be an appropriate method to represent construction operations.

However, there is a strong need to apply simulation to *high-level strategic decision making beyond construction operations*. Based on the analysis of 3500 projects, Morris and Hough (1987) reported that lack of strategic analysis is a major reason for the failure of many projects. Considering the complex interrelationships between processes, subcontractors, resources, etc. in a construction project, obtaining closed form solutions and determining appropriate policies are difficult, and strategic simulation, which aims to understand the system as a whole, could be a prominent means to get through this complex situation. Particularly, understanding of the impact of errors and changes need to be approached in this way because the decision making process that errors and changes usually accompany should be done based on the understanding of the whole structure of the project. No good policies can be made without its understanding. Thus, the use of simulation for high-level strategic decision making process requires a holistic approach. As such, DES models, based on reductionism and randomness, are difficult to use to set reliable policies for making managerial decisions [Martin and Raffo, 2001].

In an effort to address this issue, System Dynamics (SD) is proposed as a complementary tool to DES in the strategic decision making process regarding errors and changes. SD was developed in the late 1950's to apply control theory to the analysis of industrial systems [Richardson, 1985] and has been applied to complex industrial, economic, social, and environmental systems of all kinds [Turek, 1995]. In particular, SD's emphasis on system structure can greatly contribute to this strategic decision making process identifying how system structure generates dynamic and complex behavior during actual execution.

5.1 LITERATURE REVIEW

Halpin (1977) first introduced a simulation technique, CYCLONE, into construction, and Paulson (1983) has developed INSIGHT extending the modeling capabilities of CYCLONE with an interactive user interface. After that, significant research efforts have been made in simulation, and one of the example would be STROBOSCOPE [Martinez and Ioannou, 1997], which uses highlights the dynamic state of construction, particularly focusing on the operations in detail.

All the research that was mentioned is based on DES. After realizing the importance of the need for a strategic approach in simulation, significant research efforts in SD have been undertaken to address some of the dynamic issues, emphasizing the impact of the rework cycle on project performance [Cooper, 1980; Richardson and Pugh, 1981; Abdel-Hamid, 1984; Ford and Serman, 1998; Lyneis et al, 2001]. For example, Cooper (1980) identified the detrimental impacts of design changes on schedule and cost in the naval ship production process. In addition,

Abdel-Hamid (1984) found that a gap exists between real progress and perceived progress which results from overlooking errors in the early stages, causing poor performance in software development projects. Ford and Sterman (1998) highlight process dynamics (e.g., process concurrence and inter and intra-phase changes) for effective project management, exemplified in a semi-conductor chip development project. Lyneis et al. (2001) applied system dynamics modeling to implement strategic management in projects, showing that it reduced the potential impacts of the rework cycle in the military air defense system. In the construction arena, Park and Peña-Mora (2003) have developed a model that emphasizes the unique characteristics of construction (e.g., when errors and changes are introduced during construction, physical constraints make managers tend to prefer 'extra work' to 'rework,' since rework in construction is normally accompanied by the demolition of what has already been built).

5.2 WHY SYSTEM DYNAMICS?

In this research, SD has been used as a primary simulation mean. The following sections briefly talk about the advantages obtained from the use of SD over DES.

Feedback Process

The uncertainty and complexity of construction projects are usually driven by *feedback processes*. Understanding the feedback process is particularly important in the strategic decision making process because good policy decisions come from understanding the system. The main

idea behind SD is that dynamic and complex behaviors are derived from system structure [Richardson, 1985]. Compared with DES's emphasis on input randomness, SD's emphasis on the system structure gives great strength to the understanding of the system. This enables to make good policies and eventually, facilitates the strategic decision making process in project management [Williams, 2002].

Aggregation

When a simulation model is developed for high-level strategic simulation, modeling the whole world in detail is not necessarily a goal. Rather, if the model can accurately represent and explain the success or failure of a certain policy, it could be enough for that purpose. If the detailed analysis is necessary, the model can be more specified based on the user's needs. This is also related to the model's economics. In this sense, *aggregation* is an important requirement for high-level strategic simulation.

One of the features of SD is the ability of the aggregate representation of the world. For example, the stock and flow structure, which is a core model structure in SD, can represent the aggregate behavior. This aggregate representation can contribute to the understanding of the overall system, which in turns, enables to make good strategic decisions. Once an overall understanding is developed, a detailed decision can be supported by adjusting the level of aggregation in SD.

Soft Variables

In most construction simulation, the majority of variables are *hard* variables that are available as quantitative metrics and numerical data. However, most of what we know about the world is descriptive, impressionistic, and has never been recorded [Stermann, 2000]. For example, though the worker's fatigue has rarely been used in construction simulation, it certainly plays a crucial role in workers' productivity. Thus, *soft* variables, such as goals, perceptions, and expectations, are significant in representing the world. Particularly, in strategic construction simulation, soft variables become more important because some policies are derived from these soft variables. That's why SD encourages the use of soft variables in modeling of the strategic decision making process. Suppose an overtime policy can be allowed when the project is behind schedule. In this case, the issue would be how and when the project manager actually implements the overtime policy. In the manager's mental model, when the schedule is behind, the project manager perceives the schedule pressure and this is the moment when the manager triggers an overtime policy¹. Further, based on the level of the perceived schedule pressure, the degree of overtime (i.e., overtime ratio) and corresponding work hours will be determined. Likewise, the wide use of soft variables in SD makes us understand how a policy can be implemented and further, how it can affect construction performance, which in turn, contributes to determining a good policy.

¹ Specifically, when the schedule pressure reaches a certain threshold, overtime will be triggered.

Simulation Clock

Simulation clock is another important issue in strategic simulation. DES models recalculate their state variables only at a discrete set of points, referred to as 'event times' [Banks et al., 2001] assuming that nothing happens in the intervals between event times. Thus, intervals usually have an inconsistent step size, and this can cause unrealistic representations in strategic simulation [Han et al., 2006].

Continuing with the previous overtime example, suppose that a construction manager made a policy that overtime will be adopted when the schedule pressure reaches 1.2. In DES models, since the overtime ratio is only recalculated at each event time, a managerial action instigated by the calculations can also only be adopted at the event times. As illustrated in Figure 17, since there is no event time at the precise moment the overtime ratio reaches the threshold (i.e., 1.2), the planned managerial action (A in Figure 17) will not be triggered. Instead, the managerial action will be implemented at even time 3, when the overtime ratio has reached 1.37 (B in Figure 17). As such, the managerial action is adopted later than it should have been. This implies that there is a higher risk of process interruption from work hour shortage than the construction manager expected. As a result, inconsistent step size will make the policy for managerial actions less reliable.

On the other hand, SD models recalculate variable states with *consistent time intervals* that the user chose and thus, could be more responsive to a change in the project environment than

DES models by decreasing time step size. The use of a small and consistent time step size avoided the unrealistic representations of triggering policies that may happen in DES models.

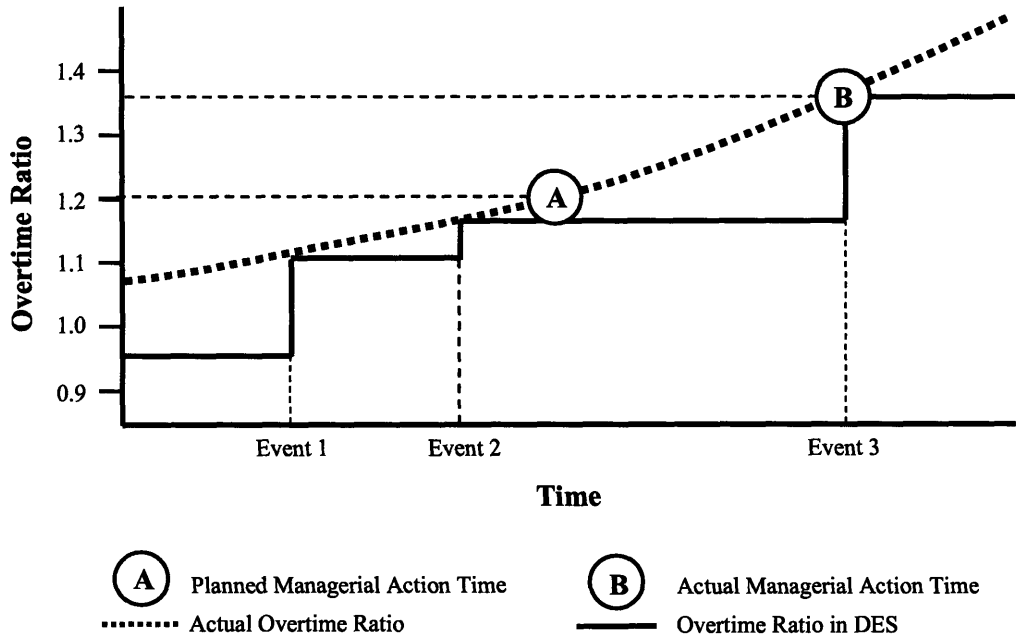


Figure 17. Inconsistent Time Step Size in DES [Adapted from Han et al., 2006].

5.3 DYNAMIC DESIGN AND CONSTRUCTION PROJECT MODEL

Despite SD's advantages to strategic simulation, it has not been widely adopted in the construction industry. In addition, as discussed in the previous literature review section, though SD contributed to an understanding of the rework cycle for effective project or product management, it is not common to find SD's application to construction. One of the main reasons is its disregard of the established concepts in practice. Focusing on this issue, the Dynamic Design and Construction Project Model (D²CPM) is proposed as a means to integrate the

fundamentals of network-based tools, which are the de-facto standard of project management modeling in the A/E/C industry [Senior and Halpin, 1998]. Thus, the current practice in the A/E/C industry can be kept, and at the same time, new features for better understanding of the dynamics of errors and changes can be added to traditional network-based tools. In addition, this model focuses on the role of additional work scope generated by rectifying errors and implementing changes as a source to disrupt the whole construction, as discussed earlier in the error and change management framework. Considering the fact that construction plans are developed based on an estimated initial work scope, adding work scope during actual construction may be very disruptive, corrupting procurement, resource allocation, man-power loading plans, and so forth. Further, when the time given to account for consequent changes is insufficient (e.g., concurrent design and construction), the impact can be even more severe. In this sense, this model aims to understand the detrimental impact of errors and changes on construction, highlighting additional work scope as a source to generate complex dynamics.

Identifying Feedback Processes

To investigate the reasons for the system behavior identified in the error and change management framework, a causal loop diagram that represents feedback processes caused by errors and changes is developed, as seen in Figure 18. In other words, this causal loop diagram can be understood as an intermediate medium that converts the error and change management framework to a working SD simulation model.

For example, one of the important inferences from the framework is that non-adjusted scope caused by hidden error and latent change may play a significant role on the deterioration of reliability and stability of an activity. The detailed procedure can be explained using the feedback process illustrated in Figure 18. First, Link A in Figure 18 shows that a hidden error can play a role on deteriorating the reliability of an activity and consequently, increase the number of errors contained in the work produced in an activity. Suppose we are installing piles. If string lines that aim to align piles are misplaced and not discovered immediately, this hidden error may also generate subsequent errors because workers may install piles according to the misplaced string lines. Second, Link B in Figure 18 shows that latent change can play a role on deteriorating the stability of an activity and consequently, increase the number of changes that may be incorporated in an activity. Suppose that an unexpected site condition triggers a change in the construction method, and this change was assumed to be the best way to deal with the differing site condition. However, suppose it was rejected by the Claim and Change Management (CCM) Group, and the originally planned method was forced into the project personnel to be used, due to the perceived increase in budget by the adoption of the proposed change. However, the originally planned method may fail to handle the differing site condition, and even more changes could be required in order to maintain the original method. Therefore, a significant increase in costs and efforts that is much more than the adoption of the proposed change may be paid for maintaining the original method. This situation has been often observed in practice, and Link B in Figure 18 illustrates it.

In addition, a hidden error could play a role in deteriorating the stability of an activity because it can later introduce changes into an activity, as denoted by Link C in Figure 18. In the

piling example, suppose that the misplaced piles were not found in the quality management process and became a hidden error. Now, suppose that the owner requested the change of building's purpose from normal office space to library space. Usually, a library requires a greater load-bearing capacity than normal office space. The CCM Group accepted the owner's request because the original structural plan had redundant dead load capacity. However, in the current project, it would be hazardous to adopt the requested change, because the misplaced piles may reduce the structural capacity of the infrastructure to bear the new planned dead load. By the time the manager notices this hidden error, the decision of the CCM Group to permit the library has already been made, so there might be subsequent changes needed to solve this hidden error. Therefore, a hidden error as well as a latent change may play a role on the deterioration of the stability of the construction process. This situation can be also applied to a latent change, which can affect the reliability of the process, as denoted by Link D in Figure 18.

The important point here is that there is a connection between the error management process and the change management process, and the connection points are adjusted scope and non-adjusted scope. Therefore, these processes cannot be dealt with separately and should be considered together in order to reduce error and change generation rate. For example, if greater reliability is required, solutions should be sought from both the error and change management processes.

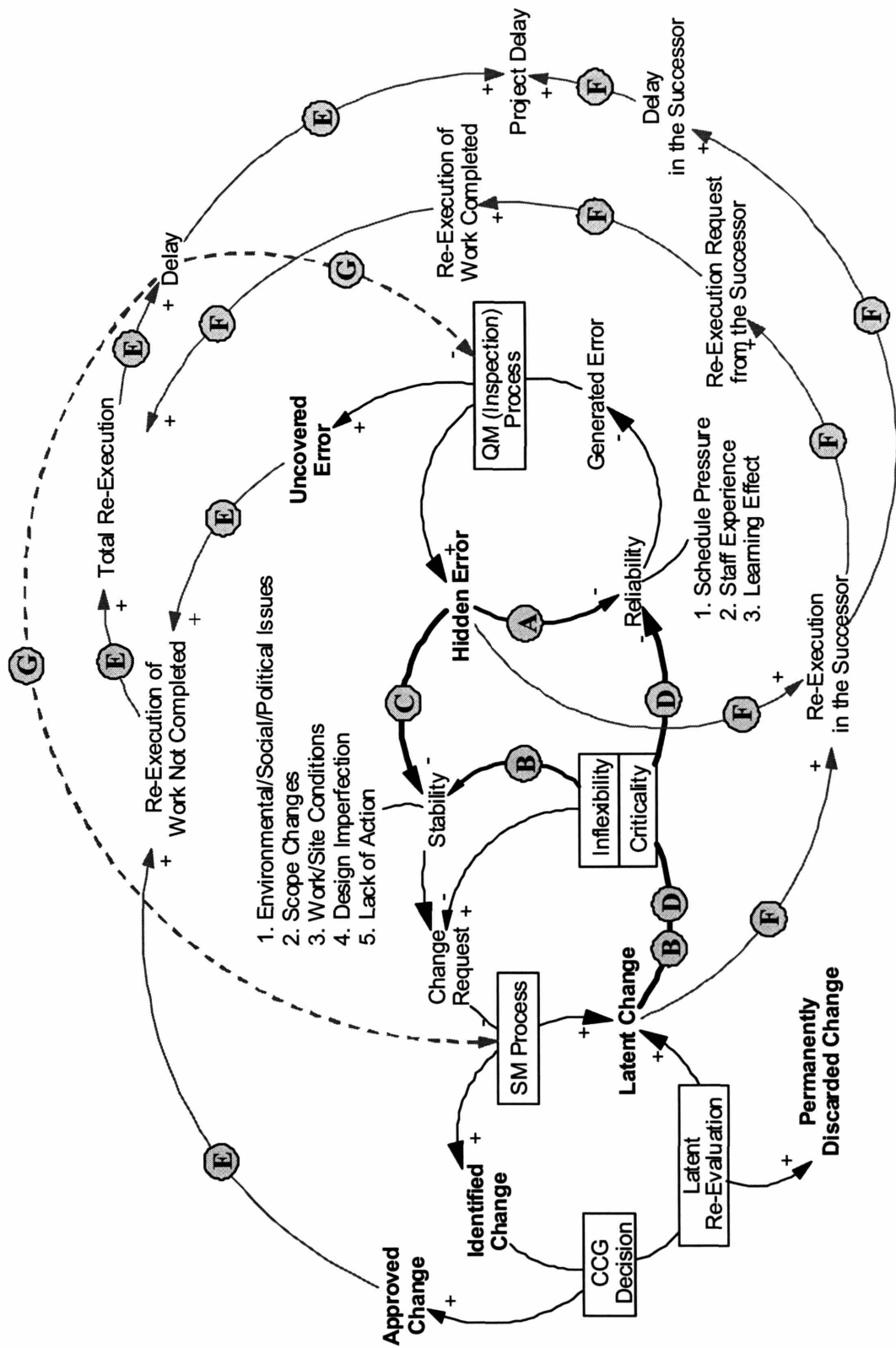


Figure 18. Feedback Processes on Error and Change Management Process

In addition, adjusted scope and non-adjusted scope finally introduce delay. Link E in Figure 18 shows delay caused by the adjusted scope that results from uncovered error and approved change. In order to deal with these perceived work scope, more time may be required and consequently, it introduces the delay. On the other hand, Link F in Figure 18 illustrates delay caused by the non-adjusted scope that results from hidden error and latent change. Delay from Link F in Figure 18 is caused by the request from successor activities because hidden error and latent change are not discovered immediately. Finally, accumulated delay may play a role on the deterioration of the quality and scope management processes, as denoted by Link G in Figure 18. This is exactly the same as the effect of schedule pressure on QM and SM, which is discussed in the Chapter 3.

Model Boundary

A clear definition of the model boundary is important in SD modeling in terms of the modeling scope and purpose. Figure 19 summarizes the primary features that are included (endogenous), assumed (exogenous), and ignored (excluded) from the model [Ogunlana et al, 2003]. Endogenous variables are the primary focus of the model and are able to be modified during the model simulation. For example, the initial work scope of an activity will be determined in proportion to a given activity duration. Then, the model simulation is performed, taking into consideration all the constraints involved in the activity processes. Examples of endogenous variables include the project progress and the fraction of hidden error and latent change. On the other hand, exogenous variables imply that they are stable during simulation. For example, given activity data, such as an initial activity duration, is not changed. Lastly, variables that are

classified as excluded are tentatively not included in the structure of the dynamic construction project model because they are beyond the focus on the current problem domain. However, once the applicability of the D²CPM is confirmed, they can be incorporated into the model in order to increase the capability to replicate real construction processes. Examples of excluded variables include multiple project development sequence, weather and seasonal effect, constraints in cash flow, and performance profiles in terms of costs, safety, and environmental impact.

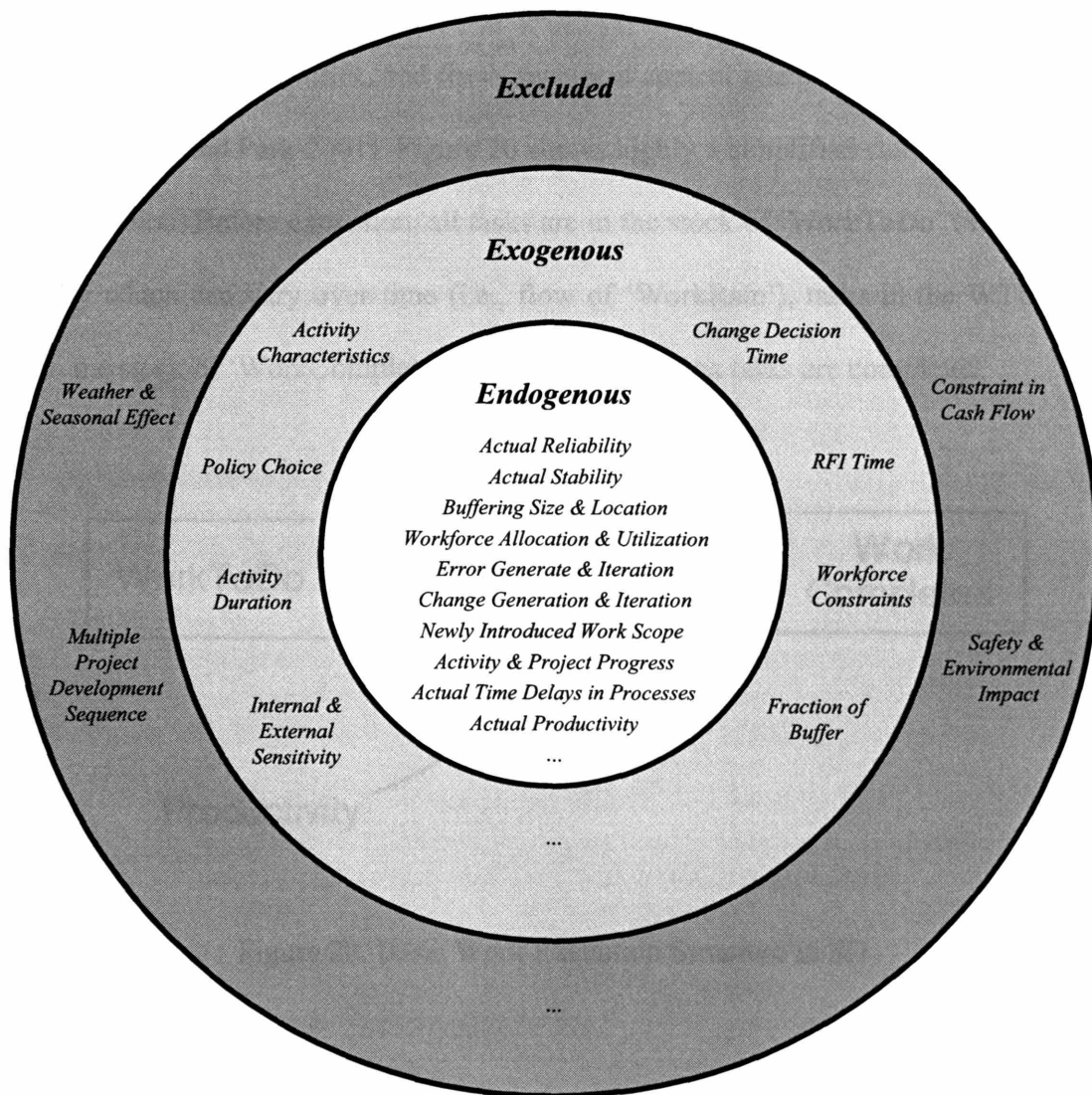


Figure 19. Model Boundaries

Basic SD Work Process Model

Figure 20 shows the basic work execution structure in SD. After conceptualizing the model structure using a causal loop diagram in Figure 18, the next step of SD modeling is model formation, which includes the identification of stock and flow structure as seen in Figure 20. Stock and flow structure characterizes the state of the system and generates the information upon which decision and actions are based, by giving the system inertia and memory [Sterman 2000]. Stocks represent stored quantities, and flows represent control quantities flowing into and out of stocks [Peña-Mora and Park 2001]. Figure 20 shows highly a simplified stock and flow structure for work execution. Before execution, all tasks are in the stock of 'WorkToDo' (WTD) Based on productivity which can vary over time (i.e., flow of 'WorkRate'), tasks in the WTD stock can move into the stock of 'WorkCompleted' (WC), which means tasks are completed.

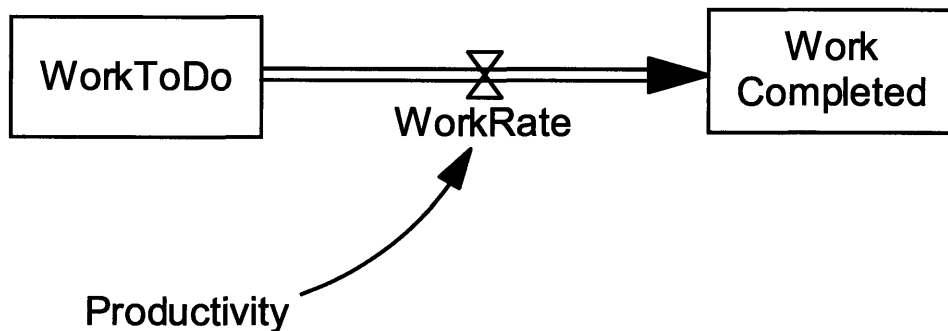


Figure 20. Basic Work Execution Structure in SD

Integration of Network-based Tools and System Dynamics Model

This basic SD model structure is integrated with network-based models such as CPM, PERT, and PDM. CPM allows the logical analysis and manipulation of a network to determine the best overall program of operation while PERT was developed to deal with uncertainty in projects by incorporating probabilities into the duration of project activities. Since the development of CPM and PERT, many network-based tools have been developed supplementing CPM and PERT. For example, PDM added different types of precedence relationships, such as Start-to-Start, Start-to-Finish, Finish-to-Start, Finish-to-Finish with leads and lags¹, between activities to the traditional CPM. Even though their rigid and static features make them difficult to represent the dynamic construction process, they have been widely used in the construction management area mainly due to their easy applicability to diverse projects [Rodrigues and Williams, 1998; Park and Peña-Mora, 2003]. In order to incorporate their applicability, the fundamentals of network-based models are integrated into the SD-based D²CPM.

In the D²CPM, concepts in network-based models can be represented as seen in Figure 21. First, productivity will be assumed to be constant in network-based models, and work constraints among activities (i.e., precedence relationships) will decide whether and when the succeeding activity can start. Thus, 'WorkIntroductionRate', which assigns the quantity of the WTD stock, is constrained by the variable 'WorkAvailable' determined by the constraints caused from precedence relationships in network-based models.

¹ If Activity A and B have Finish-to-Start relationship with a lag of 10 days, this means that Activity B can start 10 days later after the completion of Activity A.

In addition, to capture dynamic concurrency that represents work dependencies within an activity, internal concurrence [Ford and Sterman, 1998] can be adopted. Thus, ‘WorkAvailable’ can be also constrained by internal concurrence from the existing system dynamics model.

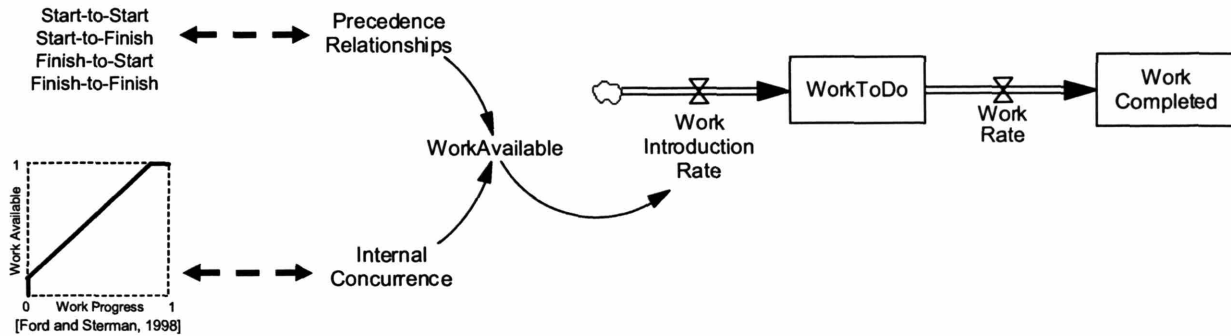


Figure 21. Modeling Work Constraints

In order to investigate whether the developed model generates the same behaviors observed in network-based models, the CPM CASE was simulated with three concurrent activities: a final design, an excavation, and a superstructure activity. Each activity has a 100 day duration and Start-to-Start relationship with a lag of 60, as seen in Figure 22. For simplicity, WU is used as a hypothetical work unit, and 1000 WU is assumed to be equivalent to 1 day’s work.

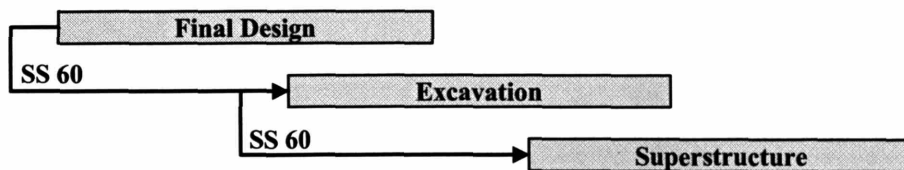


Figure 22. Activities for Simulation Experiment

This scenario follows the CPM assumptions that everything will be performed as expected with a uniform production rate. As seen in Figure 23, the CPM CASE confirms that the WTD and WC stocks are the expected straight lines and that the total duration is 220 days.

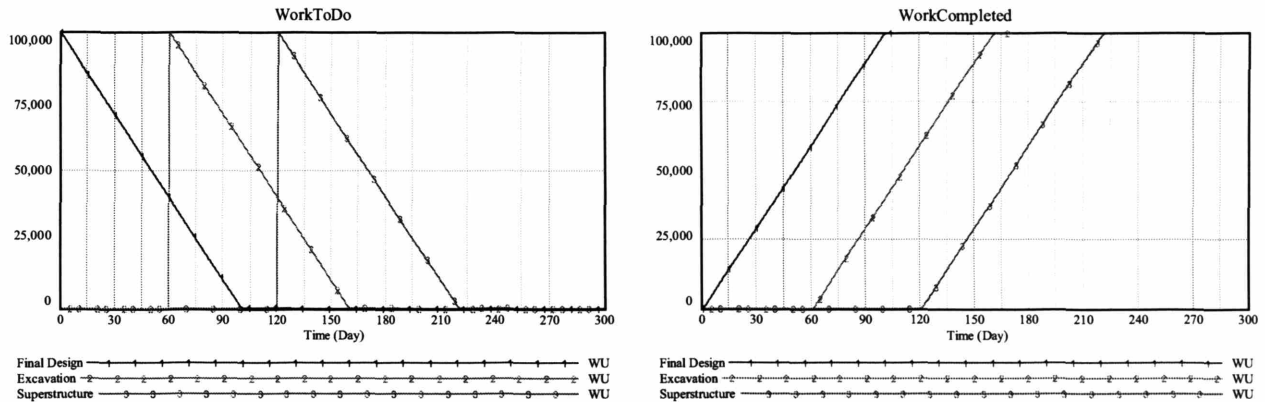


Figure 23. CPM CASE – Behavior of WorkToDo and WorkCompleted stock

Dynamic Production Rate

In this basic work execution structure, the quality management process is appended. In Figure 24, the stock of 'WorkAwaitingQM' (WAQM) is added between the WTD and WC stock in order to represent tasks completed and waiting for quality management. Additionally, dynamic production rates (e.g., fast and slow production rate) are incorporated. This was done by adding a look-up table to productivity, which controls 'WorkRate' in the work execution structure as seen in Figure 24.

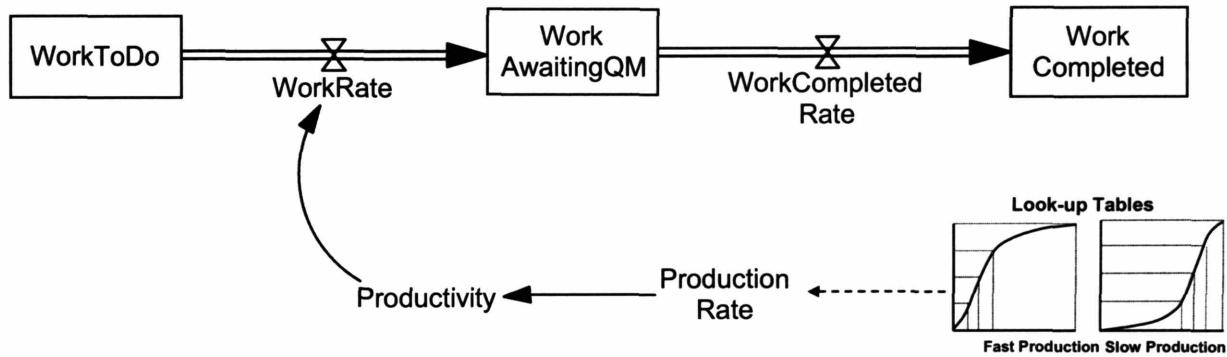


Figure 24. Dynamic Production Rate

Figure 25 shows the graphs of ‘WorkToDo’ (WTD) and ‘WorkCompleted’ (WC) stock that applied dynamic production rates (i.e., CASE 1): specifically, the final design activity for slow production rate, the excavation activity for fast production rate, and the superstructure activity for slow production rate. Unlike the CPM CASE, this CASE 1 shows that work is executed at a varying production rate.

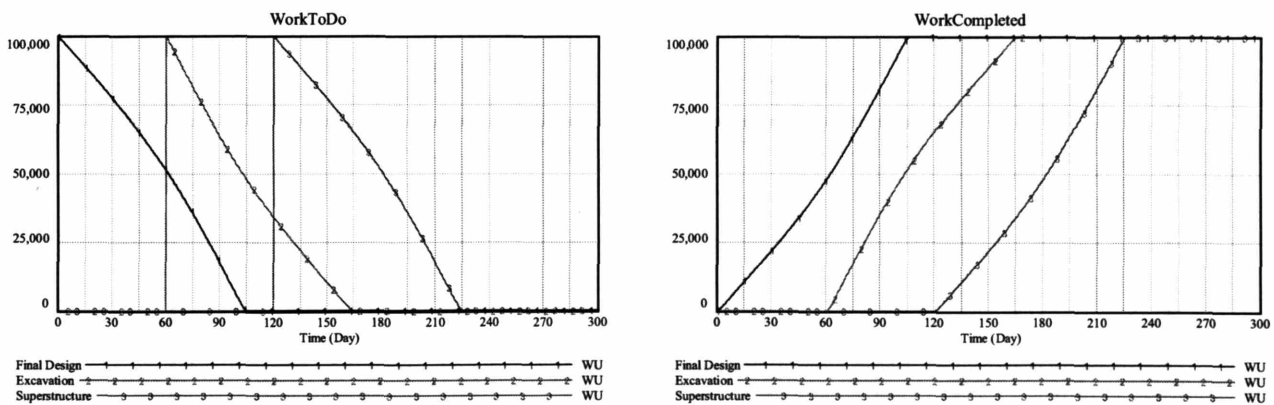


Figure 25. CASE 1 – Behavior of WorkToDo and WorkCompleted Stock

The model results, reflecting the adoption of dynamic production rates, imply that traditional assumptions do not hold true when taking into consideration the dynamic execution of what happens in reality. For example, if we consider a traditional uniform production rate, ‘WorkAwaitingQM’ (WAQM) stock (i.e., the work waiting for quality management after its execution) would be constant. However, Figure 26 shows that WAQM stock can be varied based on how work is executed. In other words, if we prepare quality management capacity based on a uniform production rate unless there is over capacity, quality management may have difficulty in dealing with the sudden work overflow for inspection, as denoted A in Figure 26. As discussed earlier, the pressure placed on inspection from work overflow is one of the main drivers to generate latency (e.g., the accelerated inspection process may cause inspectors to miss errors that they would otherwise identify).

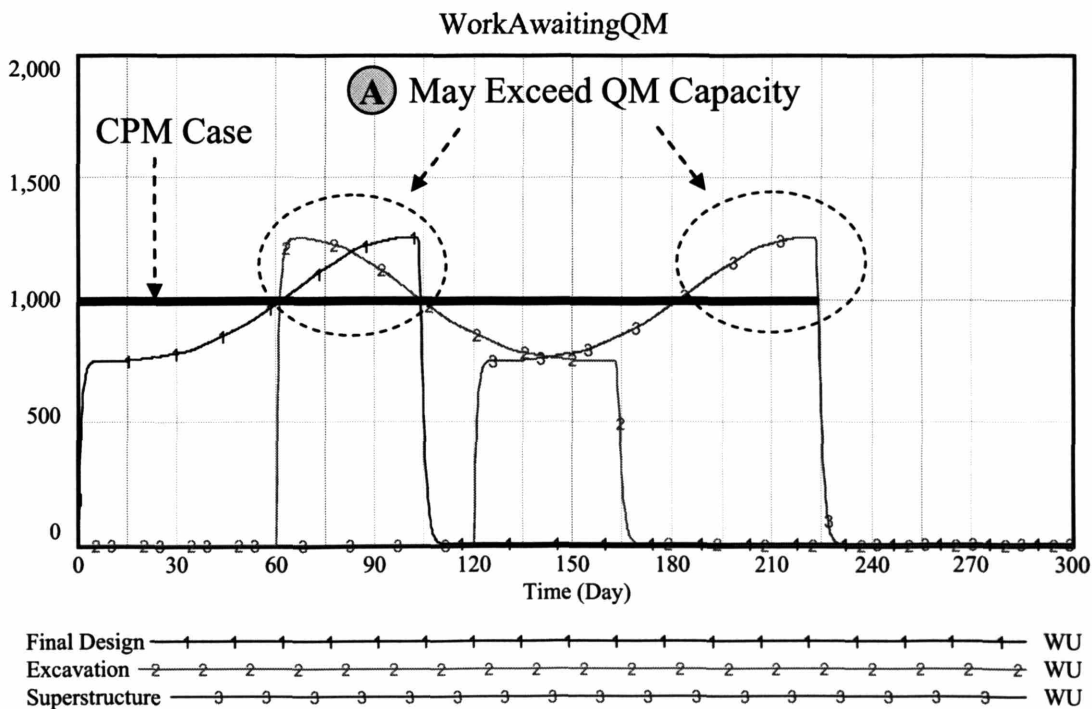


Figure 26. CASE 1 – Excess of QM Capacity

Modeling Dynamic Error and Change Cycles

Differing from the CPM assumptions, in reality there can be errors and changes. As previously discussed, errors and changes usually cause non-value-adding iterations, thereby introducing additional work scope. Concentrating on the additional work scope as the source that affects construction performance, the following sections will illustrate a model that has been designed to show the impact of errors and changes in the construction process.

Error and Change Management

The work execution associated with error discovery is modeled in Figure 27 extending the model structure in Figure 24. For example, after tasks in the WTD stock are executed with the available work rate that considers resource availability, tasks done will await Quality Management (QM). The main idea of this structure takes into account that there is no guarantee that the work will have been done correctly just because all of the work in a project has been performed. Some of the work may need to be re-executed (i.e., errors) (A in Figure 27) and this is captured by reliability (B in Figure 27). However, some portion of the errors may not be identified during the QM process and this is latency.

This latency is explained by Quality Management Thoroughness (QMTH) (C in Figure 27), the degree to which the existing quality problems of an activity have been identified during the QM process. Thus, a 'Re-ExecutionOfUncoveredErrorRate' (A in Figure 27) is governed by 'UncoveredErrorGenerationRatio' (D in Figure 27), which can be represented by reliability and

QMT. If the final design activity has 90% reliability and 80% QMTH, the actual re-execution of uncovered errors by the QM process is as much as 8% ($= 0.1 \times 0.8$) of the total work scope of the final design activity, and hidden errors are 2% ($= 0.1 \times 0.2$).

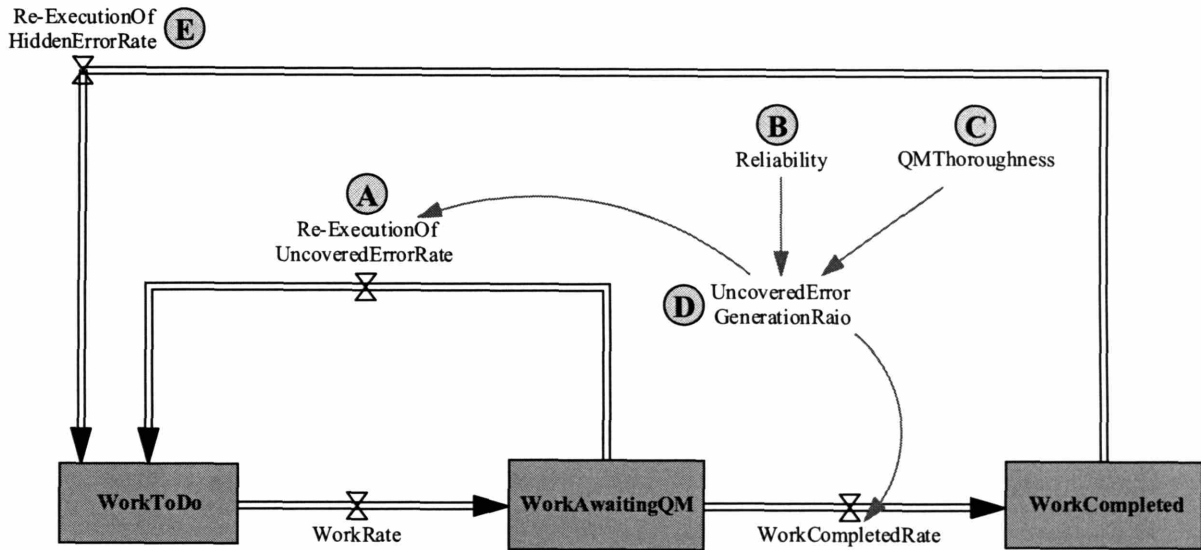


Figure 27. Work Execution with Error Management [adapted from Pugh-Roberts Associates, 1980; Ford and Sterman, 1998; Park and Peña-Mora, 2003]

On the other hand, in a case where hidden errors are identified at a later stage, there is a possibility that completed work needs to be worked again (i.e., reworked). In the model, E in Figure 27 represents this flow, and the situations will be explained in a later section.

Along with this work execution structure, a parallel co-flow structure is used to model the errors. For example, in terms of uncovered errors, the stock of ‘UncoveredError’ (UE, A in Figure 28) has an inflow, ‘UncoveredErrorDiscoveryRate’ (B in Figure 28), which can be obtained from the average error rate in the work done (i.e., ‘AvgErrorInWorkAwaitingQM’, C in

Figure 28) and the probability to uncover errors (D in Figure 28). The rest of the co-flow structure is in Appendix including the complete model structure.

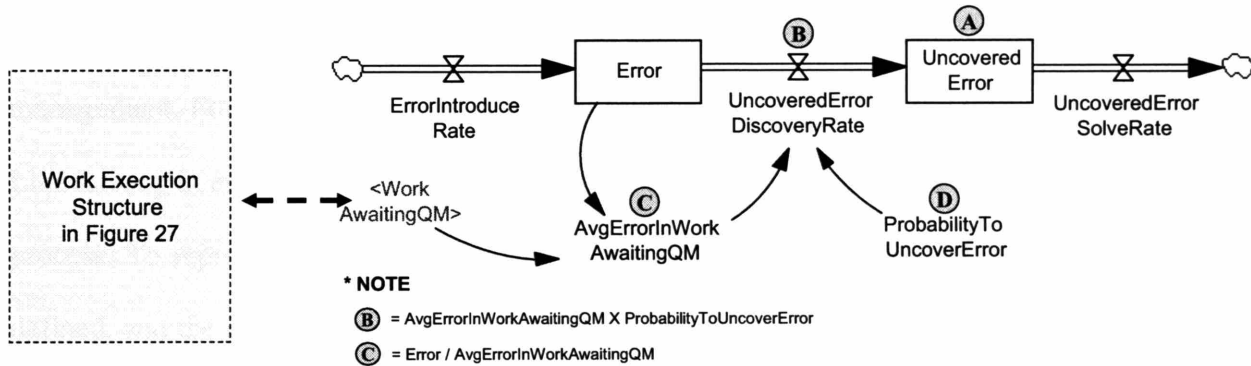


Figure 28. Co-Flow Structures for Errors

The model in Figure 27 can be extended to incorporate change (order) management. Change orders in construction are an essential mechanism for satisfying owners' construction needs throughout the project delivery process and responding effectively to errors in construction [Moselhi et al., 2005]. However, at the same time, they cause detrimental impacts on construction performance. Suppose the owner requests the change in space usage from normal office to document storage in steel frame office construction. Usually, document storage requires more structural capacity to bear the load. In this case, braces could be added to increase structural capacity. However, these braces could generate other issues which were not expected. For example, adding braces could request re-designing of the original interior design, which could accompany other changes such as furniture procurement, HVAC (Heating, Ventilating, and Air Conditioning) systems, and so forth. Also, schedule, cost, and the resource allocation

plan could be consequently changed. Thus, modeling change management is particularly important in understanding its impact on project performance.

As seen in Figure 29, change management can be modeled similarly to the model of error management. For example, instead of reliability, change is related to stability (A in Figure 29). Like latency in errors, latency in changes can occur, at this time, in the Scope Management (SM) process. To represent latency, Scope Management Thoroughness (SMTH) (B in Figure 29) is defined, and the changes can be modeled using a parallel co-flow structure, similar to the model for the errors in Figure 28.

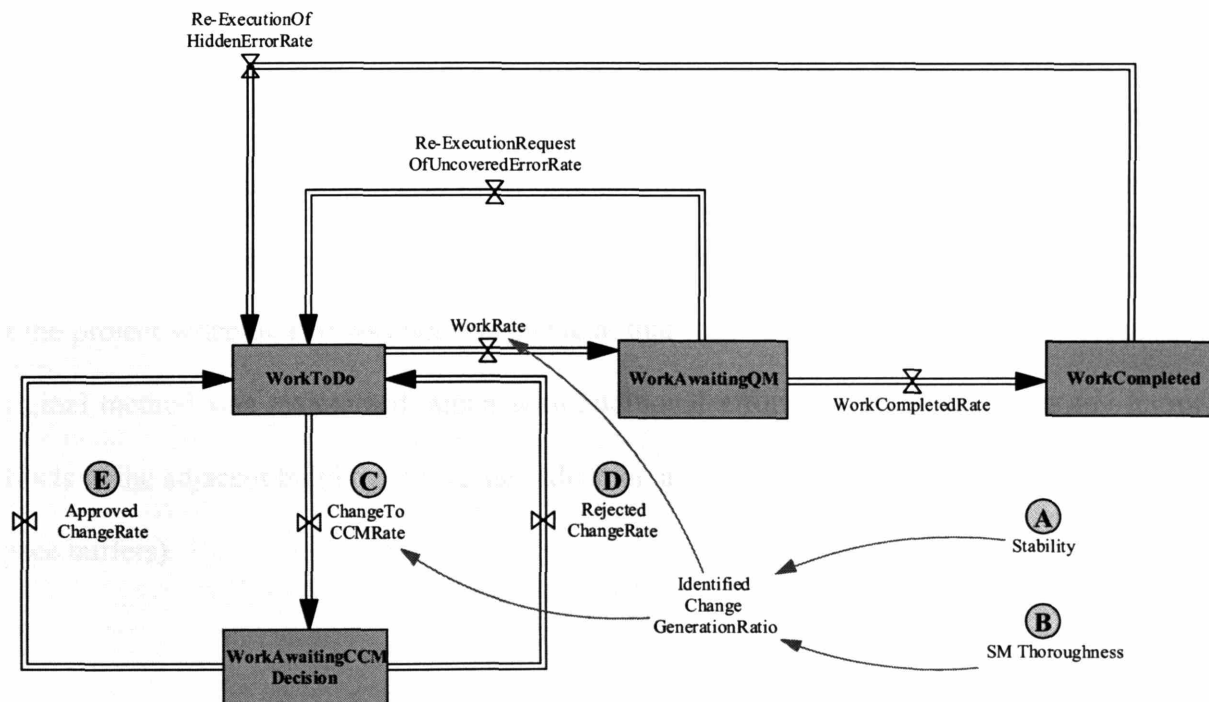


Figure 29. Work Execution with Change Management

One of the important processes in managing changes is the Claim and Change Management (CCM) process, the decision making process that determines the adoption of changes. In the model, if changes are identified, related tasks in the WTD stock are sent to the stock of ‘WorkAwaitingCCMDecision’ (WACCMGD), which needs the decision or analysis by the CCM Group. One of the major factors to affect a change decision is the feasibility of the raised issue.

For example, a blasting method (i.e., dynamite - A in Figure 30) was planned for certain excavation work. However, a closer investigation found that by utilizing the blasting method, damage could occur to adjacent buildings, due to the existence of a softer ground than expected. The usage of the scrape-and-excavate method (i.e., excavation with backhoe loaders – B in Figure 30) was requested as a change, and the request was sent for review to the CCM process (C in Figure 29). However, the CCM Group rejected this change on the grounds of its feasibility. It was determined that it would take more time to complete the job with the scrape-and-excavate method than with the blasting method, and the CCM Group felt that this would generate a delay of the project schedule that was not acceptable at that time (D in Figure 29)¹. In this context, the original method was maintained, albeit with additional effort expended to reduce any harmful effects to the adjacent buildings (e.g., the utilization and placement of additional safety braces or space buffers).

¹ Otherwise, it will be approved and back to the WTD stock to be performed, as denoted by E in Figure 27.

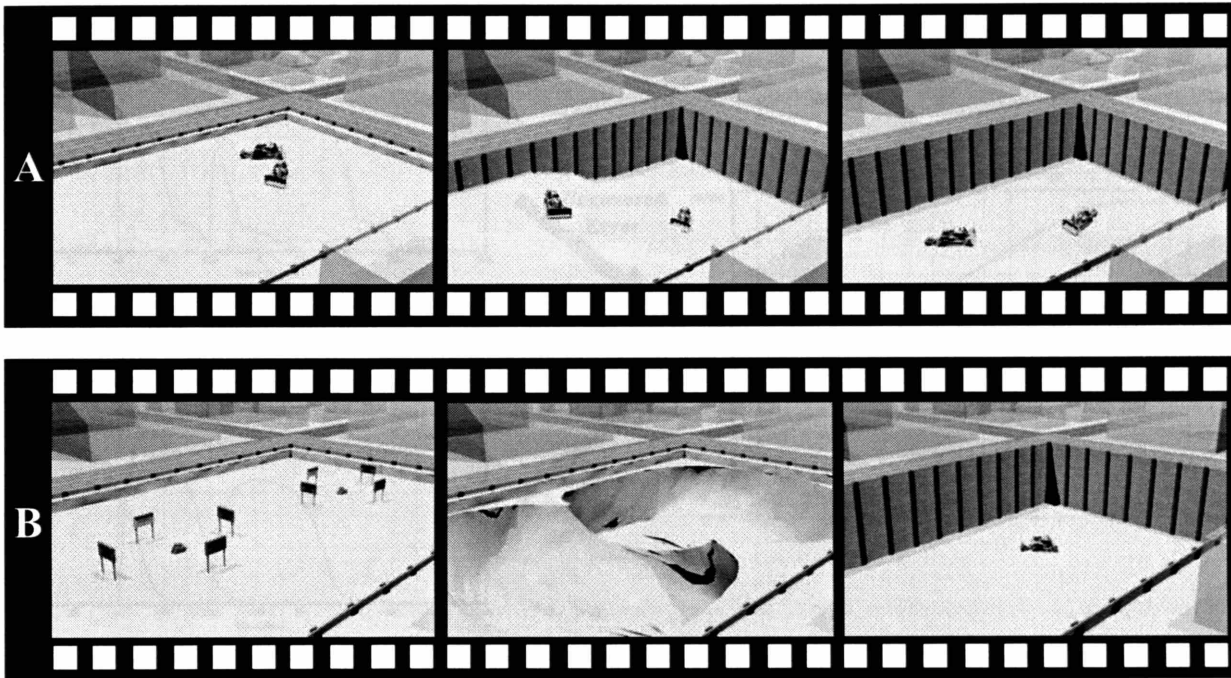


Figure 30. Illustration of Two Different Excavation Method

With this model, CASE 2 is simulated applying 85% of reliability and stability into the previous three activities. As seen in Figure 31, introducing errors and changes generates additional work scope to deal with them. If we exclude control policies such as overtime and hiring new workers, this additional work scope will delay the completion time. Furthermore, if we assume CCM Period (i.e., the time that the CCM process takes) as 7 days, the corresponding work can be delayed by as much as 7 days until the decision is made. This could contribute to the more delay of work execution.

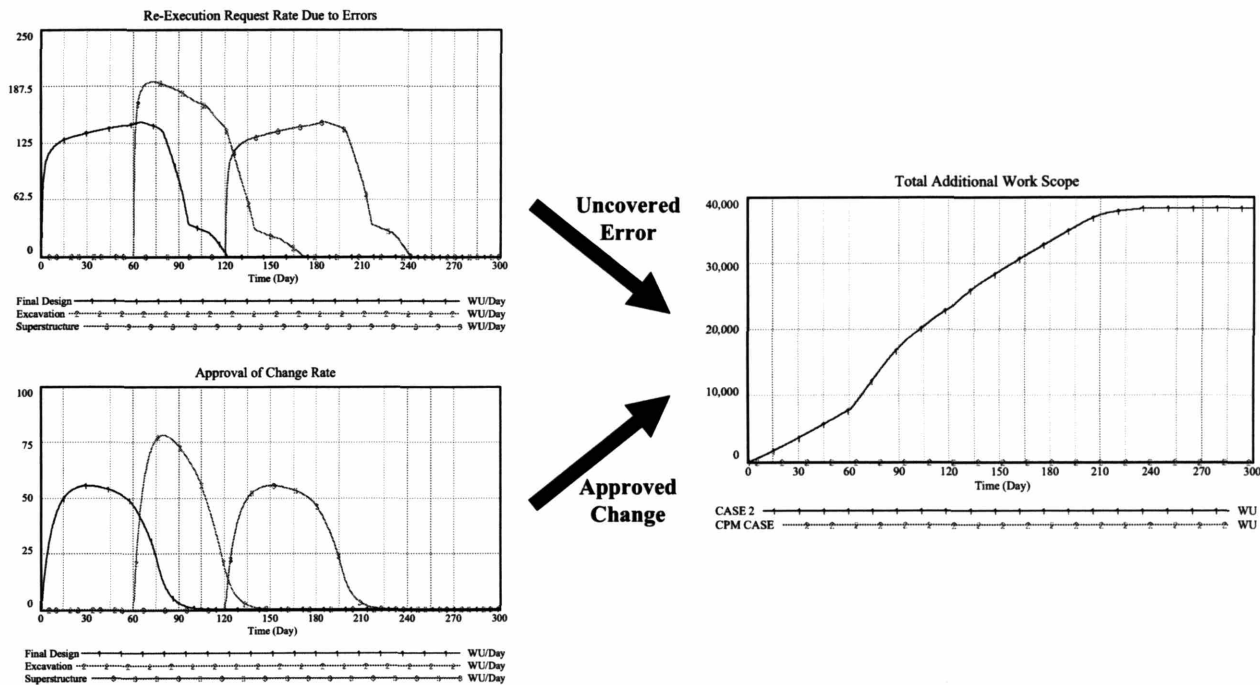


Figure 31. CASE 2 – Additional Work Scope Caused by Errors and Changes

Latency Settlement and Additional Work Scope

Errors and changes not identified or completely discarded (e.g., hidden errors and latent changes) may involve another process that coordinates them with other activities. An example of one of these processes widely used in design and construction is the Request For Information (RFI) process, which coordinates design issues between the design and the construction team. In this paper, the RFI process is extended and used to incorporate coordination with other preceding construction activities, as well as with design activities. This is because the work clarification (e.g., design error or change) process often necessitates the involvement of the preceding construction activities. Such latency and its settlement through RFI are particularly important in

concurrent design and construction. In concurrent design and construction, there is a higher possibility that errors and changes become latent than a sequential development. This is because of incomplete information from preceding activities [Tighe, 1991], the information supply problem resulting from overloaded workers [Chachere et al, 2004], and the accelerated decision making process [Lee et al., 2005]. Thus, the understanding of and modeling about latency and its settlement process is significant for successful error and change management, particularly, in concurrent design and construction.

In the model, if a hidden error or a latent change of the predecessor activity is found at the activity under study, it is passed through the RFI process, back to the predecessor activity. This is illustrated in Figure 32. The tasks identified as hidden errors and latent changes are accumulated in the stock of 'WorkAwaitingRFIReply' (WARFIR) (A in Figure 32).

At this stage, three options are available for these pending tasks. The first option is that the activity under study requests the predecessor activity to correct the hidden error or to issue a latent change (B in Figure 32). This case would arise because the origin of the hidden error and the latent change is the predecessor activity and the activity under study asks the predecessor to determine which actions could be the best way to deal with them. In this case, the associated tasks accumulate in the stock of 'WorkPendingDueToPredecessor-Change' (WPDTPC); they would thus be sent back to 'WorkToDo' (WTD) and performed after being corrected or after having changes issued in the predecessor activity (C in Figure 32).

The second option focuses on hidden errors that are sent back to WTD (D in Figure 32). A hidden error may be determined to be absorbed by the activity under study. Consider that a column erection activity is being performed after the predecessor piling activity. However, it is revealed that the location of the piles were positioned incorrectly and found as a hidden error of the predecessor activity. In this case, the manager may decide to keep the misplaced piles and proceed to install columns without requesting any correction to the predecessor activity because structural stability would be maintained and because of the lack of sufficient time necessary to correct them. Thus, tasks would proceed to the activity under study, but no action is taken to the predecessor activity.

The third option deals with latent changes, which directs the flow into the stock of WACCMGD (E in Figure 32). If the manager judges that a latent change needs urgent attention since it is significant to the whole project performance, the manager may request a decision from the CCM Group directly, rather than by way of the predecessor activity. Thus, 'WorkAwaitingCCGMDecision' (WACCMGD) actually has two inflows: one from the WTD stock (F in Figure 32), which handles the identified change and one from 'WorkAwaitingRFIRReply' (WARFIR) (E in Figure 32), which handles the latent change.

On the other hand, an additional work amount caused by errors and changes can be represented by multiplying corresponding work scope with internal and external sensitivity. There are three routes in which "requests for additional work" can be made. First, additional work can be requested by the modifier, 'TotalInternalAdoption' (TIA) (G in Figure 32). The TIA of the activity under study is used in two situations: one is when a change is approved by the

CCM Group (G_1 in Figure 32), and the other is when accommodating an error made in the predecessor activity, through RFIs (G_2 in Figure 32).

For example, even though the aligning piles may not be matched with the specification, the subsequent columns could follow the mis-aligned piles if structural stability were guaranteed and if correcting the location of piles was not feasible. As discussed earlier, construction may prefer extra work to rework because rework in construction is often related to physical demolition of a non-repeatable item. Thus, modifying design and specification in successors is common to accommodate the predecessors' errors, and is exactly what G_2 in Figure 32 tries to capture. This results in additional work (G_3 in Figure 32) or as a re-execution on already completed tasks of the activity under study (G_4 in Figure 32). As an amplifier that reflects the degree of inter-relationship within an activity, internal sensitivity (H in Figure 32) is multiplied to acquire the actual amount.

The second modifier is called 'TotalExternalAdoption' (TEA) (I in Figure 32) from the predecessor and the successor activities, which could generate additional work (I_1 in Figure 32) or re-execution (I_2 in Figure 32) through the respective external sensitivity. In other words, TIA of predecessors and successors could affect related tasks of the activity under study, being amplified by the degree of inter-relationships among activities (i.e., external sensitivity, J in Figure 32). Continuing with the above example, in order to keep the modified location of columns in the predecessor activity, additional work (e.g., the addition of beams and girders) that increases structural stability would also have to be performed in the successor activity.

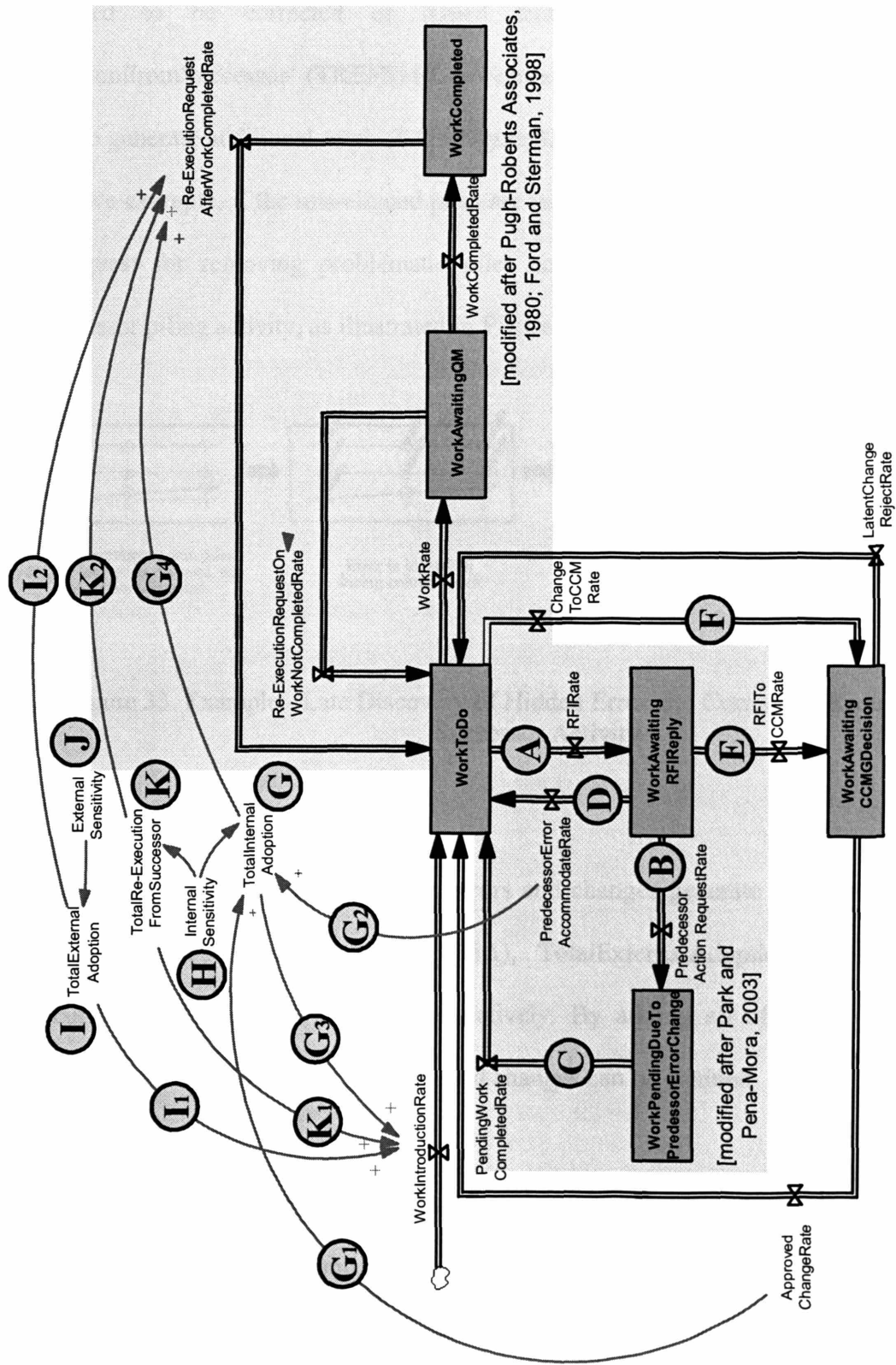


Figure 32. Work Execution with the RFI process [adapted from Pugh-Roberts Associates, 1980; Ford and Sterman, 1998; Park and Peña-Mora, 2003]

Lastly, as we observed earlier, when hidden errors or latent changes are discovered and requested to be corrected or issued from the successor activity, the ‘TotalRe-ExecutionFromSuccessor’ (TREFS) (K in Figure 32) modifier multiplied by internal sensitivity can also generate additional work (K_1 in Figure 32), as well as re-execution (K_2 in Figure 32). In the above example, if the mis-aligned piles are identified in column work and are not allowable, the request for removing problematic piles and digging new piles could be done in the predecessor piling activity, as illustrated in Figure 33.

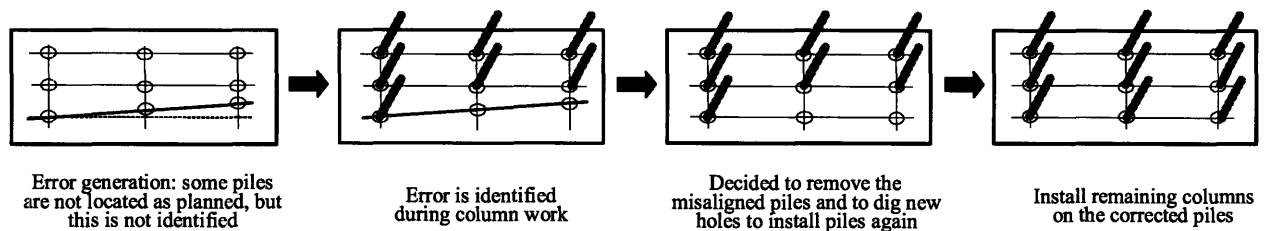


Figure 33. Example - Late Discovery of Hidden Error and Correction Request from the Successor Activity

In conclusion, the means by which errors and changes generate additional work scope was identified by TotalInternalAdoption (TIA), TotalExternalAdoption (TEA), and TotalRe-ExecutionFromSuccessor (TREFS), respectively. By adding all of these modifiers, a newly introduced work scope caused by errors and changes can be attained.

CASE 3 simulates the application of sensitivity in order to capture a propagation impact in scope. In this experiment, sensitivity of 50% was applied, indicating that errors and changes could generate as much as 50% of additional work scope to the other related activity. For

example, if an error generates 10 WU (Work Unit) in the preceding excavation activity, it can cause as much as 5 WU of additional work to the succeeding superstructure activity. Figure 34 shows the total additional work scope that incorporates the impact of sensitivity compared with that in CASE 2. Sensitivity implies that we need to consider a propagation impact in order to deal with unexpected additional work scope and, moreover, anticipate the capacity necessary to react to such additional work scope.

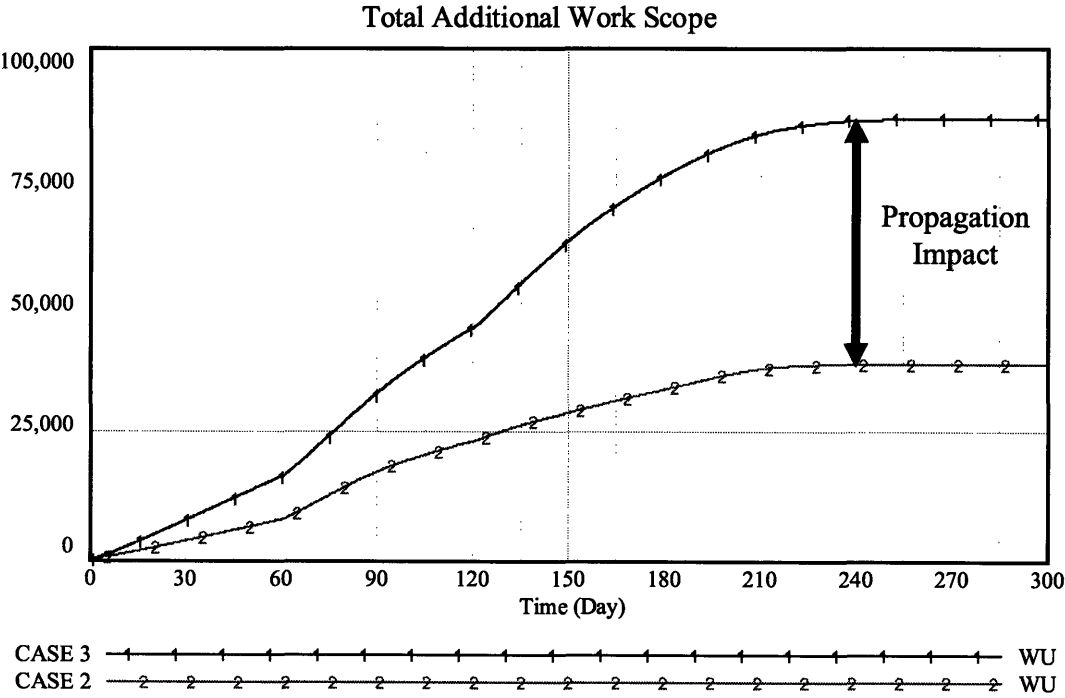


Figure 34. CASE 3 – Propagation Impact

In addition, CASE 4 is simulated applying latency. In this simulation, it is assumed that QM and SM thoroughness are both 90% which means 10% of errors and changes could be hidden or latent. The left side of Figure 35 shows that work can be re-executed even though it is perceived

that work is completed. In other words, the late discovery of hidden errors or the late approval of latent changes could demand a request for the additional work which was already completed. The right side of Figure 35 shows that a significant amount of work is waiting for an RFI response, which tries to identify and implement hidden errors and latent changes. Together with the CCM period (e.g., 7 days in this simulation), RFI response time (i.e., the time to be taken for RFI response) can also delay work execution because the RFI response time is assumed to be 5 days in this simulation.

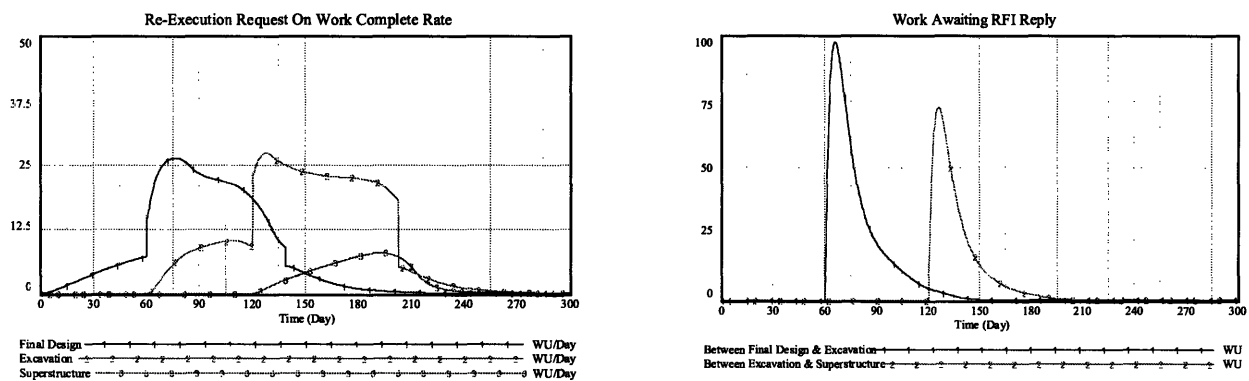


Figure 35. CASE 4 – Impact of Latency

Figure 35 shows each activity's additional work scope generated from errors and changes. One notable point is that when we apply latency, it generates less additional work in the preceding final design activity (A in Figure 36) and more additional work in the succeeding excavation activity (B in Figure 36). Due to latency, some portion of additional work scope may not be identified early and may be identified at later stages. Thus, latency reinforces the discussed idea of the 90% syndrome.

project management [Sterman, 2000]. Possible policy options that can be taken in order to improve capacity utilization are:

- Option 1: Adding service capacity (e.g., hiring additional workers)
- Option 2: Increasing work weeks (e.g., applying overtime)
- Option 3: Spending less time on each task (e.g., working faster)
- Option 4: Reducing the arrival rate of new tasks or canceling some pending tasks (e.g., controlling workflow)

Options 1 and 2 are widely used in construction when additional work is introduced. However, Option 3 (spending less time on each task) and Option 4 (reducing the arrival rate of new tasks or canceling some pending tasks) have somewhat different characteristics in construction.

For example, Option 3 is more closely related to a project personnel's first 'response' when they encounter an additional work amount, rather than a policy taken through the decision process. Suppose welding is performed on a pipe installation activity during a building's construction, and ultrasonic testing is used as a quality management technique to inspect the welded parts' performance. In this situation, if the inspectors have more work to inspect than expected and feel the pressure of meeting the schedule, their first response is usually to make an effort to speed up the progress of the testing activity. If this effort is not successful, they may request other actions from the management team to deal with it. Actually, this 'working faster'

creates diverse ‘inside dynamics’ within a project. In this sense, DPM will consider Option 3 as an embedded natural characteristic of a project when an additional work amount is introduced. In addition, Option 4 is often difficult or even impossible to execute [Sterman, 2000]. Reducing the arrival rate of new tasks or canceling some pending tasks in construction may exceed the corresponding party’s authority if it crosses a related organizations’ work responsibility. In this context, Option 4 is excluded from the current modeling scope.

As a modeling example of these options, Figure 38 shows the process of adjusting the current workweek (i.e., work hours per week) comparing with a desired workweek level by adopting an overtime policy. First, a required work rate is calculated by dividing the remaining work by available time to completion. Then, the schedule pressure is determined by comparing the calculated required work rate against the normal work rate. In the model, when the required work rate is greater than the normal work rate, it is assumed that contractors and project managers perceive the schedule pressure with a time delay. Based on the perceived schedule pressure, the overtime ratio and workweek are determined. Here, the overtime ratio ranges from 1.0 to 2.0, which means that construction workers can work up to 80 hours per week due to the degree of the schedule pressure, if we assume a normal workweek is 40 hours per week (considering that the overtime ratio can vary depending on construction conditions and policies, the model can have a different overtime ratio range based on a user’s choice). This overtime ratio is used to calculate a actual workweek and consequently, the gap between a actual workweek and a normal workweek can be used to calculate the fatigue of the workforce, which will result in productivity loss. Like this example, supporting model structures are added to represent the diverse

construction policies with their softer aspects, which basically are generated from introduction of errors and changes.

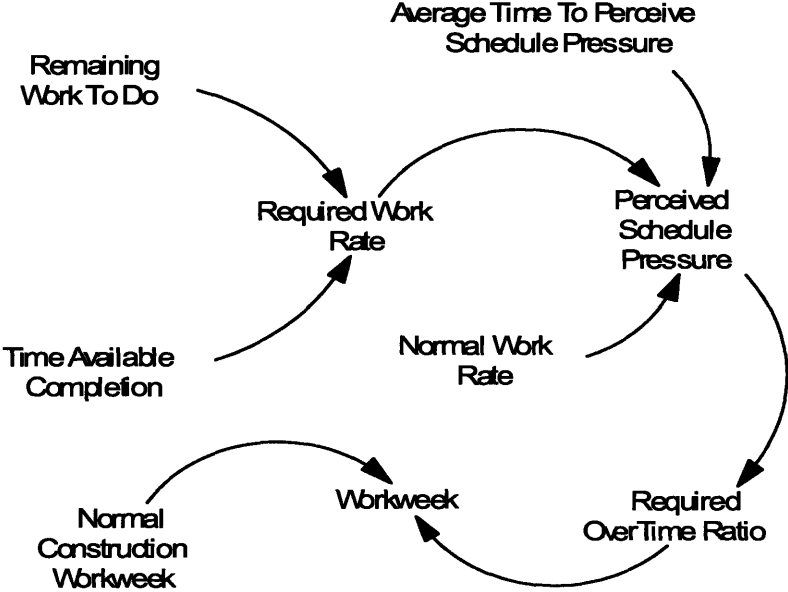


Figure 38. Example of Supporting Model Structure

5.4 VALIDATION

This model has been validated in terms of its usefulness in identifying the impacts of errors and changes on construction performance. Table 1 outlines tests that have been applied to the developed model.

Table 1. Applied Test Techniques

Test	Purpose of Test	Tools & Procedures Used at Model Testing
Boundary Adequacy	To assess the appropriateness of the model boundary in terms of the model purpose	Data collected from project documents and records, interviews from industrial partners: InteCap Inc., Modern Continental, and Barletta Heavy Division, and literature reviews are used to confirm boundary adequacy - to see whether there are potentially important feedback omitted from the model
		Subsystem diagrams, causal loop diagrams, stock and flow maps, and direct inspection of model equations - to test the boundary used in the model
Structure Assessment	To determine whether the model is consistent with knowledge of the real system relevant to the model purpose	Subsystem diagrams and stock and flow maps - to reveal the level of aggregation
		Casual diagrams - to confirm the information cues used in each decision
		Direct inspection of the equations - to check the heuristic assumed at each decision point
		The developed model is evolved from several proven model structures [Cooper, 1980; Richardson and Pugh, 1981; Abdel-Hamid, 1984; Ford and Sterman, 1998; Rodrigues and Williams, 1998; Lyneis et al, 2001; Park and Peña-Mora, 2003]
Dimensional Consistency	To ensure that all variables are mathematically consistent and further, to have variables that have realistic meanings	Automated dimensional analysis function, provided by Vensim™ (system dynamics simulation package used in this research) - to check dimensional errors
		Direct inspection of the equations - to insure that every equation must be dimensionally consistent with real world scenarios
Parameter Assessment	To estimate the values of each parameter in terms of its reasonability	Formal statistical estimation - to get available numerical data - (e.g., 'RFI Response Time' is set to 7 days as a default duration from the statistical mean of the data from different design and construction projects)
		Judgmental estimation - to supplement unavailable numerical data – (e.g., The degree of the project completion will be used to update the different value based on the data collected on the project. It can be set to take the mean as quartile of the actual data, such as 25%, 50%, 75%, and 100%) (e.g., ' RFI Response Time' can be modified by users since project type or communication technology may change its duration)
Extreme Conditions	To determine whether the model behaves in a realistic fashion no matter how extreme the inputs or policies imposed on the	Putting the extreme value - to test the robust behavior of the model – [e.g., If reliability and stability are 100% (i.e., no error and change), the simulated duration is the same as CPM estimation because CPM does not consider the error and change generation during actual execution]

	system are	
Integration Error	To make sure that the model is not sensitive to the choices of time step or integration method	Try different time steps and run the model again - to show that there is no major difference among them - (e.g., No major difference in the model - 0.1, 0.25, 0.5)
Behavior Reproduction	To assess a model's ability to reproduce the behavior of interest in the system	Compare the variables' behavior with the known - to see how well it reproduces the historical behavior - (e.g., comparison with the actual progress curve from the case projects)
Sensitivity Analysis	To test the robustness of the model behavior in uncertain conditions	Numerical, behavior mode, and policy sensitivity analysis - to know if a change in assumptions changes the numerical values of the results, the patterns of behavior, and the impacts of a proposed policy, respectively - (e.g., different progress curves by different options)

As an example of diverse test techniques, the simulation result should produce CPM calculations if a simulation setting follows CPM assumptions. In other words, if we assign 100% reliability and stability (i.e., no error and change), a straight-line production rate (i.e., work execution with a uniform rate), and no delay (i.e., instant time for the quality management process), the project duration generated by the model should be identical to the one calculated by the CPM method. The execution of the simulation confirms that these results are identical. In addition, the author conducted a couple of case studies applying the developed model to real world construction projects. In the Chapter 7, validation effort on real world projects are covered in detail.

CHAPTER 6

WEB-BASED ERROR AND CHANGE MANAGEMENT SYSTEM

In this chapter, a system's perspective of DPM is introduced, which eventually resulted in a web-based error and change management system. In short, when errors and changes are introduced, a coordination activity is involved to discuss and solve errors and changes. However, involved parties in a project, such as design professionals, subcontractors, material suppliers, consultants, and the owner, are often geographically distributed. To facilitate their coordination process, a supporting mechanism is explored in the perspective of the system.

6.1 LITERATURE REVIEW

Several error and change management systems have previously been developed. For example, Ahmed et al. (1992) developed an integrated environment for computer-aided engineering, a blackboard representation that integrates a global database, several knowledge modules, and a control mechanism to systemize object changes, and Peltonen et al. (1993) proposed an engineering document management system for changes, which supports document approval and release procedures. In addition, Spooner and Hardwick (1993) developed a rule-based system to

coordinate concurrent changes and to resolve conflict, and Ganeshan et al. (1994) developed a system to capture the design process history, initiate backtracking, and determine the decisions that might be affected when changes are made in the spatial design of residential buildings. Krishnamurthy and Law (1995) and Mokhtar et al. (1998) presented a change management system that supports collaborative design environments. Soh and Wang (2000) proposed a constraint methodology to coordinate design consistency between different geometric models and to facilitate design change management. Hegazy et al. (2001) introduced an information model to help the coordination of design as well as the management of design changes. Karim and Adeli (1999) implemented a decision support system that dealt with schedule reviews, progress monitoring, and cost-time trade-off analysis for change order approval. In addition, motivated by the potential of Internet technology in managing construction projects, Charoenngam et al (2003) developed a web-based change management system that supports documentation practice, communication, and cooperation among project team members.

All the research, mentioned above, has contributed to managing the inherent changes in the design and construction process. However, their focus is primarily on the recording and coordination of design changes, after these changes have occurred. These systems do not incorporate a comprehensive system that analyzes the impacts of errors and changes in advance (e.g., how errors and changes affect construction performance and how policies can be designed to deal with their detrimental impacts) with the support of a collaborative environment. In this context, the DPM system is developed to be capable of analysis of error and change cycles as well as be applicable in a practical and smooth manner to real-world projects.

6.2 SYSTEM OVERVIEW

The DPM system intends to *integrate several existing tools* to achieve a flexible application for diverse situations. Particularly, network-based tools are incorporated into the system dynamics model in order to increase their representation accuracy (i.e., simulation capability) while maintaining their broad applicability, as discussed in the Chapter 5. This needs be supported by the system's perspective, as well. Figure 39 illustrates how DPM integrates these different tools and how the integrated tools interact with each other.

One of the important issues in integrating network-based tools with the system dynamics model is that different kinds and types of input and output are used for their application. Thus, how to communicate with users and even among themselves becomes critical to achieve a seamless integration. In this context, DPM utilizes and creates simple algorithms and visual techniques to help organize and share gathered information.

More specifically, a *smart cell* is implemented, which receives, stores, and organizes the data input by the user. Then, the smart cell is integrated with the Dependency Structure Matrix (DSM) in order to visually represent the relationships among the different activities. Though the main objective of DSM is to provide an ordering of sequence of acquired information, DSM is utilized as a visual input interface in this research. After the smart cell receives all the necessary information from the user, it transfers that information to the system dynamic model (A in Figure 39). The system dynamics model that includes several ideas from network scheduling techniques,

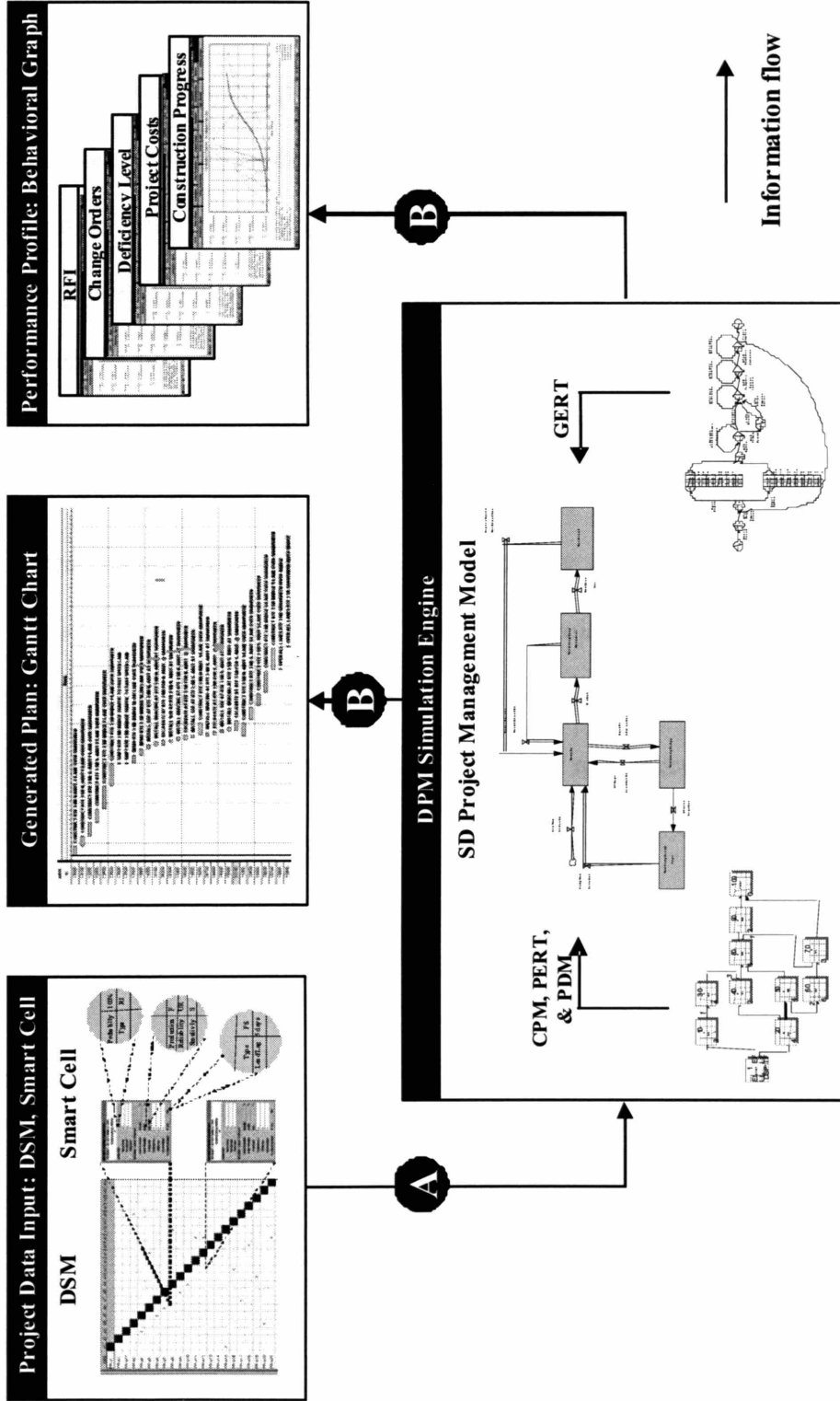


Figure 39. DPM System Overview

simulates the impact of iterative cycles and the applied policy options based on the necessary factors given through the smart cells. The generated simulation output provides performance profiles that capture the impact of unanticipated events on project performance (B in Figure 39) in an appropriate *graphical format* (i.e., Gantt chart and Behavioral graph). In addition, *the simulation output is translated into a familiar network* so that the impact of the different policies can be compared with the original network. In an illustration of the above interactions, Figure 40 presents one of the activity diagrams in the Unified Modeling Language (UML), which describes how the simulated result can be compared with the original plan generated. Functionalities for project control are also implemented. For instance, DPM gets the current progress of actual execution as an input and compares it with the stored simulated performance that had been completed before a project had actually started. Based on this comparison, DPM could provide future policy candidates with better project profiles. Or, it could re-simulate and forecast project performance by adding the actual information it previously had retrieved. Once actual data are obtained, simulated data are refined for more accurate planning and forecasting.

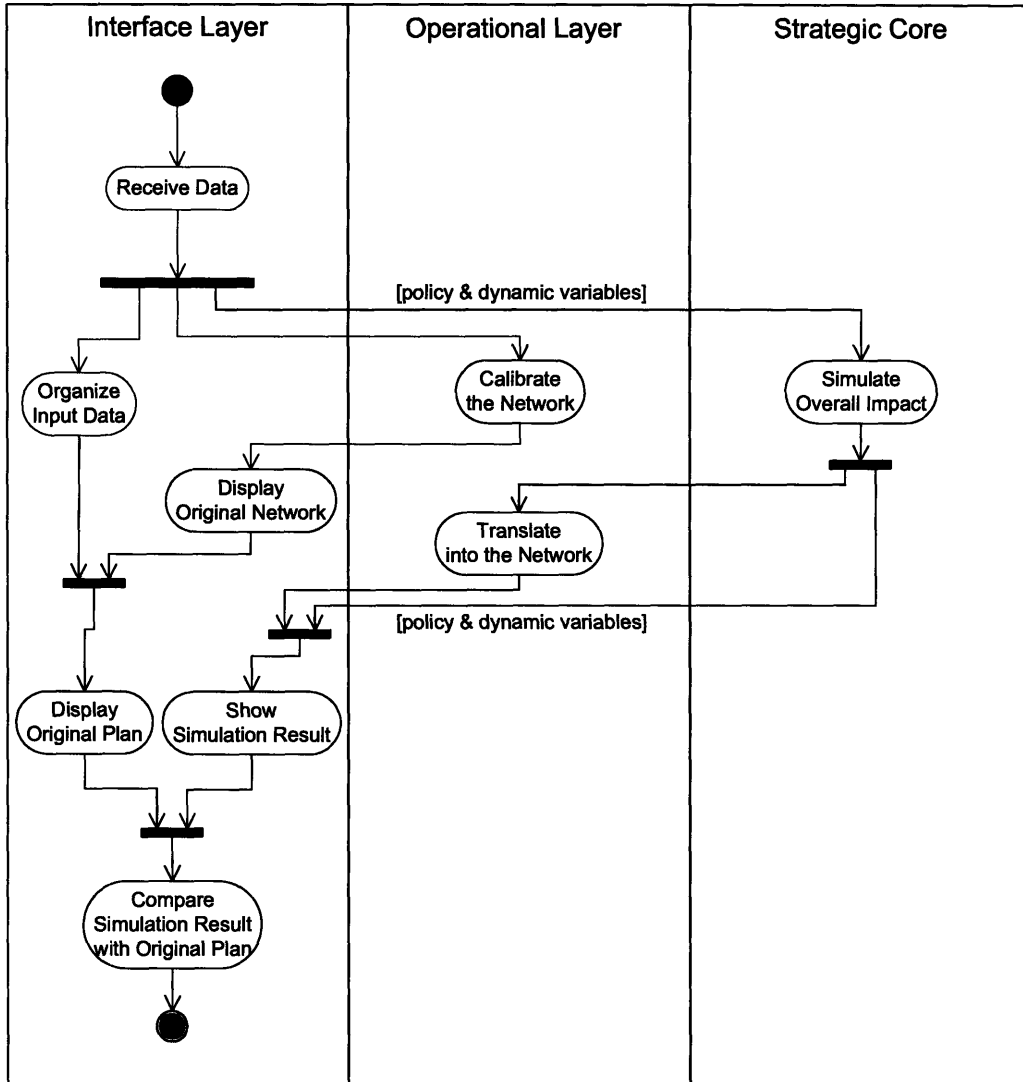


Figure 40. UML Activity Diagram with Swimlanes – Comparing Simulation Result with Original Plan

6.3 COLLABORATIVE ENVIRONMENT

Construction is performed in open environments by a temporary alliance among multiple organizations [Slaughter, 1998]. In this sense, sharing on-site solutions to problems quickly and

reliably can facilitate smooth transactions among involved organizations. Further, as projects often take place at geographically distributed places, quick and reliable information sharing becomes crucial for project success.

To address these issues, DPM is required to be built on a platform where easy and *real-time information access* can be achieved without hardware and software restrictions, as illustrated in Figure 41. For example, in some places, on-site work conditions do not allow the use of desktop computers or wire connections; rather, portable devices and wireless connection are preferred. Therefore, the platform for heterogeneous devices such as Personal Computers (PC), Personal Digital Assistants (PDA), and mobile phones with both wire and wireless networking connections, can contribute to the improvement of construction productivity. In this manner, on-site engineers could collaborate visually with off-site project managers and designers. In addition, considering the aspect of temporary alliance in construction, installing different software packages for every different project could be an annoying task and may create a situation or data format incompatibility. Thus, the proposed web-based DPM provides a universal web interface which could be accessed without hardware and software restrictions using web server-client architecture. Another important aspect is security. The sharing of reliable information is implemented using a Public Key Infrastructure (PKI). PKI uses public and private cryptographic key pairs obtained from a trusted Certificate Authority (CA) making it possible to exchange data securely. In the DPM architecture, the CA associated with the web server, distributes the digital certificate including public key to the multiple devices. In this way, the security can be maintained even though a public network like the Internet is used.

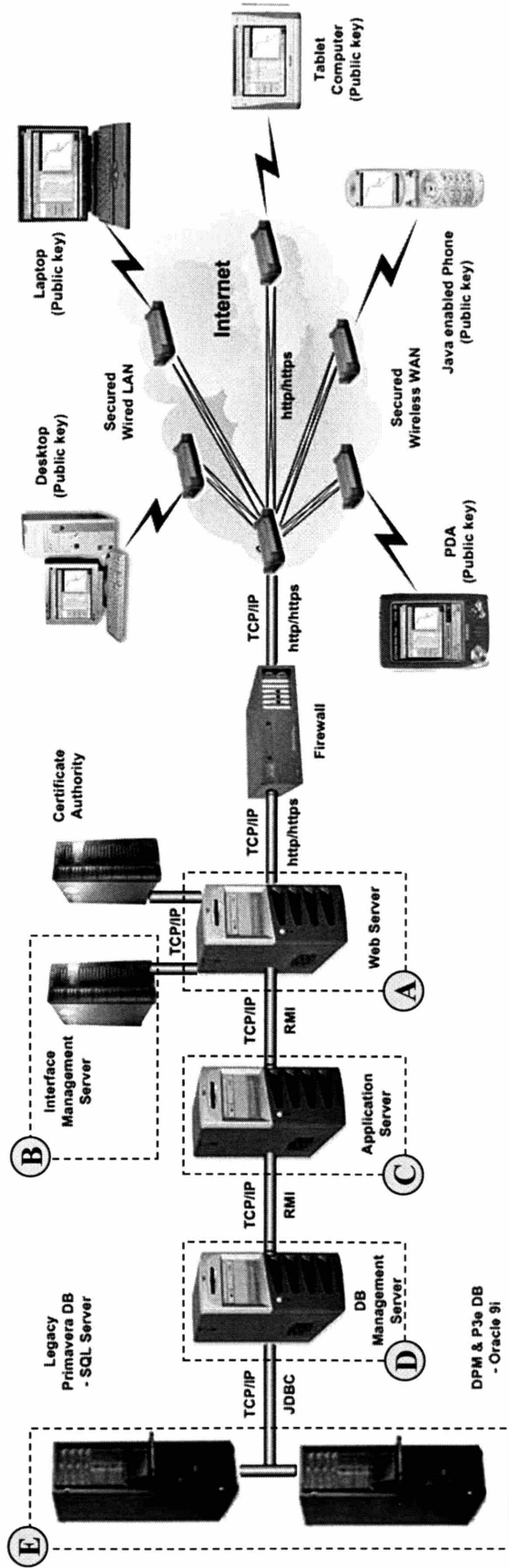


Figure 41. DPM System Architecture for Collaborative Environment

6.4 IMPLEMENTATION OF THE DPM SYSTEM

Based on several issues in the DPM system, the implementation of a web-based DPM is described in this section.

Database

The data model in DPM has been developed using a Relational Database Management System (RDMS), rather than an Object Database Management System (ODMS), as seen in Figure 42. The RDMS was utilized in order to fit the integrated server system architecture, which will be introduced later. The developed data model aims at not only capturing the project status at a give time, but also maintaining the ‘history’ of the project. This is because one project can have multiple schedule updates during execution, for instance, due to an owner’s change request. These maintained schedules can be used in various ways, such as helping the participants involved in a project share quick and compatible information about these changes. In this data model, for example, that functionality is achieved by differentiating ‘PROJECT’ and ‘VERSION’ entities, which are associated with one-to-many relationships.

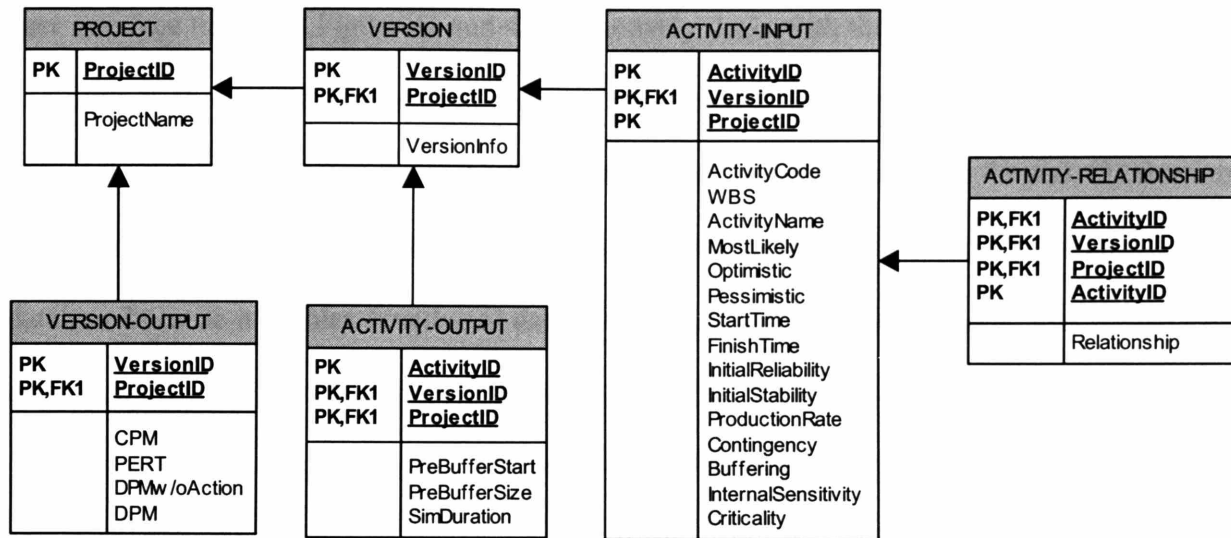


Figure 42. Sample Data Model in DPM

System Architecture

Figure 43 illustrates a UML deployment diagram that shows the physical relationships among software and hardware components in a developed system [Fowler and Scott, 2000]. DPM system architecture is designed as a five-tier scheme that is implemented by adding a client-server aspect to both the user interface tier (A in Figure 41 and 43) and the data store tier (E in Figure 41 and 43) maintaining the application logic tier separate (C in Figure 41 and 43). Thus, maximum flexibility can be achieved. For example, the DPM system requires handling the diverse look and feel of the interface, since heterogeneous devices (e.g., PC, PDA, and mobile phone) have different screen real estates. A PDA may have a (320×320) screen resolution, but Java-enabled phone could be (176×208). Adding a client-server aspect to the user interface layer, such as separating the interface management server (B in Figure 41 and 43) from the original

user interface tier (A in Figure 41 and 43), effectively deals with the different screen real estates and formats through an independent presentation tier. A similar concept is also applied to the data store tier by separating the database management server (D in Figure 41 and 43) from the data store tier (E in Figure 41 and 43). DPM adopts Oracle 9i (Oracle, 2002) as the main database because it enables Java-based database application, making it an ideal tool for database centric applications [Peña-Mora and Dwivedi, 2002]. Oracle 9i is also used for P3e/c™ (Primavera P3e/c for Construction – Project Planner, Primavera Co., 2003) database so that DPM can freely import existing project data from a widely-used commercial software. In addition, the DPM system needs to handle the legacy Primavera database, which uses Microsoft SQL Server (Microsoft Corporation, 2000), in order to seamlessly use project data of those legacy systems. Therefore, the architecture to deal with multiple databases with different schema is necessary. In order to support multiple databases, the database management server becomes an independent tier in the current implementation of DPM.

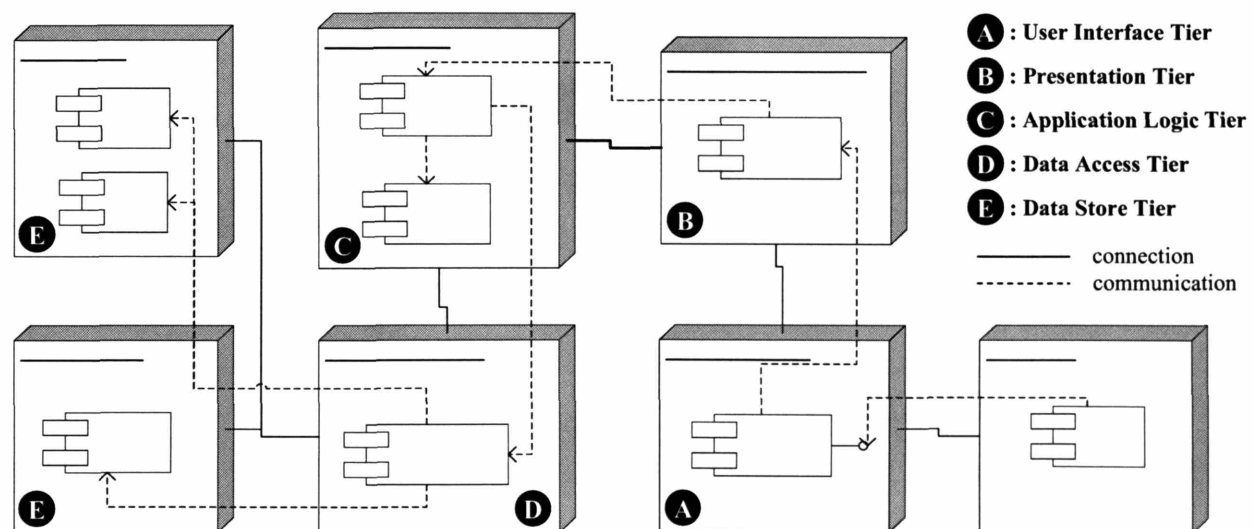


Figure 43. UML Deployment Diagram

System Development

Among several development tools, the Java language is adopted due to its cross-platform portability, which can facilitate the development of the Internet-based collaborative system. In more detail, efficient communication among the distributed tiers should be smoothed due to frequent object exchanges. Java enables the system to communicate seamlessly by providing Java Remote Method Invocation (RMI) and Java Data Base Connectivity (JDBC) as object-based middleware for the remote communication. In addition, the evolution of Java 2 Platform Micro Edition (J2ME) enables Java applications to be used for diverse portable devices without the limitation of operating systems. Lastly, Vensim™ (Ventana Software, Inc., 2002), the software used for system dynamics modeling in this research, supports Java to communicate with the system dynamics model through Dynamic Link Library (DLL), as illustrated in Figure 44. Java VensimConn class of the application logic tier connects and operates the system dynamics model in Vensim™, by loading 'venjava.dll'.

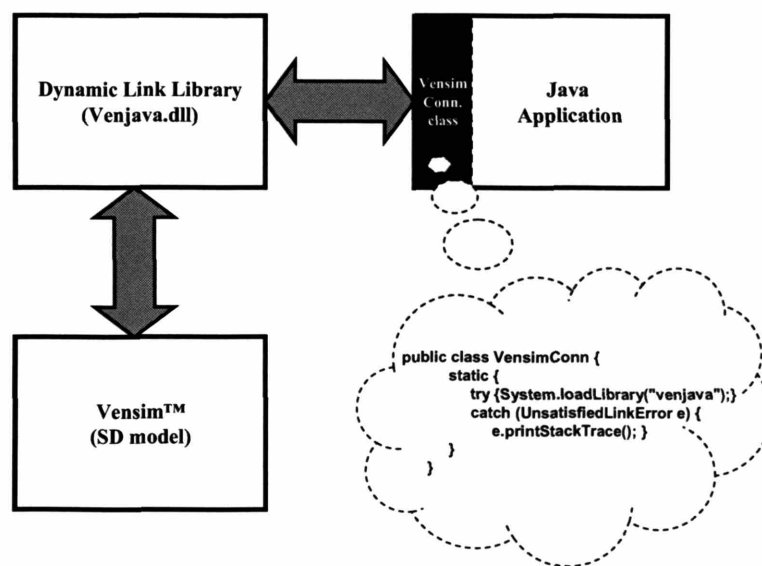


Figure 44. Inside Application Logic Tier

6.5 FUNTIONALITIES OF THE DPM SYSTEM

Based on the suggested system guidelines, a web-based DPM system¹ was developed, as seen in Figure 45, which is its main window. In this section, the functionalities of the DPM systems introduced, such as its input and output mechanism, tracking schedule changes, game, and import functionality.

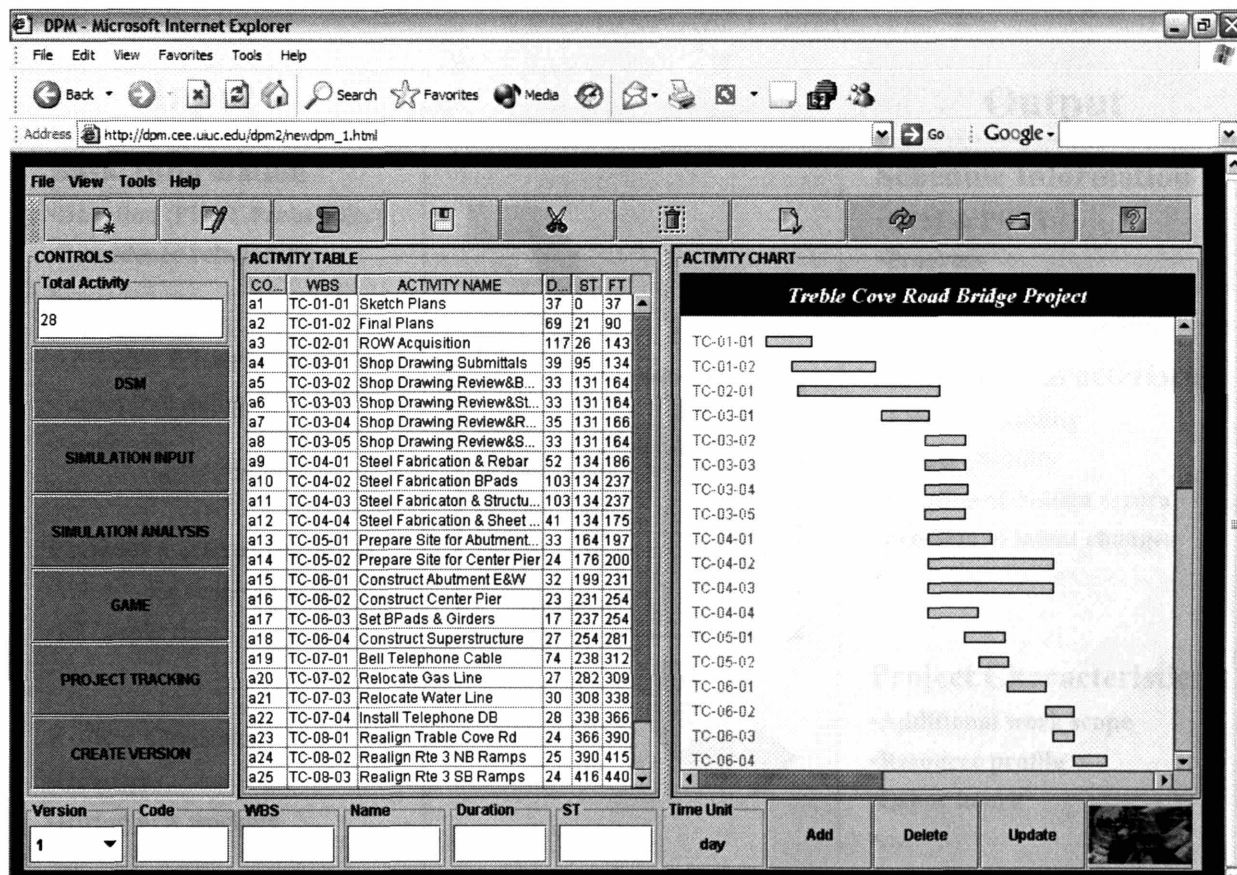


Figure 45. DPM Main Window

¹ <http://star.mit.edu/dpm> and <http://dpm.cee.uiuc.edu/dpm2>

Input and Output

Figure 46 is the overview of input and output of the DPM system. The basic idea is that based on the data that DPM gets, the corresponding output of DPM will be generated. For example, if DPM get an input for activity duration and precedence relationship, the result will be same as CPM. However, if activity characteristics such as reliability and stability are input, the DPM simulation engine will generate the corresponding results considering these characteristics.

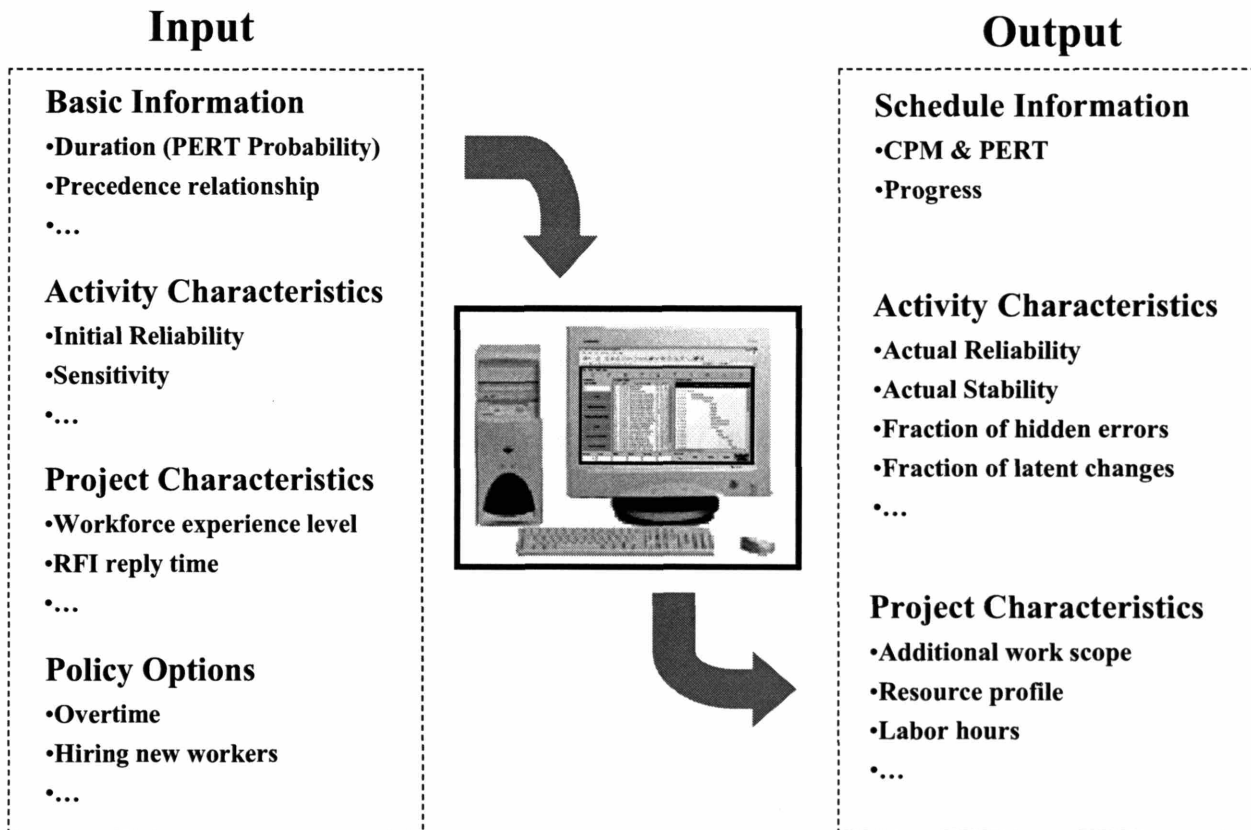


Figure 46. Overview of Input and Output of DPM

In terms of input, DPM requires two kinds of data; the basic data that will be permanently used for a variety of functions and the temporary simulation data that will be temporarily used only for simulation.

Basic Data

Basic data in DPM refers to activity and network information which represents the design and construction process. An example of this basic data includes the probabilistic duration spectrum from PERT (classified as most likely, optimistic, and pessimistic – the system also allows for the deterministic duration approach of CPM), the production rate (fast or slow), initial reliability, initial stability, the internal and external sensitivity, and precedence relationships with leads or lags. In order to help organize such diverse types of data, DPM utilizes DSM (Dependency Structure Matrix) to visualize the external relationships among activities and smart cells to store detailed activity data. For example, DSM in Figure 47 marks ‘X’ on a cell when the precedence relationship between two activities is specified. The ‘X’ mark indicates that those two activities have a precedence relationship. When the user clicks a cell which has a ‘X’ mark, detailed information about external relationship information (e.g., Start-to-Start relationship, lag 5 days, and externally sensitive) as well as internal activity information (e.g., most likely duration 45 days, slow production rate, and internally sensitive) is displayed in the left side of the window in Figure 47. Cells containing this ‘pop-up’ functionality are smart cells.

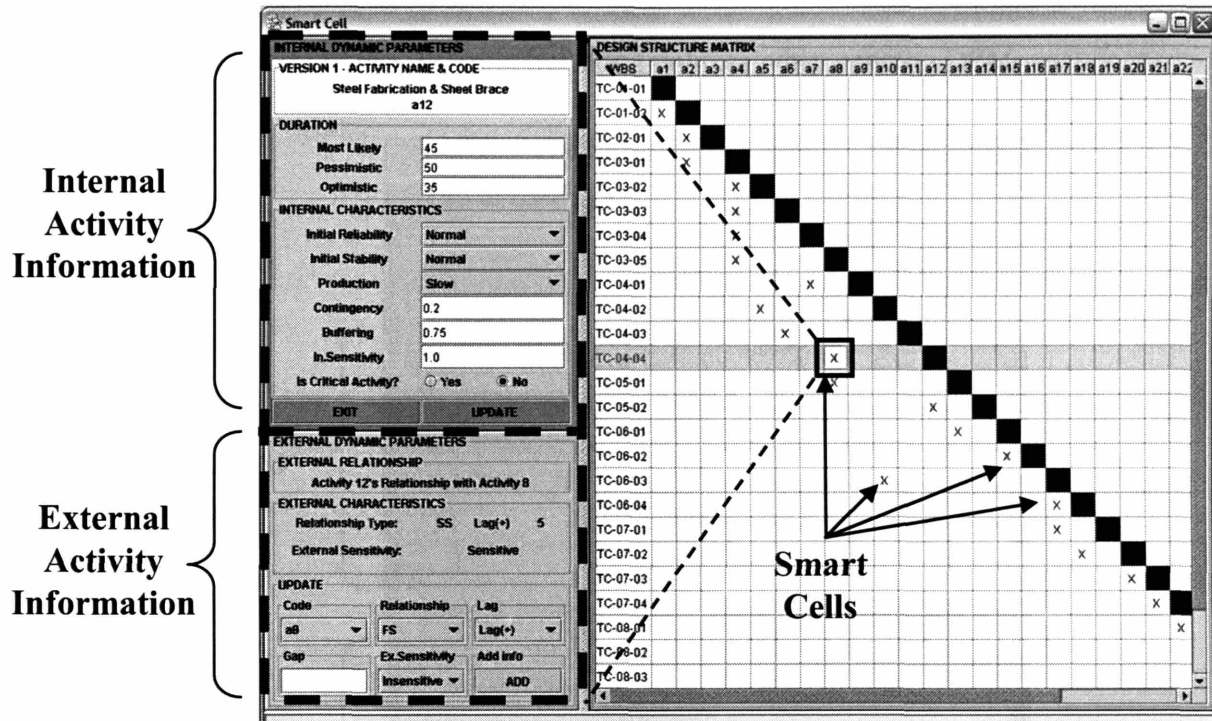


Figure 47. Basic Data - Dependency Structure Matrix with Smart Cells

Simulation Input

Simulation input, as seen in Figure 48 is temporary data that provides project policy-related information for the dynamic DPM simulation. Examples of this input include policy options, when an additional work amount is introduced during actual execution, such as Option 1 (i.e., adding service capacity; i.e., flexible head count control) and Option 2 (i.e., increasing work weeks; i.e., overtime). In addition, an average Request For Information (RFI) period, workforce experience level, and different buffering policies can be specified through this simulation input window. In terms of buffering policies, DPM can support the reliability and stability buffering

approach as well as traditional contingency buffering. With this mechanism, the user can compare the impact of these two different buffering policies when running DPM.

The screenshot shows a 'Simulation Input' window with the following structure:

- COMPARATIVE METHODS:** CPM, PERT, DPM W/O Action
- DYNAMIC SIMULATION:**
 - * POLICY OPTION:**
 - Control Headcount
 - Adopt Overtime
 - QM Familiarity: 1.0
 - SM Familiarity: 1.0
 - RFI Period: 7
 - CCB Period: 10
 - Workforce Experience Level: 1.0
 - Time to Increase Workforce: 7
 - * BUFFERING OPTION:**
 - Reliability Buffering Activation
 - Buffering for PJ: 0.5
 - Known Contingency Factor for PJ: 0.2
- Buttons:** Simulate, Exit
- Notice:** DPM will launch output window automatically after simulation
- Footer:** Java Applet Window

Annotations on the left side of the image:

- Policy-related Options:** A bracket groups the 'POLICY OPTION' section.
- Buffering Options:** A bracket groups the 'BUFFERING OPTION' section.

Figure 48. Simulation Data: Project Policy-Related Information

Output

Based on the discussed input, DPM can generate four kinds of output; *CPM*, *PERT*, *DPM w/o Action*, and *DPM w/ Action cases*. CPM and PERT cases are provided as a comparative objective, and DPM w/o Action and DPM w/ Action cases are actual simulation results that consider the impact of errors and changes. More specifically, DPM w/o Action case is the base

run that reflects the impact of the current construction characteristics and applied policies. On the other hand, DPM w/ Action case is the simulation run that applies different characteristics and policies in order to explore the varying ways to minimize the effect of errors and changes.

Figure 49 shows the simulation output of this case project in terms of a project level and activity level. The project level output illustrates the overall progress completion curve (A in Figure 49) and the Gantt chart of DPM w/ Action case (B in Figure 49). The progress completion curve (A in Figure 49) represents all four cases (CPM, PERT, DPM w/o Action, DPM w/ Action) for the purpose of comparing the impact of different policies. It also helps determine which activity could cause a bottleneck in the project and where managerial attention should be paid.

More detailed information is given through an activity level output. A Gantt chart (C in Figure 49) represents CPM, PERT, DPM w/o Action, and DPM w/ Action cases together so that they can be easily compared with each other. It also reflects the change of network information, such as duration and precedence relationships. If a user selects any activity in this Gantt chart for detailed information, the behavior of selected variables (D in Figure 49) is displayed, such as the activity progress curve, reliability, stability, and workforce utilization ratio. These behavioral graphs help users understand the diverse impacts of selected variables on each activity's performance.

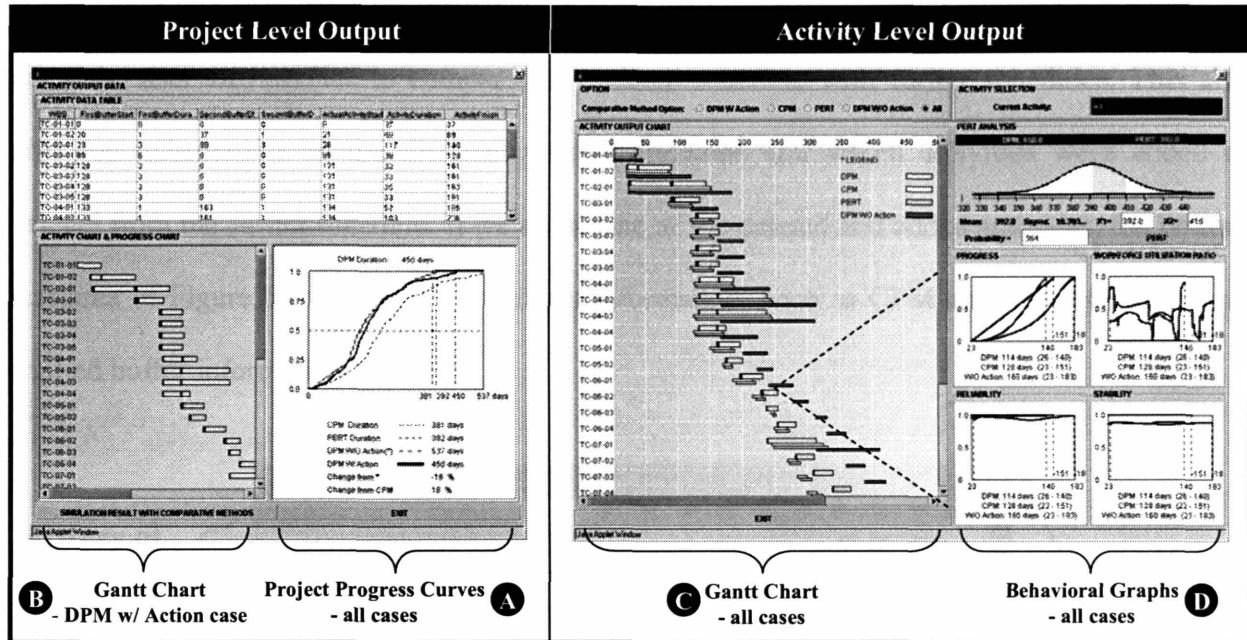


Figure 49. Project and Activity Level Output

Tracking Schedule Changes

Usually, an original schedule can be frequently updated, particularly, as a project becomes larger and more complex. Although the updated schedule is used for execution, schedules that are developed before updating are also important since they can provide valuable information for future decisions. In addition, sharing this update promptly with other participants helps them to make compatible decisions, which contribute to avoidance of possible disputes. In this context, DPM implements the functionality to track and compare the change to project schedules, as illustrated in Figure 50.

Suppose two major updates were made due to owner's requests. The Gantt chart in Figure 50 gives the user an interface to compare the updated schedule with earlier schedules. This also provides information on the different project durations and which activities were added or deleted from the earlier schedule. If we select one of the deleted and added activities, the bottom text area of Figure 50 shows each activity's information, such as CPM and DPM duration and applied buffer information.

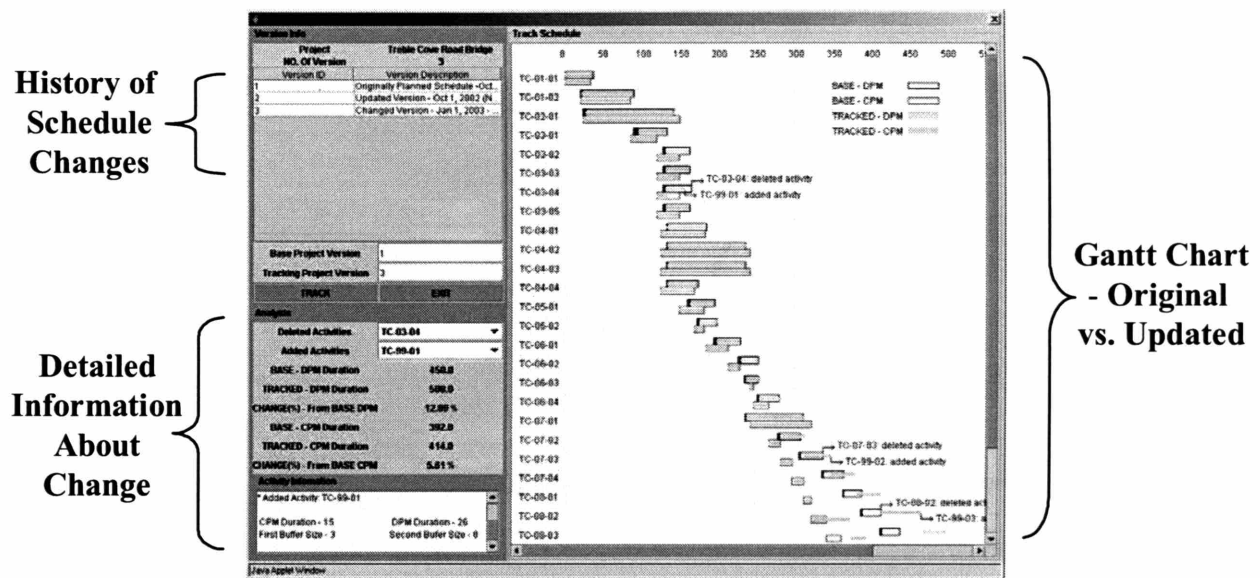


Figure 50. Tracking Schedule Changes

Game

Game can be explained as the instant simulation that can be done with free control of the simulation time step, as displayed in Figure 51. Therefore, the user can choose different construction characteristics and project policies anytime during the simulation. By implementing

a game, the user immediately compares the impacts of different characteristics and policies on the remaining project's performance. For example, though flexible hiring of additional workers was not allowed at earlier stages, it could be allowed at later stages. Game helps the user determine how Option 1 will affect the remaining project performance. In summary, game provides the following benefits; (1) immediate learning is achieved from the instant result of different variable settings and (2) project control is facilitated due to its effective capability to simulate using a flexible time step.

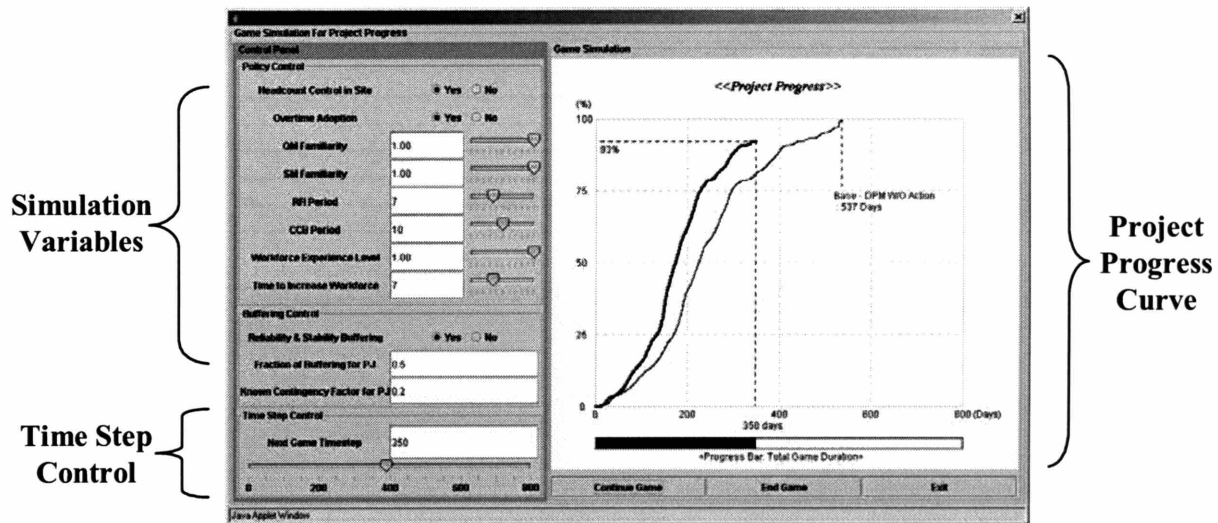


Figure 51. Game (Interactive Simulation)

Import Functionality

DPM provides the functionality to import project information from existing commercial software packages (e.g., P3e/c™) to avoid double entry of information. For example, DPM can import projects that exist in P3e/c™ by connecting to its database, as seen in Figure 52. Thus, the basic

data of projects (e.g., activity duration, precedence relationship, and lead or lag information) can be easily obtained.

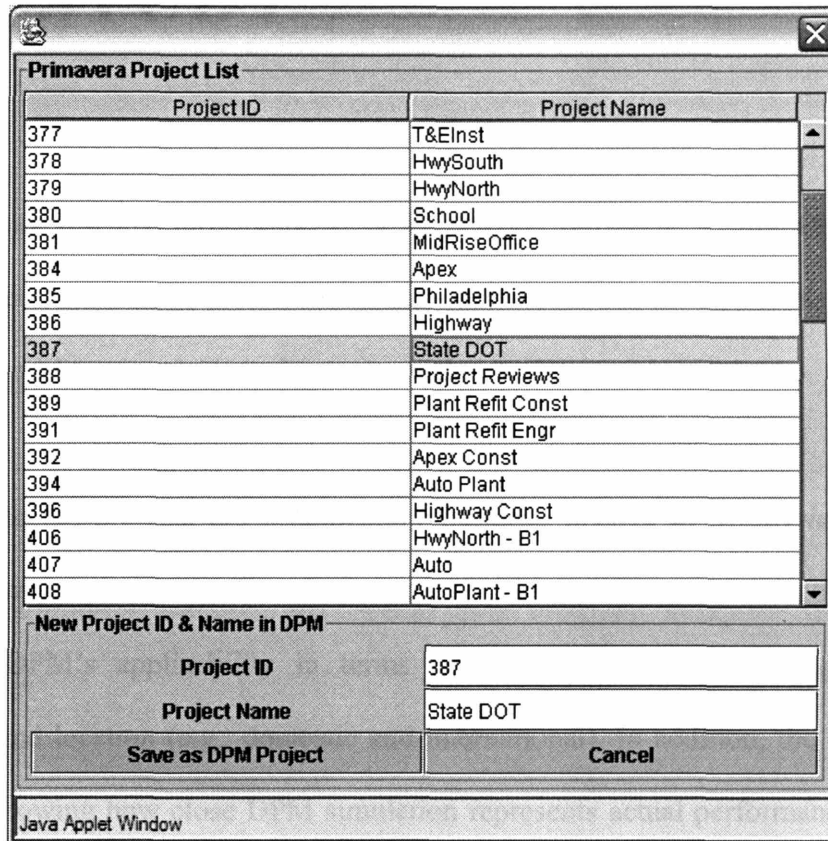


Figure 52. Information Exchange with Primavera Database

CHAPTER 7

CASE STUDY

So far, four major research developments (a framework, buffering, a simulation model, and a web-based system) have been presented. In this chapter, all the developments are applied to a couple of real world case projects to demonstrate their validity and usefulness.

The two selected projects are a *laboratory building project in Malaysia* and a *highway infrastructure project in Massachusetts*. One of the primary reasons for selecting these projects is to explore DPM's applicability in terms of the construction type (e.g., building and infrastructure) and location (e.g., domestic and international). In addition, the former was used for validation showing how close DPM simulation represents actual performance, and the latter was used for analysis showing how helpful DPM simulation is to understand and manage the impacts of errors and changes. Before going into the details of these case projects, the following section discusses the input elicitation process for their case study.

7.1 INPUT ELICITATION PROCESS

In order to determine the simulation input (i.e., model parameters), extensive individual interviews, surveys, and focus groups are conducted. Through these channels, construction

managers, superintendents, schedulers, and workers are asked to estimate values, such as forecasted stability, forecasted reliability¹, and sensitivity, based on their experience and historical data. At the initial data gathering stage, it is not easy to convey to participants what precise meaning of parameters and how they can be measured because they have never used such parameters into their project planning. In addition, the use of unfamiliar parameters may cause subjective assessment. To overcome this situation, the author *divided the input elicitation process into the three steps* shown in Figure 53. First, we conducted a semi-structured interview to collect the preliminary data in an attempt to understand project and activity characteristics (A in Figure 53). Not only does the semi-structured format generate consistent answers to a pre-determined questionnaire, it also has the flexibility to incorporate interviewees' opinions that may be unanticipated by our prepared questionnaire. Figure 54 shows the example of the pre-determined questionnaire used in one of the case projects. Here, the questionnaire was specifically intended to address different activity characteristics. In addition, for some parameters' behaviors, interviewees were asked to draw the shape of behavior over time if necessary. Then, all answers were transcribed for preliminary data analysis. Preliminary data analysis, denoted by B in Figure 53, was then done using statistical methods. For example, the average RFI response time for this case project and its approximate probability distribution was obtained. Finally, with this preliminary data analysis and the input guide explaining how to choose numerical input values, agreement with the interviewees was reached in order to determine the reasonable numerical input values (C in Figure 53).

¹ After inputting forecasted reliability, actual reliability will be generated through model simulation. This is because many factors can affect actual reliability during execution. For instance, learning would increase initial reliability



Figure 53. Input Elicitation Process

PREDECESSOR-RELATED QUESTIONS	
Q1:	What are task predecessors? Any related utility work?
Q2:	What are necessary material deliveries? What are related shop drawing submittal approvals?
Q3:	What if a predecessor activity was changed from the original plan, how will this activity be affected?
Q4:	If its predecessor activity was done incorrectly, how will this activity be affected?
TASK SPECIFIC QUESTIONS	
Q5:	What are the labor and equipment needs? Are they readily available?
Q6:	Is the task production instantaneously, completed near the start of the activity, completed near the end of the activity, or generally linear?
Q7:	Name factors that would create high productivity; Name factors that would create low productivity. What factors are most likely to be encountered? What are the potential weather impacts and their likely impact to the schedule?
Q8:	Would an additional labor crew double the expected productivity? Do you expect labor shortage problems?
Q9:	What is the degree of confidence that this schedule activity will be done correctly the first time? Name factors that could contribute to poor workmanship.
Q10:	How susceptible is this activity to change, both owner-initiated and due to changed field conditions? If a change occurs, how far will this change ripple through the schedule?
Q11:	Given a changed field condition, will redesign be necessary? Will an owner-directive be necessary? What kind of documentation of the change is necessary? Could work progress in the field uninhibited by the change? Will a corrective action plan be drafted and reviewed? Will traffic management plans need revision? Will community notification be necessary? Could change require revised equipment and labor resource needs? Could additional procurement be necessary?

Figure 54. Example of the Pre-determined Questionnaire for the Input Elicitation Process

because workers get used to their work as time goes by.

This input elicitation process has another benefit. It helps participants better understand their activities and projects. For example, one of the interviewees in the Malaysia project confirms:

I implicitly know that more additional work due to errors and changes will be introduced. However, there are no tools to address and further, incorporate this idea in planning. Actually, answering the questionnaire helps me better understand how I need to prepare a resource profile such as workforce allocation in case that unexpected more work is introduced.

- Project Manager A in the Malaysia Project

The explicit use of such dynamic activity characteristics, which are in their mental database [Forrester, 1994], provides a different perspective which enables participants to better understand their activities and projects.

7.2 LABORATORY BUILDING PROJECT IN MALAYSIA

The first case study project is a laboratory building project in Malaysia. This is a multipurpose building project from SIRIM Berhad as shown in Figure 55 (hereinafter, this project is called SIRIM project). The heading of the project is “Proposed Centre of Industrial Automation & Laboratory for Development & Research of Reliability & Quality at Bukit Jalil for SIRIM Berhad”. The contract sum for this project is about RM 30 Million¹. The date of site possession

¹ Approximately, it is equivalent to \$8 Million US as of April 20, 2006.

was on May 7th, 2004, and the date of completion is by August 5th, 2005, a total period of 15 months. The defects liability period is 12 months; start from the date of completion (Acceptant of Completion), which might differ from the planned completion date. The Liquidated and Ascertained Damage (LAD) is set as RM 8000 per calendar day that will be imposed if any delays occur. Contractors are expected to purchase the Workmen's Compensation Insurance and Contractor's All Risk Insurance as well as Performance Bond before the commencing of the project.

This case project serves two purposes: one is for validation by confirming whether the simulation results predict the behavior observed as this project actually progresses, and the other is for analysis of how errors and changes affect this project.

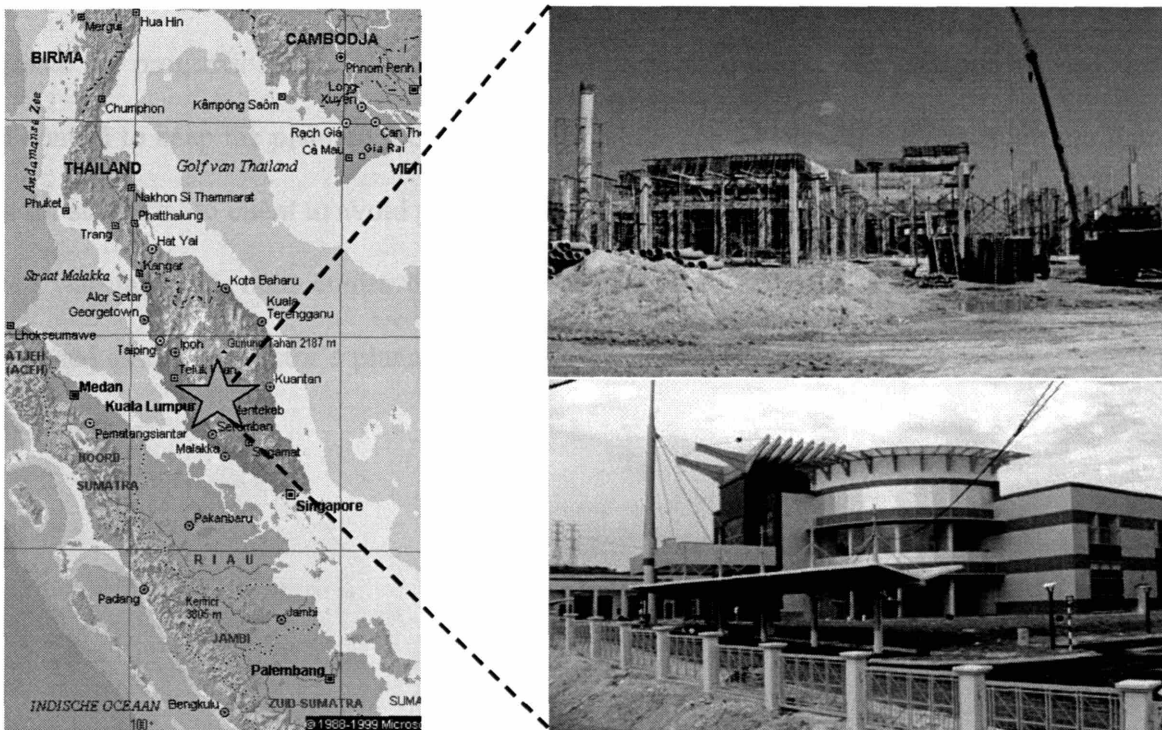


Figure 55. Project Location and Site Photo

Project Input

Among several buildings in this project, major activities in Main Building will be used as the input data for DPM simulation and analysis. On the preliminary stage, activities in Main Building beginning from excavation and end with mechanical and electrical (M & E) work will be summed up as total of 29 hammock activities as in Table 2. As this research was done only on part of the SIRIM project, the results do not imply the actual project schedule. Rather, it only represents performance of the Main Building in SIRIM project.

Through the input elicitation process discussed earlier, simulation input is attained, and Table 2 lists some of them. One thing to note is that the main contractor kept two schedules, an Internal Schedule (IS) and External Schedule (ES), because of the uncertainty and incapability to estimate the progress. IS, which assumes a tight project duration, was used for self-monitoring and control to keep the planned project progress while ES, which sets a longer duration than IS, was for submittal to client to avoid possible disputes. Thus, ES is an official schedule designed to give a warning to the main contractor in advance when there is a sign of delay. In this simulation, IE is used as the contractor's planned schedule because it more closely reflects their estimation of the progress.

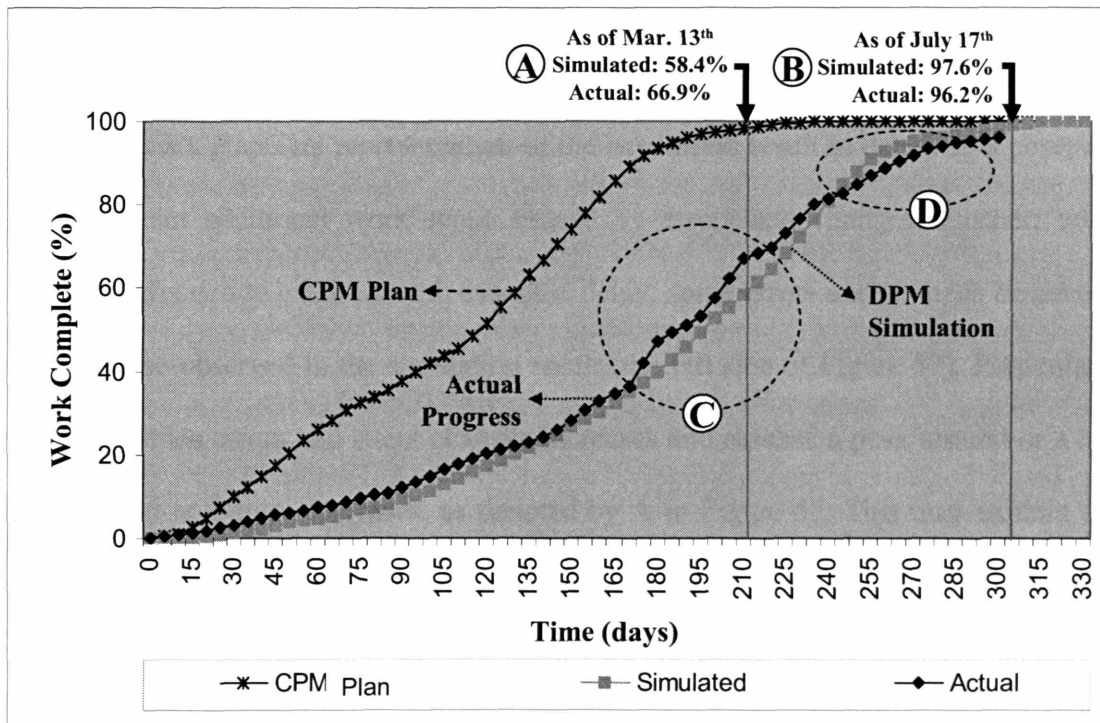
Table 2. Simulation Input for SIRIM Project

Code	Activity Name	Duration (days)	Contingency Buffering (days)	Production Type	Reliability	Sensitivity (Internal)	Sensitivity (External)	Stability
1	Excavation	35	5	L	UR	SE	SE	UST
2	Cut Off Pile	30	5	L	HR	SE	SE	ST
3	Pile Caps & Stump	70	10	S	R	SE	SE	UST
4	Ground Floor Beam	75	10	S	R	SE	SE	N
5	Ground Floor Slab	60	10	F	R	SE	SE	N
6	Ground Floor Column	105	20	S	R	SE	SE	N
7	1st Floor Beam & Slab	110	20	S	R	SE	SE	N
8	1st Floor Column	45	5	S	R	SE	SE	N
9	Roof Beam	51	11	S	R	HSE	HSE	N
10	Upper Roof Beam	60	10	S	R	SE	SE	ST
11	Metal Roof Structure	125	15	L	R	SE	SE	ST
12	Metal Roof Covering	88	18	F	R	SE	SE	ST
13	Brickwork	98	18	S	R	SE	SE	ST
14	Door & Window Frame	82	12	F	R	SE	SE	ST
15	Glazing Works (Tempered Glass)	113	13	F	R	N	N	ST
16	Internal Plastering	90	10	S	R	SE	SE	ST
17	External Plastering	91	31	F	R	ISE	ISE	ST
18	Screening Works	84	24	S	R	ISE	ISE	ST
19	Tiling Works	64	24	S	R	N	N	N
20	Ceiling Works	89	29	S	R	N	N	N
21	Aluminum Cladding Works	125	35	F	R	ISE	ISE	N
22	Staircase Finishes	4	0	L	R	SE	SE	ST
23	Door & Window Leaves	83	23	F	R	ISE	ISE	ST
24	Painting	114	24	F	HR	ISE	ISE	N
25	Lay Carpet	48	18	F	N	N	N	N
26	Plumbing	82	12	F	R	HSE	HSE	N
27	Electrical	92	12	F	R	HSE	HSE	N
28	Air Con	79	19	F	R	HSE	HSE	N
29	Fire Protection	90	15	F	R	HSE	HSE	N

*Note: S-Slow, F-Fast, L-Linear, R-Reliable, HR-Highly Reliable, N-Normal, UR-Unreliable, HSE-Highly Sensitive, SE-Sensitive, ISE-Insensitive, ST-Stable, UST-Unstable

Validation

As one of the validation steps, this section introduces the comparison of the actual Percentage of Work Complete (PWC) obtained weekly from the site with the simulated PWC generated by the developed DPM model. Such comparison is illustrated in Figure 56 with the initially planned PWC¹. As of March 13th, 2005 (A in Figure 56), while the actual PWC is far behind the planned PWC, the simulated PWC from the developed model is almost following the actual PWC with only 3.38% of Root Mean Square Error (RMSE).



Till When	# of Datasets	Root Mean Square Error (%)
March 13 th , 2005	43	3.38
July 17 th , 2005	61	3.42

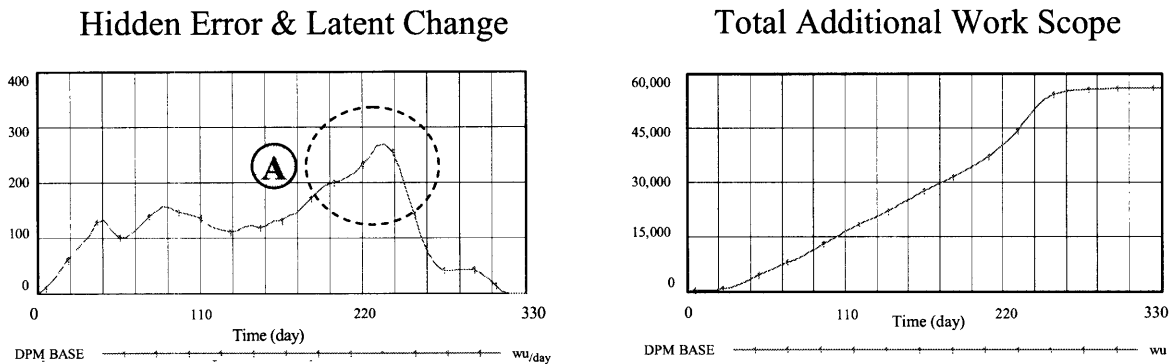
Figure 56. Planned, Actual, and Simulation Percentage of Work Complete

¹ This is Internal Schedule.

Until March 13th, 2005, this project had adopted overtime, which is one of their contingency actions when the actual progress is behind the planned. However, implementing overtime was not very effective, as seen from the gap between the planned PWC and the actual PWC in Figure 56. As such, the need for more workers was on the rise, and on March 13th, 2005, new Indonesian workers were hired to accelerate project progress. To comply with this change, the new policy was incorporated into continuous simulation, and the remaining progress is generated. As of July 17th, 2005 (B in Figure 56), RMSE was estimated as 3.42%.

Analysis

Figure 57 contains a graphical representation of the simulation result in detail. It is observed that there is significant additional work scope caused by errors and changes. Further, when the project was accelerated to catch up with schedule delay, some errors and changes became latent, which can be also observed in the simulation result (the left side of Figure 57). Particularly, the generation of hidden errors and latent changes increases and reaches a peak around or a bit after March 13th, 2005 and then, decreases, as denoted by A in Figure 57. This may explain the fact that latency introduces the gap between perceived and real performance (here, it can be said as PWC) in this project. In Figure 56, the simulated PWC is behind the actual PWC around March 13th (C in Figure 56), and this relationship is reversed (i.e., the actual PWC is behind the simulated PWC) at a later stage of the project (D in Figure 56). In other words, while the actual PWC can be considered as perceived performance, the simulated PWC can be real performance. Thus, this gap around March 13th (C in Figure 56) may be caused by hidden errors and latent changes, and they were incorporated at a later stage of the project (D in Figure 56).



* Note: WU is used as a hypothetical work unit. 1 day's work is assumed to be worth of 1,000 WU.

Figure 57. Total Additional Work Scope Caused by Hidden Error and Latency Changes

In addition, errors and changes, particularly with latency, introduced significant additional work to this project, as seen in the right side of Figure 57. The construction manager and scheduler expected that some activities had a high probability to be delayed due to errors and changes. However, the problem is that they could not incorporate this into their schedule planning (i.e., adding realism) because there is no systematic tool to address it. As a result, the construction manager and scheduler created two schedules, Internal Schedule (IS) and External Schedule (ES), respectively. ES used a large fraction of contingency schedule buffers based on the construction manager's and scheduler's intuition (e.g., 20 – 40% of activity duration). Actually, this was their answer for the expected schedule delay in order to avoid liability issues. The lack of a mechanism to incorporate negative impacts of errors and changes into schedule planning resulted in the waste of time and effort in keeping two different schedules.

Lastly, Figure 58 shows that there was a large fraction of work waiting for the Request For Information (RFI) process and Change and Claim Management (CCM) decision, which decided if identified changes were approved or not. Based on the follow-up survey of this project, the average RFI response time is estimated as 19.6 days at a later stage of the project while 10 days at an early stage of the project. This long RFI response time further delayed the resolution of errors and changes in this case project.

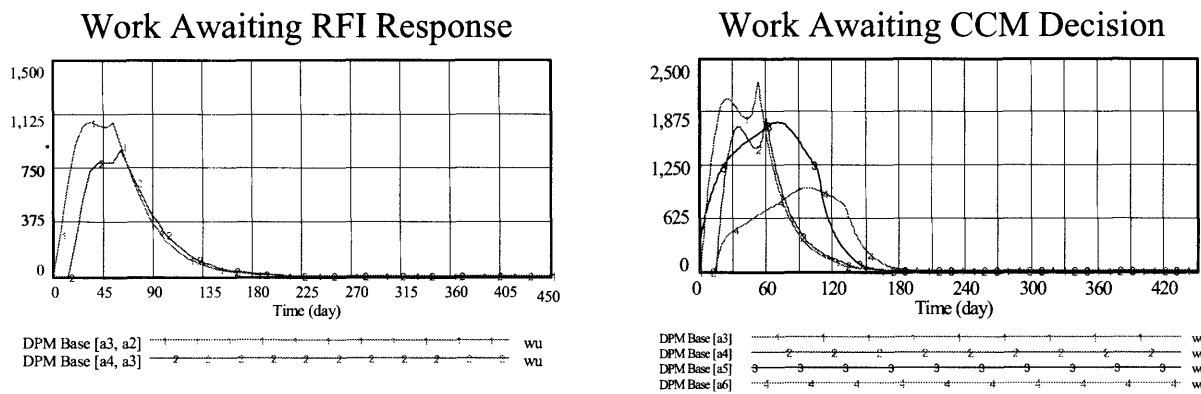


Figure 58. Coordination Work To Resolve Errors and Changes

In summary, this project experienced many errors and changes, and their significant portion became hidden or latent. Particularly, when they were re-appeared in a later stage of the project, almost doubled RFI response time (e.g., 10 days → 19.6 days) further delayed their resolution, which in turn, cause negative impacts on performance.

7.3 HIGHWAY BRIDGE INFRASTRUCTURE PROJECT IN MASSACHUSETTS

The second case project is a high bridge infrastructure project in Massachusetts. In August 2000, the Massachusetts Highway Department (MHD) awarded a \$385 million Design-Build-Operate-Transfer (DBOT) contract to Modern Continental Companies, Inc. (MCC) for roadway improvements along State Route 3 from its intersection with State Route 128 in Burlington, MA north to its terminus at the State of New Hampshire border. The design and construction phase is expected to span 42 months with beneficial occupancy of the entire roadway achieved in February, 2004. The project scope includes the renovation of 15 different underpasses and 12 distinct overpass bridges. In this paper, the Treble Cove Road Bridge construction, which is one of the overpass bridges, is presented, as seen in Figure 59¹.

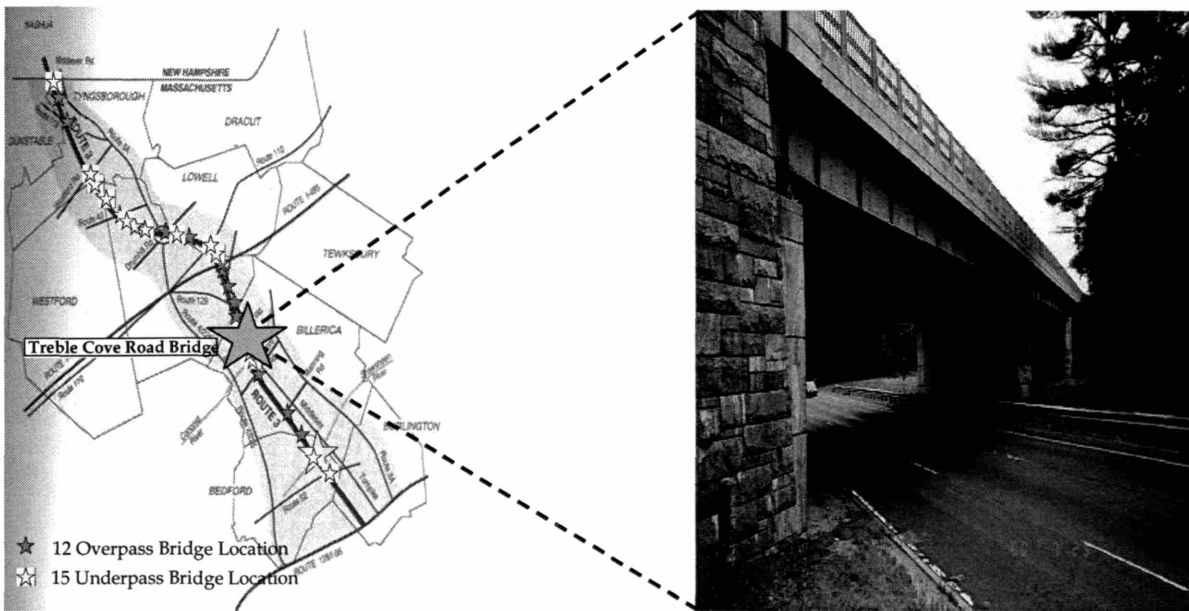


Figure 59. Project Location and Site Photo

¹ The right side of figure is the existing bridge.

With this case project, how errors and changes affect construction performance and how reliability and stability can be effective in dealing with them are explored.

Project Input

Treble Cove Road Bridge is consists of 28 design and construction activities and their simulation input is attained through the discussed input elicitation process. Table 3 is the example of the data gathered.

Table 3. Simulation Input for Treble Cove Project

Code	Activity Name	Duration (days)	Contingency Buffering (days)	Production Rate	Reliability	Sensitivity (Internal)	Sensitivity (External)	Stability
1	Sketch Plans	33	5	S	HU	ISE	ISE	HUS
2	Final Plans	66	5	S	HU	SE	SE	HUS
3	ROW Acquisition	130	10	S	R	SE	ISE	N
4	Shop Drawing Submittals	35	10	S	R	SE	SE	N
5	Shop Drawing Review/BFPads	30	10	S	U	SE	ISE	N
6	Shop Drawing Review/Struct Steel	30	20	S	U	SE	ISE	N
7	Shop Drawing Review/Rebar	30	20	S	U	SE	ISE	N
8	Shop Drawing Review/SOE Plans	30	5	S	U	SE	ISE	N
9	Steel Fabrication/Rebar	60	11	S	N	HSE	SE	N
10	Steel Fabrication/BFPads	120	10	S	N	SE	SE	ST
11	Steel Fabrication/Structural Steel	120	15	S	N	SE	SE	ST
12	Steel Fabrication/Sheet & Brace	45	18	S	N	SE	SE	ST
13	Prepare Site for Abutment E/W	33	18	F	R	SE	ISE	ST
14	Prepare Site for Center Pier	13	12	S	R	SE	ISE	ST
15	Construct Abutment E/W	30	13	S	N	N	SE	ST
16	Construct Center Pier	15	10	S	N	SE	ISE	ST
17	Set BFPads and Girders	5	31	S	N	ISE	ISE	ST
18	Construct Superstructure	20	24	S	N	ISE	ISE	ST
19	Bell Telephone Cable	80	24	S	U	N	ISE	N
20	Relocate Gas Line	15	29	S	U	N	SE	N
21	Relocate Water Line	15	35	S	U	ISE	SE	N
22	Install Telephone /db	15	0	S	U	SE	SE	ST
23	Realign Treble Cove Rd	10	23	S	R	ISE	SE	ST
24	Realign Rte 3 NB Ramps	20	24	F	R	ISE	SE	N
25	Realign Rte 3 SB Ramps	20	18	F	R	N	SE	N
26	Demolish Existing Ctr Span	10	12	S	R	HSE	ISE	N
27	Demolish Existing EAbut	10	12	S	R	HSE	ISE	N
28	Demolish Existing WAbut	10	19	S	R	HSE	ISE	N

*Note: S-Slow, F-Fast, L-Linear, R-Reliable, HR-Highly Reliable, N-Normal, UR-Unreliable, HSE-Highly Sensitive, SE-Sensitive, ISE-Insensitive, ST-Stable, UST-Unstable

Analysis

Prior to the DPM dynamic simulation, the DPM system generates CPM and PERT durations in order to compare data with the DPM's dynamic simulation results. In this case, CPM duration was 381 days, and PERT duration was 392 days, as seen in Table 4. A base case was then simulated to show the effect of the outstanding construction characteristics and policies on this project. This is illustrated by the example in which several activities' initial reliability and stability were estimated below 100% and where overtime was applied, as seen in Table 4.

The simulated duration of the base case was 537 working days. Though CPM and PERT provide relatively short durations, they don't consider iterative cycles caused by errors and changes since they assume 100% reliability and stability. Therefore, this difference in the project completion time could be explained by a significant amount of iterative cycles caused by errors and changes.

Table 4. Diverse Project Durations of the Case Project

Case	Duration (days)	Deviation from Base Case		Key Policy Options for Base Case
		days	percentage	
Base Case	537	-	-	<ul style="list-style-type: none"> • Overtime • RFI Period = 7 (days), CCM Period = 10 (days) • Schedule Contingency = 10% • QM Familiarity X QM Implementation = 0.9 • SM Familiarity X Completeness of Sources = 0.9
CPM	381	-156	-29%	
PERT	392	-145	-27%	

The base case generated by the DPM system accurately represented what had happened in the actual execution and enabled the bottleneck of the project to be found. For example, the case project was delayed in the manner that the base case simulation had predicted. As denoted by A in Figure 60, the case project was significantly delayed when 25% of the total progress was completed. Delay is prominent at the transition stage from design to construction.

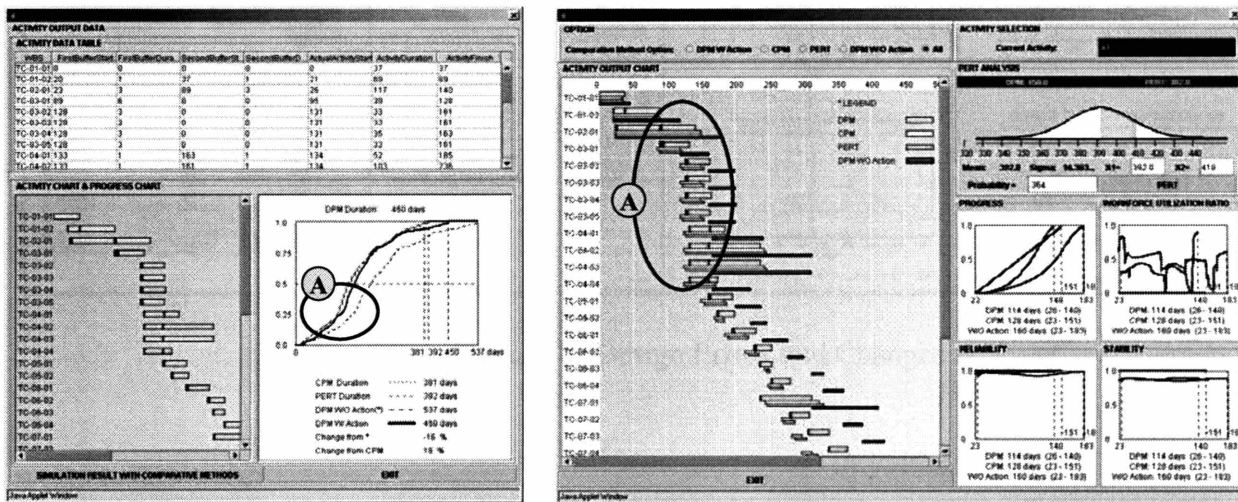


Figure 60. Simulation Results of the Case Project by Web-based DPM

More specifically, in this transition stage, the final design activity (a2) has a finish-to-start (FS) relationship with the shop drawing submittal activity (a4); and the subsequent construction activities can progress after these design activities are completed, as illustrated in Figure 61. Simulation results show that many errors and changes were generated during the final design, which introduces a large amount of additional work (Column A in Figure 61). Moreover, it is observed that a large fraction of errors and changes were not identified and became hidden errors and latent changes (Column B in Figure 61). Consequently, this situation generates a large

coordinating work amount, for example, waiting for a RFI reply and a decision about the change, between the final design activity and the shop drawing submittal activity (Column C in Figure 61).

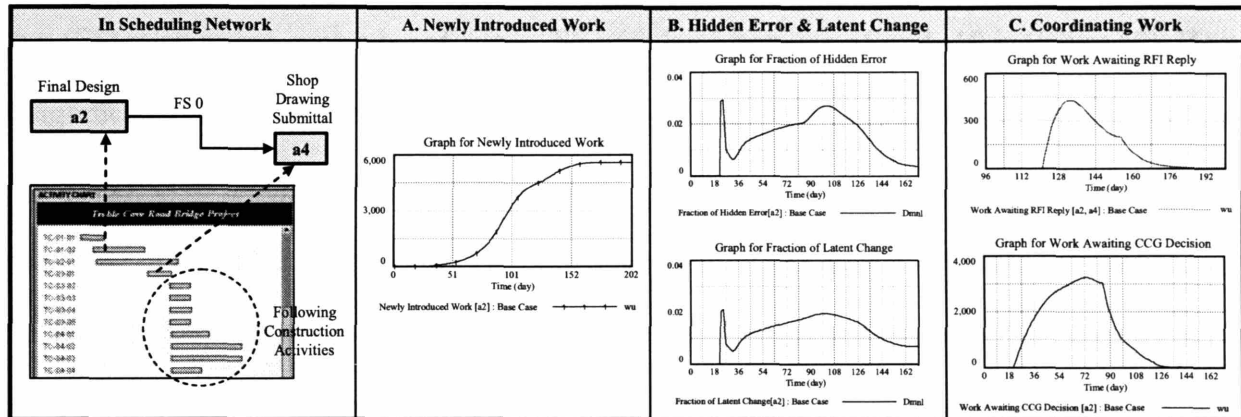


Figure 61. Impacts of Design Errors and Changes

Follow-up interviews and surveys confirmed these simulation results. Several reasons were revealed to explain this symptom; (1) design work was being released prior to the establishment of a detailed work scope. Therefore, requests for design clarifications and changes were frequently made by the contractor, which resulted in iterative design corrections, (2) the team did not have much experience on working together as part of a design-build team¹, thereby experiencing a difficulty in coordinating emerging issues. Actually, this case project was the first design-build contract for the members of the development team, from the owner's side. This led to a corresponding delay as of 25% of the completion of the total progress (A in Figure 60).

¹ A single contract between owner and design-builder and thus, one in which design and construction are performed by one team.

Errors and changes rooted in the final design activity (i.e., hidden errors and latent changes) also cause consequent detriment to the following construction activities. For example, each of the shop drawing review activities (e.g., a5 – a8 in Figure 62) and each of the corresponding fabrication activities (e.g., a9 – a12 in Figure 62) were developed concurrently; with a start-to-start (SS) relationship, with a lag of 5 (e.g., SS with lag 5 between a5 and a10 in Figure 62). Here, a significant amount of additional work was introduced with the fabrication activities (A in Figure 62), even though they have higher initial reliability (e.g., 95%) and stability (e.g., 95%) than those of the preceding shop drawing submittal activities (e.g., 90%). This is mostly because the final design, the source that introduced the original errors and changes, had already been completed; and in turn, the fabrication activities have to incorporate these design errors and changes. This situation clearly shows the detrimental impacts of hidden errors and latent changes on the remaining construction performance (i.e., the late discovery of errors and changes).

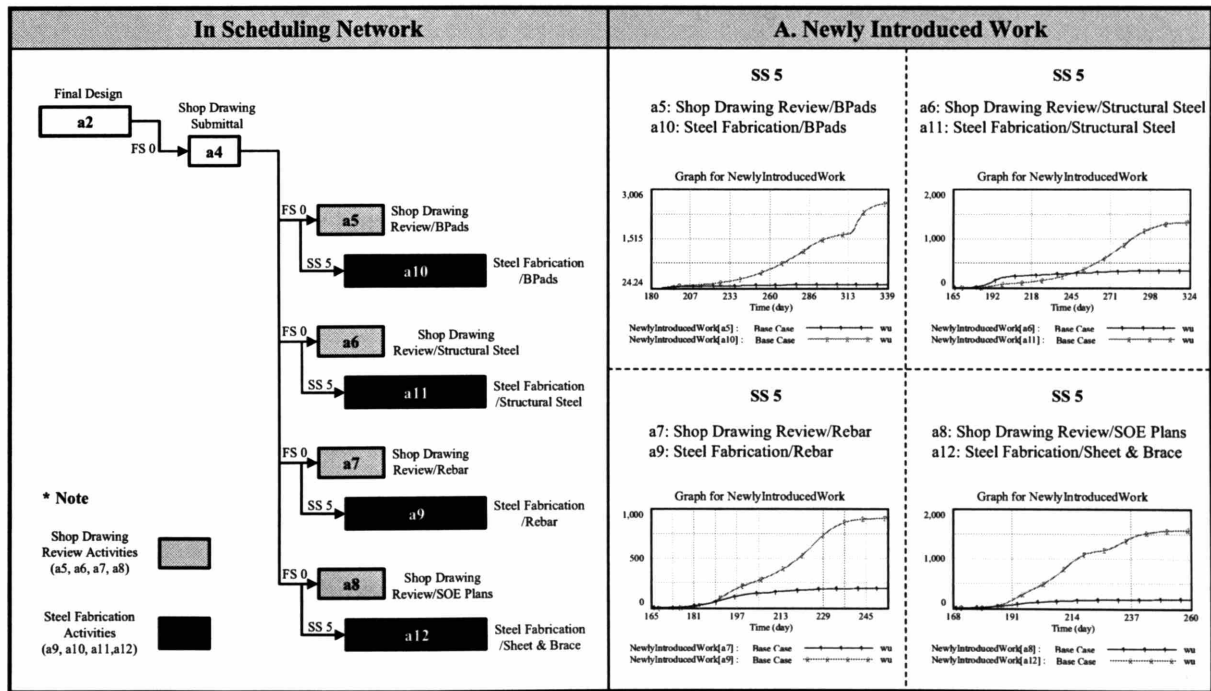


Figure 62. Increased Work Scope in Succeeding Activities

On the other hand, the impacts of different policy options on project completion time and cumulative labor hours, this project was simulated by applying diverse policies, as seen in Table 5. Extending the policy boundary to managerial efforts, a flexible headcount policy (workforce can be hired whenever required during actual execution) can help reduce project completion time more than the outstanding overtime policy (519 days in Table 5). Although overtime has been widely adopted in construction, its effect may not be strongly positive in some situations, in terms of project duration and cumulative labor hours as seen in Table 5. This is because applying overtime may generate a productivity loss, as the workforce's fatigue becomes accumulated. Thus, it may generate more cumulative labor hours than the case when overtime is not applied (1.241M → 1.332M workers x hours in Table 5). Although, the application of overtime could reduce project completion time (552 → 537 in Table 5). Previous studies from the construction industry (and other manual labor contexts) indicate that long work hours begin to reduce productivity after a week or two, with the full effect requiring a somewhat longer duration [Oliva, 1996]. In addition, reduction of the RFI response time (RFI Period) and change decision time (CCM Period) can contribute to shortening the project completion time (499 days in Table 5) and reducing cumulative labor hours (1.126M workers x hours in Table 5). In reality, it is difficult to achieve such a reduction because this is related to the whole process management at the organizational level. However, by increasing the level of coordination among project functions, the RFI response time and change decision time could be decreased. One of the important lessons learned from these diverse policy simulations is that the DPM simulation results could be more effective when combined with other managerial efforts such as applying different working policies and increasing the level of coordination.

Table 5. Effects of Diverse Policy Options

Cases				Project Duration (days)	Labor Hours (workers x hour)
Overtime	Flexible Head Count Control	RFI Period (days)	CCM Period (days)		
No	No	7	10	552	1.241M
Yes	No	7	10	537	1.332M
No	Yes	7	10	519	1.179M
Yes	Yes	7	10	517	1.167M
Yes	Yes	5	7	499	1.126M

Effectiveness of Reliability and Stability Buffering

In this section, the means through which reliability and stability buffering can be effective in absorbing the impact of errors and changes is explored. First, the effectiveness of reliability and stability buffering is investigated with the experimentation of the hypothetical case and then, it is applied to this case project.

Experimentation with the Hypothetic Case

Prior to applying to this case project, three different cases were simulated with two simple activities, a final design and an excavation activity, as seen in Figure 63. The detailed simulation setting is listed in Figure 63, showing duration, precedence relationship, production type, reliability, stability, sensitivity, the RFI period, the CCM period, and latency. The base case

utilizes contingency buffering (A in Figure 63) while the other two cases incorporate a single buffer case (reliability and stability buffering without split, B in Figure 63) and a dual buffer case (reliability and stability buffering with split, C in Figure 63), with the assumption that the rest of simulation settings are identical to the others. In these buffering cases, 10% of the activity duration (i.e., 6 days) is used for schedule contingency since this is the common practice of the companies with which the research team worked.

Simulation Variable		Final Design	Excavation
Duration (Days)		60	60
Precedence Relationship		Start-to-Start 30 (Lag)	
Production Type		Slow	Slow
Internal & External Sensitivity		1	1
RFI Period (Day)		5	5
CCM Period (Day)		7	7
Quality Management Thoroughness		0.9	0.9
Scope Management Thoroughness		0.9	0.9
Policy Option		No overtime	
		Inflexible Headcount	
Buffering Setting	Schedule Contingency	10%	10%
	Fraction of Buffering	NA	67%
	Single Buffer: First Buffer Size	NA	4 days
	Dual Buffer: First Buffer Size	NA	2 days
	Dual Buffer: Second Buffer Size	NA	2 days
Simulation Result	Base Case (144 days)	90 days	114 days
	Single Buffer Case (137 days)	85 days	107 days
	Dual Buffer Case (120 days)	78 days	90 days

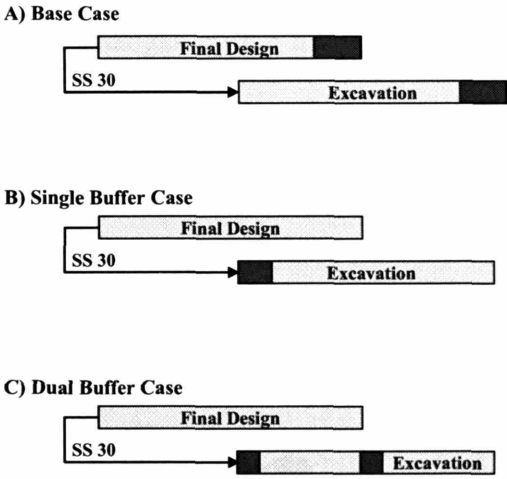


Figure 63. Simulation Setting and Results

The cases applying the reliability and stability buffer (single buffer case – 137 days and dual buffer case – 120 days) shows better performance over the simulated base case (144 days), as seen at the bottom of Figure 63. This improvement results mainly because error and change impacts of the final design activity are absorbed during the buffer periods in the excavation activity. In addition, it was noted that the dual buffer reduced the total completion time by as much as 17% from the base case (144 → 120 days) and by 14% in the single buffer case (137 →

120 days). This improvement was achieved because the second buffer dealt with the error and change impact of the later part of the final design activity in concurrent development, where the first buffer couldn't.

In more detail, the hidden errors and latent changes of the final design activity are reduced by virtue of their early discovery during the first and second buffer period, which is denoted by A and B in Figure 64, respectively. Here, the fraction of hidden errors and latent changes are arrived at by dividing the amount of hidden errors or latent changes by the total work scope. Such reduction in hidden errors and latent changes could save time and effort for coordination, which is mainly achieved through the RFI process and the claim and change management process.

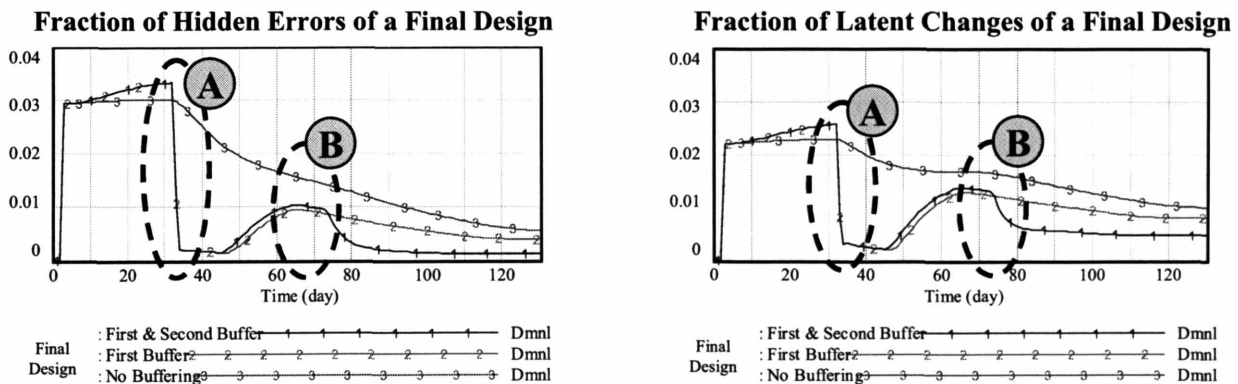


Figure 64. Reduction of Hidden Errors and Latent Changes during the Buffer Period

Further, reduction in hidden errors and latent changes also helps to avoid harmful ripple effects on the remaining project performance. For example, it is observed that reliability and

stability of the successor excavation are improved during the first and second buffer period because of such reduction, which is denoted by A and B in Figure 65, respectively. These reduced fractions of hidden errors and latent changes, together with the recovery of and increase in reliability and stability, ultimately improve project performance by decreasing overall errors and changes.

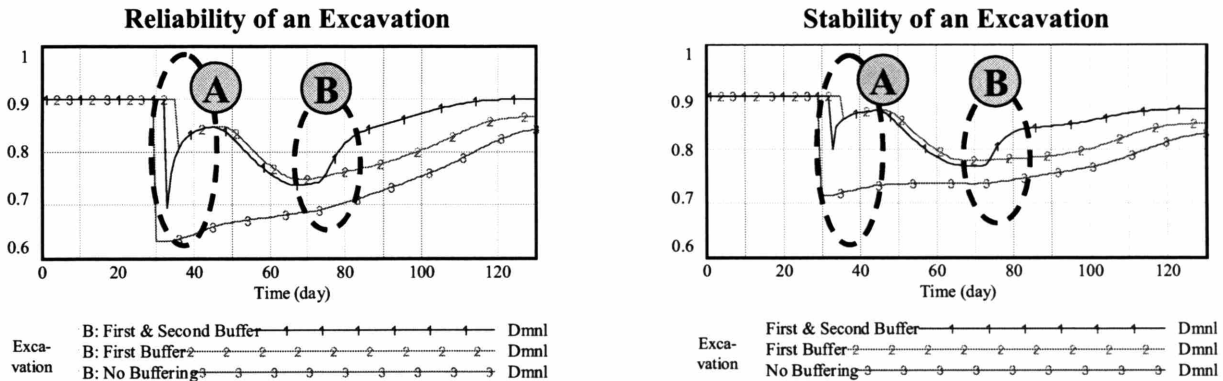
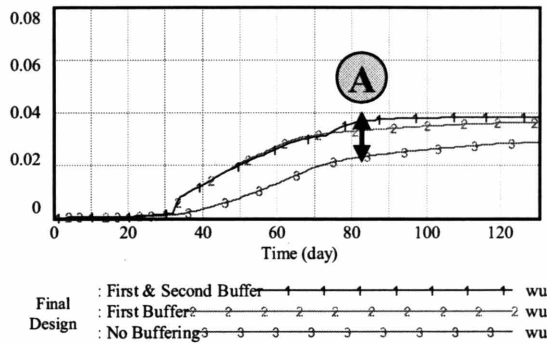


Figure 65. Improvement of Reliability and Stability during the Buffer Period

On the other hand, the amount of newly introduced work for the predecessor final design activity in the dual buffer case is much greater than that in the original case, as denoted by A in Figure 66. This is because proactive reliability and stability buffering enables early adoption of errors and changes. In later stages, however, it makes the amount of newly introduced work for the successor excavation activity less than that in the original case, as denoted by B in Figure 66. This effect circumvents the 90% syndrome, and ultimately, enables a reduction in total project duration.

Newly Introduced Work of a Final Design



Newly Introduced Work of an Excavation

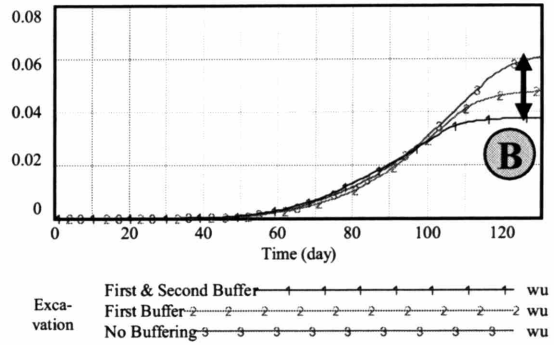


Figure 66. Early Adoption of Errors and Changes

Experimentation with the Case Project

To investigate the effectiveness of reliability and stability buffering on a case project, an evaluation was made on both the case that applied traditional contingency buffering (base case) and the case that applied reliability and stability buffering (RSB case). The former, the base case, also represents the currently outstanding construction characteristics and policies while the RSB case was designed to study the effect of reliability and stability buffering on the case project by replacing the traditional contingency buffering with reliability and stability buffering. By doing this, a comparison could be made on how reliability and stability buffering affected the current project performance.

In order to reduce this detrimental impact, the reliability and stability buffering (RSB) case is simulated, while applying the reliability and stability buffer. Buffers are located at the beginning of activities and buffer splitting is implemented where activities are concurrently developed. In

addition, a different fraction of the taken-off contingency buffers is assigned to activities based on their characteristics. The simulation result of this case resulted with 477 working days, which was shorter than the base case by as much as 60 days, as shown in Table 4 and Figure 67. This reduction is enabled by the fact that the amount of hidden errors and latent changes from the predecessor activities was reduced by virtue of the reliability and stability buffer's proactive location, allowing for more time and effort to discover errors and changes (e.g., reduction of hidden errors in the final design activity - A in Figure 67). Moreover, the buffer split helped identify the predecessors' errors and changes when these activities are developed concurrently (e.g., the amount of 'Work Awaiting RFI Reply' between the fabrication activity (a10) and the shop drawing activity (a5) shows two sharp increases due to assigning two buffers in the fabrication activity - B in Figure 67).

On the other hand, although this early identification of hidden errors and latent changes could generate more additional work than the base case (e.g., early introduction of additional work in the final design activity - C in Figure 67), the fact that they are identified early prevents these errors and changes from propagating to the succeeding activities, and ultimately, reduces their being influenced by a detrimental ripple effect. In addition, it provides the mechanism to coordinate these errors and changes prior to construction, thereby facilitating their processing.

In summary, the case project was vulnerable to errors and changes due to several reasons, as previously discussed. If errors and changes are inevitable, implementation of a redundant capacity which is based on the systematic approach, could protect the planned performance

against the uncertainty caused by errors and change. The results of the case project, which applied the proposed buffering approach, confirmed that the strategic use of buffers could be effective in protecting the planned performance.

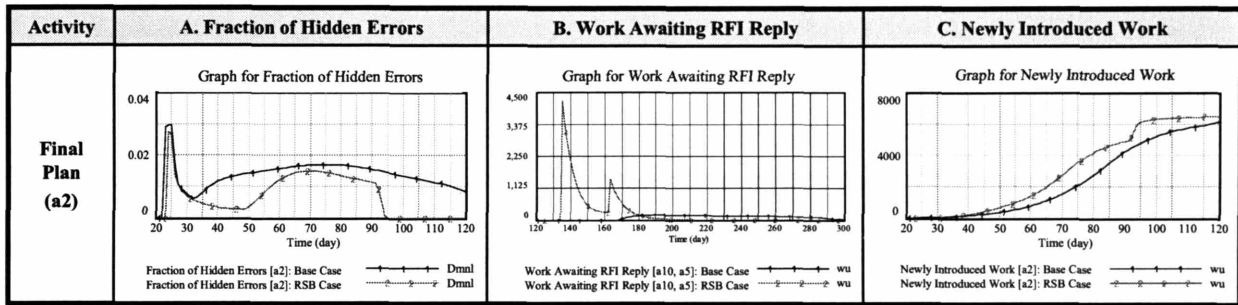


Figure 67. Effectiveness of Reliability and Stability Buffer

CHAPTER 8

CONCLUSIONS

Based on diverse simulation experiments and application to a couple of real-world case projects, this chapter discusses how we can understand and manage the behavior generated from errors and changes and further, what their implications are as a concluding remark.

Realistic Planning

Errors and changes most likely occur in construction and further, can be latent. Thus, taking into account errors and changes has a significant meaning in managing a project because additional work scope generated by errors and changes could entirely disrupt the entire prepared plan, particularly in concurrent design and construction. Thus, there is a strong need to be well-prepared for such uncertainties and complexities. However, prior to the discussion of managing these detrimental errors and changes, the more significant point would be the understanding of the mechanism associated with errors and changes. This is because if we have a good understanding of how they work, we will have a better chance to make good policies against them. The developed framework discussed in Chapter 3 serves exactly for this purpose.

Based on the clear understanding of errors and changes, the next step would be the

development of realistic contingency plans. In the SIRIM project, its contingency plan against the increase of work scope was the adoption of overtime. However, due to a lack of the ability to estimate the actual need for overtime in advance, the scope issue was not handled well. Thus, the management team decided to hire new workers from an adjacent foreign country, a decision not in their contingency plan. Although the hiring process proved time consuming, when the new workers finally did start, the results were positive due to the increase in the work done. However, some time later, the newly hired workers called a strike when they realized that they had received considerably less payment than the domestic workers. The strike had the potential to cause severe schedule and cost overrun. Fortunately, this strike was resolved early, but it cost a lot. This case example shows the importance of the early consideration of additional work scope that can occur during actual execution. This ultimately contributes to the development of a realistic contingency plan based on an accurate estimate of project performance.

Efficient Coordination

From the previous simulation experiments, it can be surmised that the time taken for the RFI and CCM processes could delay work execution. This implies that implementing an efficient coordination process can reduce the detrimental impact of errors and changes. If a project cannot avoid the generation of errors and changes, the second best policy would be to improve the coordination process that deals with errors and changes. The SIRIM project and the Treble Cove project indicate that the RFI response time and the CCM period could take on average over 20 days. In such cases, the impact could be more detrimental than simply delaying the project

completion. Considering the involvement of many multiple temporary organizations (e.g., subcontractors) in a construction project, implementing and maintaining an efficient coordination process also contributes to streamlining information flow. Particularly, since the existence of errors and changes in uncertain and complex projects is inevitable, providing an efficient mechanism to coordinate errors and changes would be an appropriate means to rectify them.

As discussed earlier, the Treble Cove project involved a design-build contract. In this contract, the coordination process between design and construction was supposed to work well in this contract because a design team could work continuously to correct their design errors and to implement changes without causing any additional cost, contract, or liability issues. However, it turned out that this project was the first design-build contract in which these two groups worked together. Due to the lack of experience working with a design-build contract, there was a significant delay in coordinating the settlement of errors and changes through the RFI and CCM processes. This example highlights the importance of accounting for the increase in the level of coordination among project functions (i.e., reducing RFI response time and CCM decision time) in managing errors and changes, even though the contractual solution (e.g., a design-build contract) might in theory provide for easier coordination. Particularly, in case of latency, there is a heavy communication and coordination load to resolve the issues so that providing an efficient mechanism to coordinate errors and changes would rectify them.

Proactive Approach

As discussed earlier, latency can explain the sudden work overflow at a later stage of a project. This detrimental impact of latency implies that current contingency plans need to be approached differently. In other words, most contingency plans in construction are based on a reactive approach; acting in response to something. Contingency plans can help absorb the detrimental impact of latency by accelerating the progress, but they do not provide for any preventions of a possible disruption; rather, they just give more resources to recover from a disruption. One possible way to overcome such situations is the adoption of a proactive buffering that aims to look ahead at possible problems and issues. Such proactive buffering could be implemented in the form of a collaborative meeting with the design group and the involved subcontractors before an activity starts. The meeting would be designed to share and discuss potential problems, such as clarifying ill-defined tasks (e.g., designs), ensuring the accuracy and constructability of drawings, and securing resource procurement. Through collaborative efforts directed at reducing potential problems in advance, opportunities for identifying hidden errors and latent changes would be increased, ultimately reducing the number of time consuming RFIs.

System Dynamic Model-based Project Management

Network-based tools have been widely used in the construction industry, particularly, due to their ease of applicability. However, they assume that the attributes of project activities such as duration are known at the beginning of a project and do not change during the project execution.

Thus, they are not adequate for representing the actual project progress, which results in frequent manual updates to the schedule to reflect the actual performance in the schedule [Martinez and Ioannou, 1997].

By contrast, SD's focus on the system structure to understand dynamics behavior, can greatly contribute to management of a project. Taking into account frequent errors and changes and the resultant deviation between expected and actual performance, the understanding of the system structure can provide a great opportunity to make good management policy. This clearly differs from network based tools' rigid and static structure. Further, the introduction of the system structure (e.g., how errors and changes affect construction performance) to the project manager, helped them to understand their project in-depth and to incorporate this understanding into managing their project. This is particularly true when a project is constituted by heavy concurrency or complex inter-relationships among phases, because these situations make it difficult to identify the consequences of cause and effect. In this case, the SD project model, which is based on the feedback process, can be beneficially utilized to manage latency by virtue of its capability to identify the complex system structure and to simulate possible scenarios applying different policies.

Nonetheless, the applicability of SD to the real world projects is still in question because of the lack of detailed operational explanations [Rodrigues and Bowers, 1996]. To overcome this situation, this research integrated traditional network-based tools and a SD-based project model. Not only does the SD model incorporate concepts in network-based tools, but also the output of

the SD model is translated into the network-based tools. For example, the developed web-based system can represent the impact of errors and changes on schedule network¹ as well as dynamic behaviors generated by the SD model. As Rodrigues and Bowers [1996] pointed out, incorporating quantitative data from SD simulation into the network-based tools provided industry practitioners with more familiarity with SD simulation.

On the other hand, SD, rather than DES, was used to model and simulate the construction process associated with errors and changes. Through this research, it is concluded that SD has a great potential to be used in construction simulations, particularly in strategic simulation due to SD's emphasis on the model structure, the aggregate behavior, soft variables, and consistent time step in simulation clock.

8.1 POTENTIAL IMPACT

The research findings obtained show that DPM has great potential impact. First, it can increase the planning and control capability of large-scale concurrent design and construction projects providing a robust plan and policy guideline against uncertainties. Considering uncertainties and complexities when the project becomes large-scaled and its activities are heavily overlapped, there is a strong need to incorporate such uncertainties and complexities into planning and control. DPM's ability to represent construction realistically and to manage it proactively can enhance its robustness. Second, DPM can enhance learning in projects with an analytic

¹ SD's analysis of the impact of errors and changes was translated into schedule network.

framework and consequent quantitative simulation. It is known that learning has rarely accumulated across construction projects [Park, 2003]. One of the major reasons is that the dynamic nature of construction and its mechanisms have not been comprehensively addressed. Still, the majority of planning tools in industry is based on the static and rigid network-based tools. In this context, DPM's ability to capture the dynamic nature of the construction process and its underlying framework that describes the construction process associated with errors and changes can greatly contribute to learning in managing the project. Third, DPM shifts the project management paradigm from a prevalent impromptu and reactive approach to a proactive approach. If everything is static and well-predicted, there would be no need to take a proactive approach. However, if the dynamics and uncertainty heavily matter, there is a strong need to look ahead for possible problems and prepare a realistic contingency plan to deal with them. This will eventually save efforts and costs in the long run. Finally, all of these features of DPM enable it to contribute to society by providing an efficient process to deliver safe built environments, especially in critically needed infrastructure projects such as hospitals, schools, roads, bridges, transportation centers and utility systems.

CHAPTER 9

FUTURE RESEARCH

Based on the research discussed so far, this chapter discusses the future research opportunities. Design and construction are very dynamic and complex processes. When large-scale, concurrent development and globalization are considered, the associated uncertainty and complexity can introduce serious schedule delay, cost overrun, quality degradation, and safety concerns. In an effort to address these issues, future research opportunities are directed to understand the dynamics and complexity of design and construction processes, and the development of mechanisms to absorb these impacts. Considering the current challenges that the A/E/C industry' is confronting due to such a dynamic and complex environment, the author believes that the research findings in these areas would contribute to the improvement of the industry's performance, while increasing the quality and safety of its operations.

To realize these objectives, the followings are proposed: (1) *comprehensive simulation framework* to systematically understand design and construction performance in large-scale concurrent design and construction, (2) the *application of latency and a proactive buffering approach* to manage uncertainty and complexity in design and construction in advance, and (3) the implementation of mechanisms to *effectively visualize voluminous construction information* to facilitate the coordination process.

9.1 COMPREHENSIVE SIMULATION FRAMEWORK

Considering complexity and uncertainty in design and construction projects, they need to be approached holistically from both a strategic level, which focuses on feedback dynamics, long term trends, and strategic decisions; as well as from an operational level. Understanding how a project is structured, what kind of environment a project belongs to (e.g., organizational and financial structure), and how a project is carried out (i.e., operational detail), is as equally important as obtaining a thorough understanding of a construction project.

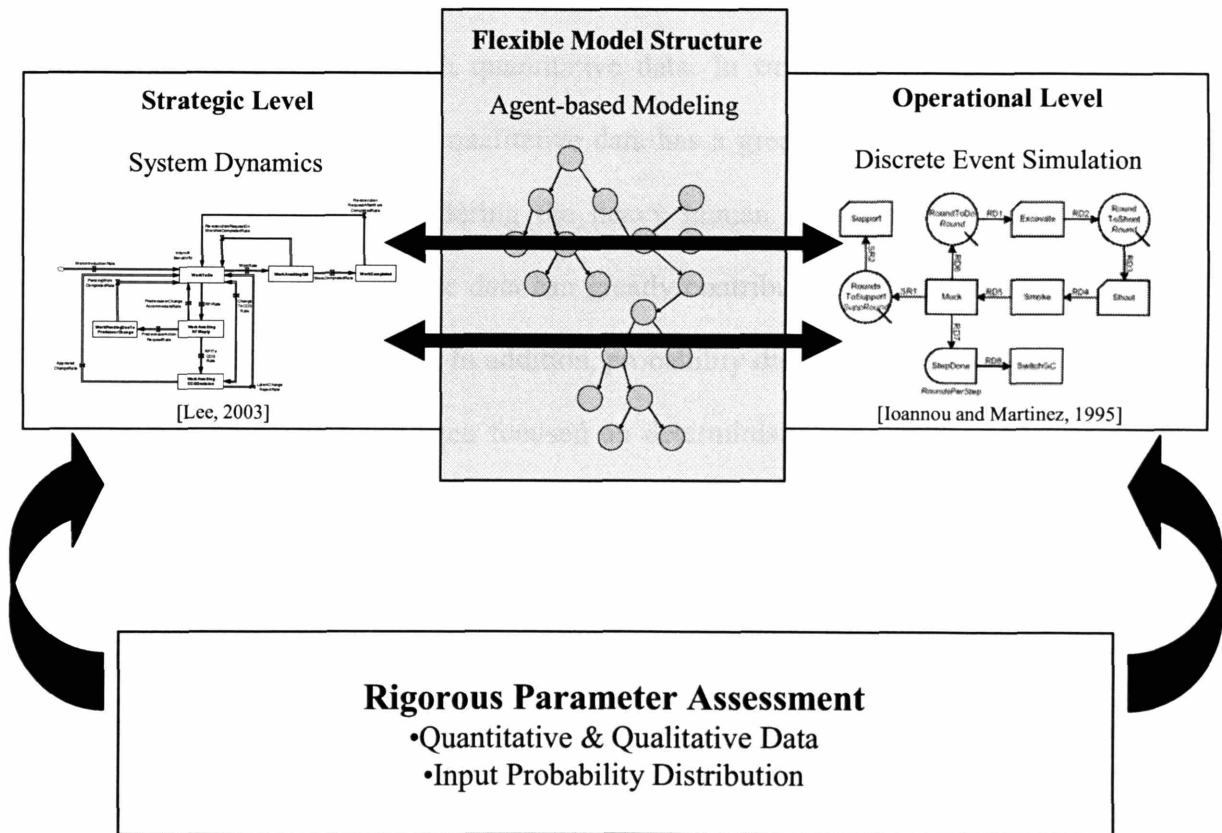


Figure 68. Comprehensive Simulation Framework

To achieve this, a comprehensive simulation framework is proposed integrating different simulation methodologies (i.e., hybrid simulation) and implementing rigorous parameter assessment, as illustrated in Figure 68. Specifically, hybrid simulation aims to take advantage of the strengths of three disciplines: the strategic insight from system dynamics, the adaptive and autonomous model structure from agent-based modeling, and the operational accuracy from discrete event simulation. Thus, more realistic representation of a large-scale project can be done incorporating its context and environment.

On the other hand, in terms of rigorous parameter assessment, qualitative data can be used for simulation input together with quantitative data. In traditional engineering, most data is quantitative. However, the use of qualitative data has a great potential to put quantitative data into context. In particular, considering the heavy human involvement in most construction projects, the adoption of qualitative data can greatly contribute to the understanding of human-related project performance issues. In addition, probability distribution can be used for stochastic simulation. Thus far, DPM has been focused on deterministic simulation. However, DPM can also do stochastic simulation incorporating associated uncertainties. For example, as seen in Figure 69, probability distribution can be applied to the variables in the developed SD model. In this example, RFI response time (A in Figure 69) is assumed to approximately follow beta distribution, which is obtained from one of the author's case projects while QM Thoroughness (B in Figure 69) and Productivity (C in Figure 69) follow uniform distribution [Chong and Low, 2005] and lognormal distribution [Maio et al., 2000] that are obtained from the literature. Based on these probability distributions, multi-variate Monte Carlo simulation was performed with 10,

000 runs.

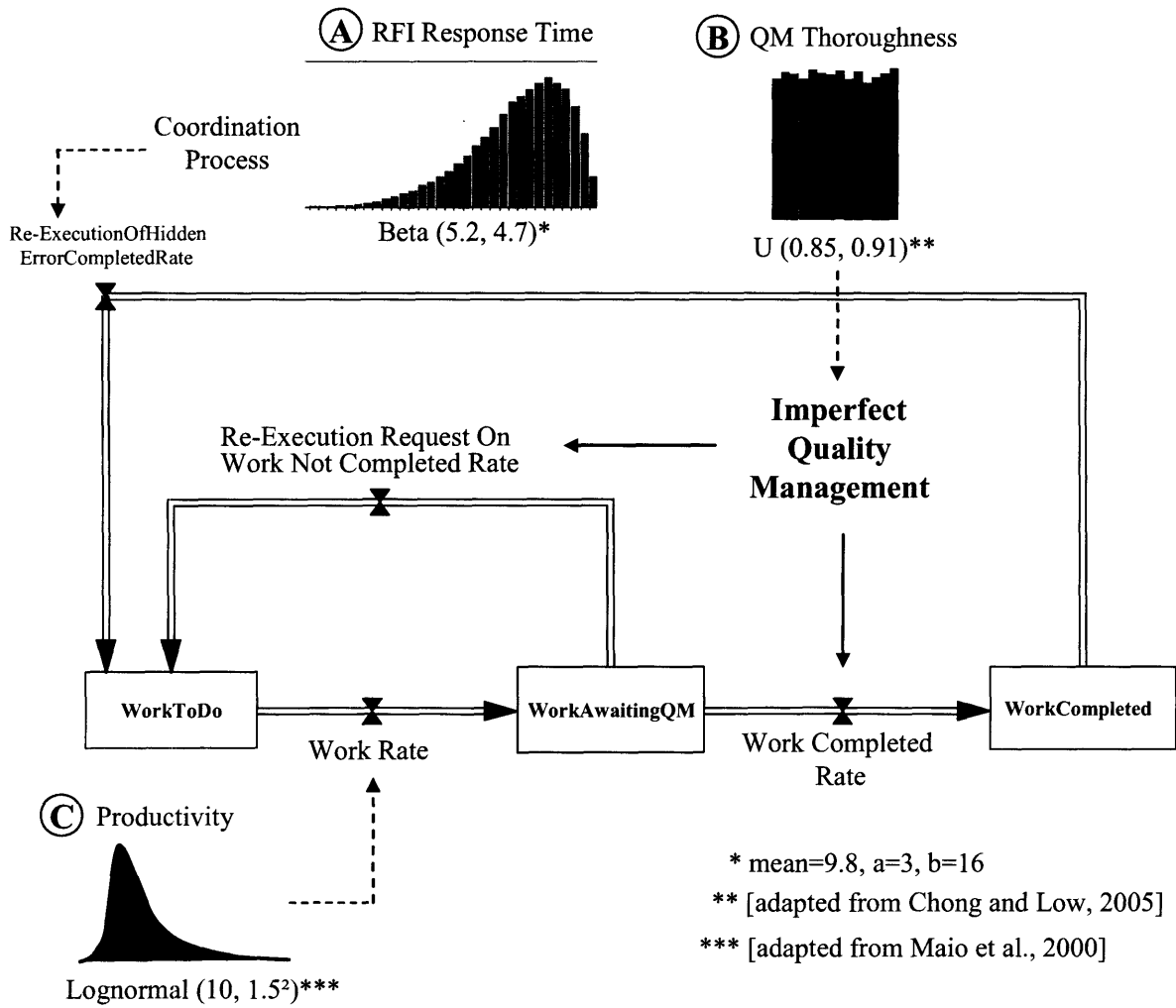


Figure 69. DPM's Extension to Stochastic Simulation

Figure 70 shows the simulation results obtained from this stochastic simulation. The left side graph shows total additional work scope with a certain confidence bound regarding the time. On the other hand, the right side graph represents the frequency, the number of simulation runs,

as related to total additional work scope.

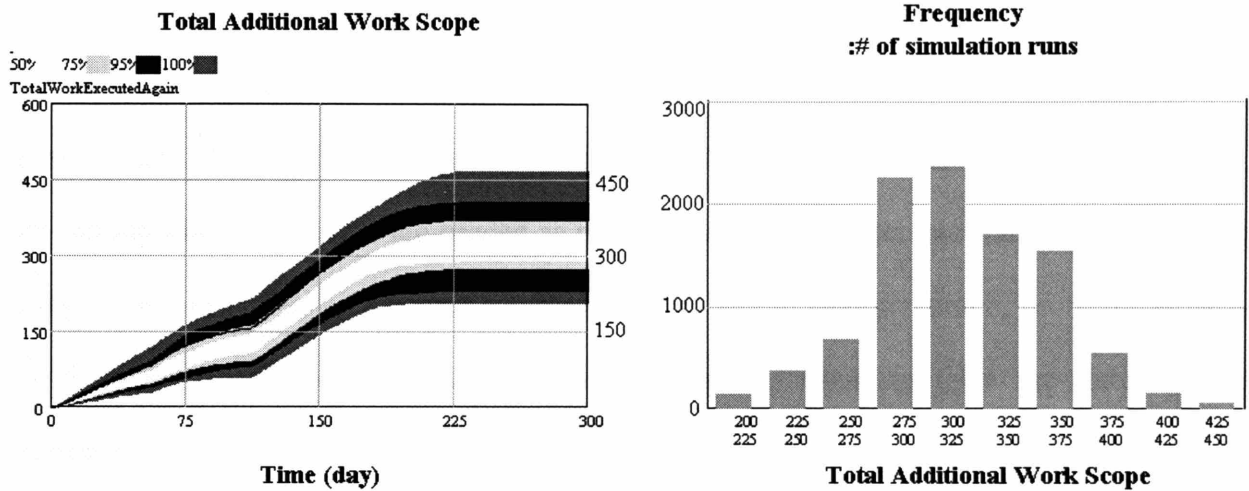


Figure 70. Examples of Stochastic Simulation Results

In summary, taking into account complexity and uncertainty in modern design and construction projects, the comprehensive simulation framework, which includes hybrid simulation and rigorous parameter assessment, is proposed to help the understanding and management of uncertainty.

9.2 APPLICATION OF LATENCY AND PROACTIVE BUFFERING

Another potential future research area would be the implementation of a proactive mechanism to absorb and prevent possible problems (e.g., latency) associated with uncertainty and complexity in design and construction projects. In particular, latency is not a phenomenon which is confined to performance in construction. We can see many situations where latency causes an information

gap in managing a project. For example, suppose it is estimated that 2,000 bricks are needed in installing masonry wall and an error is uncovered equivalent to 100 bricks. In this case, we can say that perceived resource (i.e., the number of bricks) is 2,100 bricks while planned resource is 2,000 bricks. Further, if we assume that there is a hidden error equivalent to another 100 bricks, we can say that 2,200 bricks are needed as real resource to complete this wall. In this case, the proactive use of buffers on resource can greatly contribute to identification of problems as early as possible and protect the succeeding resources implementing their pre-planning and mobilization.

On the other hand, the information gap caused by latency is also observed in other areas. The writers conducted research with an automobile company to figure out the cause for their chronic problem in a vehicle prototype build process (for confidentiality reasons, the company's name was not stated). The problem is that 300 to 400 changes to the global Bill Of Materials (BOM) are introduced every week during the last sixteen to twenty weeks of the vehicle prototype-build process. These changes cause major delays on the prototype initial build and generate over \$3 Million of scrap parts per month. More importantly, these changes could also raise significant quality issues in the prototype initial build.

Based on the preliminary analysis, the impact of latency on BOM was identified as a major reason for delay and scrap parts. Due to frequent (last-minute) design changes with the concurrent development of several tasks, most tasks are often forced to proceed without complete information from precedent tasks and accelerate the decision making process. Thus,

many changes became latent. Needless to say the gap between planned BOM and perceived BOM due to changes, the gap between perceived BOM and real BOM was introduced and became large during the last weeks of the prototype build process. To catch up with this gap caused by latency, more scrap parts and schedule delay were generated, which consequently degrades the quality in this build process.

As seen in these examples, latency can be found in and applied to many situations and particularly, when time and resource constraint is high, latency is prevalent. In this situation, the proactive use of buffers can greatly contribute to the management of these complex issues, such as production control, logistics, and equipment constraints.

9.3 VISUALIZATION BASED ON THE USER'S CAPACITY AND NEED

During the project, we often observe that there is a discrepancy between planned and actual progress. In this case, appropriate control actions need be made in a timely manner in order to minimize the discrepancy. In this sense, an efficient means of representing this discrepancy is one of the keys to support the decision making process for control actions. Furthermore, this decision making process often involves parties who may not have much knowledge of construction situations such as the owner and the end user. Thus, it may take a considerable time before the problem is understood in the decision making process. One study reports that 90% of the time in construction meetings devoted to describing the problem, explaining the rationale of decisions, and evaluating goals to be sure requirements are being met. Only 10% of time is spent

on the real decision-making such as the discussion about what-if scenarios [ENR, 2005]. One of the major reasons for this is the limitation of mediums that represent this discrepancy. For example, text information (e.g., description of inspection results), graphs or charts (e.g., progress S curve), and still photos are typically used to monitor results in the construction industry. However, these may not be sufficient to intuitively show any discrepancy. The fact that it is difficult to understand the situation clearly and quickly with the current formats is particularly troubling, considering the need for frequent remote and quick decisions in construction. As such, the effective representation of progress discrepancy is needed to save time spent on time-consuming preparations for the actual discussion in the meeting.

As an effort to address this issue, the effective visualization of construction progress is proposed. Visualization has been reported as an effective monitoring process [Kamat and Martinez, 2003] and currently, considerable research is ongoing. In this section, diverse visualization techniques are suggested to intuitively identify the discrepancy between as-planned and as-built data. Particularly, the main focus of this research opportunity is visualization based on the user's capacity and need. As discussed earlier, certain construction information such as complex schedule networking may not be helpful to the involved parties who don't have sufficient knowledge base. Thus, when visualization takes into account the user's capacity and need, the benefit of visualization can be maximized. Keeping this in mind, this research explores diverse visualization techniques and discusses their application to progress monitoring.

Metaphor

In some cases, excessively detailed visualization may not be necessary. For example, when high level management personnel are interested in the progress of a project, their main concern would be projects' general performance as measured by schedule and cost. In this case, metaphor can be used to represent the status of schedule and cost of the projects [Song et al., 2005]. Suppose the chief manager in the Department of Transportation (DOT) is taking care of a bridge project. The schedule and cost performance of a bridge can be represented by a visual motion metaphor in a small window in a computer screen: weather represents schedule and earthquake represents cost. As seen in Figure 71, if it rains, there is schedule overrun. In particular, how hard it is raining represents the severity of schedule overrun. Additionally, if the bridge is shaking due to an earthquake there is cost overrun. Of course, the degree of shaking represents its severity. Thus, in Figure 71 A denotes the situation where schedule is overrun while cost is not. Both rain and earthquake come from common perceptions in construction: they are bad for construction. In addition, the SPI (Schedule Performance Index) and CPI (Cost Performance Index) in Earned Value Analysis are displayed for further detail. Given this generally information, the chief manager can take consequent actions accordingly (e.g., getting detailed information and calling for the meeting). Thus, metaphor can be used in many situations where information in detail is not necessary.

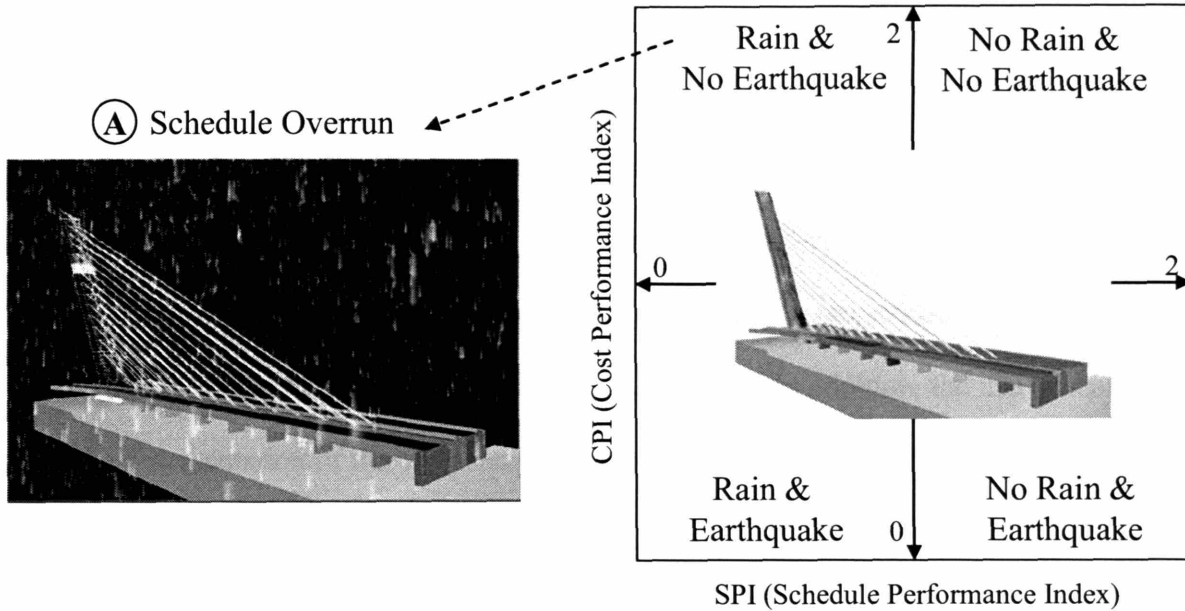


Figure 71. Visual Motion Metaphor [extended from Song et al., 2005]

Augmented Reality

Another way of representing construction progress is the use of Augmented Reality (AR). AR means putting virtual objects into real immersive environments. Applications of the blend of a virtual and a real environment are the recent focus of attention in construction. For example, Kamat and El-Tawill (2005) proposed rapid post-disaster evaluation of building damage using AR. Wang and Dunston (2005) used mixed reality for the purpose of on-the-job and off-site training programs and facilitating the collaboration process for design and construction. However, applications of mixed reality (or augmented reality) to the progress monitoring have not yet been addressed.

In progress monitoring, an as-planned image, obtained from a 3D model, is superimposed on the as-built image, obtained from the construction site. For example, Figure 72 illustrates the planned 3D model of the building superimposed on the real construction image as of today. Given the superimposition of the images, work completed and work remaining can clearly be visualized. Thus, superimposing the images provides a clear comparison between what was intended (i.e., as-planned) and the current state (i.e., as-built). In addition, the use of real images enables the representation of the temporary structure at the site so it can be useful to understand the current situation better.

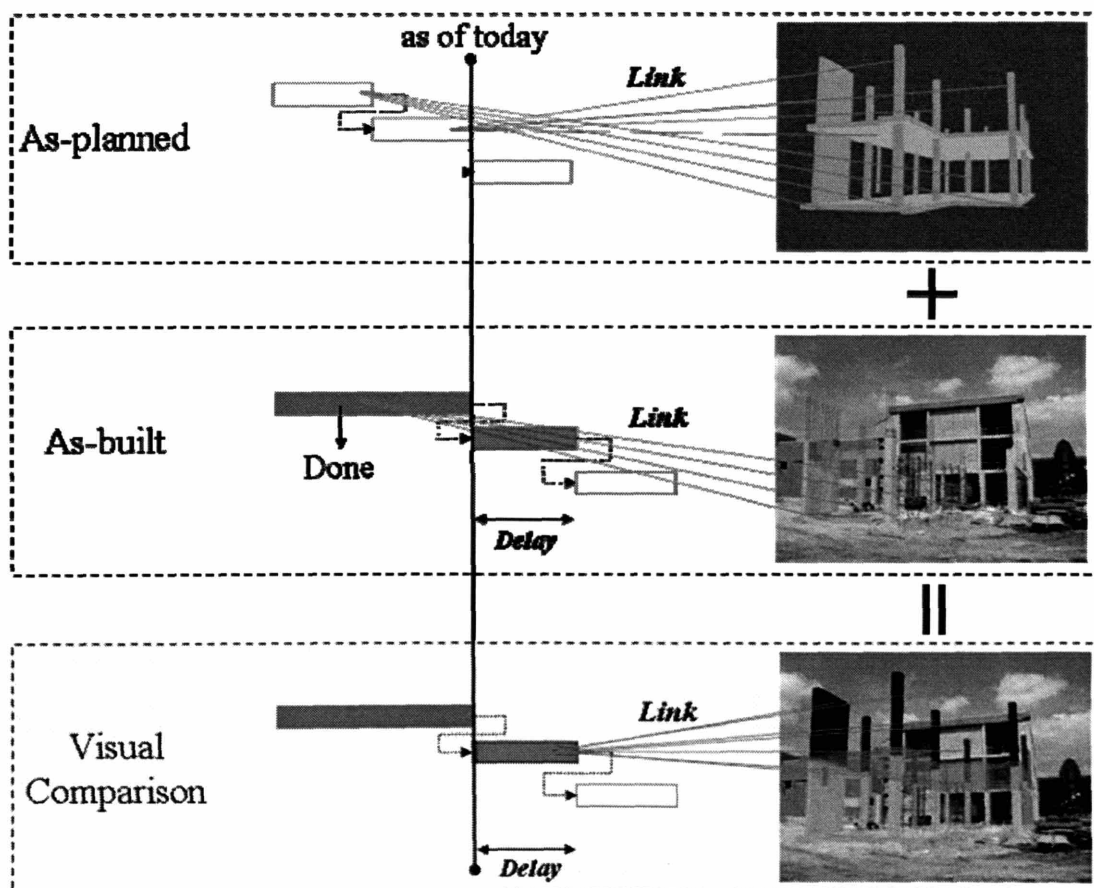

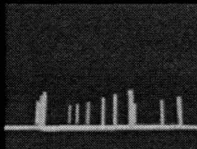
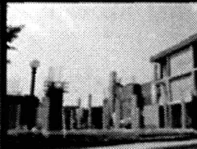


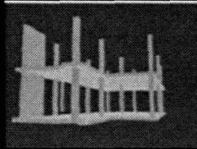
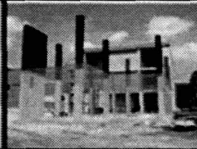
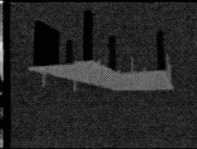


Figure 72. Augmented Reality-based Progress Monitoring

This AR image can also be linked to the schedule so that deviation in schedule can be quantified. For example, Figure 72 further illustrates that the second floor and columns that are supposed to already be installed, are the second activity in the network. Thus, we can easily quantify the deviation and in turn, trigger other information related to this activity (e.g., resource and budget information). Figure 73 illustrates a visualized report of progress monitoring. In particular, different colors are used to represent behind and ahead of schedule. For example, in Figure 73 green means ahead of schedule, and red means behind schedule. Ultimately, these benefits from AR-based progress monitoring can facilitate the coordination process by reducing the time to inform the participants as to what the situation is though they don't have much knowledge on construction systems.

	As-Built	As-Planned	Comparison	Deviation	Quantification
Week 1					4 days (-)
Week 2					7 days (+)
...					

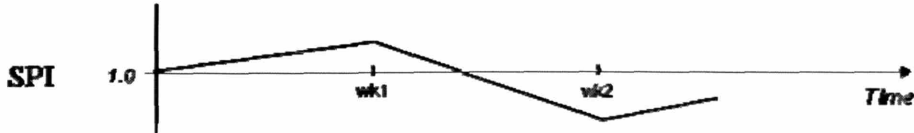


Figure 73. Visualized Progress Monitoring History

On the other hand, as-built data obtained from the laser scanner can be used for this AR-based progress monitoring. In this case, more accurate comparison and quantification will be possible because the laser scanned image can be converted into an accurate 3D as-built model.

Color and color Gradient

Another visualization technique is the use of color and color gradients. Color has been widely used for visualization of construction information [Songer and Hays, 2003; Song et al., 2005]. In progress monitoring, color can be combined with the previous AR imaging to represent diverse information in a single image. Different from the use of color in the previous section, color can be used for work packages while gradients represent work sequences in each work package. In Figure 74, two work packages are exemplified: one is the work package for the second floor in the office entrance denoted as A (red) and the other is for the second floor in the main building denoted as B (yellow). In addition, each work package's work sequence is represented by a color gradient. For example, as the gradient grows darker, there are more preceding activities. By differentiating work packages and their sequences by color and gradient, a single image can be rich in information. Along with intuitive progress comparison by AR, color and gradients tell us that how activities are associated with the comparison between as-planned and as-built data. Thus, color and gradients can give rich information which helps quickly understand the current situation and decide what to do if there is a need to reduce the gap between as-planned and as-built.

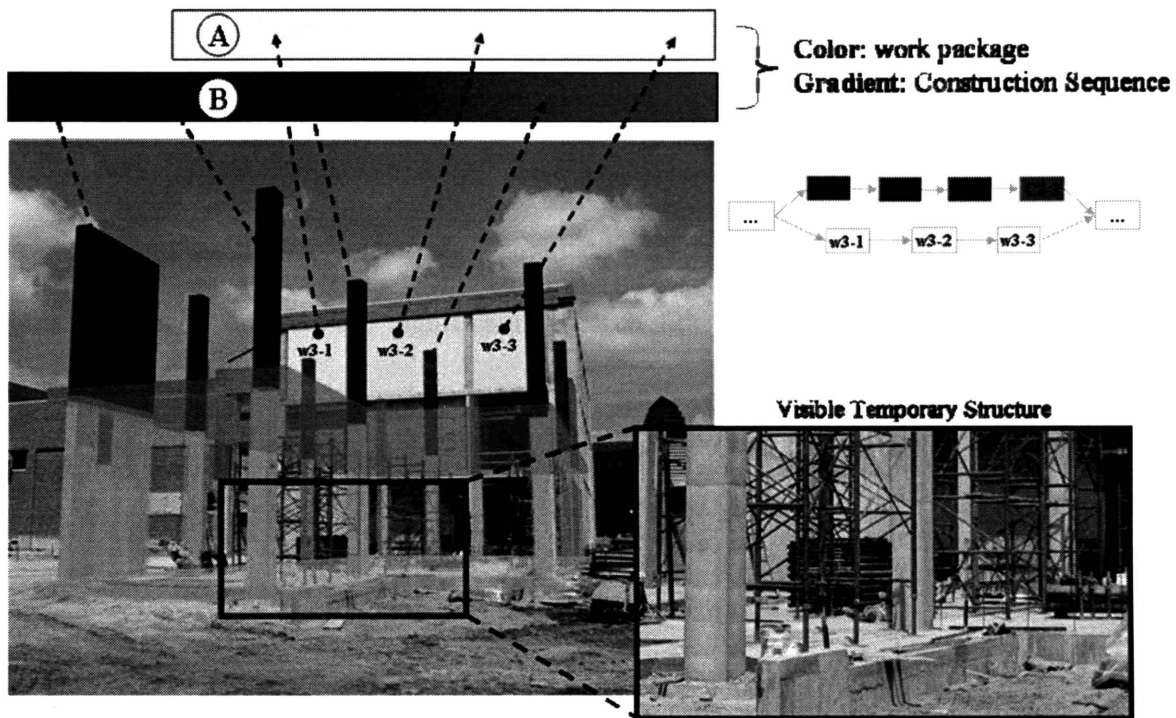


Figure 74. Use of Color and Color Gradient

Like these examples, visualization techniques for progress monitoring, particularly for helping people who do not have much knowledge of or experience with construction situations, can contribute to the coordination and the decision making process during execution. As a starting point, several visualization ideas and possible applications to progress monitoring were discussed as a future research opportunity.

In summary, taking into account the uncertainty and complexity of current construction, these future research endeavors are to gain a comprehensive understanding of these influences. By applying a systematic and proactive approach, it becomes possible to effectively manage

them. This varies from the prevalent reductionism and the commonly used reactive and impromptu approach currently taken in construction. The approach that will evolve from my research would provide the A/E/C industry with various possibilities and solutions to complex problems associated with uncertainty. Combined with the implementation of computer-integrated human-oriented construction, significant research findings can be achieved, which would be applicable to the industry immediately and, ultimately, contribute to effective construction of the safe built environment.

REFERENCES

- Abdel-Hamid, T. (1984), "The Dynamics of Software Development Project Management: An Integrative System Dynamics Perspective" Thesis (PhD), Sloan School of Management, MIT, Cambridge, MA.
- Ahmed, S., Sriram, D., and Logcher, R., 1992. Transaction-management issues in collaborative engineering. *J. Computing in Civil Engrg.* ASCE, 6 (1), pp. 85-105.
- Alarcón, L. and Ashley, B. (1999), "Playing Games: Evaluating the Impact of Lean Production Strategies on Project Cost and Schedule." International Group of Lean Construction 7th Annual Conference, July 26-28, 1999, Berkeley, CA.
- Badiru, A. and Pulat, S. (1995), "Comprehensive Project Management: Integrating Optimization Models, Management Principles, and Computers" Prentice-Hall, Upper Saddle River, NJ, pp. 107- 115.
- Ballard, G. and Howell, G. (1995), "Toward Construction JIT." The 11th Annual ARCOM Conference, September, 2003, Sheffield, UK, Association of Researchers in Construction Management (ARCOM).
- Brown, C. (2003), "Just-In-Time to Just-In-Case." Graziadio Business Report, Los Angeles, CA, January 2003, Vol. 6, No. 2.
- Chachere, J., Kunz, J., and Levitt R. (2004), "Observation, Theory, and Simulation of Integrated Concurrent Engineering: Risk Analysis Using Formal Models of Radical Project Acceleration." CIFE Working Paper #WP088, August, 2004, Stanford University, Palo Alto, CA
- Charoengam, C., Coquince, S. T. and Hadikusumo, B. H. W., 2003. Web-based application for managing change orders in construction projects. *J. Construction Innovation*, 3, pp. 197-215.
- Chong, W. and Low, S. (2005), "Assessment of Defects at Construction and Occupancy Stages" *Journal of Performance of Constructed Facilities*, ASCE, Reston, Virginia, November, 2005, Vol. 19, No. 4, pp. 283-289.
- Chou, D. and Shen, L. (2001), "Construction Modeling and Buffer Management with Integrated Production Scheduler" International Group of Lean Construction 9th Annual Conference, August 6-8, 2001, Singapore, IGLC.
- Cooper, K. (1980), "Naval Ship Production: A Claim Settled and A Framework Built. *Interfaces*, Vol. 10, No. 6, pp. 20-36.

- Cox, I.D., Morris, J.P., Rogerson, J.H., and Jared, G.E. (1999), "A Quantitative Study of Post Contract Award Design Changes in Construction" *Construction Management and Economics*, E & FN Spon Ltd., London, UK, July, 1999, Vol. 17, No. 4, pp. 427-439
- ENR (2005). McGraw-Hill, New York, NY, 2005, Vol. 255, No. 2, pp. 28-33.
- Eppinger, S. (1997), "Three Concurrent Engineering Problems in Product Development Seminar." Sloan School of Management, MIT, Cambridge, MA.
- Eppinger, S. and Krishnan, V. (1992), "Overlapping Product Development Activities by Analysis of Information Transfer Practice.", Sloan School of Management, MIT, Cambridge, MA, Working Paper 3478-92.
- Ford, D. and Sterman, J. (2003), "Overcoming the 90% Syndrome: Iteration Management in Concurrent Development Projects." *Concurrent Engineering Research and Applications*, September, 2003, Vol. 111, No. 3, pp. 177-186.
- Forrester, J. (1961), "Industrial Dynamics." Pegasus Communications, Waltham, MA, pp. 60-66.
- Fowler, M. and Scott, K. (2000), "UML Distilled: a Brief Guide to the Standard Object Modeling Language." Addison-Wesley, Inc., Upper Saddle River, NJ, pp. 129-144.
- Ganeshan, R., Garrett, J. and Finger, S., 1994. A framework for representing design intent. *Design Studies*, 15 (1), pp 59-84.
- Goldratt, E. M. (1997), "Critical Chain" North River Press, Great Barrington, MA.
- Han, S. Peña-Mora, F., Lee, S., Park, M. (2006), "Hybrid Simulation for Strategic-Operational Construction Management." Joint International Conference on Computing and Decision Making in Civil and Building Engineering, Montreal, Canada.
- Hanna, A. S., Russell, J. S., and Vandenberg, P.J. (1999), "The Impact of Change Orders on Labor Efficiency for Mechanical Construction." *Journal of Construction and Management*, ASCE, Reston, Virginia, May/June, 1999, Vol. 125, No. 3, pp. 176-184.
- Hegazy, T. (1999), "Optimization of Resource Allocation and Leveling Using Genetic Algorithms." *Journal of Construction Engineering and Management*, ASCE, Reston, Virginia, May/June, 1999, Vol. 125, No. 3, pp. 167-175.
- Hegazy, T., Zanelidin, E. and Grierson, D., 2001. Improving design coordination for building projects – I: Information model. *J. Constr. Engrg. and Mgmt.* ASCE, 127 (4), pp. 322-329.
- Hester, W.T., Kuprenas, J. A. and Chang, T. C., 1991. Construction changes and change orders: their magnitude and impact. Construction Industry Institute (CII), Source document 66, CII, Austin, Tex.

- Horman, M., Messner, J., Riley, D., and Pulaski, M. (2003), "Using Buffers To Manage Production: A Case Study of the Pentagon Renovation Project." International Group of Lean Construction 11th Annual Conference, July 22-24, 2003, Blacksburg, VA, IGLC.
- Horman, M. and Kenley R. (1998), "Process Dynamics: Identifying a Strategy for the Deployment of Buffers in Building Projects." International Journal of Logistics: Research & Applications, Vol. 1, No. 3, pp. 221-237.
- Howell, G., Laufer, A., and Ballard, G. (1993), "Interaction between Subcycles: One Key to Improved Methods." Journal of Construction Engineering and Management, ASCE, Reston, VA, December, 1993, Vol. 119, No. 4, pp. 714-728.
- Ibbs, W. (1997), "Quantitative Impacts of Project Change: Size Issues." Journal of Construction Engineering and Management, ASCE, Reston, Virginia, September/October, 1997, Vol. 123, No. 3, pp. 308-311.
- Ibbs, W. (2005), "Impact of Change's Timing on Labor Productivity" Journal of Construction Engineering and Management, ASCE, Reston, Virginia, November, 2005, Vol. 131, No. 11, pp. 1219-1223.
- Ibbs, W., Lee, S., and Li, M. (1998), "Fast-tracking's Impact on Project Change." Project Management Journal, Project Management Institute, Newtown Square, PA, 1998, Vol. 29, No. 4, pp. 35-42.
- Josephson, P. and Hammarlund, Y. (1999), "The Causes and Costs of Defects in Construction: A Study of Seven Building Projects." Automation in Construction, Elsevier, Oxford, UK, August, 1999, Vol. 8, No. 6, pp. 681-687.
- Kamat, V. R., and El-Tawil, S. (2005). "Rapid Post-Disaster Evaluation of Building Damage Using Augmented Situational Visualization", Proceedings of the 2005 Construction Research Congress, American Society of Civil Engineers, Reston, VA.
- Karim, A. and Adeli, H., 1999. CONSCOM: An OO construction scheduling and change management system. J. Constr. Engrg. and Mgmt. ASCE, 125 (5), pp. 368-376.
- Krishnamurthy, K. and Law, K., 1995. A data management model for design change control. Concurrent engineering: Res. and applications, 3(4), pp. 329-343.
- Krishnan, V., Eppinger, S., and Whitney, D. (1995), "Accelerating Product Development by the Exchange of Preliminary Product Design Information." Journal of Mechanical Design, ASME, December, 1995, Vol. 117, pp. 491-498.
- Lee, M., Hanna, A. S., and Loh, W., 2004. Decision tree approach to classify and quantify cumulative impact of change orders on productivity. J. Computing in Civil Engrg. ASCE, 18 (2), pp. 132-144.

- Lee, S. (2003), "Dynamic Quality and Change Management For Large Scale Concurrent Design and Construction Projects", Thesis (M.S.), Dept. of Civil and Environmental Engineering, MIT, Cambridge, MA.
- Lee, S., Peña-Mora, F., and 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, Vol. 131, No. 8, pp. 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, Oxford, UK, January, 2006, Vol. 15, No. 1, pp. 84-97.
- Lee, S., Peña-Mora, F., and Park, M. (2006), "Reliability and Stability Buffering Approach: Focusing on the Issues of Errors and Changes in Concurrent Design and Construction Projects." *Journal of Construction Engineering and Management*, ASCE, Reston, VA, May, 2006, Vol. 132, No.5, pp.452-464.
- Lee, S., Peña-Mora, F., Park, M. (2006), "Web-enabled System Dynamics Model for Error and Change Management on Concurrent Design and Construction Projects." *Journal of Computing in Civil Engineering*, American Society of Civil Engineers, Reston, VA. July/August, 2006. (In Press)
- Lyneis, J., Cooper, K., and Els, S. (2001), "Strategic Management of Complex Projects: A Case Study using System Dynamics." *System Dynamics Review*, System Dynamics Society, Albany, NY, Fall, 2001, Vol. 17, No. 3, pp. 237-260.
- Maio, C., Schexnayder, C., Knutson, K., and Weber, S. (2000), "Probability Distribution Functions for Construction Simulation." *Journal of Construction Engineering and Management*, ASCE, Reston, Virginia, July/August, 2000, Vol. 126, No. 4, pp. 285-292.
- Martin, R. and Raffo, D. (2001). "Application of a Hybrid Process Simulation Model to a Software Development Project." *The Journal of Systems and Software*, Volume 59, PP. 237-246.
- Martinez, J.C. and Ioannou, P.G., (1997), "State-based Probabilistic Scheduling Using STROBOSCOPE's CPM Add-On", *Proceedings of Construction Congress V*, ASCE, Reston, VA.
- Microsoft Corporation. (2000), Microsoft SQL Server, <<http://www.microsoft.com/sql/>>, Last Visit: May, 2004.
- Mokhtar, A., Bedard, C, and Fazio, P., 1998. Information model for managing design changes in a collaborative environment. *J. Computing in Civil Engrg.* ASCE, 12 (2), pp. 82-92.
- Morris, P., and Hough, G. (1987). *The Anatomy of Major Projects – A Study of the Reality of Project Management*. John Wiley & Sons.

- Moselhi, O., Assem, I., and El-Rayes, K. (2005), "Change Orders Impact of Labor Productivity" *Journal of Construction Engineering and Management*, ASCE, Reston, Virginia, March, 2005, Vol. 131, No. 3, pp. 354-359.
- Ogunlana, S., Li, H., and Sukhera, F (2003), "System Dynamics Approach to Exploring Performance Enhancement in a Construction Organization." *Journal of Construction Engineering and Management*, ASCE, Reston, Virginia, September/October, 2003, Vol. 129, No. 5, pp. 528-536.
- Oliva, R. (1996), "A Dynamic Theory of Service Delivery: Implication for Managing Service Quality." *Doctoral Thesis*, Sloan School of Management, MIT, Cambridge, MA, June, 1996, pp.23-38.
- Oracle Corporation. (2002), Oracle 9i Database, <<http://www.oracle.com/technology/products/oracle9i/>>, Last Visit: May, 2004.
- Park, M. (2001), "Dynamic Planning and Control Methodology for Large-Scale Concurrent Construction Projects." Thesis (PhD), Department of Civil & Environmental Engineering, MIT, Cambridge, MA, June, 2001.
- Park, M. and Peña-Mora, F. (2003), "Dynamic Change Management for Construction: Introducing Change Cycle into Model-based Project Management." *System Dynamics Review*, System Dynamics Society, Albany, NY, Fall, 2003, Vol. 19, No. 3, pp. 213-242.
- Park, M. and Peña-Mora, F. (2004), "Reliability Buffering for Construction Projects." *Journal of Construction Engineering and Management*, ASCE, Reston, VA, October, 2004, Vol. 130, No. 5, pp. 626-637.
- Peltonen, H., Mannisto, T., Alho, K., and Sulonen, R., 1993. An engineering document management system. *Proc. ASME Winter Annu. Meeting*.
- Peña-Mora, F. and Dwivedi, G. (2002), "Multiple Device Collaborative and Real Time Analysis System for Project Management in Civil Engineering." *Journal of Computing in Civil Engineering*, ASCE, Reston, Virginia, January, 2002, Vol. 16, No. 1, pp. 23-38.
- Peña-Mora, F. and Li, M. (2001), "A Robust Planning and Control Methodology for Design-Build Fast-Track Civil Engineering and Architectural Projects." *Journal of Construction Engineering and Management*, ASCE, Reston, Virginia, January/February, 2002, Vol. 127, No. 1, pp. 1-17.
- Peña-Mora, F. and Park, M. (2001), "Dynamic Planning for Fast-Tracking Building Construction Projects." *Journal of Construction Engineering and Management*, ASCE, Reston, Virginia, November/December, 2001, Vol. 127, No. 6, pp. 445-456.
- Primavera System Inc. (2003), Primavera Project Planner, <<http://www.primavera.com>>, Last Visit: May, 2004.

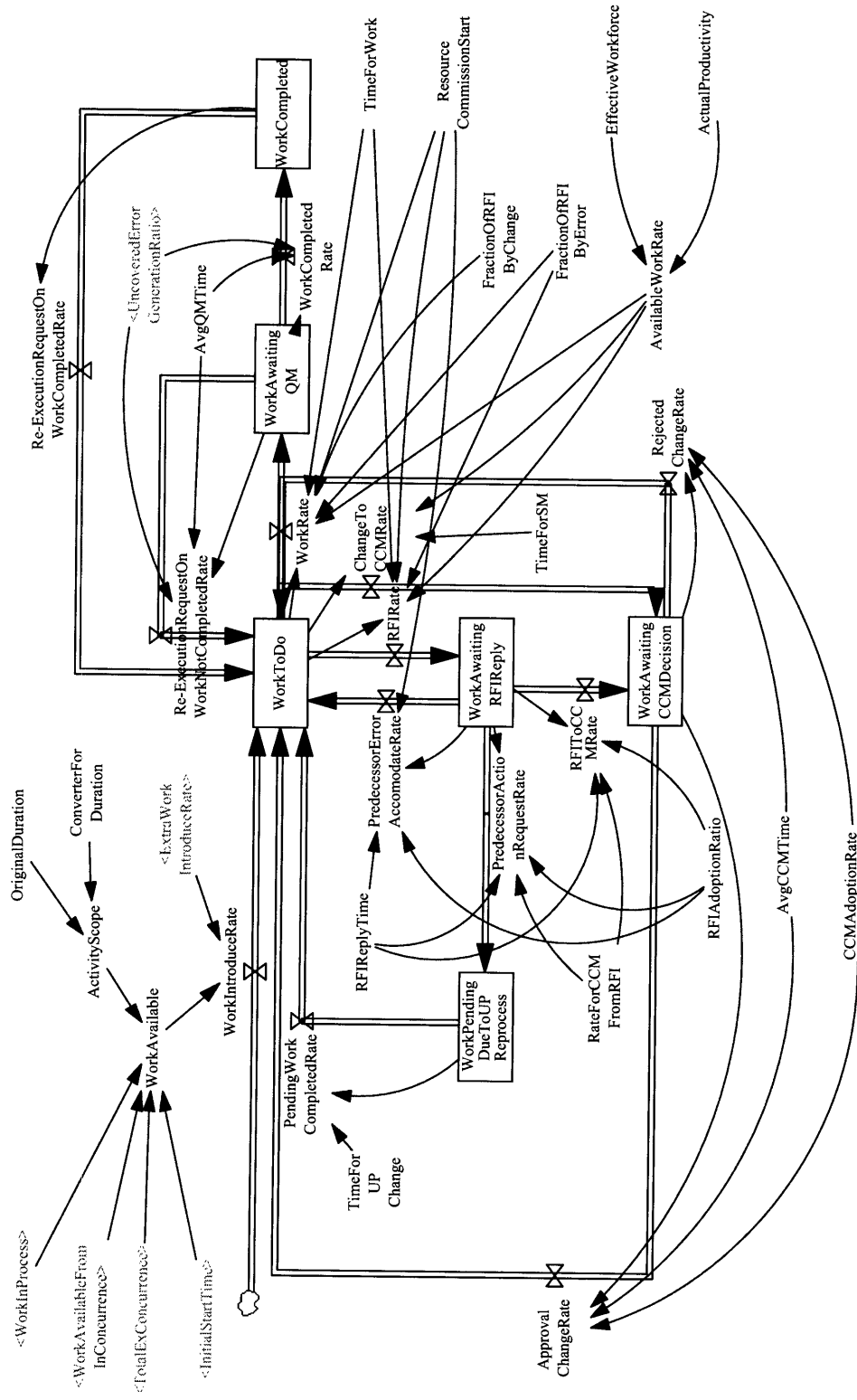
- Richardson, G. (1985), "Introduction to the System Dynamics Review." *System Dynamics Review*, Albany, NY, Summer, 1985, Vol. 1, No. 1, pp. 1-3.
- Richardson G. and Pugh, AL III (1981), "Introduction to System Dynamics Modeling with Dynamo." MIT Press, Cambridge, MA.
- Rodrigues, A. and Bowers, J. (1996), "System Dynamics in Project Management: A Comparative Analysis with Traditional Methods.' *System Dynamics Review*, System Dynamics Society, Albany, NY, Summer, 1996, Vol. 12, No. 2, pp. 121-139.
- Rodrigues, A. and Williams, T. (1998), "System Dynamics in Project Management: Assessing the Impacts of Client Behavior on Project Performance." *Journal of the Operational Research Society*, Operational Research Society, Birmingham, UK, January, 1998, Vol. 49, No. 1, pp. 2-15.
- Sakamoto, M., Horman, M., and Randolph, T. (2002), "A Study of the Relationship between Buffers and Performance in Construction." *International Group of Lean Construction 10th Annual Conference*, August 6-8, 2002, Gramado, Brazil.
- Senior, B. and Halpin, D. (1998), "Simplified Simulation System for Construction Projects." *Journal of Construction Engineering and Management*, ASCE, Reston, VA, January/February, 1998, Vol. 124, No. 1, pp. 72-81.
- Slaughter, S. (1998). "Models of Construction Innovation." *Journal of Construction Engineering and Management*, ASCE, Reston, Virginia, May/June, 1998, Vol. 124, No. 3, pp. 226-231.
- Soh, C. and Wang, Z., 2000. Parametric coordinator for engineering design. *J. Computing in Civil Engrg.* ASCE, 14 (4), pp. 233-240.
- Song, K., Pollalis, S., and Pena-Mora, F. (2005). "Project Dashboard: Concurrent Visual Representation Method of Project Metrics on 3D Building Models", *Proceedings of 2005 ASCE International Conference on Computing in Civil Engineering*, ASCE, Reston, VA.
- Songer, A. and Hays, B. (2003). "A Framework for Multi-dimensional Visualization of Project Control Data", *Proceedings of the 2003 Construction Research Congress*, American Society of Civil Engineers, Reston, VA.
- Spooner, D. and Hardwick, M., 1993. Using persistent object technology to support concurrent engineering systems. *Concurrent engineering: Methodology and applications*, Elsevier Science, Amsterdam, pp. 205-234.
- Sterman, J. (1992), "System Dynamics Modeling for Project Management." Sloan School of Management, MIT, On-line Publication, <<http://web.mit.edu/jsterman/www/>>, Last Visit: October, 2002.

- Sterman, J. (2000), "Business Dynamics: System Thinking and Modeling for a Complex World." McGraw-Hill Companies, New York, NY, pp.55-61, 469-511, and 587-595.
- Tighe, J. (1991), "Benefits of Fast Tracking are a Myth." International Journal of Project Management, Elsevier, Oxford, UK, Vol. 9, No. 1, pp. 49-51.
- Tommelein, I. and Weissenberger, M. (1999), "More Just-In-Time: Location of Buffers in Structural Steel Supply and Construction Processes." International Group of Lean Construction 7th Annual Conference, July 26-28, 1999, Berkeley, CA, IGLC.
- Trauner, T. (1992), "Managing the Construction Project." John Wiley & Sons, Inc., New York, NY, pp. 126-142.
- Turek, M. (1995), "System Dynamics Analysis of Financial Factors in Nuclear Power Plant Operations.", Thesis (M.S.), Dept. of Nuclear Engineering, MIT, Cambridge, MA
- Ventana Software Inc. (2002), Vensim Software, <<http://www.vensim.com/software.html>>, Last Visit: May, 2004.
- Wang, X., and Dunston, P. (2005). "Heavy Equipment Operator Training via Virtual Modeling Techniques", Proceedings of the 2005 Construction Research Congress, American Society of Civil Engineers, Reston, VA.
- Williams, T. (2000). Safety regulation changes during projects: the use of system dynamics to quantify the effects of change. I. J. of Project Mgmt, 18 (1), pp.23 – 31.
- Williams, T. (2002). Modeling Complex Projects. John Wiley & Sons.
- Womack, J. and Jones, D. (1996), "Lean Thinking: Banish Waste and Create Wealth in Your Corporation." Simon & Shuster, New York, NY.

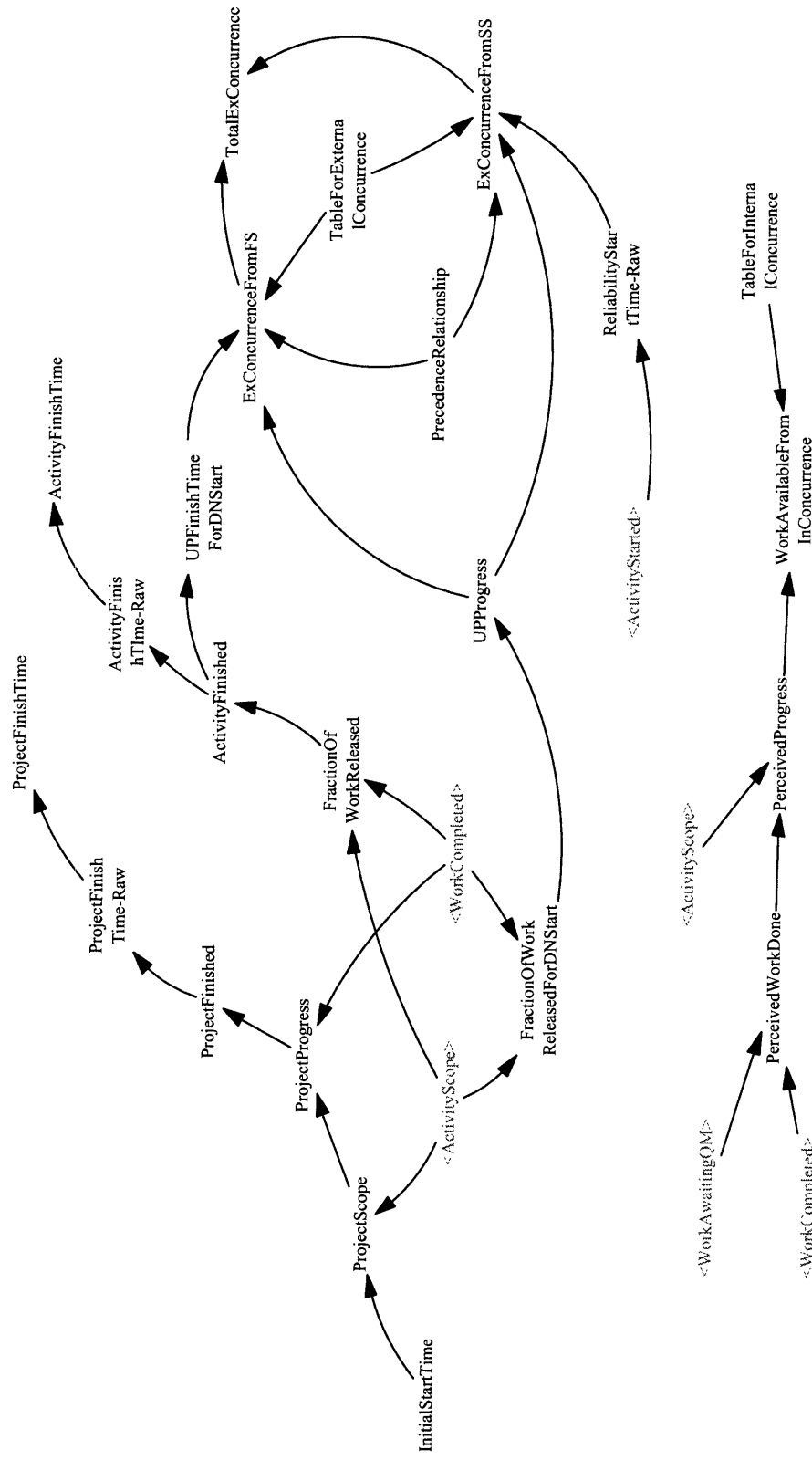
APPENDIX I

SYSTEM DYNAMICS MODEL STRUCTURES

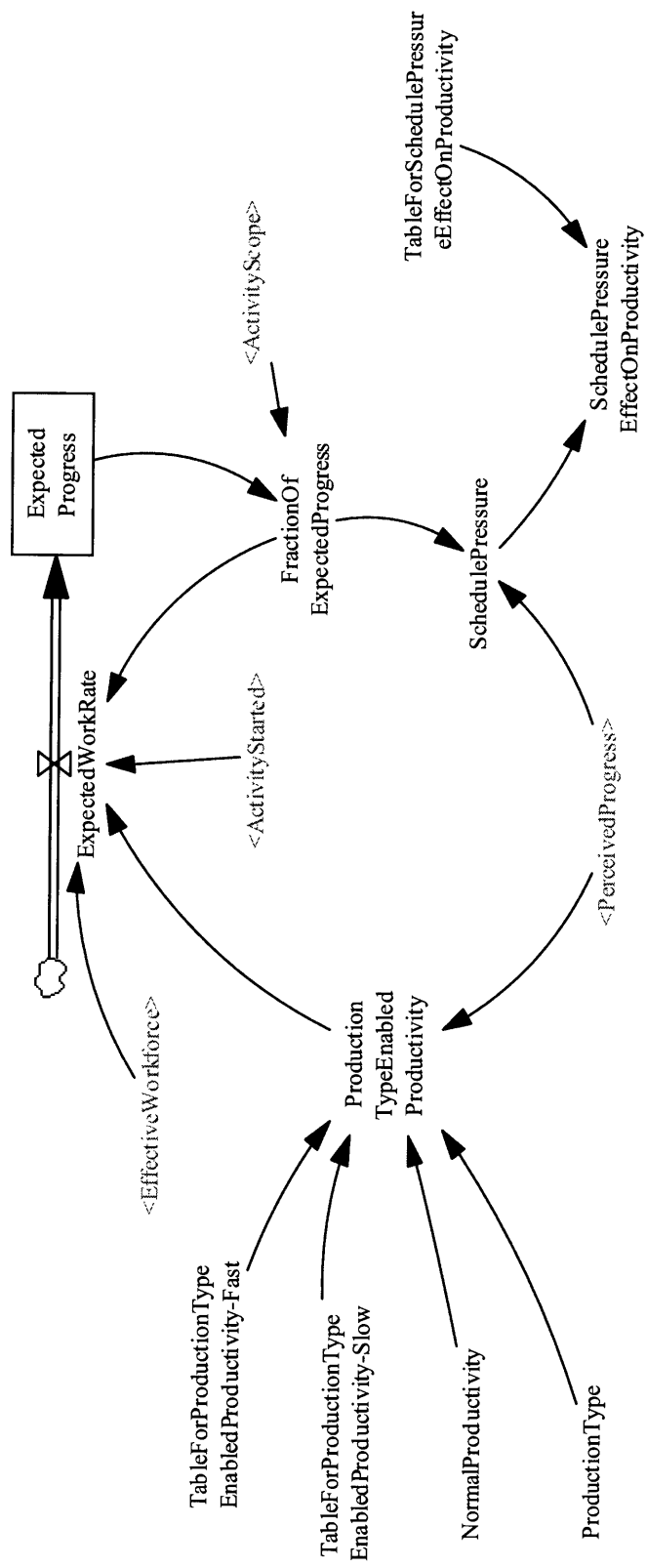
A system dynamics model introduced in Appendix is the one used for hypothetic experiments in the Chapter 6. The model used for cast study can be obtained from contacting the author.



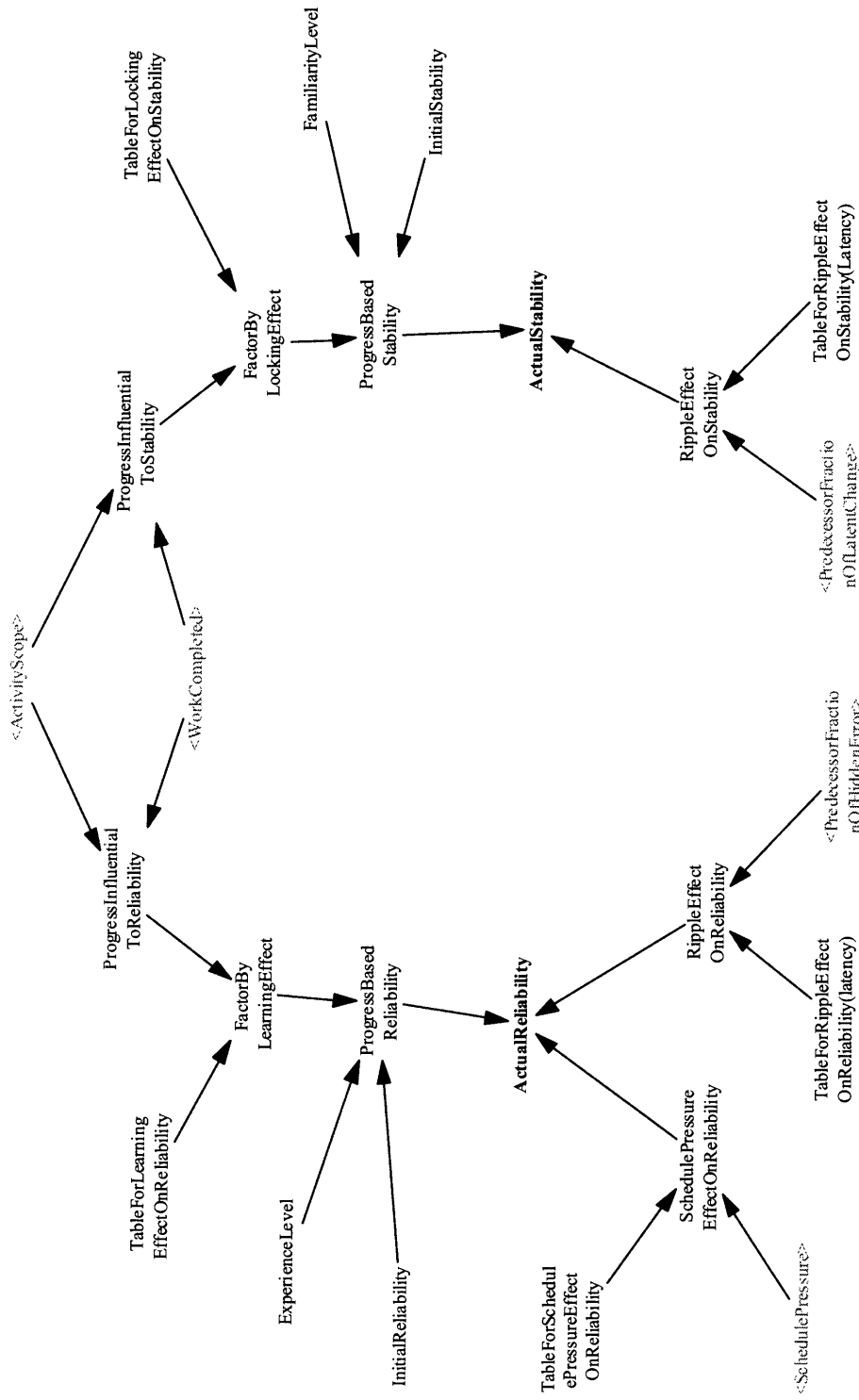
Generic Construction Process



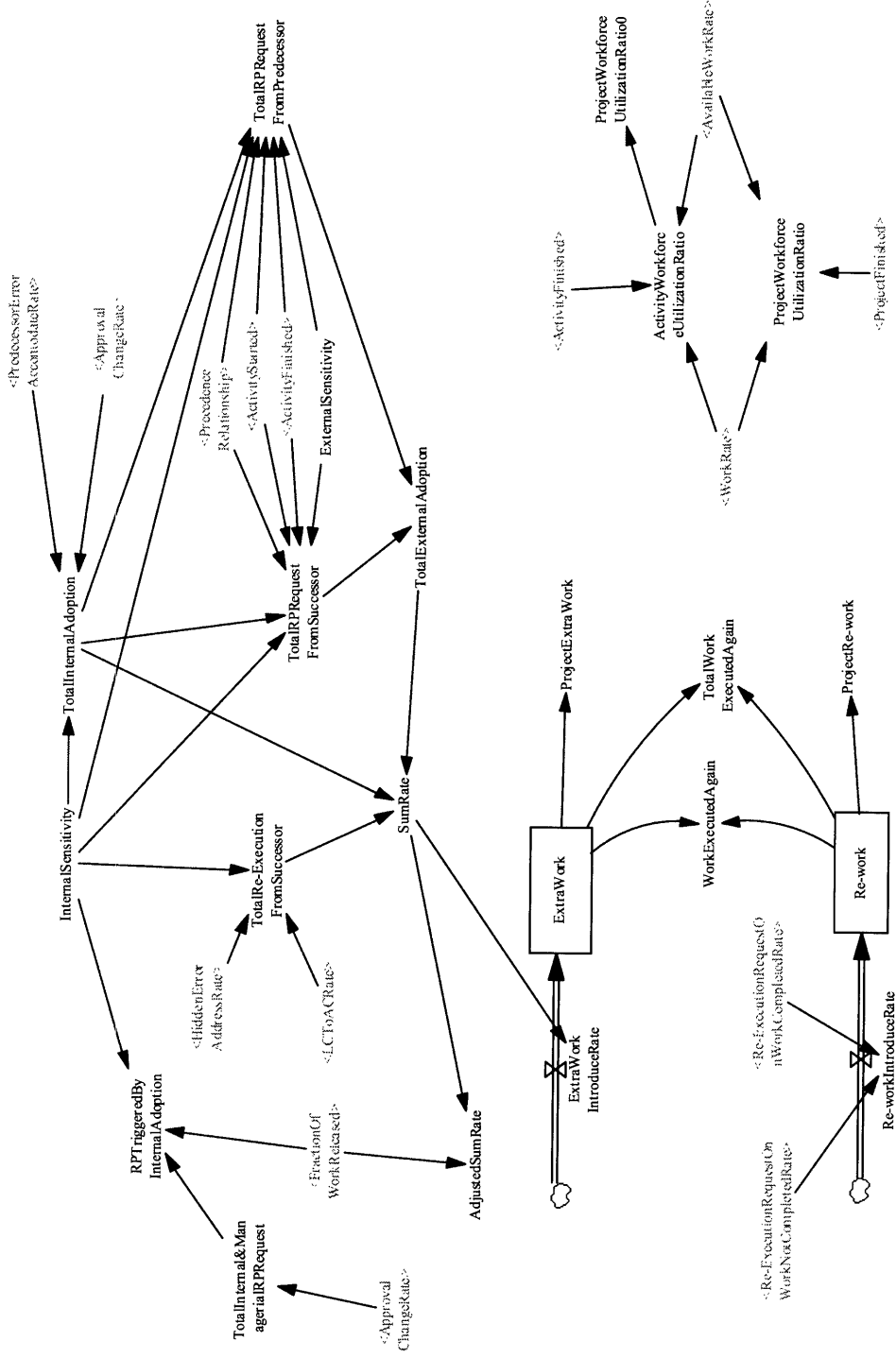
Project Progress and Work Constraint



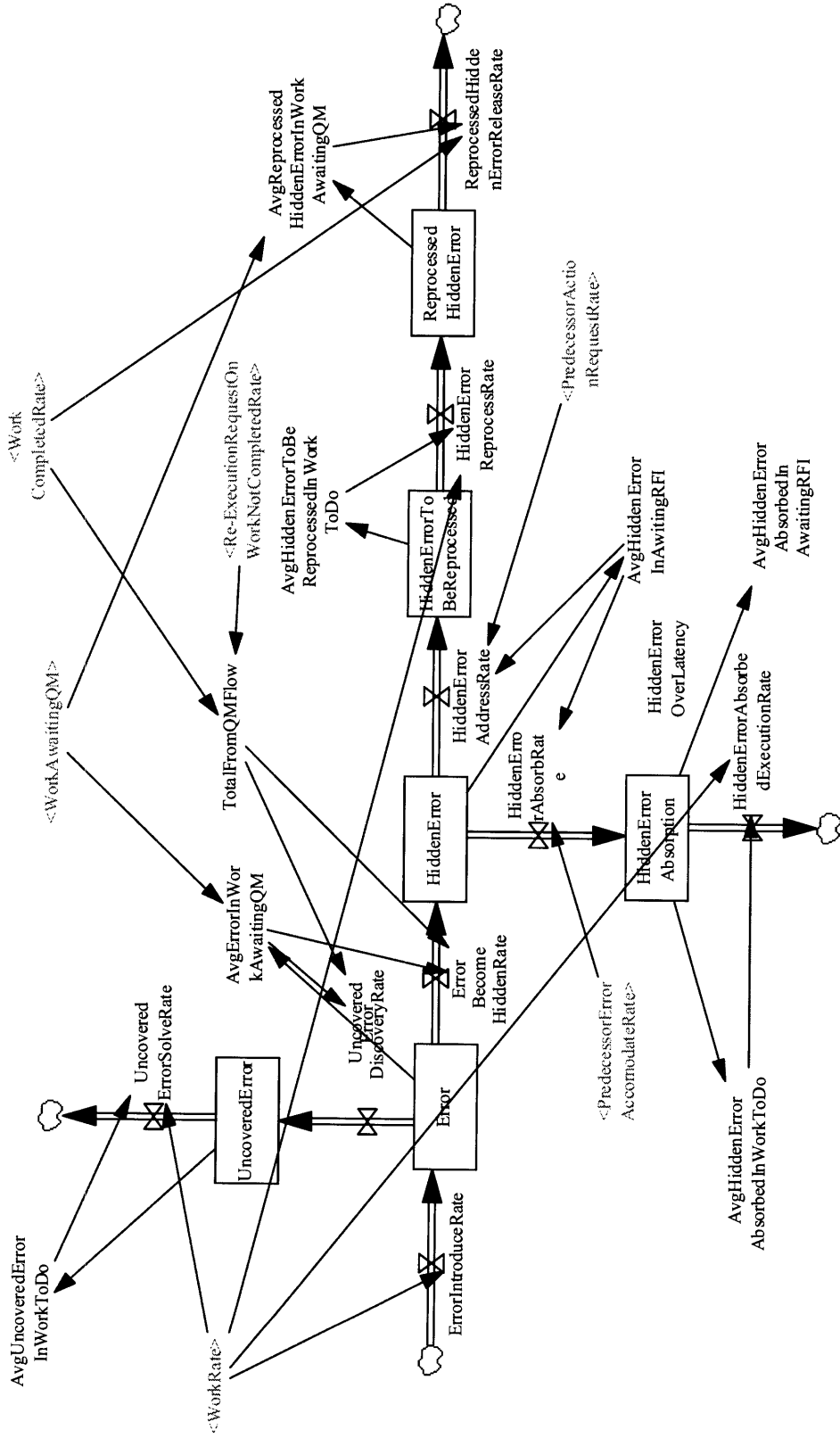
Productivity and Schedule Pressure



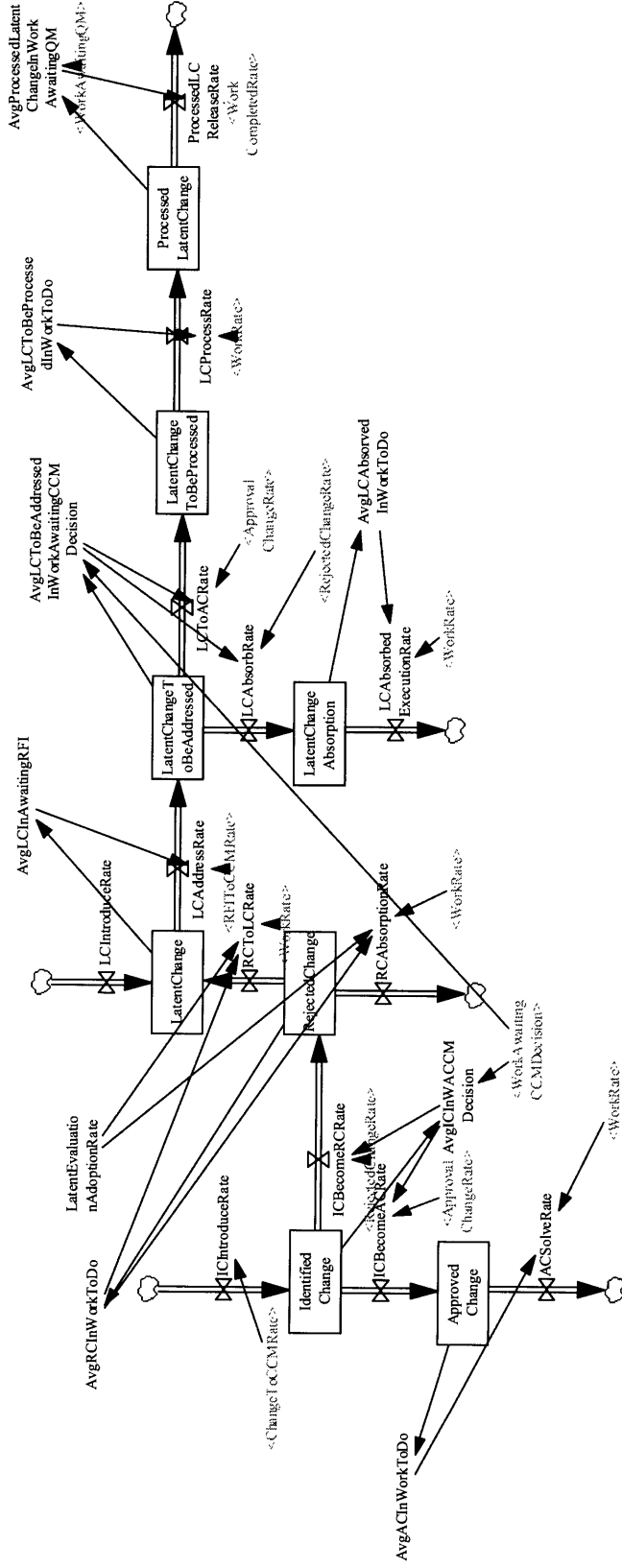
Reliability and Stability



Additional Work Scope and Workforce Utilization Ratio



Hidden Error



Latent Change

APPENDIX II

SYSTEM DYNAMICS MODEL EQUATIONS

The following equations are the ones used for hypothetic experiments in the Chapter 6. However, based on the characteristics of experiments, some constants can be changed based on the need. For example, 'ExternalSensitivity' is written as '0' in this Appendix, it can have non-zero value in case of CASE 3 simulation, which considers the propagation impact of errors and changes in the Chapter 6.

activity:¹

(a1-a3)

ACsolveRate[activity]=

AvgACInWorkToDo[activity]*WorkRate[activity]

Units: WU/Day

ActivityFinished[activity]=

SAMPLE IF TRUE (FractionOfWorkReleased[activity]>0.991,0,1)

Units: Dmnl

ActivityFinishTime[activity]=

INTEGER ("ActivityFinishTime-Raw"[activity])

Units: Day

"ActivityFinishTime-Raw"[activity]= INTEG (

IF THEN ELSE (ActivityFinished[activity]=1,1,0),0)

Units: Day

ActivityScope[activity]=

OriginalDuration[activity]*ConverterForDuration

Units: WU

¹ Subscript used for activity under study. The number of activity can be customized based on the need.

ActivityStarted[activity]=
SAMPLE IF TRUE (WorkAvailable[activity]>1,1,0)

Units: Dmnl

ActivityWorkforceUtilizationRatio[activity]=
IF THEN ELSE (ActivityFinished[activity]=0,0 , IF THEN ELSE (AvailableWorkRate
[activity]=0, 0, WorkRate[activity]/AvailableWorkRate[activity]))

Units: Dmnl

ActualProductivity[activity]=
ProductionTypeEnabledProductivity[activity]

Units: WU/(Day*Worker)

ActualReliability[activity]=
ProgressBasedReliability[activity]*RippleEffectOnReliability[activity]*SchedulePressureEff
ectOnReliability[activity]

Units: Dmnl

ActualStability[activity]=
ProgressBasedStability[activity]*RippleEffectOnStability[activity]

Units: Dmnl

AdjustedSumRate[activity]=
FractionOfWorkReleased[activity]*SumRate[activity]

Units: WU/Day

ApprovalChangeRate[activity]=
CCMAAdoptionRate[activity]*WorkAwaitingCCMDecision[activity]/AvgCCMTime[activity]

Units: WU/Day

ApprovedChange[activity]= INTEG (
ICBecomeACRate[activity]-ACSolveRate[activity], 0)

Units: WU

AvailableWorkRate[activity]=
IF THEN ELSE (ActivityStarted[activity]=1:AND:ActivityFinished[activity]=1,
EffectiveWorkforce[activity]*ActualProductivity[activity],0)

Units: WU/Day

AvgACInWorkToDo[activity]=
IF THEN ELSE (WorkToDo[activity]=0, 0, ApprovedChange[activity]/ WorkToDo[activity])
Units: Dmnl

AvgCCMTime[activity]=
7
Units: Day

AvgErrorInWorkAwaitingQM[activity]=
IF THEN ELSE (WorkAwaitingQM[activity]=0, 0, Error[activity]/
WorkAwaitingQM[activity])
Units: Dmnl

AvgHiddenErrorAbsorbedInAwaitingRFI[preceding]=
IF THEN ELSE (VMAX(WorkAwaitingRFIReply[activity!,preceding])<=0, 0,
HiddenErrorAbsorption[preceding]/VMAX(WorkAwaitingRFIReply [activity!,preceding]))
Units: Dmnl

AvgHiddenErrorAbsorbedInWorkToDo[activity]=
IF THEN ELSE (WorkToDo[activity]<=0, 0, HiddenErrorAbsorption[activity]/
WorkToDo[activity])
Units: Dmnl

AvgHiddenErrorInAwitingRFI[preceding]=
IF THEN ELSE (VMAX (WorkAwaitingRFIReply[activity!, preceding])<=0, 0,
HiddenError[preceding]/VMAX(WorkAwaitingRFIReply[activity!, preceding]))
Units: Dmnl

AvgHiddenErrorToBeReprocessedInWorkToDo[activity]=
IF THEN ELSE (WorkToDo[activity]<=0, 0, HiddenErrorToBeReprocessed[activity]
/WorkToDo[activity])
Units: Dmnl

AvgICInWACCMDecision[activity]=
IF THEN ELSE (WorkAwaitingCCMDecision[activity]=0,0,IdentifiedChange[activity]
/WorkAwaitingCCMDecision[activity])
Units: Dmnl

AvgLCAbsorbedInWorkToDo[activity]=
IF THEN ELSE (WorkToDo[activity]<=0, 0, LatentChangeAbsorption[activity]/
WorkToDo[activity])

Units: Dmnl

AvgLCInAwaitingRFI[preceding]=
IF THEN ELSE (VMAX(WorkAwaitingRFIReply[activity!,preceding])<=0, 0,
LatentChange[preceding]/VMAX(WorkAwaitingRFIReply[activity!,preceding]))

Units: Dmnl

AvgLCToBeAddressedInWorkAwaitingCCMDecision[activity]=
IF THEN ELSE (WorkAwaitingCCMDecision[activity]=0, 0, LatentChangeToBe
Addressed[activity]/WorkAwaitingCCMDecision[activity])

Units: Dmnl

AvgLCToBeProcessedInWorkToDo[activity]=
IF THEN ELSE (WorkToDo[activity]<=0,0,LatentChangeToBeProcessed[activity]/
WorkToDo[activity])

Units: Dmnl

AvgProcessedLatentChangeInWorkAwaitingQM[activity]=
IF THEN ELSE (WorkAwaitingQM[activity]<=0,0,ProcessedLatentChange[activity]
/WorkAwaitingQM[activity])

Units: Dmnl

AvgQMTime[activity]=

1

Units: Day

AvgRCInWorkToDo[activity]=
IF THEN ELSE (WorkToDo[activity]=0, 0, RejectedChange[activity]/ WorkToDo[activity])

Units: Dmnl

AvgReprocessedHiddenErrorInWorkAwaitingQM[activity]=
IF THEN ELSE (WorkAwaitingQM[activity]<=0,0,ReprocessedHiddenError[activity]
/WorkAwaitingQM[activity])

Units: Dmnl

AvgUncoveredErrorInWorkToDo[activity]=
IF THEN ELSE (WorkToDo[activity]=0, 0, UncoveredError[activity]/ WorkToDo[activity])

Units: Dmnl

CCMAAdoptionRate[activity]=
0.5

Units: Dmnl

ChangeToCCMRate[activity]=
MIN (AvailableWorkRate[activity], WorkToDo[activity]/TimeForSM[activity])*
IdentifiedChangeGenerationRatio[activity]

Units: WU/Day

ConverterForDuration=
1000

Units: WU/Day

DurationCheck[activity]= INTEG (
IF THEN ELSE (ActivityStarted[activity]=1:AND:ActivityFinished[activity]=1,1,0),0)

Units: Day

EffectiveWorkforce[activity]=
100

Units: Worker

Error[activity]= INTEG (
ErrorIntroduceRate[activity]-ErrorBecomeHiddenRate[activity]-
UncoveredErrorDiscoveryRate[activity], 0)

Units: WU

"Error&IdentifiedChange"[activity]=
Error[activity]+IdentifiedChange[activity]

Units: WU

ErrorBecomeHiddenRate[activity]=
AvgErrorInWorkAwaitingQM[activity]*TotalFromQMFlow[activity]*(1-QMThoroughness
[activity])

Units: WU/Day

ErrorIntroduceRate[activity]=
WorkRate[activity]*(1-ActualReliability[activity])

Units: WU/Day

ExConcurrenceFromFS[activity]=
VMIN (IF THEN ELSE (PrecedenceRelationship[activity,preceding!]>=-0:AND:
PrecedenceRelationship[activity,preceding!]<1000, IF THEN ELSE (Time>=
UPFinishTimeForDNStart[preceding!]+PrecedenceRelationship[activity, preceding!],
TableForExternalConcurrence[activity,preceding!](UPProgress[preceding!]), 0) ,1))

Units: Dmnl

ExConcurrenceFromSS[activity]=
VMIN (IF THEN ELSE (PrecedenceRelationship[activity,preceding!] >= 1000: AND:
PrecedenceRelationship[activity, preceding!]<2000,IF THEN ELSE (Time >=
"ReliabilityStartTime-Raw"[preceding!]+MAX(TIME STEP, PrecedenceRelationship
[activity,preceding!]-1000), TableForExternalConcurrence [activity,preceding!]
(UPProgress[preceding!] , 0) ,1))

Units: Dmnl

ExpectedProgress[activity]=INTEG (
ExpectedWorkRate[activity], 0)

Units: WU

ExpectedWorkRate[activity]=
IF THEN ELSE (FractionOfExpectedProgress[activity]>=1,0 , IF THEN ELSE
(ActivityStarted[activity]=1, EffectiveWorkforce[activity] * ProductionTypeEnabled
Productivity[activity], 0))

Units: WU/Day

ExperienceLevel[activity]=
1

Units: Dmnl

ExternalSensitivity[activity,preceding]=
0

Units: Dmnl

ExtraWork[activity]= INTEG (

ExtraWorkIntroduceRate[activity], 0)

Units: WU

ExtraWorkIntroduceRate[activity]=

SumRate[activity]

Units: WU/Day

FactorByLearningEffect[activity]=

TableForLearningEffectOnReliability(ProgressInfluentialToReliability[activity])

Units: Dmnl

FactorByLockingEffect[activity]=

TableForLockingEffectOnStability(ProgressInfluentialToStability[activity])

Units: Dmnl

FamiliarityLevel[activity]=

1

Units: Dmnl

FINAL TIME = 800

Units: Day

FractionOfExpectedProgress[activity]=

ExpectedProgress[activity]/(ActivityScope[activity]+ExtraWork[activity])

Units: Dmnl

FractionOfHiddenError[activity]=

IF THEN ELSE (WorkCompleted[activity]=0,0,HiddenError[activity]/WorkCompleted
[activity])

Units: Dmnl

FractionOfLatentChange[activity]=

IF THEN ELSE (WorkCompleted[activity]=0 , 0 , MAX(0,(LatentChange[activity])/
WorkCompleted[activity]))

Units: Dmnl

FractionOfRFIByChange[activity,preceding]=

IF THEN ELSE (PrecedenceRelationship[activity,preceding]>-1, 1, 0) *

FractionOfLatentChange[preceding]*(1+ExternalSensitivity[activity,preceding])*SMThoroughness[activity]

Units: Dmnl

FractionOfRFIByError[activity,preceding]=

IF THEN ELSE (PrecedenceRelationship[activity,preceding]>-1,1,0)*

FractionOfHiddenError[preceding]*(1+ExternalSensitivity[activity,preceding])*(QMThoroughness[activity])

Units: Dmnl

FractionOfWorkReleased[activity]=

WorkCompleted[activity]/(ActivityScope[activity]+ExtraWork[activity])

Units: Dmnl

FractionOfWorkReleasedForDNStart[activity]=

MIN(1,WorkCompleted[activity]/(ActivityScope[activity]+ExtraWork[activity]))

Units: Dmnl

HiddenError[activity]= INTEG (

ErrorBecomeHiddenRate[activity]-HiddenErrorAddressRate[activity]-
HiddenErrorAbsorbRate[activity],0)

Units: WU

HiddenErrorAbsorbedExecutionRate[activity]=

AvgHiddenErrorAbsorbedInWorkToDo[activity]*WorkRate[activity]

Units: WU/Day

HiddenErrorAbsorbRate[preceding]=

AvgHiddenErrorInAwaitingRFI[preceding]*VMAX(PredecessorErrorAccomodateRate
[activity!,preceding])

Units: WU/Day

HiddenErrorAbsorption[activity]= INTEG (

HiddenErrorAbsorbRate[activity]-HiddenErrorAbsorbedExecutionRate[activity], 0)

Units: WU

HiddenErrorAddressRate[preceding]=

AvgHiddenErrorInAwaitingRFI[preceding]*VMAX(PredecessorActionRequestRate [activity!,

preceding))

Units: WU/Day

HiddenErrorGenerationRatio[activity]=

IF THEN ELSE (ActivityFinished[activity]=1:AND:ActivityStarted[activity]=1,
(1-ActualReliability[activity])*(1-QMThoroughness[activity]), 0)

Units: Dmnl

HiddenErrorOverLatency[activity]=

0.7

Units: Dmnl

HiddenErrorReprocessRate[activity]=

AvgHiddenErrorToBeReprocessedInWorkToDo[activity]*WorkRate[activity]

Units: WU/Day

HiddenErrorToBeReprocessed[activity]= INTEG (

HiddenErrorAddressRate[activity]-HiddenErrorReprocessRate[activity], 0)

Units: WU

ICBecomeACRate[activity]=

AvgICInWACCMDecision[activity]*ApprovalChangeRate[activity]

Units: WU/Day

ICBecomeRCRate[activity]=

AvgICInWACCMDecision[activity]*RejectedChangeRate[activity]

Units: WU/Day

ICIntroduceRate[activity]=

ChangeToCCMRate[activity]

Units: WU/Day

IdentifiedChange[activity]= INTEG (

ICIntroduceRate[activity]-ICBecomeACRate[activity]-ICBecomeRCRate[activity],0)

Units: WU

IdentifiedChangeGenerationRatio[activity]=

IF THEN ELSE (ActivityStarted[activity]=1:AND:ActivityFinished[activity]=1, (1-

$$\text{ActualStability}[\text{activity}] * \text{SMThoroughness}[\text{activity}], 0)$$

Units: Dmnl

$$\text{INITIAL TIME} = 0$$

Units: Day

$$\text{InitialReliability}[\text{activity}] =$$

0.8

Units: Dmnl

$$\text{InitialStability}[\text{activity}] =$$

1

Units: Dmnl

$$\text{InitialStartTime}[\text{activity}] =$$

1e+006

Units: Day

$$\text{InternalSensitivity}[\text{activity}] =$$

0

Units: Dmnl

$$\text{LatentChange}[\text{activity}] = \text{INTEG} ($$

LCIntroduceRate[activity]+RCtoLCRate[activity]-LCAddressRate[activity], 0)

Units: WU

$$\text{LatentChangeAbsorption}[\text{activity}] = \text{INTEG} ($$

LCAbsorbRate[activity]-LCAbsorbedExecutionRate[activity], 0)

Units: WU

$$\text{LatentChangeToBeAddressed}[\text{activity}] = \text{INTEG} ($$

LCAddressRate[activity]-LCAbsorbRate[activity]-LCtoACRate[activity], 0)

Units: WU

$$\text{LatentChangeToBeProcessed}[\text{activity}] = \text{INTEG} ($$

LCtoACRate[activity]-LCProcessRate[activity], 0)

Units: WU

LatentEvaluationAdoptionRate[activity]=

0.5

Units: Dmnl

LCAbsorbedExecutionRate[activity]=

AvgLCAbsorbedInWorkToDo[activity]*WorkRate[activity]

Units: WU/Day

LCAbsorbRate[preceding]=

AvgLCToBeAddressedInWorkAwaitingCCMDecision[preceding]*RejectedChangeRate[preceding]

Units: WU/Day

LCAddressRate[preceding]=

AvgLCInAwaitingRFI[preceding]* VMAX(RFIToCCMRate[activity!,preceding])

Units: WU/Day

LCIntroduceRate[activity]=

MIN(AvailableWorkRate[activity], WorkToDo[activity]/TimeForSM[activity])*(1-ActualStability[activity])*(1-SMThoroughness[activity])

Units: WU/Day

LCProcessRate[activity]=

AvgLCToBeProcessedInWorkToDo[activity]*WorkRate[activity]

Units: WU/Day

LCToACRate[activity]=

AvgLCToBeAddressedInWorkAwaitingCCMDecision[activity]*ApprovalChangeRate[activity]

Units: WU/Day

NormalProductivity[activity]=

10

Units: WU/(Day*Worker)

OriginalDuration[activity]=

100

Units: Day

PendingWorkCompletedRate[activity,preceding]=
WorkPendingDueToUPReprocess[activity,preceding]/TimeForUPChange[preceding]

Units: WU/Day

PerceivedProgress[activity]=
MAX(0,PerceivedWorkDone[activity]/(ActivityScope[activity]+ExtraWork[activity]))

Units: Dmnl

PerceivedWorkDone[activity]=
WorkAwaitingQM[activity]+WorkCompleted[activity]

Units: WU

PrecedenceRelationship[activity,preceding]=
-1

Units: Day

preceding:¹
(a1-a3)

PredecessorActionRequestRate[activity,preceding]=
WorkAwaitingRFIReply[activity,preceding]*RFIAdoptionRatio[activity]*(1-
RateForCCMFromRFI[activity])/RFIReplyTime[activity]

Units: WU/Day

PredecessorErrorAccomodateRate[activity,preceding]=
IF THEN ELSE (ResourceCommissionStart[activity]=0, 0, 1)*
(WorkAwaitingRFIReply[activity,preceding]/RFIReplyTime[activity])*(1-
RFIAdoptionRatio[activity])

Units: WU/Day

PredecessorFractionOfHiddenError[activity]=
IF THEN ELSE (ActivityStarted[activity]=1:AND:ActivityFinished[activity]=1,SUM (IF
THEN ELSE (PrecedenceRelationship[activity,preceding!]>-1, 1,0)*
(FractionOfHiddenError[preceding!]),0)

Units: WU

PredecessorFractionOfLatentChange[activity]=

¹ Subscript used for preceding activities.

IF THEN ELSE (ActivityStarted[activity]=1:AND:ActivityFinished[activity]=1, SUM (IF THEN ELSE (PrecedenceRelationship[activity,preceding!]>-1, 1, 0)* (FractionOfLatentChange[preceding!])),0)

Units: Dmnl

ProcessedLatentChange[activity]= INTEG (LCPProcessRate[activity]-ProcessedLCReleaseRate[activity], 0)

Units: WU

ProcessedLCReleaseRate[activity]= WorkCompletedRate[activity]* AvgProcessedLatentChangeInWorkAwaitingQM[activity]

Units: WU/Day

ProductionType[activity]= 0

Units: Dmnl

ProductionTypeEnabledProductivity[activity]= NormalProductivity[activity]*(IF THEN ELSE (ProductionType[activity]=0,1,IF THEN ELSE (ProductionType[activity]=1, "TableForProductionTypeEnabled Productivity - Fast"[activity](PerceivedProgress[activity]), "TableForProductionType EnabledProductivity-Slow"[activity](PerceivedProgress[activity]))))

Units: WU/(Day*Worker)

ProgressBasedReliability[activity]= MIN (1,MAX(0, InitialReliability[activity]*MAX(1, FactorByLearningEffect[activity] * ExperienceLevel[activity])))

Units: Dmnl

ProgressBasedStability[activity]= MIN (1,MAX(0, InitialStability[activity]*MAX(1, FactorByLockingEffect[activity] *FamiliarityLevel[activity])))

Units: Dmnl

ProgressInfluentialToReliability[activity]= MIN (1,WorkCompleted[activity]/(ActivityScope[activity]+Extra Work[activity]))

Units: Dmnl

ProgressInfluentialToStability[activity]=
MIN (1,WorkCompleted[activity]/(ActivityScope[activity]+ExtraWork[activity]))

Units: Dmnl

ProjectExtraWork=
SUM (ExtraWork[activity!])

Units: WU

ProjectFinished=
IF THEN ELSE (ProjectProgress>0.999,0,1)

Units: Dmnl

ProjectFinishTime=
INTEGER ("ProjectFinishTime-Raw")

Units: Day

"ProjectFinishTime-Raw"= INTEG (
IF THEN ELSE (ProjectFinished=1,1,0), 0)

Units: Day

ProjectProgress=
IF THEN ELSE (ProjectScope=0, 0, SUM(WorkCompleted[activity!])/ProjectScope)

Units: Dmnl

"ProjectRe-work"=
SUM ("Re-work"[activity!])

Units: WU

ProjectScope=
SUM (IF THEN ELSE (InitialStartTime[activity!]=1e+006, 0 ,ActivityScope[activity!] +
Extra Work[activity!]))

Units: WU

ProjectWorkforceUtilizationRatio=
IF THEN ELSE (ProjectFinished=0, 0, IF THEN ELSE (SUM (AvailableWorkRate
[activity!])=0,0 , SUM(WorkRate[activity!])/SUM(AvailableWorkRate[activity!])))

Units: Dmnl

QMThoroughness[activity]=

1

Units: Dmnl

RateForCCMFromRFI[activity]=

0.5

Units: Dmnl

RCAbsorptionRate[activity]=

AvgRCInWorkToDo[activity]*WorkRate[activity]*(1-LatentEvaluationAdoptionRate
[activity])

Units: WU/Day

RCToLCRate[activity]=

AvgRCInWorkToDo[activity]*WorkRate[activity]*LatentEvaluationAdoptionRate[activity]

Units: WU/Day

"Re-ExecutionRequestOnWorkCompletedRate"[activity]=

MIN (WorkCompleted[activity]/TIME STEP, AdjustedSumRate[activity])

Units: WU/Day

"Re-ExecutionRequestOnWorkNotCompletedRate"[activity]=

(1+InternalSensitivity[activity])*WorkAwaitingQM[activity]*UncoveredErrorGenerationRatio[activity]/AvgQMTime[activity]

Units: WU/Day

"Re-work"[activity]= INTEG (

"Re-workIntroduceRate"[activity],0)

Units: WU

"Re-workIntroduceRate"[activity]=

"Re-ExecutionRequestOnWorkNotCompletedRate"[activity]+
"Re-ExecutionRequestOnWorkCompleted Rate" [activity]

Units: WU/Day

RejectedChange[activity]= INTEG (

ICBecomeRCRate[activity]-RCAbsorptionRate[activity]-RCToLCRate[activity],0)

Units: WU

RejectedChangeRate[activity]=
(1-CCMAoptionRate[activity])*WorkAwaitingCCMDecision[activity]/AvgCCMTime
[activity]

Units: WU/Day

"ReliabilityStartTime-Raw"[activity]= INTEG (
IF THEN ELSE (ActivityStarted[activity]=0, 1 ,0), 0)

Units: Day

ReprocessedHiddenError[activity]= INTEG (
HiddenErrorReprocessRate[activity]-ReprocessedHiddenErrorReleaseRate[activity],0)

Units: WU

ReprocessedHiddenErrorReleaseRate[activity]=
WorkCompletedRate[activity]*AvgReprocessedHiddenErrorInWorkAwaitingQM[activity]

Units: WU/Day

ResourceCommissionStart[activity]=
IF THEN ELSE (ActivityStarted[activity]=1,1,0)

Units: Dmnl

RFIAdoptionRatio[activity]=
0.5

Units: Dmnl

RFIRate[activity,preceding]=
IF THEN ELSE (ResourceCommissionStart[activity]=0, WorkToDo[activity]/
TimeForWork[activity], MIN(AvailableWorkRate[activity], WorkToDo[activity]/
TimeForWork[activity]))*(FractionOfRFIByError[activity,preceding])

Units: WU/Day

RFIReplyTime[activity]=
5

Units: Day

RFIToCCMRate[activity,preceding]=
RFIAdoptionRatio[activity]*RateForCCMFromRFI[activity]*WorkAwaitingRFIReply
[activity,preceding]/RFIReplyTime[activity]

Units: WU/Day

RippleEffectOnReliability[activity]=

"TableForRippleEffectOnReliability(latency)"(PredecessorFractionOfHiddenError
[activity])

Units: Dmnl

RippleEffectOnStability[activity]=

"TableForRippleEffectOnStability(Latency)"(PredecessorFractionOfLatentChange
[activity])

Units: Dmnl

RPTriggeredByInternalAdoption[activity]=

MAX (0,"TotalInternal&ManagerialRPRequest"[activity]*InternalSensitivity[activity]
*FractionOfWorkReleased[activity])

Units: WU/Day

SchedulePressure[activity]=

SMOOTH (IF THEN ELSE (PerceivedProgress[activity]=0,1, FractionOfExpected
Progress[activity]/PerceivedProgress[activity]), 5)

Units: Dmnl

SchedulePressureEffectOnProductivity[activity]=

TableForSchedulePressureEffectOnProductivity(SchedulePressure[activity])

Units: Dmnl

SchedulePressureEffectOnReliability[activity]=

TableForSchedulePressureEffectOnReliability(SchedulePressure[activity])

Units: Dmnl

SMThoroughness[activity]=

1

Units: Dmnl

SumRate[activity]=

TotalExternalAdoption[activity]+TotalInternalAdoption[activity]+
"TotalRe-Execution FromSuccessor"[activity]

Units: WU/Day

TableForExternalConcurrence[activity,preceding](
[(0,0)-(1,1)],(0,1),(0.25,1),(0.5,1),(0.75,1),(1,1))

Units: Dmnl

TableForInternalConcurrence[activity](
[(0,0)-(20,1)],(0,1),(1,1),(10,1))

Units: Dmnl

TableForLearningEffectOnReliability(
[(0,1)-(1,1.25)],(0,1),(0.249123,1.01535),(0.385965,1.02961),(0.501754,1.0625),
(0.617544,1.14035),(0.677193,1.19408),(0.754386,1.22478),(0.859649,1.24123), (1,1.25))

Units: Dmnl

TableForLockingEffectOnStability(
[(0,1)-(1,1.3)],(0,1),(0.133333,1.01447),(0.249123,1.03684),(0.322807,1.05789),
(0.421053,1.12237),(0.5,1.2),(0.568421,1.24079),(0.677193,1.26579),(0.824561
,1.28684),(1,1.3))

Units: Dmnl

"TableForProductionTypeEnabledProductivity-Fast"[activity](
[(0,0)-(1,2)],(0,1.25),(1,0.75))

Units: Dmnl

"TableForProductionTypeEnabledProductivity-Slow"[activity](
[(0,0)-(1,2)],(0,0.75),(1,1.25))

Units: Dmnl

"TableForRippleEffectOnReliability(latency)"(
[(0,0)-(1,1)],(0,1),(0.05,0.9),(0.1,0.8),(0.2,0.6),(0.3,0.45),(0.5,0.25),(1,0))

Units: Dmnl

"TableForRippleEffectOnStability(Latency)"(
[(0,0)-(1,1)],(0,1),(0.05,0.9),(0.1,0.8),(0.2,0.6),(0.3,0.45),(0.5,0.25),(1,0))

Units: Dmnl

TableForSchedulePressureEffectOnProductivity(
[(1,0)-(4,1.25)],(1,1),(1.17895,1.10746),(1.43158,1.20066),(1.63158,1.22807),
(2,1.25),(2.36842,1.20066),(2.76842,1.07456),(4,0.5))

Units: Dmnl

TableForSchedulePressureEffectOnReliability(
[(0,0)-(2,1)],(0,1),(0.5,1),(1,1),(1.12,1),(1.25,0.98),(1.4,0.95),(1.7,0.9),
(2,0.875),(2.5,0.825),(5,0.4),(10,0.04))

Units: Dmnl

TimeForSM[activity]=

1

Units: Day

TimeForUPChange[activity]=

5

Units: Day

TimeForWork[activity]=

1

Units: Day

TotalExConcurrence[activity]=

MIN (ExConcurrenceFromFS[activity], ExConcurrenceFromSS[activity])

Units: Dmnl

TotalExternalAdoption[activity]=

TotalRPRequestFromPredecessor[activity]+TotalRPRequestFromSuccessor[activity]

Units: WU/Day

TotalFromQMFlow[activity]=

WorkCompletedRate[activity]+"Re-ExecutionRequestOn WorkNotCompletedRate" [activity]

Units: WU/Day

"TotalInternal&ManagerialRPRequest"[activity]=

ApprovalChangeRate[activity]

Units: WU/Day

TotalInternalAdoption[activity]=

(ApprovalChangeRate[activity]+SUM(PredecessorErrorAccomodateRate[activity,
preceding!]))*InternalSensitivity[activity]

Units: WU/Day

"TotalRe-ExecutionFromSuccessor"[activity]=
(HiddenErrorAddressRate[activity]+LCtoACRate[activity])*InternalSensitivity
[activity]

Units: WU/Day

TotalRPRequestFromPredecessor[activity]=
IF THEN ELSE (ActivityStarted[activity]=1:AND:ActivityFinished[activity]=1,
SUM (IF THEN ELSE (PrecedenceRelationship[activity, preceding!]>-1,1,0)*
(TotalInternalAdoption[preceding!]))*VMAX(ExternalSensitivity[activity,
preceding!])*(1+InternalSensitivity[activity]),0)

Units: WU/Day

TotalRPRequestFromSuccessor[activity]=
IF THEN ELSE (ActivityStarted[activity]=1:AND:ActivityFinished[activity]=1,
SUM (IF THEN ELSE (PrecedenceRelationship[preceding!,activity]>-1, 1 ,0)*
(TotalInternalAdoption[preceding!]))*VMAX(ExternalSensitivity[preceding!,activity])*(1+I
nternalSensitivity[activity]),0)

Units: WU/Day

TotalWorkAgainRequested= INTEG (
SUM ("Re-ExecutionRequestOnWorkNotCompletedRate"[activity!]),0)

Units: WU

TotalWorkComplete=
SUM (WorkCompleted[activity!])

Units: WU

TotalWorkDone= INTEG (
SUM (WorkRate[activity!]), 0)

Units: WU

TotalWorkExecutedAgain=
SUM (ExtraWork[activity!]+"Re-work"[activity!])

Units: WU

TotalWorkInProgress=

SUM (WorkInProgress[activity!])

Units: WU

UncoveredError[activity]= INTEG (

UncoveredErrorDiscoveryRate[activity]-UncoveredErrorSolveRate[activity], 0)

Units: WU

UncoveredErrorDiscoveryRate[activity]=

AvgErrorInWorkAwaitingQM[activity]*TotalFromQMFlow[activity]*QMThoroughness[activity]

Units: WU/Day

UncoveredErrorGenerationRatio[activity]=

IF THEN ELSE (ActivityFinished[activity]=1:AND:ActivityStarted[activity]=1,(1-ActualReliability[activity])*QMThoroughness[activity], 0)

Units: Dmnl

UncoveredErrorSolveRate[activity]=

AvgUncoveredErrorInWorkToDo[activity]*WorkRate[activity]

Units: WU/Day

UPFinishTimeForDNStart[activity]=

SAMPLE IF TRUE (ActivityFinished[activity]=1,Time+TIME STEP,0)

Units: Day

UPProgress[activity]=

FractionOfWorkReleasedForDNStart[activity]

Units: Dmnl

WorkAvailable[activity]=

MAX(0,MIN(TotalExConcurrence[activity], WorkAvailableFromInConcurrence[activity])*ActivityScope[activity]-WorkInProgress[activity]) *IF THEN ELSE (InitialStartTime[activity]<=Time, 1 , 0)

Units: WU

WorkAvailableFromInConcurrence[activity]=

TableForInternalConcurrence[activity](PerceivedProgress[activity])

Units: Dmnl

WorkAwaitingCCMDecision[activity]= INTEG (
 +ChangeToCCMRate[activity]+SUM(RFIToCCMRate[activity,preceding!])-
 ApprovalChangeRate[activity]-RejectedChangeRate[activity], 0)

Units: WU

WorkAwaitingQM[activity]= INTEG (
 +WorkRate[activity]-"Re-ExecutionRequestOnWorkNotCompletedRate"[activity]-
 WorkCompletedRate[activity],0)

Units: WU

WorkAwaitingRFIReply[activity,preceding]= INTEG (
 RFIRate[activity,preceding]-RFIToCCMRate[activity,preceding]-PredecessorError
 AccomodateRate[activity,preceding]-PredecessorActionRequestRate[activity, preceding],0)

Units: WU

WorkCompleted[activity]= INTEG (
 +WorkCompletedRate[activity]-"ReExecutionRequestOnWorkCompletedRate" [activity],0)

Units: WU

WorkCompletedRate[activity]=
 (1-UncoveredErrorGenerationRatio[activity])*WorkAwaitingQM[activity]/
 AvgQMTime[activity]

Units: WU/Day

WorkExecutedAgain[activity]=
 ExtraWork[activity]+"Re-work"[activity]

Units: WU

WorkInProgress[activity]=
 WorkAwaitingCCMDecision[activity]+WorkAwaitingQM[activity]+SUM(WorkAwaitingRFI
 Reply[activity,preceding!])+SUM(WorkPendingDueToUPReprocess[activity,preceding!])+W
 orkToDo[activity]+WorkCompleted[activity]

Units: WU

WorkIntroduceRate[activity]=
 WorkAvailable[activity]/TIME STEP+ExtraWorkIntroduceRate[activity]

Units: WU/Day

WorkPendingDueToUPReprocess[activity,preceding]= INTEG (
 +PredecessorActionRequestRate[activity,preceding]-PendingWorkCompletedRate
 [activity,preceding],0)

Units: WU

WorkRate[activity]=
 MAX(0,IF THEN ELSE (ResourceCommissionStart[activity]=1, MIN (Available
 WorkRate[activity], (WorkToDo[activity]/TimeForWork[activity])) * (1-Identified
 ChangeGenerationRatio[activity]-SUM(FractionOfRFIByError[activity,preceding!])-
 SUM(FractionOfRFIByChange[activity,preceding!])), IF THEN ELSE
 (ActivityStarted[activity]=1, MIN(AvailableWorkRate[activity], (WorkToDo[activity]/
 TimeForWork[activity]))*(1-IdentifiedChangeGenerationRatio[activity]
 SUM(FractionOfRFIByError[activity,preceding!])-SUM(FractionOfRFIByChange
 [activity,preceding!])),0)))

Units: WU/Day

WorkToDo[activity]= INTEG (
 ApprovalChangeRate[activity]-ChangeToCCMRate[activity]+RejectedChangeRate[
 activity]+SUM(PendingWorkCompletedRate[activity,preceding!])+"Re-Execution
 RequestOnWorkNotCompletedRate"[activity]+"Re-ExecutionRequestOnWork
 CompletedRate"[activity]+SUM(PredecessorErrorAccomodateRate[activity,preceding!])+W
 orkIntroduceRate[activity]-SUM(RFIRate[activity,preceding!])-WorkRate [activity], 0)

Units: WU

APPENDIX III

JAVA CODES FOR THE WEB-BASED DPM SYSTEM

Java codes are divided as major two parts: a server-side application and a client-side application in order to achieve flexibility. Java code examples are also provided.

Server-Side Packages and Classes (or Interfaces)

package `gpm.database`: connects DPM and Primavera database (Oracle 9i)

- `DatbaseListener`
- `OracleDatbaseConnection`
- `PrimaveraOracleDatbaseConnection`

package `gpm.interfaces.impl`: deals with core server implementation connecting Oracle 9i and Vensim

- `ClientEventHandlerImpl`
- `CloseWindowAndExit`
- `DatabaseEventDispathcerImpl`
- `DatabaseInitializerImpl`
- `DPMEventHandlerImpl`
- `Graph`
- `MainServer`
- `PrimaveraDB`
- `PrimaveraDBImpl`
- `ServerImpl`
- `Vensim`
- `VensimGraph`

package `gpm.interfaces.rmi`; deals with RMI

- (Interfaces)
- `ClinetEvnetHandler`
- `DatabaseEventDispathcer`

DatabaseInitializer

Server

package gpm.s.util; represents utility classes to represent a project

Activity

Event

Precedence

Project

Rectangle

SupplementInfo

Version

Client-Side Packages and Classes

package gpm.s.client.gui: deals with graphical user interface

Client

CompareDialog

GameData

GameDialog

GameOutputCanvas

ImportProjectDialog

NewProjectDialog

OpenProjectDialog

OutputSummaryDialog

ProjectOutputDialog

RelationDialog

SimulationDialog

SummaryOutputDialog

TrackingProjectDialog

VersionTrackChart

package gpm.s.client.output: deals with graphical output interface

ActivityChart

ActivityOutputChart

Bar DrawGraph

Graph

MessageFrame

OutputCanvas1

OutputCanvas2
OutputCanvas3
OutputCanvas4
ProjectOutput

package gpms.client.tablemodels: deals with table models

ActivityTableModel
DSMTableModel
PrecedenceTableModel
PrimaveraProjectTableModel
ProjectTableModel
VersionTableModel

Code examples follow. Due to space limitation, several classes are omitted. For example, only 'Client' class is introduced representing a package 'gpms.client.gui'.

```

package gpms.database;

import gpms.interfaces.impl.ServerImpl;
import java.io.BufferedReader;
import java.io.PrintStream;
import java.net.*;

public class DatabaseListener extends Thread
{
    public DatabaseListener(ServerImpl serverimpl)
    {
        RUNNING = true;
        socket = null;
        in = null;
        server = serverimpl;
        initSocket();
        start();
    }
    public void dispatch(String s)
    {
        try
        {
            server.dispatch(s);
        }
        catch(Exception _ex)
        {
        }
    }
    public void initSocket()
    {
        try
        {
            socket = new DatagramSocket(PORT);
        }
        catch(SocketException _ex)
        {
        }
    }
    public void run()

```

```

{
    while (RUNNING)
    {
        byte abyte0[] = new byte[256];
        DatagramPacket datagrampacket = new
            DatagramPacket (abyte0, abyte0.length);
        try
        {
            socket.receive (datagrampacket);
        }
        catch (Exception _ex)
        {
        }
        String s = new String (datagrampacket.getData ());
        int i = s.lastIndexOf ("*");
        String s1 = s.substring (0, i);
        dispatch (s1);
    }
    socket.close ();
}
private ServerImpl server;
private boolean RUNNING;
private DatagramSocket socket;
private BufferedReader in;
public static int PORT = 8888;
}

```



```

package gpms.database;

import java.io.PrintStream;
import java.sql.*;
import oracle.jdbc.driver.OracleDriver;

public class OracleDatabaseConnection
{
    public OracleDatabaseConnection()
    {
        connStr = "jdbc:oracle:thin:@cee-zzrd:1521:PMDB";
        connStrUser = "ADMUSER";
        connStrPass = "admuser";
    }
    public OracleDatabaseConnection(int k)
    {
        if(k == 1)
        {
            connStr = "jdbc:oracle:thin:@cee-zzrd:1521:PMDB";
            connStrUser = "ADMUSER";
            connStrPass = "admuser";
        }
    }
    public void close()
    {
        try
        {
            rset.close();
            stmt.close();
        }
        catch(SQLException se)
        {
            se.printStackTrace();
        }
    }
    public void connect()
    {
        try

```

```

    {
        DriverManager.registerDriver(new OracleDriver());
        conn = DriverManager.getConnection(connStr,
            connStrUser, connStrPass);
    }
    catch(SQLException se)
    {
        se.printStackTrace();
    }
}
public void disconnect()
{
    try
    {
        rset.close();
        stmt.close();
        conn.close();
    }
    catch(SQLException se)
    {
        se.printStackTrace();
    }
}
public ResultSet executeSQL(String sqlStatement)
{
    if(conn == null)
        connect();
    try
    {
        stmt = conn.createStatement();
        rset = stmt.executeQuery(sqlStatement);
        return rset;
    }
    catch(SQLException se)
    {
        se.printStackTrace();
    }
    return null;
}

```

```

    }
    public String getConnStr()
    {
        return connStr;
    }
    public String getConnStrPass()
    {
        return connStrPass;
    }
    public String getConnStrUser()
    {
        return connStrUser;
    }
    public void setConnStr(String newConnStr)
    {
        connStr = newConnStr;
    }
    public void setConnStrPass(String newConnStrPass)
    {
        connStrPass = newConnStrPass;
    }
    public void setConnStrUser(String newConnStrUser)
    {
        connStrUser = newConnStrUser;
    }
    private String connStr;
    private String connStrUser;
    private String connStrPass;
    private Connection conn;
    private ResultSet rset;
    private Statement stmt;
}

```

```

package gpms.database;

import java.io.PrintStream;
import java.sql.*;
import oracle.jdbc.driver.OracleDriver;

public class OracleDatabaseConnectionPri
{
    public OracleDatabaseConnectionPri()
    {
        connStr = "jdbc:oracle:thin:@cee-zzrd:1521:PMDB";
        connStrUser = "ADMUSER";
        connStrPass = "admuser";
    }
    public OracleDatabaseConnectionPri(int k)
    {
        if(k == 1)
        {
            connStr = "jdbc:oracle:thin:@cee-zzrd:1521:PMDB";
            connStrUser = "ADMUSER";
            connStrPass = "admuser";
        }
    }
    public void close()
    {
        try
        {
            rset.close();
            stmt.close();
        }
        catch(SQLException se)
        {
            se.printStackTrace();
        }
    }
    public void connect()
    {
        try

```

```

    {
        DriverManager.registerDriver(new OracleDriver());
        conn = DriverManager.getConnection(connStr,
            connStrUser, connStrPass);
    }
    catch(SQLException se)
    {
        se.printStackTrace();
    }
}
public void disconnect()
{
    try
    {
        rset.close();
        stmt.close();
        conn.close();
    }
    catch(SQLException se)
    {
        se.printStackTrace();
    }
}
public ResultSet executeSQL(String sqlStatement)
{
    if(conn == null)
        connect();
    try
    {
        stmt = conn.createStatement();
        rset = stmt.executeQuery(sqlStatement);
        return rset;
    }
    catch(SQLException se)
    {
        se.printStackTrace();
    }
    return null;
}

```

```

    }
    public String getConnStr()
    {
        return connStr;
    }
    public String getConnStrPass()
    {
        return connStrPass;
    }
    public String getConnStrUser()
    {
        return connStrUser;
    }
    public void setConnStr(String newConnStr)
    {
        connStr = newConnStr;
    }
    public void setConnStrPass(String newConnStrPass)
    {
        connStrPass = newConnStrPass;
    }
    public void setConnStrUser(String newConnStrUser)
    {
        connStrUser = newConnStrUser;
    }
    private String connStr;
    private String connStrUser;
    private String connStrPass;
    private Connection conn;
    private ResultSet rset;
    private Statement stmt;
}

```

```
package gpms.interfaces.impl;

import gpms.interfaces.rmi.ClientEventHandler;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class ClientEventHandlerImpl extends UnicastRemoteObject
    implements ClientEventHandler
{
    public ClientEventHandlerImpl()
        throws RemoteException
    {
    }
    public void handle(String s)
        throws RemoteException
    {
    }
}
```

```
package gpms.interfaces.impl;

import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class CloseWindowAndExit extends WindowAdapter
{
    public void windowClosing(WindowEvent windowevent)
    {
        System.exit(0);
    }
    public CloseWindowAndExit()
    {
    }
}
```



```

package gpms.interfaces.impl;

import gpms.database.OracleDatabaseConnection;
import gpms.database.OracleDatabaseConnectionPri;
import gpms.interfaces.rmi.DatabaseEventDispatcher;
import gpms.util.Event;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class DatabaseEventDispatcherImpl extends UnicastRemoteObject
    implements DatabaseEventDispatcher
{
    public DatabaseEventDispatcherImpl(OracleDatabaseConnection
    ODBCConnection) throws RemoteException
    {
        this.ODBCConnection = ODBCConnection;
    }
    public DatabaseEventDispatcherImpl(OracleDatabaseConnectionPri
    ODBCConnectionPri) throws RemoteException
    {
        this.ODBCConnectionPri = ODBCConnectionPri;
    }
    public void dispatch(Event event1)
        throws RemoteException
    {
    }
    public void dispatch(String sql)
    {
        OracleDatabaseConnection ODBCConnection = new
            OracleDatabaseConnection();
        ODBCConnection.executeSQL(sql);
        ODBCConnection.disconnect();
    }
    public void dispatchPri(Event event1)
        throws RemoteException
    {
    }
}

```

```
public void dispatchPri(String sql)
{
    OracleDatabaseConnectionPri ODBCConnectionPri = new
        OracleDatabaseConnectionPri();
    ODBCConnectionPri.executeSQL(sql);
    ODBCConnectionPri.disconnect();
}
private OracleDatabaseConnection ODBCConnection;
private OracleDatabaseConnectionPri ODBCConnectionPri;
}
```

```

package gpms.interfaces.impl;

import gpms.client.output.ActivityOutput;
import gpms.client.output.ProjectOutput;
import gpms.database.OracleDatabaseConnection;
import gpms.database.OracleDatabaseConnectionPri;
import gpms.interfaces.rmi.DatabaseInitializer;
import gpms.util.*;
import java.io.PrintStream;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Vector;

public class DatabaseInitializerImpl extends UnicastRemoteObject
    implements DatabaseInitializer
{
    public DatabaseInitializerImpl()
        throws RemoteException
    {
    }
    public DatabaseInitializerImpl(OracleDatabaseConnection
    oracledatabaseconnection) throws RemoteException
    {
        ODBCConnection = oracledatabaseconnection;
    }
    public DatabaseInitializerImpl(OracleDatabaseConnectionPri
    oracledatabaseconnectionpri) throws RemoteException
    {
        ODBCConnectionPri = oracledatabaseconnectionpri;
    }
    public Vector getActivity(int i)
        throws RemoteException
    {
        Vector vector = new Vector();
        OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    }
}

```

```

ResultSet resultset =
    oracledatabaseconnection.executeSQL("SELECT ID, CODE, WBS,
    NAME, DURATION, ST, FT, DMOSTLIKELY, DOPTIMISTIC,
    DPessimistic, RELIABILITY, STABILITY, COMPLEXITY,
    CONTINGENCY, BUFFERING, INTERNALSENSITIVITY,
    ISCRITICALACTIVITY, PRIACTID, SIMST, SIMDURATION, SIMORNOT
    FROM DPM_ACTIVITY WHERE PROJID = '" + i + "'");
if(resultset != null)
    try
    {
        int j;
        String s;
        String s1;
        String s2;
        int k;
        int l;
        int i1;
        int j1;
        int k1;
        int l1;
        double d;
        double d1;
        int i2;
        double d2;
        double d3;
        double d4;
        int j2;
        int k2;
        int l2;
        int i3;
        int j3;
        int k3;
        for(; resultset.next(); vector.add(new Activity(j, s,
            s1, s2, k, l, i1, j1, k1, l1, d, d1, i2, d2, d3, d4,
            j2, k2, l2, i3, j3, k3)))
        {
            j = resultset.getInt(1);
            s = resultset.getString(2);

```

```

        s1 = resultset.getString(3);
        s2 = resultset.getString(4);
        k = resultset.getInt(5);
        l = resultset.getInt(6);
        i1 = resultset.getInt(7);
        j1 = resultset.getInt(8);
        k1 = resultset.getInt(9);
        l1 = resultset.getInt(10);
        d = resultset.getDouble(11);
        d1 = resultset.getDouble(12);
        i2 = resultset.getInt(13);
        d2 = resultset.getDouble(14);
        d3 = resultset.getDouble(15);
        d4 = resultset.getDouble(16);
        j2 = resultset.getInt(17);
        k2 = resultset.getInt(18);
        l2 = resultset.getInt(19);
        i3 = resultset.getInt(20);
        j3 = resultset.getInt(21);
        k3 = resultset.getInt(22);
    }
}
catch(SQLException _ex)
{
}
oracledatabaseconnection.disconnect();
return vector;
}
public Vector getActivityOutput(int i, int j)
throws RemoteException
{
    Vector vector = new Vector();
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeQuery("SELECT * FROM
        DPM_ACTIVITYOUTPUT WHERE PROJID = '" + i + "' AND VER_ID =
        '" + j + "'");
}

```

```

if(resultset != null)
    try
    {
        int k;
        int l;
        int i1;
        int j1;
        int k1;
        int l1;
        int i2;
        int j2;
        int k2;
        String s;
        int l2;
        int i3;
        int j3;
        int k3;
        int l3;
        String s1;
        double d;
        double d1;
        int i4;
        int j4;
        for(; resultset.next(); vector.add(new
            ActivityOutput(k, s, l, i1, j1, k1, l2, l1, i2, j2,
                k2, i3, j3, k3, l3, s1, d, d1, i4, j4)))
        {
            k = resultset.getInt(1);
            l = resultset.getInt(2);
            i1 = resultset.getInt(3);
            j1 = resultset.getInt(4);
            k1 = resultset.getInt(5);
            resultset.getInt(6);
            l1 = resultset.getInt(7);
            i2 = resultset.getInt(8);
            j2 = resultset.getInt(9);
            k2 = resultset.getInt(10);
            s = resultset.getString(11);

```

```

        resultset.getInt(12);
        l2 = resultset.getInt(13);
        i3 = resultset.getInt(14);
        j3 = resultset.getInt(15);
        k3 = resultset.getInt(16);
        l3 = resultset.getInt(17);
        s1 = resultset.getString(18);
        d = resultset.getDouble(19);
        d1 = resultset.getDouble(20);
        i4 = resultset.getInt(21);
        j4 = resultset.getInt(22);
    }
}
catch(Exception _ex)
{
}
oracledatabaseconnection.disconnect();
return vector;
}
public Vector getActivityPri(int i)
throws RemoteException
{
Vector vector = new Vector();
OracleDatabaseConnectionPri oracledatabaseconnectionpri = new
OracleDatabaseConnectionPri();
ResultSet resultset =
oracledatabaseconnectionpri.executeQuery("SELECT TASK_ID,
TARGET_DRTN_HR_CNT, TASK_NAME FROM TASK WHERE PROJ_ID = '" +
i + "'");
if(resultset != null)
try
{
for(int j = 0; resultset.next(); j++)
{
int k = resultset.getInt(1);
int l = resultset.getInt(2);
String s = resultset.getString(3);
int il = 1;

```

```

        String s1 = "a" + (new Integer(j + 1)).toString();
        String s2 = null;
        vector.add(j, new Activity(k, s1, s, s2, 1, 0, il,
            1, 1, 1, 1.0D, 1.0D, 1, 0.20000000000000001D,
            0.0D, 0.5D, 1, k, 0, 0, 0, 1));
    }
    oracledatabaseconnectionpri.disconnect();
}
catch(Exception _ex)
{
}
return vector;
}
public Vector getPredActivity(int i)
    throws RemoteException
{
    Vector vector = new Vector();
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeQuery("SELECT A.ID,
        A.PREDACTIVITYID, A.RELATION, A.LAG, A.SENSITIVITY, B.CODE
        FROM DPM_ACTIVITYRELATION A, DPM_ACTIVITY B WHERE A.ID = '"
        + i + "' AND B.ID = A.PREDACTIVITYID");
    if(resultset != null)
        try
        {
            int j;
            int k;
            String s;
            int l;
            int il;
            String s1;
            for(; resultset.next(); vector.add(new Precedence(j, k,
                s, l, il, s1)))
            {
                j = resultset.getInt(1);
                k = resultset.getInt(2);

```



```

        s = resultSet.getString(3);
        l = resultSet.getInt(4);
        i1 = resultSet.getInt(5);
        s1 = resultSet.getString(6);
    }
}
catch(SQLException _ex)
{
}
oracledatabaseconnection.disconnect();
return vector;
}
public ProjectOutput getProjectOutput(int i, int j)
throws RemoteException
{
    ProjectOutput projectoutput = null;
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultSet =
        oracledatabaseconnection.executeSQL("SELECT * FROM
        DPM_PROJECTOUTPUT WHERE PROJID = '" + i + "'AND VER_ID = '"
        + j + "'");
    if(resultSet != null)
        try
        {
            while(resultSet.next())
            {
                resultSet.getInt(1);
                resultSet.getInt(2);
                int k = resultSet.getInt(3);
                int l = resultSet.getInt(4);
                int i1 = resultSet.getInt(5);
                int j1 = resultSet.getInt(6);
                resultSet.getInt(7);
                int k1 = resultSet.getInt(8);
                resultSet.getInt(9);
                projectoutput = new ProjectOutput(k1, k, l, i1,
                    j1);
            }
        }
    }
}

```

```

        }
    }
    catch(Exception _ex)
    {
    }
    oracledatabaseconnection.disconnect();
    return projectoutput;
}
public Vector getProjects()
    throws RemoteException
{
    Vector vector = new Vector();
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeQuery("SELECT PROJID, PROJNAME,
        CPM, PERT, DPMWOACTION FROM DPM_PROJECT");
    if(resultset != null)
        try
        {
            int i;
            String s;
            int j;
            int k;
            int l;
            for(; resultset.next(); vector.add(new Project(i, s, j,
                k, l)))
            {
                i = resultset.getInt(1);
                s = resultset.getString(2);
                j = resultset.getInt(3);
                k = resultset.getInt(4);
                l = resultset.getInt(5);
            }
        }
    catch(SQLException _ex)
    {
    }
}

```

```

        oracledatabaseconnection.disconnect();
        return vector;
    }
    public Vector getProjectsPri()
        throws RemoteException
    {
        Vector vector = new Vector();
        OracleDatabaseConnectionPri oracledatabaseconnectionpri = new
            OracleDatabaseConnectionPri();
        ResultSet resultset =
            oracledatabaseconnectionpri.executeSQL("SELECT PROJ_ID,
            PROJ_SHORT_NAME FROM PROJECT");
        if(resultset != null)
            try
            {
                int i;
                String s;
                int j;
                int k;
                int l;
                for(; resultset.next(); vector.add(new Project(i, s, j,
                    k, l)))
                {
                    i = resultset.getInt(1);
                    s = resultset.getString(2);
                    j = 0;
                    k = 0;
                    l = 0;
                }
            }
            catch(SQLException _ex)
            {
            }
        oracledatabaseconnectionpri.disconnect();
        return vector;
    }
    private OracleDatabaseConnection ODBCConnection;
    private OracleDatabaseConnectionPri ODBCConnectionPri;

```

}

```

package gpms.interfaces.impl;

import gpms.client.gui.Client;
import gpms.util.Activity;
import java.io.PrintStream;
import java.rmi.RemoteException;
import java.util.StringTokenizer;
import java.util.Vector;

public class DPMEventHandlerImpl extends ClientEventHandlerImpl
{
    public DPMEventHandlerImpl(Client client1)
        throws RemoteException
    {
        client = client1;
    }
    public synchronized void handle(String s)
        throws RemoteException
    {
        StringTokenizer stringtokenizer = new StringTokenizer(s,
            "##");
        int i = Integer.parseInt(stringtokenizer.nextToken());
        if(i == 1000)
        {
            int j = Integer.parseInt(stringtokenizer.nextToken());
            int k = Integer.parseInt(stringtokenizer.nextToken());
            if(k == client.projectId)
            {
                if(j == 2000)
                {
                    int l =
                        Integer.parseInt(stringtokenizer.nextToken());
                    String s1 = new
                        String(stringtokenizer.nextToken());
                    String s3 = new
                        String(stringtokenizer.nextToken());
                    String s5 = new
                        String(stringtokenizer.nextToken());
                }
            }
        }
    }
}

```

```

int i2 =
    Integer.parseInt(stringtokenizer.nextToken());
int k2 =
    Integer.parseInt(stringtokenizer.nextToken());
int i3 =
    Integer.parseInt(stringtokenizer.nextToken());
Activity activity = new Activity(l, s1, s3, s5, i2,
    k2, i3, i2, i2, i2, 0.0D, 0.0D, 1,
    0.20000000000000001D, 0.0D, 0.5D, 1, 0, 0, 0, 0,
    1);
boolean flag = true;
int i4 = client.activityVector.size();
if(((Activity)client.activityVector.elementAt(j4)).g
etId()== l)
    {
        flag = false;
        return;
    }
if(flag)
    client.activityVector.add(activity);
} else
if(j == 2001)
{
    int i1 =
        Integer.parseInt(stringtokenizer.nextToken());
    int k1 = client.activityVector.size();
    for(int l1 = 0; l1 < k1; l1++)
        if(((Activity)client.activityVector.elementAt(l1)).g
etId()==i1)client.activityVector.removeElementAt(l1)
        ;
} else
if(j == 2002)
{
    int j1 =
        Integer.parseInt(stringtokenizer.nextToken());
    String s2 = new
        String(stringtokenizer.nextToken());
    String s4 = new

```

```

        String(stringtokenizer.nextToken());
String s6 = new
        String(stringtokenizer.nextToken());
int j2 =
        Integer.parseInt(stringtokenizer.nextToken());
int l2 =
        Integer.parseInt(stringtokenizer.nextToken());
int j3 =
        Integer.parseInt(stringtokenizer.nextToken());
int k3 = client.activityVector.size();
for(int l3 = 0; l3 < k3; l3++)

if(((Activity)client.activityVector.elementAt(l3)).g
etId() == j1)
{
    ((Activity)client.activityVector.elementAt(l3)).setD
uration(j2);
    ((Activity)client.activityVector.elementAt(l3)).setC
ode(s2);
    ((Activity)client.activityVector.elementAt(l3)).setW
bs(s4);
    ((Activity)client.activityVector.elementAt(l3)).setS
t(l2);
    ((Activity)client.activityVector.elementAt(l3)).setF
t(j3);
    ((Activity)client.activityVector.elementAt(l3)).setN
ame(s6);
}
}
client.update();
}
}
private Client client;
}

```

```

package gpms.interfaces.impl;

import java.awt.Color;
import java.awt.Graphics;
import java.io.PrintStream;

public class Graph
{
    public Graph()
    {
    }
    public Graph(float af[], float af1[], int i, int j, int k, int l,
        String s)
    {
        xValues = new float[af.length];
        for(int i1 = 0; i1 < af.length; i1++)
            xValues[i1] = af[i1];
        yValues = new float[af1.length];
        for(int j1 = 0; j1 < af1.length; j1++)
            yValues[j1] = af1[j1];
        xPos = i;
        yPos = j;
        width = k;
        height = l;
        name = s;
        minXValue = xValues[0];
        maxXValue = xValues[0];
        minYValue = yValues[0];
        maxYValue = yValues[0];
        try
        {
            for(int k1 = 0; k1 < xValues.length; k1++)
            {
                if(xValues[k1] < minXValue)
                    minXValue = xValues[k1];
                if(xValues[k1] > maxXValue)
                    maxXValue = xValues[k1];
            }
        }
    }
}

```



```

        for(int l1 = 0; l1 < yValues.length; l1++)
        {
            if(yValues[l1] < minYValue)
                minYValue = yValues[l1];
            if(yValues[l1] > maxYValue)
                maxYValue = yValues[l1];
        }
    }
    catch(ArrayIndexOutOfBoundsException _ex)
    {
    }
}

public void drawGraph(Graphics g)
{
    g.setColor(color);
    xLength = maxXValue - minXValue;
    yLength = maxYValue - minYValue;
    float f = (float)width / xLength;
    float f1 = (float)height / yLength;
    try
    {
        for(int i = 0; i < xValues.length - 1; i++)
            g.drawLine(xPos + (int)((xValues[i] - minXValue) * f),
                (yPos + height) - (int)((yValues[i] - minYValue) *
                    f1), xPos + (int)((xValues[i + 1] - minXValue) * f),
                (yPos + height) - (int)((yValues[i + 1] - minYValue)
                    * f1));
    }
    catch(ArrayIndexOutOfBoundsException _ex)
    {
    }
    g.setColor(Color.black);
    g.drawLine(xPos, yPos + height, xPos + width, yPos + height);
    g.drawLine(xPos, yPos, xPos, yPos + height);
    g.drawString(Float.toString(minXValue), xPos, yPos + height +
        15);
    g.drawString(Float.toString(maxXValue), (xPos + width) - 10,
        yPos + height + 15);
}

```

```

        g.drawString(Float.toString(minYValue), xPos - 90, yPos +
            height);
        g.drawString(Float.toString(maxYValue), xPos - 90, yPos +10);
        g.drawString(name, xPos + 20, yPos - 10);
    }
    public void setTheColor(Color color1)
    {
        color = color1;
    }
    private float xValues[];
    private float yValues[];
    private float minXValue;
    private float minYValue;
    private float maxXValue;
    private float maxYValue;
    private float xLength;
    private float yLength;
    private int xPos;
    private int yPos;
    private int width;
    private int height;
    private int minXLabel;
    private int maxXLabel;
    private Color color;
    private String name;
}

```

```

package gpms.interfaces.impl;

import gpms.database.DatabaseListener;
import gpms.database.OracleDatabaseConnection;
import gpms.interfaces.rmi.*;
import java.io.PrintStream;
import java.rmi.Naming;

public class MainServer
{
    public MainServer()
    {
        HOST = "cee-zzrt.cee.uiuc.edu";
        PORT = "1099";
        try
        {
            OracleDatabaseConnection ODBCConnectionA = new
                OracleDatabaseConnection();
            OracleDatabaseConnection ODBCConnectionB = new
                OracleDatabaseConnection();
            DatabaseInitializerImpl databaseInitializer = new
                DatabaseInitializerImpl(ODBCConnectionA);
            Naming.rebind("rmi://" + HOST + ":" + PORT +
                "/DatabaseInitializer", (DatabaseInitializer) database
                Initializer);
            DatabaseEventDispatcherImpl databaseEventDispatcher = new
                DatabaseEventDispatcherImpl(ODBCConnectionB);
            Naming.rebind("rmi://" + HOST + ":" + PORT +
                "/DatabaseEventDispatcher", DatabaseEventDispatcher) d
                atabaseEventDispatcher);
            ServerImpl server = new ServerImpl(databaseInitializer,
                databaseEventDispatcher);
            Naming.rebind("rmi://" + HOST + ":" + PORT + "/Server",
                (Server) server);
            new DatabaseListener(server);
        }
        catch(Exception e)
        {

```

```
        System.out.println(e);
    }
}
public static void main(String args[])
{
    new MainServer();
}
public String HOST;
public String PORT;
}
```

```

package gpms.interfaces.impl;

import java.awt.*;
import java.io.PrintStream;

public class OutputCanvas extends Canvas
{
    public OutputCanvas()
    {
    }
    public void draw(float af[], float afl[], int i, int j, int k,
        int l, String s)
    {
        xValues = af;
        yValues = afl;
        xPos = i;
        yPos = j;
        graphwidth = k;
        graphheight = l;
        name = s;
        repaint();
    }
    public void paint(Graphics g)
    {
        if(xValues != null)
        {
            Graph graph = new Graph(xValues, yValues, xPos, yPos,
                graphwidth, graphheight, name);
            graph.setTheColor(Color.red);
            graph.drawGraph(g);
        }
    }
    public float xValues[];
    public float yValues[];
    public int xPos;
    public int yPos;
    public int graphwidth;
    public int graphheight;
}

```

```
    public String name;  
}
```

```

package gpms.interfaces.impl;

import gpms.database.OracleDatabaseConnection;
import gpms.util.*;
import java.io.PrintStream;
import java.rmi.RemoteException;
import java.sql.*;
import java.util.*;

public class PrimaveraDB
{
    public PrimaveraDB()
        throws RemoteException
    {
        props = new Properties();
        props.put("user", "privuser");
        props.put("password", "privuser");
        props.put("db", "pmdb");
        props.put("server", "18.58.1.222:1433");
        try
        {
            myDriver =
                (Driver)Class.forName("weblogic.jdbc.mssqlserver4.Driver"
                ). newInstance();
            conn = myDriver.connect("jdbc:weblogic:mssqlserver4",
                props);
        }
        catch(Exception _ex)
        {
        }
    }
    public void closeConnection()
        throws RemoteException
    {
        try
        {
            conn.close();
        }
    }
}

```

```

        catch(Exception exception)
        {
            exception.printStackTrace();
        }
    }

    public int findDPMId(int i)
    {
        int j = 0;
        OracleDatabaseConnection oracledatabaseconnection = new
            OracleDatabaseConnection();
        ResultSet resultset =
            oracledatabaseconnection.executeQuery("SELECT ID FROM
            ACTIVITY WHERE PRIACTID = '" + i + "'");
        if(resultset != null)
            try
            {
                while(resultset.next())
                    j = resultset.getInt(1);
            }
            catch(Exception _ex)
            {
            }
        oracledatabaseconnection.disconnect();
        return j;
    }

    public String findPredCode(int i)
    {
        String s = "a1";
        OracleDatabaseConnection oracledatabaseconnection = new
            OracleDatabaseConnection();
        ResultSet resultset =
            oracledatabaseconnection.executeQuery("SELECT CODE FROM
            ACTIVITY WHERE ID = '" + i + "'");
        if(resultset != null)
            try
            {

```



```

        while(resultset.next())
            s = resultset.getString(1);
    }
    catch(Exception _ex)
    {
    }
    oracledatabaseconnection.disconnect();
    return s;
}

public Vector getPrimaveraActivity(int i)
    throws RemoteException
{
    Vector vector = new Vector();
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    try
    {
        Statement statement = conn.createStatement();
        ResultSet resultset = statement.executeQuery("select
            task_id, target_drtn_hr_cnt, task_name from Task
            where proj_id = '" + i + "'");
        if(resultset != null)
            try
            {
                for(int j = 0; resultset.next(); j++)
                {
                    int k = resultset.getInt(1);
                    int l = resultset.getInt(2);
                    String s = resultset.getString(3);
                    int i1 = l;
                    String s1 = "a" + (new Integer(j +
                        1)).toString();
                    vector.add(j, new Activity(k, s1, " ", s, l, 0,
                        i1, 1, 1, 1, 1.0D, 1,
                        0.20000000000000001D, 0.0D, 0.5D, 1, k, 0,
                        0, 0));
                }
            }
        oracledatabaseconnection.disconnect();
    }
}

```

```

        }
        catch(Exception _ex)
        {
        }
        statement.close();
        conn.close();
    }
    catch(Exception exception)
    {
        exception.printStackTrace();
    }
    return vector;
}
public Vector getPrimaveraProject()
    throws RemoteException
{
    Vector vector = new Vector();
    try
    {
        Statement statement = conn.createStatement();
        ResultSet resultset = statement.executeQuery("select
            proj_short_name, proj_id from Project");
        if(resultset != null)
            try
            {
                String s;
                int i;
                for(; resultset.next(); vector.add(new Project(i,
                    s, 0, 0, 0)))
                {
                    s = resultset.getString(1);
                    i = resultset.getInt(2);
                }
            }
            catch(Exception _ex)
            {
            }
        statement.close();
    }
}

```

```

        conn.close();
    }
    catch(Exception exception)
    {
        exception.printStackTrace();
    }
    return vector;
}
public Vector getPrimaveraRelationship(int i)
{
    Vector vector = new Vector();
    try
    {
        Statement statement = conn.createStatement();
        ResultSet resultset = statement.executeQuery("select
            task_id, pred_task_id, lag_hr_cnt, pred_type from
            Taskpred where proj_id = '" + i + "'");
        if(resultset != null)
            try
            {
                for(int j = 1; resultset.next(); j++)
                {
                    int k = resultset.getInt(1);
                    int l = resultset.getInt(2);
                    int il = resultset.getInt(3);
                    String s = null;
                    if(resultset.getString(4).equals("PR_FS"))
                        s = "FS";
                    else
                    if(resultset.getString(4).equals("PR_FF"))
                        s = "FF";
                    else
                    if(resultset.getString(4).equals("PR_SS"))
                        s = "SS";
                    else
                    if(resultset.getString(4).equals("PR_RW"))
                        s = "RW";
                    int j1 = 1;

```

```

        int k1 = findDPMId(k);
        int l1 = findDPMId(l);
        String s1 = findPredCode(l);
        vector.add(new Precedence(k1, l1, s, i1, j1,
            s1));
    }
}
catch(Exception exception)
{
    exception.printStackTrace();
}
statement.close();
conn.close();
}
catch(Exception _ex)
{
}
return vector;
}
private Properties props;
private Driver myDriver;
private Connection conn;
private DatabaseEventDispatcherImpl ded;
private ServerImpl server;
}

```

```

package gpms.interfaces.impl;

import java.sql.*;
import java.util.Properties;
import java.util.*;
import java.rmi.*;
import gpms.util.*;

public class PrimaveraDBImpl {
    private Properties props;
    private Driver myDriver;
    private Connection conn;
public PrimaveraDBImpl() throws RemoteException {
    super();
    props = new Properties();
    props.put("user", "sa");
    props.put("password", "drim");
    props.put("db", "pmdb");
    props.put("server", "18.58.2.76:1433");
    try {
        myDriver=(Driver)Class.forName("weblogic.jdbc.mssqlserver4.Dri
ver").newInstance();
        conn = myDriver.connect("jdbc:weblogic:mssqlserver4", props);
    } catch(Exception se) {
    }
public void closeConnection() throws RemoteException {
    try{
        conn.close();
    } catch(Exception e) {
        e.printStackTrace();
    }
}
public Vector getPrimaveraActivity(int projectId) throws
RemoteException{
    Vector vectorTwo = new Vector();
    try {
        Statement statement = conn.createStatement();
        ResultSet rset = statement.executeQuery("SELECT ID, CODE,

```

```

NAME, DURATION, ST, FT, DMOSTLIKELY, DOPTIMISTIC,
DPESSIMISTIC, RELIABILITY, COMPLEXITY, CONTINGENCY,
BUFFERING, INTERNALSENSITIVITY, ISCRITICALACTIVITY FROM
ACTIVITY WHERE PROJID = '"' + projectId + '"');
    if(rset!=null)
    {
        try {
            while(rset.next()) {
                int id = rset.getInt(1);
                String code = rset.getString(2);
                String name = rset.getString(3);
                int duration = rset.getInt(4);
                int st = rset.getInt(5);
                int ft = rset.getInt(6);
                int durationMostLikely =
                    rset.getInt(7);
                int durationOptimistic =
                    rset.getInt(8);
                int durationPessimistic =
                    rset.getInt(9);
                double reliability =
                    rset.getDouble(10);
                int complexity = rset.getInt(11);
                double contingency =
                    rset.getDouble(12);
                double buffering = rset.getDouble(13);
                double internalSens =
                    rset.getDouble(14);
                int isCriticalActivity =
                    rset.getInt(15);
                vectorTwo.add(new Activity(id, code,
                    name,
                    duration, st, ft, durationMostLikely,
                    durationOptimistic,
                    durationPessimistic, reliability,
                    complexity, contingency, buffering,
                    internalSens, isCriticalActivity));
            }
        }
    }

```

```

        } catch(Exception se) {
        }
    }
    statement.close();
} catch (Exception e) {
    e.printStackTrace();
}
return vectorTwo;
}

public Vector getPrimaveraProject() throws RemoteException {
    Vector vectorOne = new Vector();
    try {
        Statement statement = conn.createStatement();
        ResultSet result = statement.executeQuery("select
proj_name, proj_id from Project");
        if(result!=null)
        {
            try {
                while(result.next()) {
                    String name = result.getString(1);
                    int id = result.getInt(2);
                    vectorOne.add(new Project(id, name));
                }
            } catch(Exception se) {
            }
        }
        statement.close();
        conn.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return vectorOne;
}
}
}

```

```

package gpms.interfaces.impl;

import gpms.client.gui.GameData;
import gpms.client.output.SimulationData;
import gpms.database.OracleDatabaseConnection;
import gpms.interfaces.rmi.ClientEventHandler;
import gpms.interfaces.rmi.Server;
import gpms.util.*;
import java.io.PrintStream;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Enumeration;
import java.util.Vector;

public class ServerImpl extends UnicastRemoteObject
    implements Server
{
    public ServerImpl(DatabaseInitializerImpl
        databaseinitializerimpl, DatabaseEventDispatcherImpl
        databaseeventdispatcherimpl)
        throws RemoteException
    {
        tempduration = 800;
        timeValue = null;
        resultValue = null;
        timeValue01 = null;
        resultValue01 = null;
        timeValue02 = null;
        resultValue02 = null;
        timeValue1 = null;
        resultValue1 = null;
        timeValue2 = null;
        resultValue2 = null;
        timeValue3 = null;
        resultValue3 = null;
        resultValueCPM = null;
    }
}

```



```

    resultValueWA = null;
    resultValueCPM1 = null;
    resultValueWA1 = null;
    resultValueCPM2 = null;
    resultValueWA2 = null;
    timeValueCPM = null;
    timeValueWA = null;
    alreadySimTime = null;
    alreadySimData = null;
    resultValue03 = null;
    resultValue4 = null;
    resultValueCPM3 = null;
    resultValuePERT = null;
    resultValuePERT1 = null;
    resultValuePERT2 = null;
    resultValuePERT3 = null;
    resultValueWA3 = null;
    timeValue03 = null;
    timeValue4 = null;
    timeValuePERT = null;
    databaseInitializer = databaseinitializerimpl;
    ded = databaseeventdispatcherimpl;
    vector = new Vector();
    vectorOne = new Vector();
    clientVector = new Vector();
    vensim = new Vensim();
    result = vensim.command("SPECIAL>LOADMODEL|dpm.vmf");
}

public synchronized void attach(ClientEventHandler
    clienteventhandler) throws RemoteException
{
    if(!clientVector.contains(clienteventhandler))
    {
        clientVector.add(clienteventhandler);
    }
}

public void createProject(String s, String s1, int i, int j, int
    k) throws RemoteException

```

```

{
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    oracledatabaseconnection.executeSQL("INSERT INTO DPM_PROJECT
        VALUES('" + s + "','" + s1 + "','" + i + "','" + j +
            "','" + k + "','" + l + "')");
    oracledatabaseconnection.executeSQL("INSERT INTO
        DPM_PROJECTOUTPUT VALUES('" + s + "', 0,0,0,0,0,0,0,1)");
    oracledatabaseconnection.disconnect();
}
public void createRelationship(int i, int j, String s, int k, int
    l, String s1) throws RemoteException
{
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    oracledatabaseconnection.executeSQL("INSERT INTO
        DPM_ACTIVITYRELATION VALUES('" + i + "','" + j + "','" +
        s + "','" + k + "','" + l + "')");
    oracledatabaseconnection.disconnect();
}
public synchronized void detach(ClientEventHandler
    clienteventhandler) throws RemoteException
{
    for(Enumeration enumeration = vector.elements();
        enumeration.hasMoreElements();)
    {
        ClientEventHandler clienteventhandler1 =
            (ClientEventHandler)enumeration.nextElement();
        if(clienteventhandler.equals(clienteventhandler1))
            clientVector.remove(clienteventhandler1);
    }
}
public synchronized void dispatch(String s)
    throws RemoteException
{
    int i = 0;
    try
    {

```

```

        for(i = 0; i < clientVector.size(); i++)
        {
            ((ClientEventHandler)clientVector.elementAt(i)).handle(s);
        }
    }
    catch(Exception _ex)
    {
        clientVector.remove(i);
        dispatch(s);
    }
}

public String findPredCode(int i)
{
    String s = "a1";
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeQuery("SELECT CODE FROM
        DPM_ACTIVITY WHERE ID = '" + i + "'");
    if(resultset != null)
        try
        {
            while(resultset.next())
                s = resultset.getString(1);
        }
        catch(Exception _ex)
        {
        }
    oracledatabaseconnection.disconnect();
    return s;
}

public void gameSimulation(GameData gamedata, SimulationData
    simulationdata, int i, boolean flag) throws RemoteException
{
    float af[] = new float[801];
    float af1[] = new float[801];
    if(i == 1)
    {

```

```

result = vensim.command("SIMULATE>RUNNAME|game");
Vector vector1 = new Vector();
int l = simulationdata.getProjectId();
try
{
    vector1 = getActivity(l, 1);
}
catch(Exception _ex)
{
}
int j1 = vector1.size();
if(j1 > 0)
{
    vensimCommand =
        "SIMULATE>SETVAL|WillingnessToControlHeadCount=" +
        simulationdata.getHc();
    result = vensim.command(vensimCommand);
    vensimCommand =
        "SIMULATE>SETVAL|WillingnessToAdoptOvertime=" +
        simulationdata.getOt();
    result = vensim.command(vensimCommand);
    vensimCommand =
        "SIMULATE>SETVAL|DurationDividerforRFIReplyTime=" +
        simulationdata.getRfi();
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|ExperienceLevel=" +
        gamedata.getExperience();
    result = vensim.command(vensimCommand);
    vensimCommand =
        "SIMULATE>SETVAL|TimetoIncreaseWorkforce=" +
        simulationdata.getIncrease();
    result = vensim.command(vensimCommand);
    vensimCommand =
        "SIMULATE>SETVAL|ReliabilityBufferingActivated=" +
        simulationdata.getReliability();
    result = vensim.command(vensimCommand);
    vensimCommand =

```

```

        "SIMULATE>SETVAL|SecondBufferActivated=" +
        simulationdata.getReliability();
result = vensim.command(vensimCommand);
vensimCommand =
    "SIMULATE>SETVAL|FractionOfBufferingPJ=" +
    simulationdata.getBuffer();
result = vensim.command(vensimCommand);
vensimCommand =
    "SIMULATE>SETVAL|KnownContingencyFactorPJ=" +
    simulationdata.getContingency();
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|WorkHoursPerDay=" +
    gamedata.getWorkhours();
result = vensim.command(vensimCommand);
}
for(int l1 = 0; l1 < j1; l1++)
{
    int l2 = ((Activity)vector1.get(l1)).getId();
    int i3 = ((Activity)vector1.get(l1)).getST();
    String s1 = ((Activity)vector1.get(l1)).getCode();
    int j3 = ((Activity)vector1.get(l1)).getComplexity();
    double d =
        ((Activity)vector1.get(l1)).getReliability();
    double d1 =
        ((Activity)vector1.get(l1)).getStability();
    int k3 = ((Activity)vector1.get(l1)).getDuration();
    double d2 =
        ((Activity)vector1.get(l1)).getContingency();
    double d3 =
        ((Activity)vector1.get(l1)).getBuffering();
    double d4 =
        ((Activity)vector1.get(l1)).getInternalSensitivity();
    int l3 =
        ((Activity)vector1.get(l1)).getIsCriticalActivity();
    vensimCommand = "SIMULATE>SETVAL|InitialStartTime[" +
        s1 + "]" + i3;
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|FastEvolution[" + s1

```

```

    + "]" + j3;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|GivenReliability[" +
    s1 + "]" + d;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|GivenStability[" + s1
    + "]" + d1;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|OriginalDuration[" +
    s1 + "]" + k3;
result = vensim.command(vensimCommand);
vensimCommand =
    "SIMULATE>SETVAL|KnownContingencyFactor[" + s1 +
    "]" + d2;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|FractionOfBuffering["
    + s1 + "]" + d3;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|InternalSensitivity["
    + s1 + "]" + d4;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|IsItCritical[" + s1 +
    "]" + l3;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|QMFamiliarity[" + s1
    + "]" + simulationdata.getQMth();
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|SMFamiliarity[" + s1
    + "]" + simulationdata.getSMth();
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|DividerTimeforQM[" +
    s1 + "]" + simulationdata.getQMPeriod();
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|TimeForCCBDecision["
    + s1 + "]" + simulationdata.getCCB();
result = vensim.command(vensimCommand);
Vector vector3 = new Vector();
try

```

```

{
    vector3 = databaseInitialzer.getPredActivity(l2);
}
catch(Exception _ex)
{
}
for(int i4 = 0; i4 < vector3.size(); i4++)
{
    String s2 =
        ((Precedence)vector3.get(i4)).getRelation();
    String s3 =
        ((Precedence)vector3.get(i4)).getPredActivityC
ode
        ();
    int j4 = ((Precedence)vector3.get(i4)).getLag();
    int k4 = ((Precedence)vector3.get(i4)).getSens();
    if(s2.equals("FS"))
    {
        vensimCommand =
            "SIMULATE>SETVAL|PrecedenceRelationship["
            + s1 + ", " + s3 + "]= " + j4;
        result = vensim.command(vensimCommand);
    } else
    if(s2.equals("SS"))
    {
        j4 += 1000;
        vensimCommand =
            "SIMULATE>SETVAL|PrecedenceRelationship["
            + s1 + ", " + s3 + "]= " + j4;
        result = vensim.command(vensimCommand);
    } else
    if(s2.equals("FF"))
    {
        j4 += 2000;
        vensimCommand =
            "SIMULATE>SETVAL|PrecedenceRelationship["
            + s1 + ", " + s3 + "]= " + j4;
        result = vensim.command(vensimCommand);
    }
}

```

```

    } else
    if(s2.equals("RW"))
    {
        vensimCommand =
            "SIMULATE>SETVAL|ReprocessIterationRelati
            onship[" + s1 + "," + s3 + "]=1";
        result = vensim.command(vensimCommand);
    }
    vensimCommand =
        "SIMULATE>SETVAL|ExternalSensitivity[" + s1 +
        "," + s3 + "]= " + k4;
    result = vensim.command(vensimCommand);
    }
}
}
if(i == 2)
{
    Vector vector2 = new Vector();
    int i1 = simulationdata.getProjectId();
    try
    {
        vector2 = databaseInitializer.getActivity(i1);
    }
    catch(Exception _ex)
    {
    }
    int k1 = vector2.size();
    if(k1 > 0)
    {
        vensimCommand =
            "SIMULATE>SETVAL|gWillingnessToControlHeadcount=" +
            gamedata.getHeadcount() + "&SPECIAL>RESETINPUT";
        result = vensim.command(vensimCommand);
        vensimCommand =
            "SIMULATE>SETVAL|gWillingnessToAdoptOvertime=" +
            gamedata.getOvertime() + "&SPECIAL>RESETINPUT";
        result = vensim.command(vensimCommand);
        vensimCommand =

```



```

        "SIMULATE>SETVAL|gFractionOfBufferingPJ=" +
        gamedata.getBuffer() + "&SPECIAL>RESETINPUT";
result = vensim.command(vensimCommand);
vensimCommand =
    "SIMULATE>SETVAL|gKnownContingencyFactorPJ=" + gamed
ata.getContingency() + "&SPECIAL>RESETINPUT";
result = vensim.command(vensimCommand);
vensimCommand =
    "SIMULATE>SETVAL|gReliabilityBufferingActivated=" +
    gamedata.getReliAct() + "&SPECIAL>RESETINPUT";
result = vensim.command(vensimCommand);
vensimCommand =
    "SIMULATE>SETVAL|gSecondBufferActivated=" +
    gamedata.getReliAct() + "&SPECIAL>RESETINPUT";
result = vensim.command(vensimCommand);
vensimCommand =
    "SIMULATE>SETVAL|gDurationDividerforRFIReplyTime=" +
    gamedata.getRFI() + "&SPECIAL>RESETINPUT";
result = vensim.command(vensimCommand);
vensimCommand =
    "SIMULATE>SETVAL|gTimeToIncreaseWorkforce=" +
    gamedata.getTimeToIncrease() +
    "&SPECIAL>RESETINPUT";
result = vensim.command(vensimCommand);
vensimCommand =
    "SIMULATE>SETVAL|gTimeToDecreaseWorkforce=" +
    gamedata.getTimeToIncrease() +
    "&SPECIAL>RESETINPUT";
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|gExperienceLevel=" +
    gamedata.getExperience() + "&SPECIAL>RESETINPUT";
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|gWorkHoursPerDay=" +
    gamedata.getWorkhours() + "&SPECIAL>RESETINPUT";
result = vensim.command(vensimCommand);
    }
for(int i2 = 0; i2 < k1; i2++)
{

```

```

        ((Activity)vector2.get(i2)).getId();
        ((Activity)vector2.get(i2)).getST();
        String s = ((Activity)vector2.get(i2)).getCode();
        ((Activity)vector2.get(i2)).getComplexity();
        ((Activity)vector2.get(i2)).getReliability();
        ((Activity)vector2.get(i2)).getDuration();
        ((Activity)vector2.get(i2)).getContingency();
        ((Activity)vector2.get(i2)).getBuffering();
        ((Activity)vector2.get(i2)).getInternalSensitiviy();
        ((Activity)vector2.get(i2)).getIsCriticalActivity();
        vensimCommand = "SIMULATE>SETVAL|gQMFamiliarity[" + s
            + "]" + simulationdata.getQMth() +
            "&SPECIAL>RESETINPUT";
        result = vensim.command(vensimCommand);
        vensimCommand = "SIMULATE>SETVAL|gSMFamiliarity[" + s
            + "]" + simulationdata.getSMth() +
            "&SPECIAL>RESETINPUT";
        result = vensim.command(vensimCommand);
        vensimCommand = "SIMULATE>SETVAL|gDividerTimeforQM[" +
            s + "]" + gamedata.getQMPeriod() +
            "&SPECIAL>RESETINPUT";
        result = vensim.command(vensimCommand);
        vensimCommand = "SIMULATE>SETVAL|gTimeForCCBDecision["
            + s + "]" + gamedata.getCCBPeriod() +
            "&SPECIAL>RESETINPUT";
        result = vensim.command(vensimCommand);
        vensimCommand = "SIMULATE>SETVAL|Units_for_Time=Week";
        result = vensim.command(vensimCommand);
        vensimCommand = "SIMULATE>SETVAL|Units for Time=Week";
        result = vensim.command(vensimCommand);
    }
}
if(i == 1)
{
    result = vensim.command("MENU>GAME|O");
    if(result == 0)
    int j = getGameInterval();
    result = vensim.command("GAME>GAMEINTERVAL|" + j);
}

```

```

result = vensim.command("GAME>GAMEON");
vensim.get_data("game.vdf", "ProjectFinishTime", "time",
    af, af1, j + 1);
setFinalTime(af[j]);
vensim.get_data("game.vdf", "ProjectProgress", "time", af,
    af1, j + 1);
float af4[] = new float[j];
float af2[] = new float[j];
for(int j2 = 0; j2 < j; j2++)
{
    af4[j2] = 100F * af[j2];
    af2[j2] = af1[j2];
}
setTimeForGraph(af2);
setDataForGraph(af4);
}
if(i == 2)
{
    result = vensim.command("GAME>GAMEINTERVAL|" +
        getGameInterval());
    result = vensim.command("SPECIAL>RESETINPUT");
    result = vensim.command("GAME>GAMEON");
    int k = gamedata.getSumInterval();
    vensim.get_data("game.vdf", "ProjectFinishTime", "time",
        af, af1, k + 1);
    setFinalTime(af[k]);
    vensim.get_data("game.vdf", "ProjectProgress", "time", af,
        af1, k + 1);
    float _tmp = af[k] * 100F;
    float af5[] = new float[k];
    float af3[] = new float[k];
    for(int k2 = 0; k2 < k; k2++)
    {
        af5[k2] = 100F * af[k2];
        af3[k2] = af1[k2];
    }
    setTimeForGraph(af3);
    setDataForGraph(af5);
}

```

```

    }
    if(i == 3)
        result = vensim.command("GAME>ENDGAME");
}
public float getActivityFinalTime()
{
    return activityFinalTime;
}
public float getActivityFinalTimeCPM()
{
    return activityFinalTime1;
}

public float getActivityFinalTimeWA()
{
    return activityFinalTime2;
}
public float getActivityStartTime()
{
    return activityStartTime;
}
public float getActivityStartTimeCPM()
{
    return activityStartTime1;
}
public float getActivityStartTimeWA()
{
    return activityStartTime2;
}
public float[] getDataForGraph()
{
    return resultValue;
}
public float[] getDataForGraph2()
{
    return resultValue01;
}
public float[] getDataForGraph3()

```

```

    {
        return resultValue02;
    }
public float[] getDataForGraphCPM()
{
    return resultValueCPM;
}
public float[] getDataForGraphCPM1()
{
    return resultValueCPM1;
}
public float[] getDataForGraphCPM2()
{
    return resultValueCPM2;
}
public float[] getDataForGraphProductivity()
{
    return resultValue2;
}
public float[] getDataForGraphProgress()
{
    return resultValue1;
}
public float[] getDataForGraphQuality()
{
    return resultValue3;
}
public float[] getDataForGraphWA()
{
    return resultValueWA;
}
public float[] getDataForGraphWA1()
{
    return resultValueWA1;
}
public float[] getDataForGraphWA2()
{
    return resultValueWA2;
}

```

```

}
public float getFinalTime()
{
    return finalTime;
}
public float getFinalTime2()
{
    return finalTime01;
}
public float getFinalTime3()
{
    return finalTime02;
}
public int getFinishTimeActivity(String s)
{
    float af[] = new float[401];
    float afl[] = new float[401];
    vensim.get_data("dpm.vdf", "ActivityFinishTime[" + s + "]",
        "time", af, afl, 401);
    int i = (int)af[401];
    return i;
}
public int getGameInterval()
    throws RemoteException
{
    return interval;
}
public Vector getPrimaveraActivity(int i)
    throws RemoteException
{
    Vector vector1 = new Vector();
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeQuery("SELECT TASK_ID,
        TARGET_DRTN_HR_CNT, TASK_NAME FROM TASK WHERE PROJ_ID =
        '" + i + "'");
    if(resultset != null)

```

```

try
{
    for(int j = 0; resultset.next(); j++)
    {
        int k = resultset.getInt(1);
        int l = resultset.getInt(2);
        String s = resultset.getString(3);
        int i1 = l;
        String s1 = "a" + (new Integer(j + 1)).toString();
        String s2 = "PR-" + s1;
        vector1.add(j, new Activity(k, s1, s2, s, l, 0, i1,
            1, 1, 1, 1.0D, 1.0D, 1, 0.20000000000000001D,
            0 .0D, 0.5D, 1, k, 0, 0, 0, 1));
    }
}
catch(Exception _ex)
{
}

oracledatabaseconnection.disconnect();
return vector1;
}

public Vector getPrimaveraProjectList()
throws RemoteException
{
    Vector vector1 = new Vector();
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeQuery("SELECT PROJ_ID, P
        ROJ_SHORT_NAME FROM PROJECT");
    if(resultset != null)
        try
        {
            int i;
            String s;
            int j;
            int k;
            int l;

```

```

        for(; resultset.next(); vector1.add(new Project(i, s,
            j, k, l)))
        {
            i = resultset.getInt(1);
            s = resultset.getString(2);
            j = 0;
            k = 0;
            l = 0;
        }
    }
    catch(SQLException _ex)
    {
    }
    oracledatabaseconnection.disconnect();
    return vector1;
}

```

```

public Vector getPrimaveraRelationship(int i)
    throws RemoteException
{
    Vector vector1 = new Vector();
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeQuery("SELECT TASK_ID,
        PRED_TASK_ID, LAG_HR_CNT, PRED_TYPE FROM TASKPRED WHERE
        PROJ_ID = '" + i + "'");
    try
    {
        if(resultset != null)
            try
            {
                for(int j = 1; resultset.next(); j++)
                {
                    int k = resultset.getInt(1);
                    int l = resultset.getInt(2);
                    int il = resultset.getInt(3);
                    String s = null;

```



```

        if(resultset.getString(4).equals("PR_FS"))
            s = "FS";
        else
            if(resultset.getString(4).equals("PR_FF"))
                s = "FF";
            else
                if(resultset.getString(4).equals("PR_SS"))
                    s = "SS";
                else
                    if(resultset.getString(4).equals("PR_RW"))
                        s = "RW";
                    int j1 = 1;
                    int k1 = findDPMId(k);
                    int l1 = findDPMId(l);
                    String s1 = findPredCode(l);
                    vector1.add(new Precedence(k1, l1, s, i1, j1,
                        s1));
                }
            }
        catch(Exception exception)
        {
            exception.printStackTrace();
        }
    }
    catch(Exception _ex)
    {
    }
    return vector1;
}

public Vector getProjectList()
    throws RemoteException
{
    Vector vector1 = new Vector();
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeQuery("SELECT PROJID, P
        ROJNAME, CPM, PERT, DPMWOACTION FROM DPM_PROJECT");

```

```

if(resultset != null)
    try
    {
        int i;
        String s;
        int j;
        int k;
        int l;
        for(; resultset.next(); vector1.add(new Project(i, s,
            j, k, l)))
        {
            i = resultset.getInt(1);
            s = resultset.getString(2);
            j = resultset.getInt(3);
            k = resultset.getInt(4);
            l = resultset.getInt(5);
        }
    }
    catch(SQLException _ex)
    {
    }
    oracledatabaseconnection.disconnect();
    return vector1;
}
public float[] getSimFile_data()
{
    return alreadySimData;
}
public float getSimFile_finalltime()
{
    return alreadySimFinalTime;
}
public float[] getSimFile_time()
{
    return alreadySimTime;
}
public int getStartTimeActivity(String s)
{

```

```

float af[] = new float[401];
float afl[] = new float[401];
int i = vensim.get_data("dpm.vdf", "ProjectFinishTime",
    "time", af, afl, 401);
i = (int)af[400];
return i;
}
public float[] getTimeForGraph()
{
    return timeValue;
}
public float[] getTimeForGraph2()
{
    return timeValue01;
}
public float[] getTimeForGraph3()
{
    return timeValue02;
}
public float[] getTimeForGraphCPM()
{
    return timeValueCPM;
}
public float[] getTimeForGraphProductivity()
{
    return timeValue2;
}
public float[] getTimeForGraphProgress()
{
    return timeValue1;
}
public float[] getTimeForGraphQuality()
{
    return timeValue3;
}
public float[] getTimeForGraphWA()
{
    return timeValueWA;
}

```

```

}
public void monteCarlo()
    throws RemoteException
{
}
public void policyMaker()
    throws RemoteException
{
}
public void setActivityFinalTime(float f)
{
    activityFinalTime = f;
}
public void setActivityFinalTimeCPM(float f)
{
    activityFinalTime1 = f;
}
public void setActivityFinalTimeWA(float f)
{
    activityFinalTime2 = f;
}
public void setActivityProductivity(String s)
{
    int i = (int)getActivityStartTime();
    int j = (int)getActivityFinalTime();
    float af[] = new float[801];
    float af1[] = new float[801];
    vensim.get_data("dpm.vdf", "WorkForceUtilizationRatio-
        Activity[" + s + "]", "time", af, af1, 801);
    int k = (j - i) + 1;
    float af2[] = new float[k];
    float af3[] = new float[k];
    int l = 0;
    for(int il = 0; il < af1.length; il++)
    {
        if(af1[il] < (float)i)
            continue;
        af3[l] = af1[il];
    }
}

```

```

        af2[1] = af[i1] * 100F;
        if(++i1 == k)
            break;
    }
    setTimeForGraphProductivity(af3);
    setDataForGraphProductivity(af2);
    int j1 = (int) getActivityStartTimeCPM();
    int k1 = (int) getActivityFinalTimeCPM();
    int l1 = (k1 - j1) + 1;
    vensim.get_data("cpm.vdf", "WorkForceUtilizationRatio-
        Activity[" + s + "]", "time", af, af1, 801);
    float af4[] = new float[l1];
    float af5[] = new float[l1];
    int i2 = 0;
    for(int j2 = 0; j2 < af1.length; j2++)
    {
        if(af1[j2] < (float)j1)
            continue;
        af5[i2] = af1[j2];
        af4[i2] = af[j2] * 100F;
        if(++i2 == l1)
            break;
    }
    setDataForGraphCPM1(af4);
    int k2 = (int) getActivityStartTimePERT();
    int l2 = (int) getActivityFinalTimePERT();
    int i3 = (l2 - k2) + 1;
    vensim.get_data("pert.vdf", "WorkForceUtilizationRatio-
        Activity[" + s + "]", "time", af, af1, 801);
    float af6[] = new float[i3];
    float af7[] = new float[i3];
    int j3 = 0;
    for(int k3 = 0; k3 < af1.length; k3++)
    {
        if(af1[k3] < (float)k2)
            continue;
        af7[j3] = af1[k3];
        af6[j3] = af[k3] * 100F;
    }

```

```

        if(++j3 == i3)
            break;
    }
    setDataForGraphPERT1(af6);
    int l3 = (int) getActivityStartTimeWA();
    int i4 = (int) getActivityFinalTimeWA();
    int j4 = i4 - l3;
    vensim.get_data("dpmWoAction.vdf", "WorkForceUtilizationRatio-
    Activity[" + s + "]", "time", af, afl, 801);
    float af8[] = new float[j4];
    float af9[] = new float[j4];
    int k4 = 0;
    for(int l4 = 0; l4 < afl.length; l4++)
    {
        if(afl[l4] < (float)l3)
            continue;
        af9[k4] = afl[l4];
        af8[k4] = af[l4] * 100F;
        if(++k4 == j4)
            break;
    }
    setDataForGraphWA1(af8);
}
public void setActivityProgress(String s)
{
    float af[] = new float[801];
    float afl[] = new float[801];
    vensim.get_data("dpm.vdf", "ActualActivityStartTime[" + s +
        "]", "time", af, afl, 801);
    int i = (int)af[800];
    setActivityStartTime(i);
    vensim.get_data("dpm.vdf", "ActivityFinishTime[" + s + "]",
        "time", af, afl, 801);
    int j = (int)af[800];
    setActivityFinalTime(j);
    vensim.get_data("dpm.vdf", "FractionOfWorkReleased[" + s +
        "]", "time", af, afl, 801);
    int k = (j - i) + 1;

```

```

float af2[] = new float[k];
float af3[] = new float[k];
int l = 0;
for(int i1 = 0; i1 < af1.length; i1++)
{
    if(af1[i1] < (float)i)
        continue;
    af3[l] = af1[i1];
    af2[l] = af[i1] * 100F;
    if(++l == k)
        break;
}
setTimeForGraphProgress(af3);
setDataForGraphProgress(af2);
vensim.get_data("cpm.vdf", "ActualActivityStartTime[" + s +
    "]", "time", af, af1, 801);
int j1 = (int)af[800];
setActivityStartTimeCPM(j1);
vensim.get_data("cpm.vdf", "ActivityFinishTime[" + s + "]",
    "time", af, af1, 801);
int k1 = (int)af[800];
setActivityFinalTimeCPM(k1);
int l1 = (k1 - j1) + 1;
vensim.get_data("cpm.vdf", "FractionOfWorkReleased[" + s +
    "]", "time", af, af1, 801);
float af4[] = new float[l1];
float af5[] = new float[l1];
int i2 = 0;
for(int j2 = 0; j2 < af1.length; j2++)
{
    if(af1[j2] < (float)j1)
        continue;
    af5[i2] = af1[j2];
    af4[i2] = af[j2] * 100F;
    if(++i2 == l1)
        break;
}
setTimeForGraphCPM(af5);

```

```

setDataForGraphCPM(af4);
vensim.get_data("pert.vdf", "ActualActivityStartTime[" + s +
    "]", "time", af, af1, 801);
int k2 = (int)af[800];
setActivityStartTimePERT(k2);
vensim.get_data("pert.vdf", "ActivityFinishTime[" + s + "]",
    "time", af, af1, 801);
int l2 = (int)af[800];
setActivityFinalTimePERT(l2);
int i3 = (l2 - k2) + 1;
vensim.get_data("pert.vdf", "FractionOfWorkReleased[" + s +
    "]", "time", af, af1, 801);
float af6[] = new float[i3];
float af7[] = new float[i3];
int j3 = 0;
for(int k3 = 0; k3 < af1.length; k3++)
{
    if(af1[k3] < (float)k2)
        continue;
    af7[j3] = af1[k3];
    af6[j3] = af[k3] * 100F;
    if(++j3 == i3)
        break;
}
setTimeForGraphPERT(af7);
setDataForGraphPERT(af6);
vensim.get_data("dpmWoAction.vdf", "ActualActivityStartTime["
    + s + "]", "time", af, af1, 801);
int l3 = (int)af[800];
setActivityStartTimeWA(l3);
vensim.get_data("dpmWoAction.vdf", "ActivityFinishTime[" + s
    + "]", "time", af, af1, 801);
int i4 = (int)af[800];
setActivityFinalTimeWA(i4);
vensim.get_data("dpmWoAction.vdf", "FractionOfWorkReleased["
    + s + "]", "time", af, af1, 801);
int j4 = (i4 - l3) + 1;
float af8[] = new float[j4];

```



```

float af9[] = new float[j4];
int k4 = 0;
for(int l4 = 0; l4 < af1.length; l4++)
{
    if(af1[l4] < (float)l3)
        continue;
    af9[k4] = af1[l4];
    af8[k4] = af[l4] * 100F;
    if(++k4 == j4)
        break;
}
setTimeForGraphWA(af9);
setDataForGraphWA(af8);
}
public void setActivityQuality(String s)
{
    int i = (int) getActivityStartTime();
    int j = (int) getActivityFinalTime();
    float af[] = new float[801];
    float af1[] = new float[801];
    vensim.get_data("dpm.vdf", "ActualReliability[" + s + "]",
        "time", af, af1, 801);
    int k = (j - i) + 1;
    float af2[] = new float[k];
    float af3[] = new float[k];
    int l = 0;
    for(int il = 0; il < af1.length; il++)
    {
        if(af1[il] < (float)i)
            continue;
        af3[l] = af1[il];
        af2[l] = af[il] * 100F;
        if(++l == k)
            break;
    }
    setTimeForGraphQuality(af3);
    setDataForGraphQuality(af2);
    int j1 = (int) getActivityStartTimeCPM();

```

```

int k1 = (int) getActivityFinalTimeCPM();
int l1 = (k1 - j1) + 1;
vensim.get_data("cpm.vdf", "ActualReliability[" + s + "]",
    "time", af, af1, 801);
float af4[] = new float[l1];
int i2 = 0;
for(int j2 = 0; j2 < af1.length; j2++)
{
    if(af1[j2] < (float)j1)
        continue;
    af4[i2] = af[j2] * 100F;
    if(++i2 == l1)
        break;
}
setDataForGraphCPM2(af4);
int k2 = (int) getActivityStartTimePERT();
int l2 = (int) getActivityFinalTimePERT();
int i3 = (l2 - k2) + 1;
vensim.get_data("pert.vdf", "ActualReliability[" + s + "]",
    "time", af, af1, 801);
float af5[] = new float[i3];
int j3 = 0;
for(int k3 = 0; k3 < af1.length; k3++)
{
    if(af1[k3] < (float)k2)
        continue;
    af5[j3] = af[k3] * 100F;
    if(++j3 == i3)
        break;
}
setDataForGraphPERT2(af5);
int l3 = (int) getActivityStartTimeWA();
int i4 = (int) getActivityFinalTimeWA();
int j4 = (i4 - l3) + 1;
vensim.get_data("dpmWoAction.vdf", "ActualReliability[" + s +
    "]", "time", af, af1, 801);
float af6[] = new float[j4];
int k4 = 0;

```

```

for(int l4 = 0; l4 < af1.length; l4++)
{
    if(af1[l4] < (float)l3)
        continue;
    af6[k4] = af[l4] * 100F;
    if(++k4 == j4)
        break;
}
setDataForGraphWA2(af6);
}
public void setActivityStartTime(float f)
{
    activityStartTime = f;
}
public void setActivityStartTimeCPM(float f)
{
    activityStartTime1 = f;
}
public void setActivityStartTimeWA(float f)
{
    activityStartTime2 = f;
}
public void setCpm()
{
    float af[] = new float[801];
    float af1[] = new float[801];
    int i = vensim.get_data("cpm.vdf", "ProjectFinishTime",
        "time", af1, af, 801);
    i = (int)af1[800];
    float af2[] = new float[801];
    float af3[] = new float[801];
    vensim.get_data("cpm.vdf", "ProjectProgress", "time", af3,
        af2, 801);
    int j = 0;
    for(int k = 0; k < af2.length; k++)
    {
        if(af2[k] != (float)i)
            continue;

```

```

        j = k;
        break;
    }
    float af4[] = new float[j];
    float af5[] = new float[j];
    for(int l = 0; l < j; l++)
    {
        af5[l] = af2[l];
        af4[l] = 100F * af3[l];
    }
    setTimeForGraph2(af5);
    setDataForGraph2(af4);
    setFinalTime2(i);
}
public void setDataForGraph(float af[])
{
    resultValue = af;
}
public void setDataForGraph2(float af[])
{
    resultValue01 = af;
}
public void setDataForGraph3(float af[])
{
    resultValue02 = af;
}
public void setDataForGraphCPM(float af[])
{
    resultValueCPM = af;
}
public void setDataForGraphCPM1(float af[])
{
    resultValueCPM1 = af;
}
public void setDataForGraphCPM2(float af[])
{
    resultValueCPM2 = af;
}

```

```

public void setDataForGraphProductivity(float af[])
{
    resultValue2 = af;
}
public void setDataForGraphProgress(float af[])
{
    resultValue1 = af;
}
public void setDataForGraphQuality(float af[])
{
    resultValue3 = af;
}
public void setDataForGraphWA(float af[])
{
    resultValueWA = af;
}
public void setDataForGraphWA1(float af[])
{
    resultValueWA1 = af;
}
public void setDataForGraphWA2(float af[])
{
    resultValueWA2 = af;
}
public void setDpm()
{
    float af[] = new float[801];
    float af1[] = new float[801];
    int i = vensim.get_data("dpm.vdf", "ProjectFinishTime",
        "time", af1, af, 801);
    i = (int)af1[800];
    float af2[] = new float[801];
    float af3[] = new float[801];
    vensim.get_data("dpm.vdf", "ProjectProgress", "time", af3,
        af2, 801);
    int j = 0;
    for(int k = 0; k < af2.length; k++)
    {

```

```

        if (af2[k] != (float)i)
            continue;
        j = k;
        break;
    }
    float af4[] = new float[j];
    float af5[] = new float[j];
    for (int l = 0; l < j; l++)
    {
        af5[l] = af2[l];
        af4[l] = 100F * af3[l];
    }
    setTimeForGraph(af5);
    setDataForGraph(af4);
    setFinalTime(i);
}
public void setDpmWoAction()
{
    float af[] = new float[801];
    float af1[] = new float[801];
    int i = vensim.get_data("dpmWoAction.vdf",
        "ProjectFinishTime", "time", af1, af, 801);
    i = (int)af1[800];
    float af2[] = new float[801];
    float af3[] = new float[801];
    vensim.get_data("dpmWoAction.vdf", "ProjectProgress", "time",
        af3, af2, 801);
    int j = 0;
    for (int k = 0; k < af2.length; k++)
    {
        if (af2[k] != (float)i)
            continue;
        j = k;
        break;
    }
    float af4[] = new float[j];
    float af5[] = new float[j];
    for (int l = 0; l < j; l++)

```

```

    {
        af5[1] = af2[1];
        af4[1] = 100F * af3[1];
    }
    setTimeForGraph3(af5);
    setDataForGraph3(af4);
    setFinalTime3(i);
}
public void setFinalTime(float f)
{
    finalTime = f;
}
public void setFinalTime2(float f)
{
    finalTime01 = f;
}
public void setFinalTime3(float f)
{
    finalTime02 = f;
}
public void setGameInterval(int i)
    throws RemoteException
{
    interval = i;
}
public void setSimFile(String s)
{
    float af[] = new float[801];
    float af1[] = new float[801];
    int i = vensim.get_data(s, "ProjectFinishTime", "time", af1,
        af, 801);
    i = (int)af1[800];
    float af2[] = new float[i];
    float af3[] = new float[i];
    vensim.get_data(s, "ProjectProgress", "time", af1, af, 801);
    for(int j = 0; j < i; j++)
    {
        af3[j] = af[j];
    }
}

```

```

        af2[j] = af1[j] * 100F;
    }
    setSimFile_time(af3);
    setSimFile_data(af2);
    setSimFile_finaltime(i);
}
public void setSimFile_data(float af[])
{
    alreadySimData = af;
}
public void setSimFile_finaltime(float f)
{
    alreadySimFinalTime = f;
}
public void setSimFile_time(float af[])
{
    alreadySimTime = af;
}
public void setTimeForGraph(float af[])
{
    timeValue = af;
}
public void setTimeForGraph2(float af[])
{
    timeValue01 = af;
}
public void setTimeForGraph3(float af[])
{
    timeValue02 = af;
}
public void setTimeForGraphCPM(float af[])
{
    timeValueCPM = af;
}
public void setTimeForGraphProductivity(float af[])
{
    timeValue2 = af;
}

```



```

public void setTimeForGraphProgress(float af[])
{
    timeValue1 = af;
}
public void setTimeForGraphQuality(float af[])
{
    timeValue3 = af;
}
public void setTimeForGraphWA(float af[])
{
    timeValueWA = af;
}
public void simulation(SimulationData simulationdata, int i)
    throws RemoteException
{
    float af[] = new float[801];
    float af1[] = new float[801];
    if(i == 1)
        result = vensim.command("SIMULATE>RUNNAME|cpm");
    else
    if(i == 2)
        result = vensim.command("SIMULATE>RUNNAME|pert");
    else
    if(i == 3)
        result = vensim.command("SIMULATE>RUNNAME|dpmWoAction");
    else
    if(i == 4)
        result = vensim.command("SIMULATE>RUNNAME|dpm");
    Vector vector1 = new Vector();
    int j = simulationdata.getProjectId();
    int k = simulationdata.getVersionID();
    try
    {
        vector1 = getActivity(j, k);
    }
    catch(Exception _ex)
    {
    }
}

```

```

int l = vector1.size();
new OracleDatabaseConnection();
if(l > 0)
{
    vensimCommand =
        "SIMULATE>SETVAL|WillingnessToControlHeadcount=" +
        simulationdata.getHc();
    result = vensim.command(vensimCommand);
    vensimCommand =
        "SIMULATE>SETVAL|WillingnessToAdoptOverTime=" +
        simulationdata.getOt();
    result = vensim.command(vensimCommand);
    vensimCommand =
        "SIMULATE>SETVAL|ReliabilityBufferingActivated=" +
        simulationdata.getReliability();
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|SecondBufferActivated=" +
        simulationdata.getReliability();
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|FractionOfBufferingPJ=" +
        simulationdata.getBuffer();
    result = vensim.command(vensimCommand);
    vensimCommand =
        "SIMULATE>SETVAL|KnownContingencyFactorPJ=" +
        simulationdata.getContingency();
    result = vensim.command(vensimCommand);
    vensimCommand =
        "SIMULATE>SETVAL|DurationDividerForRFIReplyTime=" +
        simulationdata.getRfi();
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|TimeToIncreaseWorkforce=" +
        simulationdata.getIncrease();
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|ExperienceLevel=" +
        simulationdata.getWOEX();
    result = vensim.command(vensimCommand);
}
for(int i1 = 0; i1 < l; i1++)

```

```

{
    int j1 = ((Activity)vector1.get(i1)).getId();
    int k1 = ((Activity)vector1.get(i1)).getST();
    String s7 = ((Activity)vector1.get(i1)).getCode();
    int k2 = ((Activity)vector1.get(i1)).getComplexity();
    double d = ((Activity)vector1.get(i1)).getReliability();
    double d1 = ((Activity)vector1.get(i1)).getStability();
    int j7 = ((Activity)vector1.get(i1)).getDuration();
    double d2 = ((Activity)vector1.get(i1)).getContingency();
    double d3 = ((Activity)vector1.get(i1)).getBuffering();
    double d4 =
        ((Activity)vector1.get(i1)).getInternalSensitivity();
    int l9 =
        ((Activity)vector1.get(i1)).getIsCriticalActivity();
    int j10 =
        ((Activity)vector1.get(i1)).getDurationOptimistic();
    int k10 =
        ((Activity)vector1.get(i1)).getDurationPessimistic()
    ;
    vensimCommand = "SIMULATE>SETVAL|InitialStartTime[" + s7 +
        "]" + k1;
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|FastEvolution[" + s7 +
        "]" + k2;
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|GivenReliability[" + s7 +
        "]" + d;
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|GivenStability[" + s7 +
        "]" + d1;
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|OriginalDuration[" + s7 +
        "]" + j7;
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|OptimisticDuration[" + s7
        + "]" + j10;
    result = vensim.command(vensimCommand);
    vensimCommand = "SIMULATE>SETVAL|PessimisticDuration[" +

```

```

        s7 + "]" + k10;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|KnownContingencyFactor["
    + s7 + "]" + d2;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|FractionOfBuffering[" +
    s7 + "]" + d3;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|InternalSensitivity[" +
    s7 + "]" + d4;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|IsItCritical[" + s7 +
    "]" + l9;
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|QMfamiliarity[" + s7 +
    "]" + simulationdata.getQMth();
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|SMfamiliarity[" + s7 +
    "]" + simulationdata.getSMth();
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|DividerTimeforQM[" + s7 +
    "]" + simulationdata.getQMPeriod();
result = vensim.command(vensimCommand);
vensimCommand = "SIMULATE>SETVAL|TimeForCCBDecision[" + s7
    + "]" + simulationdata.getCCB();
result = vensim.command(vensimCommand);
if(i == 1)
{
    vensimCommand =
        "SIMULATE>SETVAL|IsInstaneousActivity[" + s7 + "]" +
        + 1;
    result = vensim.command(vensimCommand);
}
if(i == 2)
{
    vensimCommand = "SIMULATE>SETVAL|IsItPERT[" + s7 +
        "]" + 1;
    result = vensim.command(vensimCommand);
}

```

```

}
Vector vector2 = new Vector();
try
{
    vector2 = databaseInitializer.getPredActivity(j1);
}
catch(Exception _ex)
{
}
for(int l10 = 0; l10 < vector2.size(); l10++)
{
    String s13 =
        ((Precedence)vector2.get(l10)).getRelation();
    String s14 =
        ((Precedence)vector2.get(l10)).getPredActivityCode()
;
    int i11 = ((Precedence)vector2.get(l10)).getLag();
    int j11 = ((Precedence)vector2.get(l10)).getSens();
    if(s13.equals("FS"))
    {
        vensimCommand =
            "SIMULATE>SETVAL|PrecedenceRelationship[" + s7
            + "," + s14 + "]= " + i11;
        result = vensim.command(vensimCommand);
    } else
    if(s13.equals("SS"))
    {
        i11 += 1000;
        vensimCommand =
            "SIMULATE>SETVAL|PrecedenceRelationship[" + s7
            + "," + s14 + "]= " + i11;
        result = vensim.command(vensimCommand);
    } else
    if(s13.equals("FF"))
    {
        i11 += 2000;
        vensimCommand =

```

```

        "SIMULATE>SETVAL|PrecedenceRelationship[" + s7
        + "," + s14 + "]=1";
        result = vensim.command(vensimCommand);
    } else
    if(s13.equals("RW"))
    {
        vensimCommand =
            "SIMULATE>SETVAL|ReprocessIterationRelationship[" + s7 + "," + s14 + "]=1";
        result = vensim.command(vensimCommand);
    }
    vensimCommand = "SIMULATE>SETVAL|ExternalSensitivity[" + s7 + "," + s14 + "]=1";
    result = vensim.command(vensimCommand);
}
}
result = vensim.command("MENU>RUN|O");
if(result == 0)
if(i == 4)
{
    tpoints = vensim.get_data("dpm.vdf", "ProjectFinishTime",
        "time", af, af1, 801);
    tpoints = (int)af[800];
    float af2[] = new float[801];
    float af6[] = new float[801];
    tpoints01 = vensim.get_data("dpm.vdf", "ProjectProgress",
        "time", af2, af6, 801);
    for(int l1 = 0; l1 < af6.length; l1++)
    {
        if(af6[l1] != (float)tpoints)
            continue;
        testLength = l1;
        break;
    }
    float af10[] = new float[testLength];
    float af13[] = new float[testLength];
    for(int l2 = 0; l2 < testLength; l2++)
    {

```

```

        af13[12] = af6[12];
        af10[12] = 100F * af2[12];
    }
    setTimeForGraph(af13);
    setDataForGraph(af10);
    setFinalTime(tpoints);
    String s = "update DPM_PROJECTOUTPUT set simulatedduration
        = '" + tpoints + "' where PROJID='" + j + "' AND
        VER_ID='" + k + "'";
    try
    {
        ded.dispatch(s);
    }
    catch(Exception _ex) { }
    s = "update DPM_PROJECTOUTPUT set pdmduration = '" +
        tpoints + "' where PROJID='" + j + "' AND VER_ID
        =' " + k + "'";
    try
    {
        ded.dispatch(s);
    }
    catch(Exception _ex) { }
    s = "DELETE FROM DPM_ACTIVITYOUTPUT WHERE PROJID='" + j +
        "' AND VER_ID='" + k + "'";
    try
    {
        ded.dispatch(s);
    }
    catch(Exception _ex) { }
    for(int i3 = 0; i3 < 1; i3++)
    {
        String s8 = ((Activity)vector1.get(i3)).getCode();
        int k4 = ((Activity)vector1.get(i3)).getId();
        int j5 = vensim.get_data("dpm.vdf",
            "ReliabilityStartTime[" + s8 + "]", "time", af, af1,
            801);
        j5 = (int)af[800];
        int k6 = vensim.get_data("dpm.vdf",

```

```

        "ReliabilityBuffer[" + s8 + "]", "time", af, af1,
        801);
k6 = (int)af[800];
int k7 = vensim.get_data("dpm.vdf",
    "ActualActivityStartTime[" + s8 + "]", "time", af,
    af1, 801);
k7 = (int)af[800];
int i8 = vensim.get_data("dpm.vdf",
    "ActivityDuration[" + s8 + "]", "time", af, af1,
    801);
i8 = (int)af[800];
int k8 = vensim.get_data("dpm.vdf",
    "ActivityFinishTime[" + s8 + "]", "time", af, af1,
    801);
k8 = (int)af[800];
String s12 = ((Activity)vector1.get(i3)).getWbs();
int i9 = vensim.get_data("dpm.vdf",
    "SecondBufferStartTime[" + s8 + "]", "time", af, af1,
    801);
i9 = (int)af[800];
int j9 = vensim.get_data("dpm.vdf", "SecondBuffer[" +
    s8 + "]", "time", af, af1, 801);
j9 = (int)af[800];
int k9 = 1;
int i10 = 0;
String s1 = "INSERT INTO DPM_ACTIVITYOUTPUT VALUES ('"
    + k4 + "',''" + j5 + "',''" + k6 + "',''" + k7 + "',''"
    + i8 + "',''" + 0 + "',''" + 0 + "',''" + 0 + "',''" +
    k9 + "',''" + i10 + "',''" + s8 + "',''" + j + "',''" +
    k8 + "','0, 0, 0, 0,'" + s12 + "',''" + i9 + "',''" +
    j9 + "','0,0, '" + k + "'");
try
{
    ded.dispatch(s1);
}
catch(Exception exception)
{
    exception.printStackTrace();
}

```



```

setFinalTime2(tpoints);
String s2 = "update DPM_PROJECTOUTPUT set
            simulatedduration = '" + tpoints + "' where PROJID
            ='" + j + "' AND VER_ID ='" + k + "'";
try
{
    ded.dispatch(s2);
}
catch(Exception _ex) { }
for(int k3 = 0; k3 < 1; k3++)
{
    String s9 = ((Activity)vector1.get(k3)).getCode();
    int 14 = ((Activity)vector1.get(k3)).getId();
    vensim.get_data("cpm.vdf", "ReliabilityStartTime[" +
        s9 + "]", "time", af, af1, 801);
    int _tmp = (int)af[800];
    vensim.get_data("cpm.vdf", "ReliabilityBuffer[" + s9 +
        "]", "time", af, af1, 801);
    int _tmp1 = (int)af[800];
    int k5 = vensim.get_data("cpm.vdf",
        "ActualActivityStartTime[" + s9 + "]", "time", af,
        af1, 801);
    k5 = (int)af[800];
    int 16 = vensim.get_data("cpm.vdf",
        "ActivityDuration[" + s9 + "]", "time", af, af1,
801);
    16 = (int)af[800];
    vensim.get_data("cpm.vdf", "ActivityFinishTime[" + s9
        + "]", "time", af, af1, 801);
    int _tmp2 = (int)af[800];
    String s3 = "UPDATE DPM_ACTIVITYOUTPUT SET CPMST = '"
        + k5 + "', CPMDURATION = '" + 16 + "' WHERE ID = '"
        + 14 + "'";
    try
    {
        ded.dispatch(s3);
    }
    catch(Exception _ex) { }
}

```

```

    }
}
if(i == 2 || i == 4)
{
    tpoints = vensim.get_data("pert.vdf", "ProjectFinishTime",
        "time", af, af1, 801);
    tpoints = (int)af[800];
    float af4[] = new float[801];
    float af8[] = new float[801];
    vensim.get_data("pert.vdf", "TOTALSigma", "time", af4, af8,
        801);
    float f = af4[800];
    setPERTSigma(f);
    float af15[] = new float[801];
    float af17[] = new float[801];
    tpoints02 = vensim.get_data("pert.vdf", "ProjectProgress",
        "time", af15, af17, 801);
    for(int j4 = 0; j4 < af17.length; j4++)
    {
        if(af17[j4] != (float)tpoints)
            continue;
        testLength = j4;
        break;
    }
    float af18[] = new float[testLength];
    float af19[] = new float[testLength];
    for(int l5 = 0; l5 < testLength; l5++)
    {
        af19[l5] = af17[l5];
        af18[l5] = 100F * af15[l5];
    }

    setTimeForGraph4(af19);
    setDataForGraph4(af18);
    setFinalTime4(tpoints);
    String s4 = "update DPM_PROJECTOUTPUT set
        simulatedduration = '" + tpoints + "' where PROJID
        ='" + j + "' AND VER_ID ='" + k + "'";
}

```

```

try
{
    ded.dispatch(s4);
}
catch(Exception _ex) { }
for(int i6 = 0; i6 < 1; i6++)
{
    String s11 = ((Activity)vector1.get(i6)).getCode();
    int l7 = ((Activity)vector1.get(i6)).getId();
    vensim.get_data("pert.vdf", "ReliabilityStartTime[" +
        s11 + "]", "time", af, af1, 801);
    int _tmp3 = (int)af[800];
    vensim.get_data("pert.vdf", "ReliabilityBuffer[" + s11
        + "]", "time", af, af1, 801);
    int _tmp4 = (int)af[800];
    int j8 = vensim.get_data("pert.vdf",
        "ActualActivityStartTime[" + s11 + "]", "time", af,
        af1, 801);
    j8 = (int)af[800];
    int l8 = vensim.get_data("pert.vdf",
        "ActivityDuration[" + s11 + "]", "time", af, af1,
        801);
    l8 = (int)af[800];
    vensim.get_data("pert.vdf", "ActivityFinishTime[" +
        s11 + "]", "time", af, af1, 801);
    int _tmp5 = (int)af[800];
    String s5 = "UPDATE DPM_ACTIVITYOUTPUT SET PERTST = '"
        + j8 + "'", PERTDURATION = '" + l8 + "' WHERE ID = '"
        + l7 + "'";
    try
    {
        ded.dispatch(s5);
    }
    catch(Exception _ex) { }
}
}
if(i == 3 || i == 4)
{

```

```

tpoints = vensim.get_data("dpmWoAction.vdf",
    "ProjectFinishTime", "time", af, af1, 801);
tpoints = (int)af[800];
float af5[] = new float[801];
float af9[] = new float[801];
tpoints03 = vensim.get_data("dpmWoAction.vdf",
    "ProjectProgress", "time", af5, af9, 801);
for(int j2 = 0; j2 < af9.length; j2++)
{
    if(af9[j2] != (float)tpoints)
        continue;
    testLength = j2;
    break;
}
float af12[] = new float[testLength];
float af16[] = new float[testLength];
for(int l3 = 0; l3 < testLength; l3++)
{
    af16[l3] = af9[l3];
    af12[l3] = 100F * af5[l3];
}
setTimeForGraph3(af16);
setDataForGraph3(af12);
setFinalTime3(tpoints);
for(int i4 = 0; i4 < l; i4++)
{
    String s10 = ((Activity)vector1.get(i4)).getCode();
    int i5 = ((Activity)vector1.get(i4)).getId();
    vensim.get_data("dpmWoAction.vdf",
        "ReliabilityStartTime[" + s10 + "]", "time", af, af1,
        801);
    int _tmp6 = (int)af[800];
    vensim.get_data("dpmWoAction.vdf",
        "ReliabilityBuffer[" + s10 + "]", "time", af, af1,
        801);
    int _tmp7 = (int)af[800];
    int j6 = vensim.get_data("dpmWoAction.vdf",

```

```

        "ActualActivityStartTime[" + s10 + "]", "time", af,
        af1, 801);
j6 = (int)af[800];
int i7 = vensim.get_data("dpmWoAction.vdf",
    "ActivityDuration[" + s10 + "]", "time", af, af1,
    801);
i7 = (int)af[800];
vensim.get_data("dpmWoAction.vdf",
    "ActivityFinishTime[" + s10 + "]", "time", af, af1,
    801);
int _tmp8 = (int)af[800];
String s6 = "UPDATE DPM_ACTIVITYOUTPUT SET WOST = '" +
    j6 + "', WODURATION = '" + i7 + "' WHERE ID = '" +
    i5 + "'";
try
{
    ded.dispatch(s6);
}
catch(Exception _ex) { }
}
}
}
public float getActivityFinalTimePERT()
{
    return activityFinalTime3;
}
public float getActivityStartTimePERT()
{
    return activityStartTime3;
}
public float[] getDataForGraph4()
{
    return resultValue03;
}
public float[] getDataForGraphCPM3()
{
    return resultValueCPM3;
}
}

```

```

public float[] getDataForGraphPERT()
{
    return resultValuePERT;
}
public float[] getDataForGraphPERT1()
{
    return resultValuePERT1;
}
public float[] getDataForGraphPERT2()
{
    return resultValuePERT2;
}
public float[] getDataForGraphPERT3()
{
    return resultValuePERT3;
}
public float[] getDataForGraphStability()
{
    return resultValue4;
}
public float[] getDataForGraphWA3()
{
    return resultValueWA3;
}
public float getFinalTime4()
{
    return finalTime03;
}
public float getPERTSigma()
{
    return sigma;
}
public float[] getTimeForGraph4()
{
    return timeValue03;
}
public float[] getTimeForGraphPERT()
{

```

```

        return timeValuePERT;
    }
    public float[] getTimeForGraphStability()
    {
        return timeValue4;
    }
    public void setActivityFinalTimePERT(float f)
    {
        activityFinalTime3 = f;
    }
    public void setActivityStability(String s)
    {
        int i = (int) getActivityStartTime();
        int j = (int) getActivityFinalTime();
        float af[] = new float[801];
        float af1[] = new float[801];
        vensim.get_data("dpm.vdf", "ActualStability[" + s + "]",
            "time", af, af1, 801);
        int k = (j - i) + 1;
        float af2[] = new float[k];
        float af3[] = new float[k];
        int l = 0;
        for(int il = 0; il < af1.length; il++)
        {
            if(af1[il] < (float)i)
                continue;
            af3[l] = af1[il];
            af2[l] = af[il] * 100F;
            if(++l == k)
                break;
        }
        setTimeForGraphStability(af3);
        setDataForGraphStability(af2);
        int j1 = (int) getActivityStartTimeCPM();
        int k1 = (int) getActivityFinalTimeCPM();
        int l1 = (k1 - j1) + 1;
        vensim.get_data("cpm.vdf", "ActualStability[" + s + "]",
            "time", af, af1, 801);
    }

```



```

float af4[] = new float[l1];
int i2 = 0;
for(int j2 = 0; j2 < af1.length; j2++)
{
    if(af1[j2] < (float)j1)
        continue;
    af4[i2] = af[j2] * 100F;
    if(++i2 == l1)
        break;
}
setDataForGraphCPM3(af4);
int k2 = (int) getActivityStartTimePERT();
int l2 = (int) getActivityFinalTimePERT();
int i3 = (l2 - k2) + 1;
vensim.get_data("pert.vdf", "ActualStability[" + s + "]",
    "time", af, af1, 801);
float af5[] = new float[i3];
int j3 = 0;
for(int k3 = 0; k3 < af1.length; k3++)
{
    if(af1[k3] < (float)k2)
        continue;
    af5[j3] = af[k3] * 100F;
    if(++j3 == i3)
        break;
}
setDataForGraphPERT3(af5);
int l3 = (int) getActivityStartTimeWA();
int i4 = (int) getActivityFinalTimeWA();
int j4 = (i4 - l3) + 1;
vensim.get_data("dpmWoAction.vdf", "ActualStability[" + s +
    "]", "time", af, af1, 801);
float af6[] = new float[j4];
int k4 = 0;
for(int l4 = 0; l4 < af1.length; l4++)
{
    if(af1[l4] < (float)l3)
        continue;

```

```

        af6[k4] = af[l4] * 100F;
        if(++k4 == j4)
            break;
    }
    setDataForGraphWA3(af6);
}
public void setActivityStartTimePERT(float f)
{
    activityStartTime3 = f;
}
public void setDataForGraph4(float af[])
{
    resultValue03 = af;
}
public void setDataForGraphCPM3(float af[])
{
    resultValueCPM3 = af;
}
public void setDataForGraphPERT(float af[])
{
    resultValuePERT = af;
}
public void setDataForGraphPERT1(float af[])
{
    resultValuePERT1 = af;
}
public void setDataForGraphPERT2(float af[])
{
    resultValuePERT2 = af;
}
public void setDataForGraphPERT3(float af[])
{
    resultValuePERT3 = af;
}
public void setDataForGraphStability(float af[])
{
    resultValue4 = af;
}

```

```

public void setDataForGraphWA3(float af[])
{
    resultValueWA3 = af;
}
public void setFinalTime4(float f)
{
    finalTime03 = f;
}
public void setPert()
{
    float af[] = new float[801];
    float af1[] = new float[801];
    int i = vensim.get_data("pert.vdf", "ProjectFinishTime",
        "time", af1, af, 801);
    i = (int)af1[800];
    float af2[] = new float[801];
    float af3[] = new float[801];
    vensim.get_data("pert.vdf", "ProjectProgress", "time", af3,
        af2, 801);
    int j = 0;
    for(int k = 0; k < af2.length; k++)
    {
        if(af2[k] != (float)i)
            continue;
        j = k;
        break;
    }
    float af4[] = new float[j];
    float af5[] = new float[j];
    for(int l = 0; l < j; l++)
    {
        af5[l] = af2[l];
        af4[l] = 100F * af3[l];
    }
    float af6[] = new float[801];
    float af7[] = new float[801];
    vensim.get_data("pert.vdf", "TotalSigma", "time", af7, af6,
        801);
}

```

```

float f = af7[800];
setTimeForGraph4(af5);
setDataForGraph4(af4);
setFinalTime4(i);
setPERTSigma(f);
}
public void setPERTSigma(float f)
{
    sigma = f;
}
public void setTimeForGraph4(float af[])
{
    timeValue03 = af;
}
public void setTimeForGraphPERT(float af[])
{
    timeValuePERT = af;
}
public void setTimeForGraphStability(float af[])
{
    timeValue4 = af;
}
public void createActivity(String s, int i, String s1, String s2,
    String s3, int j, int k) throws RemoteException
{
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    oracledatabaseconnection.executeSQL("INSERT INTO DPM_ACTIVITY
        VALUES(SEQ_DPM_ACTIVITY.NEXTVAL,'" + s1 + "','" + s2 +
        "','" + s3 + "','" + j + "','" + 0 + "','" + j + "','" +
        s + "','" + j + "','" + j + "','" + j + "','" + 1, 1 ,1, 0, 0,
        1, 1,'" + k + "','0,0,0,1)");
    oracledatabaseconnection.disconnect();
}
public void createVersion(int i, int j)
    throws RemoteException
{
    String s = "Changed version as of one year later";

```

```

OracleDatabaseConnection oracledatabaseconnection = new
    OracleDatabaseConnection();
oracledatabaseconnection.executeSQL("INSERT INTO DPM_VERSION
    VALUES('" + j + "','"+ i + "','"+ s + "' )");
oracledatabaseconnection.executeSQL("INSERT INTO
    DPM_PROJECTOUTPUT VALUES('" + i + "','0,0,0,0,0,0,0,0,'" + j
    + "')");
oracledatabaseconnection.disconnect();
}
public int findDPMId(int i)
{
    int j = 0;
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeSQL("SELECT ID FROM
        DPM_ACTIVITY WHERE PRIACTID = '" + i + "'");
    if(resultset != null)
        try
        {
            while(resultset.next())
                j = resultset.getInt(1);
        }
        catch(Exception _ex)
        {
        }
    oracledatabaseconnection.disconnect();
    return j;
}
public Vector getActivity(int i, int j)
    throws RemoteException
{
    vectorOne.removeAllElements();
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeSQL("SELECT ID, CODE, WBS,
        NAME, DURATION, ST, FT, DMOSTLIKELY, DOPTIMISTIC,

```

```

DPESSIMISTIC, RELIABILITY, STABILITY, COMPLEXITY,
CONTINGENCY, BUFFERING, INTERNALSENSITIVITY,
ISCITICALACTIVITY, PRIACTID, SIMST, SIMDURATION, SIMORNOT
FROM DPM_ACTIVITY WHERE PROJID = '"' + i + '"' AND VER_ID = '"'
+ j + '"');
if(resultset != null)
    try
    {
        int k;
        String s;
        String s1;
        String s2;
        int l;
        int i1;
        int j1;
        int k1;
        int l1;
        int i2;
        double d;
        double d1;
        int j2;
        double d2;
        double d3;
        double d4;
        int k2;
        int l2;
        int i3;
        int j3;
        int k3;
        for(; resultset.next(); vectorOne.add(new Activity(k,
            s, s1, s2, l, i1, j1, k1, l1, i2, d, d1, j2, d2, d3,
            d4, k2, l2, i3, j3, k3, j)))
        {
            k = resultset.getInt(1);
            s = resultset.getString(2);
            s1 = resultset.getString(3);
            s2 = resultset.getString(4);
            l = resultset.getInt(5);

```

```

        i1 = resultset.getInt(6);
        j1 = resultset.getInt(7);
        k1 = resultset.getInt(8);
        l1 = resultset.getInt(9);
        i2 = resultset.getInt(10);
        d = resultset.getDouble(11);
        d1 = resultset.getDouble(12);
        j2 = resultset.getInt(13);
        d2 = resultset.getDouble(14);
        d3 = resultset.getDouble(15);
        d4 = resultset.getDouble(16);
        k2 = resultset.getInt(17);
        l2 = resultset.getInt(18);
        i3 = resultset.getInt(19);
        j3 = resultset.getInt(20);
        k3 = resultset.getInt(21);
    }
}
catch(SQLException _ex)
{
}
oracledatabaseconnection.disconnect();
return vectorOne;
}
public int getNumOfVersion(int i)
    throws RemoteException
{
    int j = 0;
    OracleDatabaseConnection oracledatabaseconnection = new
        OracleDatabaseConnection();
    ResultSet resultset =
        oracledatabaseconnection.executeQuery("SELECT NUMOFVERSION
        FROM DPM_PROJECT WHERE PROJID = '" + i + "'");
    if(resultset != null)
        try
        {
            while(resultset.next())
                j = resultset.getInt(1);
        }
    }
}

```

```

        }
        catch(SQLException _ex)
        {
        }
        oracledatabaseconnection.disconnect();
        return j;
    }
    public Vector getVersionList(int i)
        throws RemoteException
    {
        Vector vector1 = new Vector();
        OracleDatabaseConnection oracledatabaseconnection = new
            OracleDatabaseConnection();
        ResultSet resultset =
            oracledatabaseconnection.executeQuery("SELECT VER_ID,
            VERSIONINFO FROM DPM_VERSION WHERE PROJID = '" + i +
            "'");
        if(resultset != null)
            try
            {
                int j;
                String s;
                for(; resultset.next(); vector1.add(new Version(j,
                    s)))
                {
                    j = resultset.getInt(1);
                    s = resultset.getString(2);
                }
            }
            catch(SQLException _ex)
            {
            }
        oracledatabaseconnection.disconnect();
        return vector1;
    }
    public Vector vector;
    public Vector vectorOne;
    private Vensim vensim;

```



```

private int tpoints;
private int result;
private int tpoints01;
private int tpoints02;
private int tpoints03;
private String vensimCommand;
private DatabaseInitializerImpl databaseInitializer;
private DatabaseEventDispatcherImpl ded;
public Vector clientVector;
public int tempduration;
float timeValue[];
float resultValue[];
float timeValue01[];
float resultValue01[];
float timeValue02[];
float resultValue02[];
float timeValue1[];
float resultValue1[];
float timeValue2[];
float resultValue2[];
float timeValue3[];
float resultValue3[];
float resultValueCPM[];
float resultValueWA[];
float resultValueCPM1[];
float resultValueWA1[];
float resultValueCPM2[];
float resultValueWA2[];
float timeValueCPM[];
float timeValueWA[];
float alreadySimTime[];
float alreadySimData[];
float activityStartTime;
float activityFinalTime;
float activityStartTime1;
float activityFinalTime1;
float activityStartTime2;
float activityFinalTime2;

```

```
float activityStartTime3;
float activityFinalTime3;
float alreadySimFinalTime;
float finalTime;
float finalTime01;
float finalTime02;
float finalTime03;
int testLength;
int testLength01;
int gameLength;
int interval;
float resultValue03[];
float resultValue4[];
float resultValueCPM3[];
float resultValuePERT[];
float resultValuePERT1[];
float resultValuePERT2[];
float resultValuePERT3[];
float resultValueWA3[];
float sigma;
float timeValue03[];
float timeValue4[];
float timeValuePERT[];
}
```

```

package gpms.interfaces.impl;

import java.io.PrintStream;

public class Vensim
{
    public native int be_quiet(int i);
    public native int check_status();
    public native int command(String s);
    public native int continue_simulation(int i);
    public native int finish_simulation();
    public native int get_data(String s, String s1, String s2, float
        af[], float af1[], int i);
    public native int get_dpval(String s, double ad[]);
    public native int get_val(String s, float af[]);
    public native int show_sketch(int i, int j, int k, int l);
    public native int start_simulation(int i, int j, int k);
    public native int tool_command(String s, int i, int j);
    public native int vensim_get_dpvecvals(long al[], double ad[],
        int i);
    public native int vensim_get_info(int i, String s, int j);
    public native int vensim_get_sens_at_time(String s, String s1,
        String s2, float af[], float af1[], int i);
    public native int vensim_get_substring(String s, int i, String s1,
        int j);
    public native int vensim_get_varattrib(String s, int i, String s1,
        int j);
    public native int vensim_get_varnames(String s, int i, String s1,
        int j);
    public native long vensim_get_varoff(String s);
    public native int vensim_get_vecvals(long al[], float af[], int
        i);
    public native int vensim_set_parent_window(int i, long l, long
        ll);
    public Vensim()
    {
    }
    static

```

```
{
  try
  {
    System.loadLibrary("venjava");
  }
  catch(UnsatisfiedLinkError unsatisfiedlinkerror)
  {
  }
}
}
```

```
package gpms.interfaces.rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ClientEventHandler
    extends Remote
{
    public abstract void handle(String s)
        throws RemoteException;
}
```

```
package gpms.interfaces.rmi;

import gpms.util.Event;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface DatabaseEventDispatcher
    extends Remote
{
    public abstract void dispatch(Event event)
        throws RemoteException;
    public abstract void dispatch(String s)
        throws RemoteException;
}
```

```
package gpms.interfaces.rmi;

import gpms.client.output.ProjectOutput;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Vector;

public interface DatabaseInitializer
    extends Remote
{
    public abstract Vector getActivity(int i)
        throws RemoteException;
    public abstract Vector getPredActivity(int i)
        throws RemoteException;
    public abstract Vector getProjects()
        throws RemoteException;
    public abstract Vector getActivityOutput(int i, int j)
        throws RemoteException;
    public abstract ProjectOutput getProjectOutput(int i, int j)
        throws RemoteException;
}
```

```

package gpms.interfaces.rmi;

import gpms.client.gui.GameData;
import gpms.client.output.SimulationData;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Vector;

public interface Server
    extends Remote
{
    public abstract void attach(ClientEventHandler
clienteventhandler)
        throws RemoteException;
    public abstract void createProject(String s, String sl, int i,
        int j, int k) throws RemoteException;
    public abstract void createRelationship(int i, int j, String s,
        int k, int l, String sl) throws RemoteException;
    public abstract void detach(ClientEventHandler
        clienteventhandler) throws RemoteException;
    public abstract void dispatch(String s)
        throws RemoteException;
    public abstract String findPredCode(int i)
        throws RemoteException;
    public abstract void gameSimulation(GameData gamedata,
        SimulationData simulationdata, int i, boolean flag) throws
        RemoteException;
    public abstract float getActivityFinalTime()
        throws RemoteException;
    public abstract float getActivityFinalTimeCPM()
        throws RemoteException;
    public abstract float getActivityFinalTimeWA()
        throws RemoteException;
    public abstract float getActivityStartTime()
        throws RemoteException;
    public abstract float getActivityStartTimeCPM()
        throws RemoteException;
    public abstract float getActivityStartTimeWA()

```



```

        throws RemoteException;
public abstract float[] getDataForGraph()
        throws RemoteException;
public abstract float[] getDataForGraph2()
        throws RemoteException;
public abstract float[] getDataForGraph3()
        throws RemoteException;
public abstract float[] getDataForGraphCPM()
        throws RemoteException;
public abstract float[] getDataForGraphCPM1()
        throws RemoteException;
public abstract float[] getDataForGraphCPM2()
        throws RemoteException;
public abstract float[] getDataForGraphProductivity()
        throws RemoteException;
public abstract float[] getDataForGraphProgress()
        throws RemoteException;
public abstract float[] getDataForGraphQuality()
        throws RemoteException;
public abstract float[] getDataForGraphWA()
        throws RemoteException;
public abstract float[] getDataForGraphWA1()
        throws RemoteException;
public abstract float[] getDataForGraphWA2()
        throws RemoteException;
public abstract float getFinalTime()
        throws RemoteException;
public abstract float getFinalTime2()
        throws RemoteException;
public abstract float getFinalTime3()
        throws RemoteException;
public abstract int getGameInterval()
        throws RemoteException;
public abstract Vector getPrimaveraActivity(int i)
        throws RemoteException;
public abstract Vector getPrimaveraProjectList()
        throws RemoteException;
public abstract Vector getPrimaveraRelationship(int i)

```

```

        throws RemoteException;
public abstract Vector getProjectList()
        throws RemoteException;
public abstract float[] getSimFile_data()
        throws RemoteException;
public abstract float getSimFile_finaltime()
        throws RemoteException;
public abstract float[] getSimFile_time()
        throws RemoteException;
public abstract int getStartTimeActivity(String s)
        throws RemoteException;
public abstract float[] getTimeForGraph()
        throws RemoteException;
public abstract float[] getTimeForGraph2()
        throws RemoteException;
public abstract float[] getTimeForGraph3()
        throws RemoteException;
public abstract float[] getTimeForGraphCPM()
        throws RemoteException;
public abstract float[] getTimeForGraphProductivity()
        throws RemoteException;
public abstract float[] getTimeForGraphProgress()
        throws RemoteException;
public abstract float[] getTimeForGraphQuality()
        throws RemoteException;
public abstract float[] getTimeForGraphWA()
        throws RemoteException;
public abstract void monteCarlo()
        throws RemoteException;
public abstract void policyMaker()
        throws RemoteException;
public abstract void setActivityFinalTime(float f)
        throws RemoteException;
public abstract void setActivityFinalTimeCPM(float f)
        throws RemoteException;
public abstract void setActivityFinalTimeWA(float f)
        throws RemoteException;
public abstract void setActivityProductivity(String s)

```

```

        throws RemoteException;
public abstract void setActivityProgress(String s)
        throws RemoteException;
public abstract void setActivityQuality(String s)
        throws RemoteException;
public abstract void setActivityStartTime(float f)
        throws RemoteException;
public abstract void setActivityStartTimeCPM(float f)
        throws RemoteException;
public abstract void setActivityStartTimeWA(float f)
        throws RemoteException;
public abstract void setCpm()
        throws RemoteException;
public abstract void setDataForGraph(float af[])
        throws RemoteException;
public abstract void setDataForGraph2(float af[])
        throws RemoteException;
public abstract void setDataForGraph3(float af[])
        throws RemoteException;
public abstract void setDataForGraphCPM1(float af[])
        throws RemoteException;
public abstract void setDataForGraphCPM2(float af[])
        throws RemoteException;
public abstract void setDataForGraphProductivity(float af[])
        throws RemoteException;
public abstract void setDataForGraphProgress(float af[])
        throws RemoteException;
public abstract void setDataForGraphQuality(float af[])
        throws RemoteException;
public abstract void setDataForGraphWA(float af[])
        throws RemoteException;
public abstract void setDataForGraphWA1(float af[])
        throws RemoteException;
public abstract void setDataForGraphWA2(float af[])
        throws RemoteException;
public abstract void setDpm()
        throws RemoteException;
public abstract void setDpmWoAction()

```

```

        throws RemoteException;
public abstract void setFinalTime(float f)
        throws RemoteException;
public abstract void setFinalTime2(float f)
        throws RemoteException;
public abstract void setFinalTime3(float f)
        throws RemoteException;
public abstract void setGameInterval(int i)
        throws RemoteException;
public abstract void setSimFile(String s)
        throws RemoteException;
public abstract void setSimFile_data(float af[])
        throws RemoteException;
public abstract void setSimFile_finaltime(float f)
        throws RemoteException;
public abstract void setSimFile_time(float af[])
        throws RemoteException;
public abstract void setTimeForGraph(float af[])
        throws RemoteException;
public abstract void setTimeForGraph2(float af[])
        throws RemoteException;
public abstract void setTimeForGraph3(float af[])
        throws RemoteException;
public abstract void setTimeForGraphCPM(float af[])
        throws RemoteException;
public abstract void setTimeForGraphProductivity(float af[])
        throws RemoteException;
public abstract void setTimeForGraphProgress(float af[])
        throws RemoteException;
public abstract void setTimeForGraphQuality(float af[])
        throws RemoteException;
public abstract void setTimeForGraphWA(float af[])
        throws RemoteException;
public abstract void simulation(SimulationData simulationdata,
        int i) throws RemoteException;
public abstract void skdlPlanner(SimulationData simulationdata)
        throws RemoteException;
public abstract float getActivityFinalTimePERT()

```

```

        throws RemoteException;
public abstract float getActivityStartTimePERT()
        throws RemoteException;
public abstract float[] getDataForGraph4()
        throws RemoteException;
public abstract float[] getDataForGraphCPM3()
        throws RemoteException;
public abstract float[] getDataForGraphPERT()
        throws RemoteException;
public abstract float[] getDataForGraphPERT1()
        throws RemoteException;
public abstract float[] getDataForGraphPERT2()
        throws RemoteException;
public abstract float[] getDataForGraphPERT3()
        throws RemoteException;
public abstract float[] getDataForGraphStability()
        throws RemoteException;
public abstract float[] getDataForGraphWA3()
        throws RemoteException;
public abstract float getFinalTime4()
        throws RemoteException;
public abstract float getPERTSigma()
        throws RemoteException;
public abstract float[] getTimeForGraph4()
        throws RemoteException;
public abstract float[] getTimeForGraphPERT()
        throws RemoteException;
public abstract float[] getTimeForGraphStability()
        throws RemoteException;
public abstract void setActivityFinalTimePERT(float f)
        throws RemoteException;
public abstract void setActivityStability(String s)
        throws RemoteException;
public abstract void setActivityStartTimePERT(float f)
        throws RemoteException;
public abstract void setDataForGraph4(float af[])
        throws RemoteException;
public abstract void setDataForGraphCPM3(float af[])

```

```

        throws RemoteException;
public abstract void setDataForGraphPERT(float af[])
        throws RemoteException;
public abstract void setDataForGraphPERT1(float af[])
        throws RemoteException;
public abstract void setDataForGraphPERT2(float af[])
        throws RemoteException;
public abstract void setDataForGraphPERT3(float af[])
        throws RemoteException;
public abstract void setDataForGraphStability(float af[])
        throws RemoteException;
public abstract void setDataForGraphWA3(float af[])
        throws RemoteException;
public abstract void setFinalTime4(float f)
        throws RemoteException;
public abstract void setPert()
        throws RemoteException;
public abstract void setPERTSigma(float f)
        throws RemoteException;
public abstract void setTimeForGraph4(float af[])
        throws RemoteException;
public abstract void setTimeForGraphPERT(float af[])
        throws RemoteException;
public abstract void setTimeForGraphStability(float af[])
        throws RemoteException;
public abstract void createActivity(String s, int i, String s1,
        String s2, String s3, int j, int k) throws RemoteException;
public abstract void createVersion(int i, int j)
        throws RemoteException;
public abstract int findDPMId(int i)
        throws RemoteException;
public abstract Vector getActivity(int i, int j)
        throws RemoteException;
public abstract int getNumOfVersion(int i)
        throws RemoteException;
public abstract Vector getVersionList(int i)
        throws RemoteException;
}

```

```
package gpms.interfaces.rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ClientEventHandler
    extends Remote
{
    public abstract void handle(String s)
        throws RemoteException;
}
```

```
package gpms.interfaces.rmi;

import gpms.util.Event;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface DatabaseEventDispatcher
    extends Remote
{
    public abstract void dispatch(Event event)
        throws RemoteException;
    public abstract void dispatch(String s)
        throws RemoteException;
}
```



```
package gpms.interfaces.rmi;

import gpms.client.output.ProjectOutput;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Vector;

public interface DatabaseInitializer
    extends Remote
{
    public abstract Vector getActivity(int i)
        throws RemoteException;
    public abstract Vector getPredActivity(int i)
        throws RemoteException;
    public abstract Vector getProjects()
        throws RemoteException;
    public abstract Vector getActivityOutput(int i, int j)
        throws RemoteException;
    public abstract ProjectOutput getProjectOutput(int i, int j)
        throws RemoteException;
}
```

```

package gpms.interfaces.rmi;

import gpms.client.gui.GameData;
import gpms.client.output.SimulationData;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Vector;

public interface Server
    extends Remote
{
    public abstract void attach(ClientEventHandler
        clienteventhandler) throws RemoteException;
    public abstract void createProject(String s, String s1, int i,
        int j, int k) throws RemoteException;
    public abstract void createRelationship(int i, int j, String s,
        int k, int l, String s1) throws RemoteException;
    public abstract void detach(ClientEventHandler
        clienteventhandler) throws RemoteException;
    public abstract void dispatch(String s)
        throws RemoteException;
    public abstract String findPredCode(int i)
        throws RemoteException;
    public abstract void gameSimulation(GameData gamedata,
        SimulationData simulationdata, int i, boolean flag) throws
        RemoteException;
    public abstract float getActivityFinalTime()
        throws RemoteException;
    public abstract float getActivityFinalTimeCPM()
        throws RemoteException;
    public abstract float getActivityFinalTimeWA()
        throws RemoteException;
    public abstract float getActivityStartTime()
        throws RemoteException;
    public abstract float getActivityStartTimeCPM()
        throws RemoteException;
    public abstract float getActivityStartTimeWA()
        throws RemoteException;
}

```

```

public abstract float[] getDataForGraph()
    throws RemoteException;
public abstract float[] getDataForGraph2()
    throws RemoteException;
public abstract float[] getDataForGraph3()
    throws RemoteException;
public abstract float[] getDataForGraphCPM()
    throws RemoteException;
public abstract float[] getDataForGraphCPM1()
    throws RemoteException;
public abstract float[] getDataForGraphCPM2()
    throws RemoteException;
public abstract float[] getDataForGraphProductivity()
    throws RemoteException;
public abstract float[] getDataForGraphProgress()
    throws RemoteException;
public abstract float[] getDataForGraphQuality()
    throws RemoteException;
public abstract float[] getDataForGraphWA()
    throws RemoteException;
public abstract float[] getDataForGraphWA1()
    throws RemoteException;
public abstract float[] getDataForGraphWA2()
    throws RemoteException;
public abstract float getFinalTime()
    throws RemoteException;
public abstract float getFinalTime2()
    throws RemoteException;
public abstract float getFinalTime3()
    throws RemoteException;
public abstract int getGameInterval()
    throws RemoteException;
public abstract Vector getPrimaveraActivity(int i)
    throws RemoteException;
public abstract Vector getPrimaveraProjectList()
    throws RemoteException;
public abstract Vector getPrimaveraRelationship(int i)
    throws RemoteException;

```

```

public abstract Vector getProjectList()
    throws RemoteException;
public abstract float[] getSimFile_data()
    throws RemoteException;
public abstract float getSimFile_finaltime()
    throws RemoteException;
public abstract float[] getSimFile_time()
    throws RemoteException;
public abstract int getStartTimeActivity(String s)
    throws RemoteException;
public abstract float[] getTimeForGraph()
    throws RemoteException;
public abstract float[] getTimeForGraph2()
    throws RemoteException;
public abstract float[] getTimeForGraph3()
    throws RemoteException;
public abstract float[] getTimeForGraphCPM()
    throws RemoteException;
public abstract float[] getTimeForGraphProductivity()
    throws RemoteException;
public abstract float[] getTimeForGraphProgress()
    throws RemoteException;
public abstract float[] getTimeForGraphQuality()
    throws RemoteException;
public abstract float[] getTimeForGraphWA()
    throws RemoteException;
public abstract void monteCarlo()
    throws RemoteException;
public abstract void policyMaker()
    throws RemoteException;
public abstract void setActivityFinalTime(float f)
    throws RemoteException;
public abstract void setActivityFinalTimeCPM(float f)
    throws RemoteException;
public abstract void setActivityFinalTimeWA(float f)
    throws RemoteException;
public abstract void setActivityProductivity(String s)
    throws RemoteException;

```

```

public abstract void setActivityProgress(String s)
    throws RemoteException;
public abstract void setActivityQuality(String s)
    throws RemoteException;
public abstract void setActivityStartTime(float f)
    throws RemoteException;
public abstract void setActivityStartTimeCPM(float f)
    throws RemoteException;
public abstract void setActivityStartTimeWA(float f)
    throws RemoteException;
public abstract void setCpm()
    throws RemoteException;
public abstract void setDataForGraph(float af[])
    throws RemoteException;
public abstract void setDataForGraph2(float af[])
    throws RemoteException;
public abstract void setDataForGraph3(float af[])
    throws RemoteException;
public abstract void setDataForGraphCPM1(float af[])
    throws RemoteException;
public abstract void setDataForGraphCPM2(float af[])
    throws RemoteException;
public abstract void setDataForGraphProductivity(float af[])
    throws RemoteException;
public abstract void setDataForGraphProgress(float af[])
    throws RemoteException;
public abstract void setDataForGraphQuality(float af[])
    throws RemoteException;
public abstract void setDataForGraphWA(float af[])
    throws RemoteException;
public abstract void setDataForGraphWA1(float af[])
    throws RemoteException;
public abstract void setDataForGraphWA2(float af[])
    throws RemoteException;
public abstract void setDpm()
    throws RemoteException;
public abstract void setDpmWoAction()
    throws RemoteException;

```

```

public abstract void setFinalTime(float f)
    throws RemoteException;
public abstract void setFinalTime2(float f)
    throws RemoteException;
public abstract void setFinalTime3(float f)
    throws RemoteException;
public abstract void setGameInterval(int i)
    throws RemoteException;
public abstract void setSimFile(String s)
    throws RemoteException;
public abstract void setSimFile_data(float af[])
    throws RemoteException;
public abstract void setSimFile_finaltime(float f)
    throws RemoteException;
public abstract void setSimFile_time(float af[])
    throws RemoteException;
public abstract void setTimeForGraph(float af[])
    throws RemoteException;
public abstract void setTimeForGraph2(float af[])
    throws RemoteException;
public abstract void setTimeForGraph3(float af[])
    throws RemoteException;
public abstract void setTimeForGraphCPM(float af[])
    throws RemoteException;
public abstract void setTimeForGraphProductivity(float af[])
    throws RemoteException;
public abstract void setTimeForGraphProgress(float af[])
    throws RemoteException;
public abstract void setTimeForGraphQuality(float af[])
    throws RemoteException;
public abstract void setTimeForGraphWA(float af[])
    throws RemoteException;
public abstract void simulation(SimulationData simulationdata,
    int i) throws RemoteException;
public abstract void skdlPlanner(SimulationData simulationdata)
    throws RemoteException;
public abstract float getActivityFinalTimePERT()
    throws RemoteException;

```

```

public abstract float getActivityStartTimePERT()
    throws RemoteException;
public abstract float[] getDataForGraph4()
    throws RemoteException;
public abstract float[] getDataForGraphCPM3()
    throws RemoteException;
public abstract float[] getDataForGraphPERT()
    throws RemoteException;
public abstract float[] getDataForGraphPERT1()
    throws RemoteException;
public abstract float[] getDataForGraphPERT2()
    throws RemoteException;
public abstract float[] getDataForGraphPERT3()
    throws RemoteException;
public abstract float[] getDataForGraphStability()
    throws RemoteException;
public abstract float[] getDataForGraphWA3()
    throws RemoteException;
public abstract float getFinalTime4()
    throws RemoteException;
public abstract float getPERTSigma()
    throws RemoteException;
public abstract float[] getTimeForGraph4()
    throws RemoteException;
public abstract float[] getTimeForGraphPERT()
    throws RemoteException;
public abstract float[] getTimeForGraphStability()
    throws RemoteException;
public abstract void setActivityFinalTimePERT(float f)
    throws RemoteException;
public abstract void setActivityStability(String s)
    throws RemoteException;
public abstract void setActivityStartTimePERT(float f)
    throws RemoteException;
public abstract void setDataForGraph4(float af[])
    throws RemoteException;
public abstract void setDataForGraphCPM3(float af[])
    throws RemoteException;

```

```

public abstract void setDataForGraphPERT(float af[])
    throws RemoteException;
public abstract void setDataForGraphPERT1(float af[])
    throws RemoteException;
public abstract void setDataForGraphPERT2(float af[])
    throws RemoteException;
public abstract void setDataForGraphPERT3(float af[])
    throws RemoteException;
public abstract void setDataForGraphStability(float af[])
    throws RemoteException;
public abstract void setDataForGraphWA3(float af[])
    throws RemoteException;
public abstract void setFinalTime4(float f)
    throws RemoteException;
public abstract void setPert()
    throws RemoteException;
public abstract void setPERTSigma(float f)
    throws RemoteException;
public abstract void setTimeForGraph4(float af[])
    throws RemoteException;
public abstract void setTimeForGraphPERT(float af[])
    throws RemoteException;
public abstract void setTimeForGraphStability(float af[])
    throws RemoteException;
public abstract void createActivity(String s, int i, String s1,
    String s2, String s3, int j, int k) throws RemoteException;
public abstract void createVersion(int i, int j)
    throws RemoteException;
public abstract int findDPMId(int i)
    throws RemoteException;
public abstract Vector getActivity(int i, int j)
    throws RemoteException;
public abstract int getNumOfVersion(int i)
    throws RemoteException;
public abstract Vector getVersionList(int i)
    throws RemoteException;
}

```



```

package gpms.client.gui;

import gpms.client.output.*;
import java.net.*;
import java.rmi.*;
import java.rmi.RemoteException;
import java.util.Date;
import gpms.util.Activity;
import gpms.util.*;
import gpms.interfaces.rmi.*;
import gpms.interfaces.impl.*;
import gpms.client.tablemodels.*;
import gpms.client.tablemodels.ActivityTableModel;
import java.util.Vector;
import java.applet.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.table.*;
import javax.swing.Icon;

public class Client extends javax.swing.JApplet implements
ActionListener {
    private javax.swing.JButton dsmButton;
    private javax.swing.JButton simulationButton;
    private javax.swing.JButton SDKLPlannerButton;
    private javax.swing.JButton policyMakerButton;
    private javax.swing.JButton monteCarloButton;
    private javax.swing.JButton exitButton;
    private javax.swing.JButton helpButton;
    private javax.swing.JButton trackButton;
    private javax.swing.JButton versionButton;
    private javax.swing.JLabel pIDLabel;
    private javax.swing.JLabel pNameLabel;
    private javax.swing.JLabel ifLabel;
    private javax.swing.JLabel pfLabel;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JButton newButton;

```

```

private javax.swing.JToolBar toolBar;
private java.awt.Image newImage;
private javax.swing.JButton openButton;
private java.awt.Image openImage;
private javax.swing.JButton saveButton;
private java.awt.Image saveImage;
private javax.swing.JButton cutButton;
private java.awt.Image cutImage;
private javax.swing.JButton deleteButton;
private java.awt.Image deleteImage;
private javax.swing.JButton refreshButton;
private java.awt.Image refreshImage;
private javax.swing.JTable table;
public java.util.Vector activityVector;
private java.awt.MediaTracker mediaTracker;
public ActivityChart activityChart;
private javax.swing.JButton clearButton;
private java.awt.Image clearImage;
private javax.swing.JButton outputSummaryButton;
private javax.swing.JButton propertyButton;
private java.awt.Image propertyImage;
private javax.swing.JButton menuHelpButton;
private javax.swing.JButton browseButton;
private java.awt.Image browseImage;
private java.awt.Image helpImage;
private java.awt.Image cuteImage;
private javax.swing.ListSelectionModel alsModel;
public gpms.interfaces.rmi.DatabaseInitializer
    databaseInitializer;
private gpms.interfaces.rmi.Server server;
public gpms.interfaces.rmi.DatabaseEventDispatcher
    databaseEventDispatcher;
public int projectId = -1;
public java.lang.String projectName;
public int versionID;
private javax.swing.JButton update;
private javax.swing.JButton insert;
private javax.swing.JButton delete;

```

```

private javax.swing.JLabel versionLabel;
public javax.swing.JTextField cpmField;
public javax.swing.JTextField pertField;
public javax.swing.JTextField dpmWoActionField;
public javax.swing.JTextField simOrNotField;
private javax.swing.JTextField code;
private javax.swing.JTextField name;
private javax.swing.JTextField duration;
private javax.swing.JTextField st;
private javax.swing.JTextField totalActivity;
NewProjectDialog newProjectDial;
public gpms.client.tablemodels.ActivityTableModel
    activityTableModel;
private gpms.interfaces.impl.DPMEventHandlerImpl
    dpmEventHandlerImpl;
private int selectedRow, selectedColumn;
private int numOfActivity, numOfActivity_1;
static boolean newProje = false;
static JTextField numOfActivityField;
int index;
private TestPrimavera primavera;
protected javax.swing.JComboBox versionCombo;
private Vector versionVector;
private javax.swing.JButton gameButton;
private javax.swing.JLabel titleLabel;
private javax.swing.JPanel titlePanel;
private javax.swing.JTextField wbs;
public void actionPerformed(ActionEvent e)
{
    if(e.getSource() instanceof JButton)
    {
        String str = e.getActionCommand();
        if(str.equals("outputSummary"))
        {
            try {
                server.setCpm();
                server.setPert();
                server.setDpmWoAction();
            }
        }
    }
}

```

```

        server.setDpm();
        activityOutput();
    } catch (Exception em){
    }
}
else if(str.equals("monteCarlo"))
{
    try
    {
        server.monteCarlo();
    }
    catch(RemoteException rm){}
}
else if(str.equals("track"))
{
    versionTrack();

}
else if(str.equals("showDSM"))
{
    supplementInfo((Activity) (activityVector.elementAt
(index)));
}
else if(str.equals("simulation"))
{
    new SimulationDialog(server, this, 1);
}
else if(str.equals("policyMaker"))
{
    try
    {
        server.policyMaker();
    }
    catch(RemoteException rm){}
}
else if(str.equals("game"))
{
    new GameDialog(server, this);
}

```

```

}
else if(str.equals("version"))
{
    try {
        versionID = versionCombo.getItemCount() + 1;
        databaseEventDispatcher.dispatch("UPDATE
DPM_PROJECT SET NUMOFVERSION = '" + versionID + "'
WHERE PROJID = '"+projectId+"'");
        server.createVersion(projectId, versionID);
        Vector copyVector = new Vector();
        copyVector = server.getActivity(projectId, 1);
        for (int i=0; i<copyVector.size(); i++) {
            databaseEventDispatcher.dispatch("INSERT INTO
DPM_ACTIVITY VALUES (SEQ_DPM_ACTIVITY.NEXTVAL, '" +
((Activity) (copyVector.elementAt(i))).getCode().to
String() + "', '" +
((Activity) (copyVector.elementAt(i))).getWbs().toS
tring() + "', '" +
((Activity) (copyVector.elementAt(i))).getName().to
String() + "', '" +
((Activity) (copyVector.elementAt(i))).getDuration(
) + "', '" +
((Activity) (copyVector.elementAt(i))).getST() +
"', '" +
((Activity) (copyVector.elementAt(i))).getFt() +
"', '" + projectId + "', '" +
((Activity) (copyVector.elementAt(i))).getDurationM
ostLikely() + "', '" +
((Activity) (copyVector.elementAt(i))).getDurationO
ptimistic()+ "', '" +
((Activity) (copyVector.elementAt(i))).getDurationP
essimistic() + "', '" +
((Activity) (copyVector.elementAt(i))).getReliabili
ty() + "', '" +
((Activity) (copyVector.elementAt(i))).getStability
() + "', '" +
((Activity) (copyVector.elementAt(i))).getComplexit
y() + "', '" +

```

```

        ((Activity) (copyVector.elementAt(i))).getContingen
cy() + "', '" +
        ((Activity) (copyVector.elementAt(i))).getBuffering
() + "', '" +
        ((Activity) (copyVector.elementAt(i))).getInternalS
ensitiviy() + "', '" +
        ((Activity) (copyVector.elementAt(i))).getIsCritica
lActivity() + "', '" +
        ((Activity) (copyVector.elementAt(i))).getPrimavera
Id() + "', '" +
        ((Activity) (copyVector.elementAt(i))).getSimSt() +
"', '" +
        ((Activity) (copyVector.elementAt(i))).getSimDurati
on() + "', '" +
        ((Activity) (copyVector.elementAt(i))).getSimOrNot(
) + "', '" + versionID + "')");
    }
    activityVector = server.getActivity(projectId,
        versionID);
    activityTableModel.listOfActivity =
        activityVector;
    activityChart.listOfActivity = activityVector;
    update();
    versionCombo.setSelectedIndex(versionID-1);
    Vector copyActivityVector = new Vector();
    copyActivityVector = activityVector;
    for (int j=0; j<copyVector.size(); j++) {
        Vector copyPredVector = new Vector();
        copyPredVector =
            databaseInitializer.getPredActivity( ((Activ
ity)copyVector.elementAt(j)).getId() );
        for (int m=0; m<copyPredVector.size(); m++)
        {
            int a1 =
                ((Precedence) copyPredVector.element
At(m)).getActivityId() -
                ((Precedence) copyPredVector.elementAt(
m)).getPredActivityId();

```

```

        int a2 =
            ((Activity)copyActivityVector.elementAt(j)).getId() - a1;
            ((Precedence)copyPredVector.elementAt(m)).setActivityId( ((Activity)copyActivityVector.elementAt(j)).getId() );
            ((Precedence)copyPredVector.elementAt(m)).setPredActivityId( a2 );
            server.createRelationship(((Precedence)
            copyPredVector.elementAt(m)).getActivityId(), ((Precedence)
            copyPredVector.elementAt(m)).getPredActivityId(), ((Precedence)
            copyPredVector.elementAt(m)).getRelation(), ((Precedence)
            copyPredVector.elementAt(m)).getLag(),
            1, ((Precedence)
            copyPredVector.elementAt(m)).getPredActivityCode() );
        }
    }
    catch (RemoteException rm) {
    }
}

else if(str.equals("help"))
{
}
else if(str.equals("exit"))
{
    System.exit(0);
}
else if(str.equals("openproject"))
{
    new OpenProjectDialog(server, this);
}

```

```

else if(str.equals("newproject"))
{
    new NewProjectDialog(server, this);
}
else if(str.equals("updateactivity"))
{
    try
    {
        String sname = name.getText();
        String swbs = wbs.getText();
        String scode = code.getText();
        int sduration =
        Integer.parseInt(duration.getText());
        int sst = Integer.parseInt(st.getText());
        int sft = sst + sduration;
        int index = alsModel.getMinSelectionIndex();
        setIndexForS(alsModel.getMinSelectionIndex());
        Activity activity =
        (Activity)(activityVector.elementAt(getIndexForS()
        ));
        int id = activity.getID();
        try
        {
            activityVector =
            server.getActivity(projectId, versionID);
            activityTableModel.listOfActivity =
            activityVector;
            activityChart.listOfActivity =
            activityVector;
            update();
        }
        catch(RemoteException rm)
        {
        }
        name.setText("");
        wbs.setText("");
        code.setText("");
        duration.setText("");
    }
}

```



```

        st.setText("");
    }
    catch(NumberFormatException nfe)
    {
    }
}
else if(str.equals("insertactivity"))
{
    try
    {
        String sname = name.getText();
        String swbs = wbs.getText();
        String scode = code.getText();
        int sduration =
        Integer.parseInt(duration.getText());
        int sst = Integer.parseInt(st.getText());
        int sft = sduration + sst;
        try
        {
            databaseEventDispatcher.dispatch("INSERT INTO
                DPM_ACTIVITY
                VALUES( SEQ_DPM_ACTIVITY.NEXTVAL,'" + scode +
                "','" + swbs + "','" + sname + "','" +
                sduration + "','" + sst + "','" + sft + "','"
                + projectId + "','" + sduration + "','" +
                sduration + "','" + sduration + "', 1, 1,
                1 ,0.2, 0, 0.5, 1, 0, 0, 0, 0, '" + versionID
                + "')");
            activityVector = server.getActivity(projectId,
                versionID);
            activityTableModel.listOfActivity =
                activityVector;
            activityChart.listOfActivity = activityVector;
            update();
        }
        catch(RemoteException rm)
        {
        }
    }
}

```

```

        name.setText("");
        code.setText("");
        wbs.setText("");
        duration.setText("");
        st.setText("");
    }
    catch(NumberFormatException nfe)
    {
    }
}
else if(str.equals("deleteactivity"))
{
    setIndexForS(alsModel.getMinSelectionIndex());
    int id =
        ((Activity) (activityVector.elementAt(getIndexForS(
        )))).getID();
    try
    {
        databaseEventDispatcher.dispatch("DELETE
        FROM DPM_ACTIVITY WHERE ID = '" + id + "'");
        databaseEventDispatcher.dispatch("DELETE
        FROM DPM_ACTIVITYRELATION WHERE ID = '" + id
        + "'");
        databaseEventDispatcher.dispatch("DELETE
        FROM DPM_ACTIVITYRELATION WHERE
        PREDACTIVITYID = '" + id + "'");
        activityVector =
        server.getActivity(projectId,
        versionID);
        activityTableModel.listOfActivity =
        activityVector;
        activityChart.listOfActivity =
        activityVector;
        update();
    }
    catch(RemoteException rm)
    {
    }
}

```

```

        name.setText("");
        wbs.setText("");
        code.setText("");
        duration.setText("");
        st.setText("");
    }
    else if(str.equals("skdlPlanner"))
    {
        new SimulationDialog(server, this, 2);
    }
}
}
public void activityOutput()
{
    try
    {
        if(projectId != -1)
        {
            Vector vector = new Vector();
            Vector sortVector = new Vector();
            sortVector =
            databaseInitializer.getActivityOutput(projectId,
            versionID);
            ActivityOutput sortArray[];
            sortArray = new ActivityOutput[sortVector.size()];
            for(int i=0; i<sortVector.size(); i++) {
                for(int j=1; j<sortVector.size()+1; j++) {
                    if( ((ActivityOutput)sortVector.elementAt
                    (i)).getCode().equals("a"+j) ) {
                        sortArray[j-1] =
                        (ActivityOutput)sortVector.elementA
                        t(i);
                    }
                }
            }
            for(int k=0; k<sortArray.length; k++) {
                vector.add(k, (ActivityOutput)sortArray[k]);
            }
        }
    }
}

```

```

        new ActivityOutputDialog(server,vector);
    }
}
catch(Exception e)
{
}
}
public void destroy() {
    super.destroy();
}
public int getActivityNum() {
    return numOfActivity;
}
public int getActivityNum_1() {
    return numOfActivity_1;
}
public String getAppletInfo() {
    return "Client\n" + "\n" + "Insert the type's description
here.\n" + "Creation date: (5/15/00 10:18:37 PM)\n" +
"@author: \n" +"";
}
public int getIndexForS() {
    return index;
}
public int getProjectIDForS() {
    return projectId;
}
public Vector getRowActivity() {
    Vector rowVector = new Vector();
    for(int i=0; i<activityTableModel.getRowCount(); i++) {
        rowVector.addElement(activityTableModel.getValueAt(i,0).toStri
ng());
    }
    return rowVector;
}
public int getSelectedColumn() {
    return selectedColumn;
}
}

```

```

public int getSelectedRow() {
    return selectedRow;
}
public void getServerObjects()
{
    try
    {
        URL base = getDocumentBase();
        String hostName = "cee-zzrt.cee.uiuc.edu";
        String port = "1099";
        server = (Server)Naming.lookup("rmi://" + hostName + ":" +
+ port + "/Server");
        databaseInitializer =
        (DatabaseInitializer)Naming.lookup("rmi://" + hostName +
+ ":" + port + "/DatabaseInitializer");
        databaseEventDispatcher =
        (DatabaseEventDispatcher)Naming.lookup("rmi://" +
        hostName + ":" + port + "/DatabaseEventDispatcher");
        dpmEventHandlerImpl = new DPMEventHandlerImpl(this);
        server.attach( (ClientEventHandler)dpmEventHandlerImpl );
    }
    catch(RemoteException rm)
    {
        rm.printStackTrace();
    }
    catch(NotBoundException nbe)
    {
    }
    catch(MalformedURLException murle)
    {
    }
}
public int getTotalActivityNumber() {
    return numOfActivity;
}
public void importProject()
{
    new ImportProjectDialog(server, this);
}

```

```

}
public void init() {
    super.init();
    getServerObjects();
    loadActivityData();
    mediaTracker = new MediaTracker(this);
    loadImage();
    versionVector = new Vector();
    makeButtons();
    makeFields();
    makeLabels();
    makeMenu();
    makeToolBar();
    makeCombo();
    if (projectName == null) {
        activityChart = new ActivityChart(activityVector, "");
    }
    else {
        activityChart = new ActivityChart(activityVector,
            projectName);
    }
    activityTableModel = new ActivityTableModel(activityVector);
    table = new JTable(activityTableModel);
    table.setShowGrid(true);
    TableColumn column = null;
    for (int i = 0; i < 6; i++) {
        column = table.getColumnModel().getColumn(i);
        if (i==0)
            column.setPreferredWidth(30);
        else if ( i==1 )
            column.setPreferredWidth(55);
        else if (i == 2) {
            column.setPreferredWidth(150);
        }
        else {
            column.setPreferredWidth(20);
        }
    }
}

```

```

setActivityNum_1(table.getRowCount());
table.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        int rowSelect = table.getSelectedRow();
        int columnSelect = table.getSelectedColumn();
        if (rowSelect == -1 || columnSelect == -1)
            return;
        else {
            setSelectedColumn(columnSelect);
            setSelectedRow(rowSelect);
            setActivityNum(table.getRowCount());
        }
    }
});
alsModel = table.getSelectionModel();
table.addMouseListener(new MouseAdapter()
{
    public void mouseClicked(MouseEvent e)
    {
        int index = alsModel.getMinSelectionIndex();
        Activity activity =
        (Activity) (activityVector.elementAt(index));
        name.setText(activity.getName());
        wbs.setText(activity.getWbs());
        code.setText(activity.getCode());
        duration.setText( (new
        Integer(activity.getDuration()).toString() );
        st.setText( (new
        Integer(activity.getSt()).toString());
        if(table.getSelectedColumn() == 3)
        {supplementInfo((Activity) (activityVector.elementA
        t(index)));
        }
    }
});
ButtonCellRenderer bcr = new ButtonCellRenderer();
columnModel = table.getColumnModel();
columnModel.getColumn(3).setCellRenderer(bcr);

```

```

setUI();
try
{
    dpmEventHandlerImpl = new DPMEventHandlerImpl(this);
    server.attach( (ClientEventHandler)dpmEventHandlerImpl);
}
catch(Exception rm)
{
}
getContentPane().setVisible(true);
}
public void loadActivityData() {
    try
    {
        activityVector = new Vector();
    }
    catch(Exception rm)
    {
    }
}
public void loadImage()
{
    openImage = getImage(getCodeBase(), "open.gif");
    deleteImage = getImage(getCodeBase(), "delete.gif");
    cutImage = getImage(getCodeBase(), "cut.gif");
    newImage = getImage(getCodeBase(), "new.gif");
    refreshImage = getImage(getCodeBase(), "refresh.gif");
    saveImage = getImage(getCodeBase(), "save.gif");
    clearImage = getImage(getCodeBase(), "clear.gif");
    propertyImage = getImage(getCodeBase(), "property.gif");
    browseImage = getImage(getCodeBase(), "browse.gif");
    helpImage = getImage(getCodeBase(), "help.gif");
    cuteImage = getImage(getCodeBase(), "cute_1.jpg");
    mediaTracker.addImage(openImage, 0);
    mediaTracker.addImage(cutImage, 0);
    mediaTracker.addImage(deleteImage, 0);
    mediaTracker.addImage(newImage, 0);
    mediaTracker.addImage(refreshImage, 0);
    mediaTracker.addImage(saveImage, 0);
}

```



```

mediaTracker.addImage(clearImage, 0);
mediaTracker.addImage(propertyImage, 0);
mediaTracker.addImage(helpImage, 0);
mediaTracker.addImage(browseImage, 0);
mediaTracker.addImage(cuteImage, 0);
try
{
    mediaTracker.waitForAll();
}
catch(InterruptedException ie)
{}
}
public static void main(String[] args)
{
}
public void makeButtons()
{
    dsmButton = new JButton("DSM");
    dsmButton.setVerticalTextPosition(AbstractButton.BOTTOM);
    dsmButton.setHorizontalTextPosition(AbstractButton.CENTER);
    dsmButton.setActionCommand("showDSM");
    dsmButton.setBackground(new Color(208,128,100));
    dsmButton.setForeground(Color.black);
    dsmButton.setEnabled(true);
    dsmButton.addActionListener(this);
    dsmButton.setBorderPainted(true);
    trackButton = new JButton("PROJECT TRACKING");
    trackButton.setVerticalTextPosition(AbstractButton.BOTTOM);
    trackButton.setHorizontalTextPosition(AbstractButton.CENTER);
    trackButton.setActionCommand("track");
    trackButton.setBackground(new Color(208,128,100));
    trackButton.setForeground(Color.black);
    trackButton.setEnabled(true);
    trackButton.addActionListener(this);
    trackButton.setBorderPainted(true);
    simulationButton = new JButton("SIMULATION INPUT");
    simulationButton.setVerticalTextPosition(AbstractButton.BOTTOM
);

```

```

simulationButton.setHorizontalTextPosition (AbstractButton.CENT
ER);
simulationButton.setActionCommand("simulation");
simulationButton.setBackground(new Color(208,128,100));
simulationButton.setForeground(Color.black);
simulationButton.setEnabled(true);
simulationButton.addActionListener(this);
simulationButton.setBorderPainted(true);
SDKLPlannerButton = new JButton("SDKL Planner");
SDKLPlannerButton.setVerticalTextPosition (AbstractButton.BOTTO
M);
SDKLPlannerButton.setHorizontalTextPosition (AbstractButton.CEN
TER);
SDKLPlannerButton.setActionCommand("skdlPlanner");
SDKLPlannerButton.setBackground(new Color(208,128,100));
SDKLPlannerButton.setForeground(Color.black);
SDKLPlannerButton.setEnabled(true);
SDKLPlannerButton.addActionListener(this);
SDKLPlannerButton.setBorderPainted(true);
gameButton = new JButton("GAME");
gameButton.setVerticalTextPosition (AbstractButton.BOTTOM);
gameButton.setHorizontalTextPosition (AbstractButton.CENTER);
gameButton.setActionCommand("game");
gameButton.setBackground(new Color(208,128,100));
gameButton.setForeground(Color.black);
gameButton.setEnabled(true);
gameButton.addActionListener(this);
gameButton.setBorderPainted(true);
exitButton = new JButton("EXIT");
exitButton.setVerticalTextPosition (AbstractButton.BOTTOM);
exitButton.setHorizontalTextPosition (AbstractButton.CENTER);
exitButton.setActionCommand("exit");
exitButton.setBackground(new Color(208,128,100));
exitButton.setForeground(Color.black);
exitButton.setEnabled(true);
exitButton.addActionListener(this);
exitButton.setBorderPainted(true);
outputSummaryButton = new JButton("SIMULATION ANALYSIS");

```

```

outputSummaryButton.setVerticalTextPosition(AbstractButton.BOTTOM);
outputSummaryButton.setHorizontalTextPosition(AbstractButton.CENTER);
outputSummaryButton.setActionCommand("outputSummary");
outputSummaryButton.setBackground(new Color(208,128,100));
outputSummaryButton.setForeground(Color.black);
outputSummaryButton.setEnabled(true);
outputSummaryButton.addActionListener(this);
outputSummaryButton.setBorderPainted(true);
versionButton = new JButton("CREATE VERSION");
versionButton.setVerticalTextPosition(AbstractButton.BOTTOM);
versionButton.setHorizontalTextPosition(AbstractButton.CENTER);
;
versionButton.setActionCommand("version");
versionButton.setBackground(new Color(208,128,100));
versionButton.setForeground(Color.black);
versionButton.setEnabled(false);
versionButton.addActionListener(this);
versionButton.setBorderPainted(true);
helpButton = new JButton("Help");
helpButton.setVerticalTextPosition(AbstractButton.BOTTOM);
helpButton.setHorizontalTextPosition(AbstractButton.CENTER);
helpButton.setActionCommand("help");
helpButton.setBackground(Color.lightGray);
helpButton.setForeground(Color.black);
helpButton.setEnabled(true);
helpButton.addActionListener(this);
helpButton.setBorderPainted(true);
insert = new JButton("Add");
insert.setVerticalTextPosition(AbstractButton.BOTTOM);
insert.setHorizontalTextPosition(AbstractButton.CENTER);
insert.setActionCommand("insertactivity");
insert.setBackground(new Color(217,203,100));
insert.setForeground(Color.black);
insert.setEnabled(true);
insert.addActionListener(this);
insert.setBorderPainted(true);

```

```

delete = new JButton("Delete");
delete.setVerticalTextPosition(AbstractButton.BOTTOM);
delete.setHorizontalTextPosition(AbstractButton.CENTER);
delete.setActionCommand("deleteactivity");
delete.setBackground(new Color(217,203,100));
delete.setForeground(Color.black);
delete.setEnabled(true);
delete.addActionListener(this);
delete.setBorderPainted(true);
update = new JButton("Update");
update.setVerticalTextPosition(AbstractButton.BOTTOM);
update.setHorizontalTextPosition(AbstractButton.CENTER);
update.setActionCommand("updateactivity");
update.setBackground(new Color(217,203,100));
update.setForeground(Color.black);
update.setEnabled(true);
update.addActionListener(this);
update.setBorderPainted(true);
}
public void makeCombo() {
    versionCombo = new JComboBox(versionVector);
    versionCombo.setBackground(Color.white);
    versionCombo.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e)
        {
            int s =
                Integer.parseInt(versionCombo.getSelectedItem().toString());
            try {
                activityVector = server.getActivity(projectId, s);
            } catch (RemoteException re) {
            }
            activityTableModel.listOfActivity = activityVector;
            activityChart.listOfActivity = activityVector;
            numOfActivityField.setText((new
                Integer(activityVector.size())).toString());
            versionID = s;
            update();
        }
    });
}

```

```

        }
    });
}

public void makeFields()
{
    code = new JTextField();
    wbs = new JTextField();
    duration = new JTextField();
    name = new JTextField();
    st = new JTextField();
    totalActivity = new JTextField();
    pertField = new JTextField();
    dpmWoActionField = new JTextField();
    simOrNotField = new JTextField();
}

public void makeLabels()
{
    pIDLabel = new JLabel("Project ID",JLabel.CENTER);
    pIDLabel.setVerticalTextPosition(JLabel.BOTTOM);
    pIDLabel.setHorizontalTextPosition(JLabel.CENTER);
    pNameLabel = new JLabel("Project Name",JLabel.CENTER);
    pNameLabel.setVerticalTextPosition(JLabel.BOTTOM);
    pNameLabel.setHorizontalTextPosition(JLabel.CENTER);
    ifLabel = new JLabel("Initial Finish",JLabel.CENTER);
    ifLabel.setVerticalTextPosition(JLabel.BOTTOM);
    ifLabel.setHorizontalTextPosition(JLabel.CENTER);
    pfLabel = new JLabel("Planned Finish",JLabel.CENTER);
    pfLabel.setVerticalTextPosition(JLabel.BOTTOM);
    pfLabel.setHorizontalTextPosition(JLabel.CENTER);
}

public void makeMenu()
{
    menuBar = new JMenuBar();
    this.setJMenuBar(menuBar);
    JMenu menuFile = new JMenu("File");
    JMenuItem newProject = new JMenuItem("New Project");
    newProject.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)

```

```

        {
            newProject();
        }
    });
    menuFile.add(newProject);
    JMenuItem openProject = new JMenuItem("Open Project");
    openProject.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            openProject();
        }
    });
    menuFile.add(openProject);
    JMenuItem importFile = new JMenuItem("Import Project");
    importFile.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            importProject();
        }
    });
    menuFile.add(importFile);
    JMenuItem exportFile = new JMenuItem("Export Project");
    exportFile.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
        }
    });
    menuFile.add(exportFile);
    JMenuItem saveFile = new JMenuItem("Save");
    saveFile.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
        }
    });
    });

```

```

menuFile.add(saveFile);
JMenuItem saveAsFile = new JMenuItem("Save As");
saveAsFile.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
    }
});
menuFile.add(saveAsFile);
menuFile.addSeparator();
JMenuItem exitFile = new JMenuItem("Exit");
exitFile.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        System.exit(0);
    }
});
menuFile.add(exitFile);
JMenu menuView = new JMenu("View");
JMenuItem calendar = new JMenuItem("Calendar");
calendar.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
    }
});
menuView.add(calendar);
JMenuItem pertChart = new JMenuItem("PERT Chart");
pertChart.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
    }
});
menuView.add(pertChart);
JMenuItem ganttChart = new JMenuItem("Gantt Chart");
ganttChart.addActionListener(new ActionListener()

```

```

{
    public void actionPerformed(ActionEvent e)
    {
    }
});
menuView.add(ganttChart);
JMenuItem projectOutput = new JMenuItem("ProjectOutput");
projectOutput.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        projectOutput();
    }
});
menuView.add(projectOutput);
JMenuItem activityOutput = new JMenuItem("ActivityOutput");
activityOutput.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        activityOutput();
    }
});
menuView.add(activityOutput);
JMenu menuTool = new JMenu("Tools");
JMenuItem resources = new JMenuItem("Resources");
resources.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
    }
});
menuTool.add(resources);
JMenuItem filter = new JMenuItem("Filter");
filter.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {

```



```

        }
    });
    menuTool.add(filter);
    JMenuItem property = new JMenuItem("Property");
    property.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
        }
    });
    property.add(resources);
    JMenu menuHelp = new JMenu("Help");
    JMenuItem index = new JMenuItem("Index");
    JMenuItem search = new JMenuItem("Search");
    JMenuItem gettingStarted = new JMenuItem("Getting Started");
    JMenuItem aboutDpm = new JMenuItem("About DPM");
    menuHelp.add(index);
    menuHelp.add(search);
    menuHelp.add(gettingStarted);
    menuHelp.addSeparator();
    menuHelp.add(aboutDpm);
    menuBar.add(menuFile);
    menuBar.add(menuView);
    menuBar.add(menuTool);
    menuBar.add(menuHelp);
}
public void makeToolBar()
{
    toolBar = new JToolBar();
    toolBar.setLayout(new GridLayout(1,8));
    newButton = new JButton(new ImageIcon(newImage));
    newButton.setVerticalTextPosition(AbstractButton.BOTTOM);
    newButton.setHorizontalTextPosition(AbstractButton.CENTER);
    newButton.setActionCommand("new");
    newButton.setEnabled(true);
    newButton.addActionListener(this);
    newButton.setBorderPainted(true);
    newButton.setToolTipText("new");
}

```

```

toolBar.add(newButton);
openButton = new JButton(new ImageIcon(openImage));
openButton.setVerticalTextPosition(AbstractButton.BOTTOM);
openButton.setHorizontalTextPosition(AbstractButton.CENTER);
openButton.setActionCommand("open");
openButton.setEnabled(true);
openButton.addActionListener(this);
openButton.setBorderPainted(true);
openButton.setToolTipText("open");
toolBar.add(openButton);
propertyButton = new JButton(new ImageIcon(propertyImage));
propertyButton.setVerticalTextPosition(AbstractButton.BOTTOM);
propertyButton.setHorizontalTextPosition(AbstractButton.CENTER);
propertyButton.setActionCommand("property");
propertyButton.setEnabled(true);
propertyButton.addActionListener(this);
propertyButton.setBorderPainted(true);
propertyButton.setToolTipText("property");
toolBar.add(propertyButton);
saveButton = new JButton(new ImageIcon(saveImage));
saveButton.setVerticalTextPosition(AbstractButton.BOTTOM);
saveButton.setHorizontalTextPosition(AbstractButton.CENTER);
saveButton.setActionCommand("save");
saveButton.setEnabled(true);
saveButton.addActionListener(this);
saveButton.setBorderPainted(true);
saveButton.setToolTipText("save");
toolBar.add(saveButton);
cutButton = new JButton(new ImageIcon(cutImage));
cutButton.setVerticalTextPosition(AbstractButton.BOTTOM);
cutButton.setHorizontalTextPosition(AbstractButton.CENTER);
cutButton.setActionCommand("cut");
cutButton.setEnabled(true);
cutButton.addActionListener(this);
cutButton.setBorderPainted(true);
cutButton.setToolTipText("cut");
toolBar.add(cutButton);

```

```

deleteButton = new JButton(new ImageIcon(deleteImage));
deleteButton.setVerticalTextPosition(AbstractButton.BOTTOM);
deleteButton.setHorizontalTextPosition(AbstractButton.CENTER);
deleteButton.setActionCommand("delete");
deleteButton.setEnabled(true);
deleteButton.addActionListener(this);
deleteButton.setBorderPainted(true);
deleteButton.setToolTipText("Delete");
toolBar.add(deleteButton);

clearButton = new JButton(new ImageIcon(clearImage));
clearButton.setVerticalTextPosition(AbstractButton.BOTTOM);
clearButton.setHorizontalTextPosition(AbstractButton.CENTER);
clearButton.setActionCommand("clear");
clearButton.setEnabled(true);
clearButton.addActionListener(this);
clearButton.setBorderPainted(true);
clearButton.setToolTipText("Clear");
toolBar.add(clearButton);

refreshButton = new JButton(new ImageIcon(refreshImage));
refreshButton.setVerticalTextPosition(AbstractButton.BOTTOM);
refreshButton.setHorizontalTextPosition(AbstractButton.CENTER);
;
refreshButton.setActionCommand("refresh");
refreshButton.setBackground(Color.lightGray);
refreshButton.setForeground(Color.black);
refreshButton.setEnabled(true);
refreshButton.addActionListener(this);
refreshButton.setBorderPainted(true);
refreshButton.setToolTipText("Refresh Canvas");
toolBar.add(refreshButton);

browseButton = new JButton(new ImageIcon(browseImage));
browseButton.setVerticalTextPosition(AbstractButton.BOTTOM);
browseButton.setHorizontalTextPosition(AbstractButton.CENTER);
browseButton.setActionCommand("browse");
browseButton.setBackground(Color.lightGray);
browseButton.setForeground(Color.black);
browseButton.setEnabled(true);
browseButton.addActionListener(this);

```

```

        browseButton.setBorderPainted(true);
        browseButton.setToolTipText("Browse");
        toolBar.add(browseButton);
        menuHelpButton = new JButton(new ImageIcon(helpImage));
        menuHelpButton.setVerticalTextPosition(AbstractButton.BOTTOM);
        menuHelpButton.setHorizontalTextPosition(AbstractButton.CENTER);
        menuHelpButton.setActionCommand("menuHelp");
        menuHelpButton.setBackground(Color.lightGray);
        menuHelpButton.setForeground(Color.black);
        menuHelpButton.setEnabled(true);
        menuHelpButton.addActionListener(this);
        menuHelpButton.setBorderPainted(true);
        menuHelpButton.setToolTipText("Help");
        toolBar.add(menuHelpButton);
    }
    public void newProject()
    {
        new NewProjectDialog(server, this);
    }
    public void openProject()
    {
        new OpenProjectDialog(server, this);
    }
    public void paint(Graphics g) {
        super.paint(g);
    }
    public void projectOutput()
    {
        try
        {
            if(projectId != -1)
            {
                ProjectOutput projectOutput =
                    databaseInitializer.getProjectOutput(projectId,
                    versionID);
                if(projectOutput!=null)
                {

```

```

        new ProjectOutputDialog(projectOutput);
    }
}
catch(Exception e)
{
}
}
public void setActivityNum() {
    if(newProje == true) {
        setTotalActivityNumber(Integer.parseInt((numOfActivityField).getText()));
        newProjectDial.newProj = false;
    }
    if(newProje == false) {
        numOfActivityField = new JTextField("Default");
    }
}
void setActivityNum(int num) {
    numOfActivity = num;
}
public void setActivityNum_1(int num) {
    numOfActivity_1 = num;
}
public void setIndexForS(int in) {
    index = in;
}
void setSelectedColumn(int col) {
    selectedColumn = col;
}
void setSelectedRow(int row) {
    selectedRow = row;
}
public void setTotalActivityNumber(int num) {
    numOfActivity = num;
}
public void setUI()
{

```

```

this.setJMenuBar(menuBar);
JScrollPane scrollTableActivity = new JScrollPane(table);
JScrollPane scrollDisplayActivity = new
JScrollPane(activityChart);
JSplitPane splitPane = new
JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
JPanel tableContainer = new JPanel(new GridLayout(1,1));
tableContainer.setBorder(BorderFactory.createCompoundBorder(Bo
rderFactory.createLineBorder(Color.black),
BorderFactory.createTitledBorder("ACTIVITY TABLE")));
tableContainer.add(scrollTableActivity);
tableContainer.setPreferredSize(new Dimension(350,150));
splitPane.add(tableContainer);
JPanel chartContainer = new JPanel(new GridLayout(1,1));
chartContainer.setBorder(BorderFactory.createCompoundBorder(Bo
rderFactory.createLineBorder(Color.black),
BorderFactory.createTitledBorder("ACTIVITY CHART")));
titleLabel = new JLabel();
titleLabel.setFont(new Font("Serif", Font.BOLD + Font.ITALIC,
18));
titleLabel.setForeground(Color.white);
titleLabel = new JPanel();
titleLabel.setSize(200,30);
titleLabel.add(titleLabel);
chartContainer.setLayout(new BorderLayout());
chartContainer.add(titlePanel, BorderLayout.NORTH);
chartContainer.add(scrollDisplayActivity,
BorderLayout.CENTER);
chartContainer.setPreferredSize(new Dimension(200,150));
splitPane.add(chartContainer);
numOfActivityField = new JTextField("No selection");
JPanel totalActPanel = new JPanel();
totalActPanel.setBorder(BorderFactory.createTitledBorder("Tota
lActivity"));
totalActPanel.setLayout(new GridLayout(1,1));
totalActPanel.add(numOfActivityField);
cpmPanel.setBorder(BorderFactory.createTitledBorder("CPM

```

```

Duration"));
cpmPanel.setLayout(new GridLayout(1,1));
cpmPanel.add(cpmField);
JPanel pertPanel = new JPanel();
pertPanel.setBorder(BorderFactory.createTitledBorder("PERT
Duration"));
pertPanel.setLayout(new GridLayout(1,1));
pertPanel.add(pertField);
JPanel dpmWoActionPanel = new JPanel();
dpmWoActionPanel.setBorder(BorderFactory.createTitledBorder("D
PM W/O Action"));
dpmWoActionPanel.setLayout(new GridLayout(1,1));
dpmWoActionPanel.add(dpmWoActionField);
JPanel buttonPanel = new JPanel();
buttonPanel.setBorder(BorderFactory.createCompoundBorder (Borde
rFactory.createLineBorder(Color.white),
BorderFactory.createTitledBorder("CONTROLS")));
buttonPanel.setLayout(new GridLayout(7,1));
buttonPanel.add(totalActPanel);
buttonPanel.add(pertPanel);
buttonPanel.add(dpmWoActionPanel);
buttonPanel.add(dsmButton);
buttonPanel.add(simulationButton);
buttonPanel.add(outputSummaryButton);
buttonPanel.add(gameButton);
buttonPanel.add(trackButton);
buttonPanel.add(versionButton);
JPanel aPanel = new JPanel();
aPanel.setLayout(new GridLayout(1,11));
JPanel codePanel = new JPanel();
codePanel.setBorder(BorderFactory.createTitledBorder("Code"));
codePanel.setLayout(new GridLayout(1,1));
codePanel.add(code);
JPanel wbsPanel = new JPanel();
wbsPanel.setBorder(BorderFactory.createTitledBorder("WBS"));
wbsPanel.setLayout(new GridLayout(1,1));
wbsPanel.add(wbs);
JPanel durationPanel = new JPanel();

```

```

durationPanel.setBorder(BorderFactory.createTitledBorder("Duration"));
durationPanel.setLayout(new GridLayout(1,1));
durationPanel.add(duration);
JPanel namePanel = new JPanel();
namePanel.setBorder(BorderFactory.createTitledBorder("Name"));
namePanel.setLayout(new GridLayout(1,1));
namePanel.add(name);
JPanel stPanel = new JPanel();
stPanel.setBorder(BorderFactory.createTitledBorder("ST"));
stPanel.setLayout(new GridLayout(1,1));
stPanel.add(st);
newProjectDial.timeLabel = new JLabel(newProjectDial.times[1],
JLabel.CENTER);
newProjectDial.timeLabel.setVerticalTextPosition(JLabel.BOTTOM);
newProjectDial.timeLabel.setHorizontalTextPosition(JLabel.CENTER);
JPanel timePanel = new JPanel();
timePanel.setBorder(BorderFactory.createTitledBorder("Time Unit"));
timePanel.setLayout(new BorderLayout());
timePanel.add(newProjectDial.timeLabel, "Center");
JPanel versionPanel = new JPanel();
versionPanel.setBorder(BorderFactory.createTitledBorder("Version"));
versionPanel.setLayout(new GridLayout(1,1));
versionLabel = new JLabel("");
versionLabel.setVerticalTextPosition(JLabel.BOTTOM);
versionLabel.setHorizontalTextPosition(JLabel.CENTER);
versionPanel.add(versionCombo);
JPanel cutePanel = new JPanel();
JButton cuteButton = new JButton(new ImageIcon(cuteImage));
cuteButton.setVerticalTextPosition(AbstractButton.BOTTOM);
cuteButton.setHorizontalTextPosition(AbstractButton.CENTER);
cuteButton.setEnabled(true);
cutePanel.setLayout(new GridLayout(1,1));
cutePanel.add(cuteButton);

```



```

aPanel.add(versionPanel);
aPanel.add(codePanel);
aPanel.add(wbsPanel);
aPanel.add(namePanel);
aPanel.add(durationPanel);
aPanel.add(stPanel);
aPanel.add(timePanel);
aPanel.add(insert);
aPanel.add(delete);
aPanel.add(update);
aPanel.add(cutePanel);
JPanel mainPanel = new JPanel();
mainPanel.setLayout(new BorderLayout());
mainPanel.add(BorderLayout.NORTH, toolBar);
mainPanel.add(BorderLayout.CENTER, splitPane);
mainPanel.add(BorderLayout.WEST, buttonPanel);
mainPanel.add(BorderLayout.SOUTH, aPanel);
getContentPane().setLayout(new GridLayout(1,1));
getContentPane().add(mainPanel);
}
public void start() {
    super.start();
}
public void stop() {
    super.stop();
}
public void summaryOutput()
{
    try
    {
        if(projectId != -1)
        {
            Vector vector = new Vector();
            vector =
            databaseInitializer.getActivityOutput(projectId,
            versionID);
            new SummaryOutputDialog(server, vector);
        }
    }
}

```

```

    }
    catch(Exception e)
    {
    }
}
public void supplementInfo(Activity activity)
{
    SupplementDataDialog sup =new SupplementDataDialog(server,
    this, activity, activityVector);
    sup.setSize(975,750);
    sup.show();
}
public void update()
{
    ((ActivityTableModel)table.getModel()).fireTableDataChanged();
    titlePanel.setBackground(Color.black);
    titleLabel.setText(projectName + " Project");
    activityChart.setProjectTitle(projectName);
    activityChart.repaint();
    numOfActivityField.setText((new
    Integer(activityVector.size())).toString());
    versionLabel.setText(new Integer(versionID).toString());
    versionVector.removeAllElements();
    try {
        for(int i = 0; i < server.getNumOfVersion(projectId);
        i++) {
            versionVector.add(new Integer(i + 1));
        }
    } catch (RemoteException e) {
    }
}
public void updateTable(String codeName, String dur, int id) {
    for(int i=0; i<table.getRowCount(); i++) {
        if(table.getValueAt(i,0).toString().equals(codeName)) {
            try {
                duration.setText(dur);
                databaseEventDispatcher.dispatch("UPDATE

```

```

        DPM_ACTIVITY SET DURATION='"+dur+"'WHERE
        ID='"+id+"'");
        activityVector =
        databaseInitializer.getActivity(projectId);
        activityTableModel.listOfActivity =
        activityVector;
        activityChart.listOfActivity =
        activityVector;
        update();
    } catch (Exception tt) {
    }
}
}
}
public void updateTableInSupple(int projectId) {
    try {
        activityVector =
        databaseInitializer.getActivity(projectId);
        activityTableModel.listOfActivity = activityVector;
        activityChart.listOfActivity = activityVector;
        update();
    } catch (Exception y) {
    }
}
public void versionTrack()
{
    try
    {
        if(projectId != -1)
        {
            Vector vector = new Vector();
            Vector sortVector = new Vector();
            sortVector =
            databaseInitializer.getActivityOutput(projectId,
            1);
            ActivityOutput sortArray[];
            sortArray = new ActivityOutput[sortVector.size()];
            for(int i=0; i<sortVector.size(); i++) {

```

```

        for(int j=1; j<sortVector.size()+1; j++) {
            if( ((ActivityOutput) sortVector.elementAt
                (i)).getCode().equals("a"+j) ) {
                sortArray[j-1] =
                    (ActivityOutput) sortVector.elementAt(i
                    );
            }
        }
    }
    for(int k=0; k<sortArray.length; k++) {
        vector.add(k, (ActivityOutput) sortArray[k]);
    }
    new TrackingProjectDialog(server,
        databaseInitializer, this, vector, projectId,
        projectName);
}
}
catch(Exception e)
{
}
}
}

```

```

package gpms.client.output;

import java.io.Serializable;

public class ActivityOutput implements Serializable
{
    private int preBufferStart;
    private int preBufferSize;
    private int plannedPhaseStart;
    private int plannedPhaseDuration;
    private int postBufferStart;
    private int postBufferSize;
    private int reworkRate;
    private int resourceUtilizationRate;
    private int prodyLoss;
    private java.lang.String code;
    private int id;
    private int cpmSt;
    private int cpmDuration;
    private int woSt;
    private int woDuration;
    public java.lang.String getCode() {
        return code;
    }
    public int getCpmDuration() {
        return cpmDuration;
    }
    public int getCpmSt() {
        return cpmSt;
    }
    public int getId() {
        return id;
    }
    public int getPlannedPhaseDuration() {
        return plannedPhaseDuration;
    }
    public int getPlannedPhaseStart() {
        return plannedPhaseStart;
    }
}

```

```

}
public int getPostBufferSize() {
    return postBufferSize;
}
public int getPostBufferStart() {
    return postBufferStart;
}
public int getPreBufferSize() {
    return preBufferSize;
}
public int getPreBufferStart() {
    return preBufferStart;
}
public int getProdyLoss() {
    return prodyLoss;
}
public int getResourceUtilizationRate() {
    return resourceUtilizationRate;
}
public int getReworkRate() {
    return reworkRate;
}
public int getWoDuration() {
    return woDuration;
}
public int getWoSt() {
    return woSt;
}
public void setCode(java.lang.String newCode) {
    code = newCode;
}
public void setId(int newId) {
    id = newId;
}
public void setPlannedPhaseDuration(int newPlannedPhaseDuration) {
    plannedPhaseDuration = newPlannedPhaseDuration;
}
public void setPlannedPhaseStart(int newPlannedPhaseStart) {

```

```

        plannedPhaseStart = newPlannedPhaseStart;
    }
    public void setPostBufferSize(int newPostBufferSize) {
        postBufferSize = newPostBufferSize;
    }
    public void setPostBufferStart(int newPostBufferStart) {
        postBufferStart = newPostBufferStart;
    }
    public void setPreBufferSize(int newPreBufferSize) {
        preBufferSize = newPreBufferSize;
    }
    public void setPreBufferStart(int newPreBufferStart) {
        preBufferStart = newPreBufferStart;
    }
    public void setProdyLoss(int newProdyLoss) {
        prodyLoss = newProdyLoss;
    }
    public void setResourceUtilizationRate(int
newResourceUtilizationRate) {
        resourceUtilizationRate = newResourceUtilizationRate;
    }
    public void setReworkRate(int newReworkRate) {
        reworkRate = newReworkRate;
    }
    private int pertDuration;
    private int pertSt;
    private double secondSize;
    private double secondStart;
    private String wbs;
    public ActivityOutput(int id, String code, int preBufferStart, int
preBufferSize, int plannedPhaseStart, int plannedPhaseDuration, int
postBufferStart, int postBufferSize, int reworkRate, int
resourceUtilizationRate, int prodyLoss, int cpmSt, int cpmDuration,
int woSt, int woDuration, String wbs, double secondStart, double
secondSize, int pertSt, int pertDuration)
    {
        super();
        this.id = id;

```

```

        this.code = code;
        this.preBufferStart = preBufferStart;
        this.preBufferSize = preBufferSize;
        this.plannedPhaseStart = plannedPhaseStart;
        this.plannedPhaseDuration = plannedPhaseDuration;
        this.postBufferStart = postBufferStart;
        this.postBufferSize = postBufferSize;
        this.reworkRate = reworkRate;
        this.resourceUtilizationRate = resourceUtilizationRate;
        this.prodyLoss = prodyLoss;
        this.cpmSt = cpmSt;
        this.cpmDuration = cpmDuration;
        this.woSt = woSt;
        this.woDuration = woDuration;
        this.wbs = wbs;
        this.secondStart = secondStart;
        this.secondSize = secondSize;
        this.pertSt = pertSt;
        this.pertDuration = pertDuration;
    }
    public int getPertDuration() {
        return pertDuration;
    }
    public int getPertSt() {
        return pertSt;
    }
    public double getSecondSize() {
        return secondSize;
    }
    public double getSecondStart() {
        return secondStart;
    }
    public java.lang.String getWbs() {
        return wbs;
    }
    public void setSecondSize(double ss) {
        secondSize = ss;
    }
}

```



```
public void setSecondStart(double ss) {  
    secondStart = ss;  
}  
public void setWbs(java.lang.String newWbs) {  
    wbs = newWbs;  
}  
}
```

```

package gpms.client.output;

import gpms.client.gui.*;
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.util.Vector;
import gpms.server.*;
import gpms.interfaces.rmi.*;
import java.util.*;
import gpms.client.tablemodels.*;import javax.swing.table.*;/**

public class ActivityOutputDialog extends javax.swing.JDialog
implements
    java.awt.event.ActionListener {
    private javax.swing.JButton okayButton;
    private javax.swing.JButton compareButton;
    private ActivityOutput activityOutput;
    private java.util.Vector vector;
    private gpms.interfaces.impl.VensimGraph vensimGraph;
    private OutputCanvas oc;
    private Graph graph;
    private float[] tval;
    private float[] rval;
    private gpms.interfaces.rmi.Server server;
    private gpms.client.output.DrawGraph dGraph;
    private Vector outputVector;
    public ActivityOutputDialog(Server server, Vector vector) {
        super();
        this.vector = vector;
        this.server = server;
        makeButtons();
        setUI();
    }
    public void actionPerformed(java.awt.event.ActionEvent e)
    {
        if(e.getSource() instanceof JButton)
        {

```

```

        String str = e.getActionCommand();
        if(str.equals("okay"))
        {
            setVisible(false);
            dispose();
        }
        else if(str.equals("compare"))
        {
            new CompareDialog(server, vector);
        }
    }
}
public Vector getVector() {
    return outputVector;
}
public void makeButtons()
{
    okayButton = new JButton("EXIT");
    okayButton.setVerticalTextPosition(AbstractButton.BOTTOM);
    okayButton.setHorizontalTextPosition(AbstractButton.CENTER);
    okayButton.setActionCommand("okay");
    okayButton.setBackground(Color.lightGray);
    okayButton.setForeground(Color.black);
    okayButton.setEnabled(true);
    okayButton.addActionListener(this);
    okayButton.setBorderPainted(true);
    compareButton = new JButton("SIMULATION RESULT WITH
    COMPARATIVE METHODS");
    compareButton.setVerticalTextPosition(AbstractButton.BOTTOM);
    compareButton.setHorizontalTextPosition(AbstractButton.CENTER)
    ;
    compareButton.setActionCommand("compare");
    compareButton.setBackground(Color.lightGray);
    compareButton.setForeground(Color.black);
    compareButton.setEnabled(true);
    compareButton.addActionListener(this);
    compareButton.setBorderPainted(true);
}
}

```

```

public void setUI()
{
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout());
    panel.setBorder(BorderFactory.createTitledBorder("ACTIVITY
    OUTPUT DATA"));
    ActivityOutputTableModel activityOutputTableModel = new
    ActivityOutputTableModel(vector);
    JTable table = new JTable(activityOutputTableModel);
    table.getTableHeader().setBackground(Color.orange);
    TableColumn column = null;
    for (int i = 0; i < 7; i++) {
        column = table.getColumnModel().getColumn(i);
        if (i==0)
            column.setPreferredWidth(35);
    }
    JScrollPane jsp = new JScrollPane(table);
    ActivityOutputChart aoc = new ActivityOutputChart(vector);
    aoc.setWith(10);
    aoc.repaint();
    JScrollPane jspo = new JScrollPane(aoc);
    jspo.setPreferredSize(new Dimension(350,420));
    JPanel graphPanel = new JPanel();
    graphPanel.setLayout(new GridLayout(1,1));
    oc = new OutputCanvas();
    oc.setBackground(Color.white);
    graphPanel.add(oc);
    JScrollPane jspc = new JScrollPane(graphPanel);
    JPanel panelTwo = new JPanel();
    panelTwo.setLayout(new GridLayout(1,2));
    panelTwo.add(compareButton);
    panelTwo.add(okayButton);
    JSplitPane splitPanel = new
    JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
    splitPanel.add(jspo);
    splitPanel.add(jspc);
    JSplitPane splitPane = new
    JSplitPane(JSplitPane.VERTICAL_SPLIT);

```

```

JPanel tableContainer = new JPanel(new GridLayout(1,1));
tableContainer.setBorder(BorderFactory.createTitledBorder("ACT
IVITY DATA TABLE"));
tableContainer.add(jsp);
tableContainer.setPreferredSize(new Dimension(750,200));
splitPane.add(tableContainer);
JPanel chartContainer = new JPanel(new GridLayout(1,1));
chartContainer.setBorder(BorderFactory.createTitledBorder("ACT
IVITY CHART & PROGRESS CHART"));
chartContainer.add(splitPanel);
chartContainer.setPreferredSize(new Dimension(750,420));
splitPane.add(chartContainer);
panel.add(splitPane, BorderLayout.CENTER);
panel.add(panelTwo, BorderLayout.SOUTH);
setContentPane(panel);
pack();
setVisible(true);
try {
    int time = (int)server.getFinalTime();
    int test = 1;
    if( time >= (int)server.getFinalTime2() )
        {}
    else if (time < (int)server.getFinalTime2() ){
        time = (int)server.getFinalTime2();
        test = 2;
    }
    if( time >= (int)server.getFinalTime3() )
        {}
    else if ( time < (int)server.getFinalTime3() ){
        time = (int)server.getFinalTime3();
        test = 3;
    }
    if (test == 1 ) {
        float resultVal[], resultVal01[], resultVal02[];
        resultVal = new
        float[server.getTimeForGraph().length];
        resultVal01 = new
        float[server.getTimeForGraph().length];

```

```

resultVal02 = new
float[server.getTimeForGraph().length];
for(int j=0; j<resultVal.length; j++) {
    resultVal[j] = 100;
    resultVal01[j] =100;
    resultVal02[j] = 100;
}
float test01[];
test01 = new
float[server.getDataForGraph2().length];
test01 = server.getDataForGraph2();
for(int k=0; k<test01.length; k++) {
    resultVal[k] = test01[k];
}
float test02[];
test02 = new
float[server.getDataForGraph3().length];
test02 = server.getDataForGraph3();
for(int m=0; m<test02.length; m++) {
    resultVal01[m] = test02[m];
}
float test03[];
test03 = new
float[server.getDataForGraph4().length];
test03 = server.getDataForGraph4();
for(int m=0; m<test03.length; m++) {
    resultVal02[m] = test03[m];
}
oc.draw(server.getTimeForGraph(),
server.getDataForGraph, time, 55, 40, 270, 180,
"DPM Duration: ",resultVal, resultVal01,
resultVal02, (int)server.getFinalTime2(),
(int)server.getFinalTime3(),
(int)server.getFinalTime4(), 1);}
else if (test ==2) {
float resultVal[], resultVal01[], resultVal02[];
resultVal = new
float[server.getTimeForGraph2().length];

```

```

resultVal01 = new
float[server.getTimeForGraph2().length];
resultVal02 = new
float[server.getTimeForGraph2().length];
for(int j=0; j<resultVal.length; j++) {
    resultVal[j] = 100;
    resultVal01[j] = 100;
    resultVal02[j] = 100;
}
float test01[];
test01 = new
float[server.getDataForGraph().length];
test01 = server.getDataForGraph();
for(int k=0; k<test01.length-1; k++) {
    resultVal[k] = test01[k];
}
float test02[];
test02 = new
float[server.getDataForGraph3().length];
test02 = server.getDataForGraph3();
for(int m=0; m<test02.length-1; m++) {
    resultVal01[m] =
    server.getDataForGraph3()[m];
}
float test03[];
test03 = new
float[server.getDataForGraph4().length];
test03 = server.getDataForGraph4();
for(int m=0; m<test03.length-1; m++) {
    resultVal02[m] =
    server.getDataForGraph4()[m];
}
oc.draw(server.getTimeForGraph2(), server.getDataForG
raph2(), time, 55, 40, 270, 180, "DPM
Duration:", resultVal, resultVal01, resultVal02,
(int) server.getFinalTime(),
(int) server.getFinalTime3(),
(int) server.getFinalTime4(), 2);

```

```

}
else if (test==3) {
    float resultVal[], resultVal01[], resultVal02[];
    resultVal = new
    float[server.getTimeForGraph3().length];
    resultVal01 = new
    float[server.getTimeForGraph3().length];
    resultVal02 = new
    float[server.getTimeForGraph3().length];
    for(int j=0; j<resultVal.length; j++) {
        resultVal[j] = 100;
        resultVal01[j] =100;
        resultVal02[j] = 100;
    }
    float test01[];
    test01 = new
    float[server.getDataForGraph().length];
    test01 = server.getDataForGraph();
    for(int k=0; k<test01.length; k++) {
        resultVal[k] = test01[k];
    }
    float test02[];
    test02 = new
    float[server.getDataForGraph2().length];
    test02 = server.getDataForGraph2();
    for(int m=0; m<test02.length; m++) {
        resultVal01[m] = test02[m];
    }
    float test03[];
    test03 = new
    float[server.getDataForGraph4().length];
    test03 = server.getDataForGraph4();
    for(int m=0; m<test03.length; m++) {
        resultVal02[m] = test03[m];
    }
}
oc.draw(server.getTimeForGraph3(),server.getDataForGraph3
(),time, 55, 40, 270, 180, "DPM Duration: ",resultVal,
resultVal01, resultVal02,(int)server.getFinalTime(),

```



```
        (int)server.getFinalTime2(), (int)server.getFinalTime4(),
        3);}
    } catch(Exception rm){
    }
    oc.startSim();
    }
}
```

```

package gpms.client.output;

import java.awt.Color;
import java.awt.Graphics;
import java.io.PrintStream;

public class Graph
{
    public Graph()
    {
    }
    public Graph(float af[], float af1[], int i, int j, int k, int l,
        String s)
    {
        xValues = new float[af.length];
        for(int i1 = 0; i1 < af.length; i1++)
            xValues[i1] = af[i1];
        yValues = new float[af1.length];
        for(int j1 = 0; j1 < af1.length; j1++)
            yValues[j1] = af1[j1];
        xPos = i;
        yPos = j;
        width = k;
        height = l;
        name = s;
        minXValue = xValues[0];
        maxXValue = xValues[0];
        minYValue = yValues[0];
        maxYValue = yValues[0];
        try
        {
            for(int k1 = 0; k1 < xValues.length; k1++)
            {
                if(xValues[k1] < minXValue)
                    minXValue = xValues[k1];
                if(xValues[k1] > maxXValue)
                    maxXValue = xValues[k1];
            }
        }
    }
}

```

```

        for(int l1 = 0; l1 < yValues.length; l1++)
        {
            if(yValues[l1] < minYValue)
                minYValue = yValues[l1];
            if(yValues[l1] > maxYValue)
                maxYValue = yValues[l1];
        }
    }
    catch(ArrayIndexOutOfBoundsException _ex)
    {
    }
}
public void drawGraph(Graphics g)
{
    g.setColor(color);
    xLength = maxXValue - minXValue;
    yLength = maxYValue - minYValue;
    float f = (float)width / xLength;
    float f1 = (float)height / yLength;
    try
    {
        for(int i = 0; i < xValues.length - 1; i++)
            g.drawLine(xPos + (int)((xValues[i] - minXValue) * f),
                (yPos + height) - (int)((yValues[i] - minYValue) *
                    f1), xPos + (int)((xValues[i + 1] - minXValue) * f),
                (yPos + height) - (int)((yValues[i + 1] - minYValue)
                    * f1));
    }
    catch(ArrayIndexOutOfBoundsException _ex)
    {
    }
}
public void setTheColor(Color color1)
{
    color = color1;
}
private float xValues[];
private float yValues[];

```

```
private float minXValue;  
private float minYValue;  
private float maxXValue;  
private float maxYValue;  
private float xLength;  
private float yLength;  
private int xPos;  
private int yPos;  
private int width;  
private int height;  
private int minXLabel;  
private int maxXLabel;  
private Color color;  
private String name;  
}
```

```

package gpms.client.output;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.text.DecimalFormat;
import java.awt.geom.*;

public class OutputCanvas extends JPanel implements Runnable
{
    static int testCount = 0;
    boolean motion = false;
    Thread th;
    int count = 0;
    double dt;
    static final float dash1[] = {3.0f};
    static final BasicStroke dashed = new BasicStroke(1.0f,
        BasicStroke.CAP_BUTT, BasicStroke.JOIN_MITER, 3.0f, dash1,
        0.0f);
    private float xValues[], yValues[], minXValue, minYValue,
        maxXValue, maxYValue, xLength, yLength;
    private int xPos, yPos, width, height, minXLabel, maxXLabel;
    private String name;
    private int finalTime;
    static int timer = 0;
    static int n =1;
    private float yVal01[], yVal02[], yVal03[];
    private int cpm, pert, dpmWoAction;
    private int which;
    public OutputCanvas() {
    }
    public void paintComponent( Graphics g )
    {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;
        g2d.fill( new
            Rectangle2D.Double(xPos,yPos,xPos+width,yPos +
            height));
    }
}

```

```

g2d.setColor(Color.black);
g2d.drawLine( xPos, yPos + height, xPos + width, yPos +
height ) ;
g2d.drawLine( xPos, yPos, xPos, yPos + height);
g2d.drawLine( xPos, yPos, xPos+width, yPos);
g2d.drawLine( xPos+width,yPos, xPos+width, yPos+height);
float dashes02[] = {5};
g2d.setStroke(new BasicStroke(1, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND, 10, dashes02, 0));
g2d.drawLine( xPos, yPos + (height/4), xPos + width,
yPos + (height/4) ) ;
g2d.drawLine( xPos, yPos + (height/2), xPos + width,
yPos + (height/2) ) ;
g2d.drawLine( xPos, yPos + (height/4*3), xPos + width,
yPos + (height/4*3) ) ;
g2d.drawLine( xPos+width/2, yPos, xPos + width/2, yPos +
height ) ;
g2d.drawLine( xPos+width/4, yPos, xPos + width/4, yPos +
height ) ;
g2d.drawLine( xPos+width/4*3, yPos, xPos + width/4*3,
yPos + height ) ;
g2d.setColor(Color.orange);
g2d.drawString( "0", xPos-3, yPos + height + 15);
g2d.drawString( Integer.toString(finalTime)+" days",
xPos +
width - 10, yPos + height + 15);
g2d.drawString( Float.toString(minYValue), xPos - 30,
yPos + height + 5);
g2d.drawString("1.0", xPos - 30, yPos + 5);
g2d.drawString("0.25", xPos - 30, yPos + 5 +
height*3/4);
g2d.drawString("0.5", xPos - 30, yPos + 5 + height/2);
g2d.drawString("0.75", xPos - 30, yPos + 5 + height/4);
g2d.drawString( name,xPos + 45, yPos - 15 ) ;
float dash[] = {5};
g2d.setStroke(new BasicStroke(1, BasicStroke.CAP_ROUND,
BasicStroke.JOIN_ROUND, 10, dash, 0));
if (which == 1) {

```

```

g2d.setColor(Color.yellow);
g2d.drawString("CPM Duration", xPos + 30, yPos +
height + 60 );
g2d.drawLine(xPos+140, yPos+height+55, xPos+180,
yPos+height+55);
g2d.drawString(cpm+" days", xPos+190,
yPos+height+60);
g2d.setColor(Color.green);
g2d.drawString("PERT Duration", xPos + 30, yPos +
height + 80 );
g2d.drawLine(xPos+140, yPos+height+75, xPos+180,
yPos+height+75);
g2d.drawString(pert+" days", xPos+190,
yPos+height+80);
g2d.setColor(Color.red);
g2d.drawString("DPM W/O Action(*)", xPos + 30,
yPos + height +100 );
g2d.drawLine(xPos+140, yPos+height+95, xPos+180,
yPos+height+95);
g2d.drawString(dpmWoAction+" days", xPos+190,
yPos+height+100);
g2d.setStroke(new BasicStroke(3.0f));
g2d.setColor(Color.cyan);
g2d.drawString("DPM W/ Action", xPos + 30, yPos +
height +120 );
g2d.drawLine(xPos+140, yPos+height+115, xPos+175,
yPos+height+115);
g2d.drawString(finalTime+" days", xPos+190,
yPos+height+120);
g2d.drawString("Change from *", xPos + 30, yPos +
height +140 );
int change = (int)( ((double)(finalTime-
dpmWoAction)/(double)(dpmWoAction))*100 );
g2d.drawString(change + " %", xPos+190,
yPos+height+140);
g2d.drawString("Change from CPM", xPos + 30, yPos
+ height +160 );
int change2 = (int)( ((double)(finalTime-

```

```

        cpm)/(double) (cpm))*100 );
        g2d.drawString(change2 + " %", xPos+190,
        yPos+height+160);
    }
    else if (which == 2) {
        g2d.setColor(Color.yellow);
        g2d.drawString("CPM Duration", xPos + 30, yPos + height
        + 60 );
        g2d.drawLine(xPos+140, yPos+height+55, xPos+180,
        yPos+height+55);
        g2d.drawString(finalTime+" days", xPos+190,
        yPos+height+60);
        g2d.setColor(Color.green);
        g2d.drawString("PERT Duration", xPos + 30, yPos + height
        + 80 );
        g2d.drawLine(xPos+140, yPos+height+75, xPos+180,
        yPos+height+75);
        g2d.drawString(pert+" days", xPos+190, yPos+height+80);
        g2d.setColor(Color.red);
        g2d.drawString("DPM W/O Action(*)", xPos + 30, yPos +
        height +100 );
        g2d.drawLine(xPos+140, yPos+height+95, xPos+180,
        yPos+height+95);
        g2d.drawString(dpmWoAction+" days", xPos+190,
        yPos+height+100);
        g2d.setStroke(new BasicStroke(3.0f));
        g2d.setColor(Color.cyan);
        g2d.drawString("DPM W/ Action", xPos + 30, yPos + height
        +120 );
        g2d.drawLine(xPos+140, yPos+height+115, xPos+175,
        yPos+height+115);
        g2d.drawString(cpm+" days", xPos+190, yPos+height+120);
        g2d.drawString("Change from *", xPos + 30, yPos + height
        +140 );
        int change = (int)( ((double) (cpm-
        dpmWoAction)/(double) (dpmWoAction))*100 );
        g2d.drawString(change + " %", xPos+190,
        yPos+height+140);
    }
}

```



```

g2d.drawString("Change from CPM", xPos + 30, yPos +
height +160 );
int change2 = (int)( ((double)(cpm-
finalTime)/(double)(finalTime))*100 );
g2d.drawString(change2 + " %", xPos+190,
yPos+height+160);
}
else if (which == 3) {
g2d.setColor(Color.yellow);
g2d.drawString("CPM Duration", xPos + 30, yPos + height
+ 60 );
g2d.drawLine(xPos+140, yPos+height+55, xPos+180,
yPos+height+55);
g2d.drawString(dpmWoAction+" days", xPos+190,
yPos+height+60);
g2d.setColor(Color.green);
g2d.drawString("PERT Duration", xPos + 30, yPos + height
+80 );
g2d.drawLine(xPos+140, yPos+height+75, xPos+180,
yPos+height+75);
g2d.drawString(pert+" days", xPos+190, yPos+height+80);
g2d.setColor(Color.red);
g2d.drawString("DPM W/O Action(*)", xPos + 30, yPos +
height +100 );
g2d.drawLine(xPos+140, yPos+height+95, xPos+180,
yPos+height+95);
g2d.drawString(finalTime+" days", xPos+190,
yPos+height+100);
g2d.setStroke(new BasicStroke(3.0f));
g2d.setColor(Color.cyan);
g2d.drawString("DPM W/ Action", xPos + 30, yPos + height
+120 );
g2d.drawLine(xPos+140, yPos+height+115, xPos+175,
yPos+height+115);
g2d.drawString(cpm+" days", xPos+190, yPos+height+120);
g2d.drawString("Change from *", xPos + 30, yPos + height
+140 );
int change = (int)( ((double)(cpm-

```

```

        finalTime)/(double)(finalTime))*100 );
        g2d.drawString(change + " %", xPos+190,
        yPos+height+140);
        g2d.drawString("Change from CPM", xPos + 30, yPos +
        height +160 );
        int change2 = (int)( ((double)(cpm-
        dpmWoAction)/(double)(dpmWoAction))*100 );
        g2d.drawString(change2 + " %", xPos+190,
        yPos+height+160);
    }
    g2d.setColor(Color.orange);
    if(motion)
    {
        plotGraph(g2d, count);
    }
}
public void plotGraph( Graphics g, int c )
{
    Graphics2D g2d = (Graphics2D) g;
    float dashes[] = {5};
    if( c>1) {
        g2d.setColor(Color.cyan);
        xLength = maxXValue - minXValue;
        yLength = maxYValue - minYValue;
        float xScaler = (float) width/xLength;
        float yScaler = (float) height/yLength;
        try
        {
            int test01[], test02[], test03[], test04[],
            test05[];
            test01 = new int[c];
            test02 = new int[c];
            test03 = new int[c];
            test04 = new int[c];
            test05 = new int[c];
            for(int j=0; j<c; j++) {
                test01[j] = xPos + (int) ((xValues[j] - minXValue)
                * xScaler);
            }
        }
    }
}

```

```

test02[j] = yPos + height - (int) ((yValues[j] -
    minYValue) * yScaler);
test03[j] = yPos + height - (int) ((yVal01[j] -
    minYValue) * yScaler);
test04[j] = yPos + height - (int) ((yVal02[j] -
    minYValue) * yScaler);
test05[j] = yPos + height - (int) ((yVal03[j] -
    minYValue) * yScaler);
}
if (which == 1) {
    if(c < finalTime -3 ) {
        g2d.drawString(" "+new
            Integer(c).toString()+" days", xPos + 140,
            yPos - 15 );
    }
    if(c >= finalTime - 3) {
        g2d.drawString(" "+new
            Integer(finalTime).toString()+" days",
            xPos + 140, yPos - 15 );
    }
    g2d.setStroke(new BasicStroke(2.0f));
    g2d.drawPolyline(test01, test02, c);
    g2d.setStroke(new BasicStroke(1,
        BasicStroke.CAP_ROUND,
        BasicStroke.JOIN_ROUND, 10, dashes, 0));
    g2d.setColor(Color.yellow);
    g2d.drawPolyline(test01, test03, c);
    g2d.setColor(Color.white);
    g2d.drawPolyline(test01, test04, c);
    g2d.setColor(Color.green);
    g2d.drawPolyline(test01, test05, c);
    if(c >= finalTime - 2) {
        g2d.setColor(Color.cyan);
        g2d.drawLine( xPos+width, yPos,
            xPos+width, yPos+height);
        g2d.setColor(Color.yellow);
        g2d.drawLine(test01[cpm], test03[cpm],
            test01[cpm], test03[cpm]+height);
    }
}

```

```

        g2d.drawString( new
        Integer(cpm).toString(), test01[cpm]-10,
        yPos+height+15);
        g2d.setColor(Color.red);
        g2d.drawLine( test01[dpmWoAction],
        test04[dpmWoAction],
        test01[dpmWoAction],
        test04[dpmWoAction]+height);
        g2d.drawString( new
        Integer(dpmWoAction).toString(),
        test01[dpmWoAction]-10,
        yPos+height+15);
        g2d.setColor(Color.green);
        g2d.drawLine( test01[per],
        test05[per], test01[per],
        test05[per]+height);
        g2d.drawString( new
        Integer(per).toString(),
        test01[per], yPos+height+15);
    }
}
}
catch( ArrayIndexOutOfBoundsException ae )
{
}
}
public void run() {
    Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
    Thread cTh = Thread.currentThread();
    while(count+1<xValues.length && cTh==th) {
        count++;
        repaint();
        try {
            Thread.sleep(10);
            timer++;
        } catch (InterruptedException ie) {
            break;
        }
    }
}
}

```

```

        th= null;
    }
    public void start() {
        if(th == null)
            th = new Thread(this);
        th.start();
    }
    public void startSim() {
        motion = true;
        count = 0;
        start();
    }
    public void stop() {
        th = null;
    }
    public void draw( float xAxisValues[], float yAxisValues[],int
    finalTime, int xPos, int yPos, int graphwidth, int graphheight,
    String name, float yVal01[], float yVal02[], float yVal03[], int cpm,
    int dpmWoAction, int pert, int which)
    {
        xValues = xAxisValues;
        yValues = yAxisValues;
        this.xPos = xPos;
        this.yPos = yPos;
        width = graphwidth;
        height = graphheight;
        this.name = name;
        this.finalTime = finalTime;
        this.yVal01 = yVal01;
        this.yVal02 = yVal02;
        this.yVal03 = yVal03;
        this.cpm = cpm;
        this.dpmWoAction = dpmWoAction;
        this.pert = pert;
        this.which = which;
        timer.setCoalesce(true);
        int delay = (fps>0) ? (1000/fps) : 100;
        timer = new Timer(delay, new ActionListener() {

```

```

        public void actionPerformed(ActionEvent evt) {
            timer.start();
            repaint();
        }
    });
    xValues = new float[xAxisValues.length];
    for (int i = 0; i < xAxisValues.length; i++)
    {
        xValues[i] = xAxisValues[i];
    }
    yValues = new float[yAxisValues.length];
    for (int i = 0; i < yAxisValues.length; i++)
    {
        yValues[i] = yAxisValues[i];
    }
    this.xPos = xPos;
    this.yPos = yPos;
    this.width = width;
    this.height = height;
    this.name = name;
    maxXValue = xValues[0];
    minYValue = yValues[0];
    maxYValue = yValues[0];
    try
    {
        for (int i = 0; i < xValues.length; i++)
        {
            if (xValues[i] < minXValue)
            {
                minXValue = xValues[i];
            }
            if (xValues[i] > maxXValue)
            {
                maxXValue = xValues[i];
            }
        }
        for (int i = 0; i < yValues.length; i++)
        {

```

```
        if (yValues[i] < minYValue)
        {
            minYValue = yValues[i];
        }
        if (yValues[i] > maxYValue)
        {
            maxYValue = yValues[i];
        }
    }
}
catch( ArrayIndexOutOfBoundsException ae )
{
}
}
}
```

```

package gpms.client.tablemodels;

import gpms.util.*;
import javax.swing.*;
import java.awt.*;
import javax.swing.JTable;
import javax.swing.SwingUtilities;
import java.text.*;
import java.util.Vector;

public class ActivityTableModel extends
    javax.swing.table.AbstractTableModel
{
    public java.util.Vector listOfActivity;
    private java.util.Vector activityHeading;
public ActivityTableModel(Vector listOfActivity)
{
    super();
    this.listOfActivity = listOfActivity;
    activityHeading = new Vector();
    activityHeading.add("CODE");
    activityHeading.add("WBS");
    activityHeading.add("ACTIVITY NAME");
    activityHeading.add("DURATION");
    activityHeading.add("ST");
    activityHeading.add("FT");
}
public void fireTableDataChanged()
{
    super.fireTableDataChanged();
}
public int getColumnCount() {
    return activityHeading.size();
}
public String getColumnName(int col)
{
    return (String)activityHeading.get(col);
}
}

```



```

public int getRowCount()
{
    return listOfActivity.size();
}
public Object getValueAt(int row, int col)
{
    if (listOfActivity.get(row) != null)
    {
        if( ((Activity)listOfActivity.get(row)).getSimOrNot() ==
            0) {
            if(col == 0)
                return
                    ((Activity)listOfActivity.get(row)).getCode();
            else if(col ==1)
                return
                    ((Activity)listOfActivity.get(row)).getWbs();
            else if(col == 2)
                return
                    ((Activity)listOfActivity.get(row)).getName();
            else if(col == 3)
                return new
                    Integer(((Activity)listOfActivity.get(row)).getDur
                    ation());
            else if(col == 4)
                return new
                    Integer(((Activity)listOfActivity.get(row)).getST(
                    ));
            else if(col == 5)
                return new
                    Integer(((Activity)listOfActivity.get(row)).getST(
                    ) +
                    ((Activity)listOfActivity.get(row)).getDuration())
                    ;
        }
        else if
            ( ((Activity)listOfActivity.get(row)).getSimOrNot()
            == 1) {
            if(col == 0)

```

```

        return
        ((Activity)listOfActivity.get(row)).getCode();
    else if (col==1)
        return
        ((Activity)listOfActivity.get(row)).getWbs();
    else if(col == 2)
        return
        ((Activity)listOfActivity.get(row)).getName();
    else if(col == 3)
        return new
        Integer(((Activity)listOfActivity.get(row)).getSim
        Duration());
    else if(col == 4)
        return new
        Integer(((Activity)listOfActivity.get(row)).getSim
        St());
    else if(col == 5)
        return new
        Integer(((Activity)listOfActivity.get(row)).getSim
        St() +
        ((Activity)listOfActivity.get(row)).getSimDuration
        ());
    }
}
return "NULL";
}
}

```

```

package gpms.client.tablemodels;

import gpms.util.Precedence;
import gpms.util.Activity;
import java.util.Vector;
import javax.swing.table.AbstractTableModel;
import gpms.client.tablemodels.CellData;
import java.awt.event.*;
import java.awt.*;
import gpms.client.gui.Client;
public class DSMTTableModel extends AbstractTableModel{
    CellData data[][];
    String [] columnNames;
    int nwTable, nhTable;
    String [] rowNames;
    gpms.client.gui.Client client;
public DSMTTableModel(int w, int h) {
    super();
    initTable(w,h);
}
public void fireTableDataChanged() {
    super.fireTableDataChanged();
}
public Class getColumnClass(int c) {
    return getValueAt(0,c).getClass();
}
public int getColumnCount() {
    return columnNames.length;
}
public String getColumnName(int col) {
    return columnNames[col];
}
public int getRowCount() {
    return data.length;
}
public Object getValueAt(int row, int col) {
    return data[row][col];
}
}

```

```

private void initTable(int w, int h) {
    nwTable = w;
    nhTable = h+1;
    data = new CellData[nwTable][nhTable];
    for(int i=0; i<w; i++) {
        for(int j=0; j<nhTable; j++) {
            data[i][j] = new CellData();
        }
    }
}

public boolean isCellEditable(int row, int col) {
    for(int i=0; i<getRowCount(); i++) {
        if ((row==i) && (col==i+1))
            return false;
    }
    return true;
}

public void setColumnNames(int col) {
    columnNames = new String[col+1];
    for (int i=0; i<=col; i++) {
        if(i==0)
            columnNames[i]="WBS";
        else
            columnNames[i]="a"+i;
    }
}

void setRowName(String [] wbsName) {
    for(int i=0; i<wbsName.length; i++) {
        setValueAt(wbsName[i],i,0);
    }
}

public void setValueAt(Object value, int row, int col) {
    data[row][col].setValue(value);
    fireTableCellUpdated(row,col);
}
}

```

```

package gpms.client.tablemodels;

import gpms.util.Precedence;
import java.util.Vector;
import javax.swing.table.AbstractTableModel;

public class PrecedenceTableModel extends AbstractTableModel
{
    public PrecedenceTableModel(Vector vector)
    {
        listOfPrecedenceActivity = vector;
        precedenceHeading = new Vector();
        precedenceHeading.add("Preceding Activity");
        precedenceHeading.add("Relation");
        precedenceHeading.add("Lag");
        precedenceHeading.add("Sensitivity");
    }
    public void fireTableDataChanged()
    {
        super.fireTableDataChanged();
    }
    public int getColumnCount()
    {
        return precedenceHeading.size();
    }
    public String getColumnName(int i)
    {
        return (String)precedenceHeading.get(i);
    }
    public int getRowCount()
    {
        return listOfPrecedenceActivity.size();
    }
    public Object getValueAt(int i, int j)
    {
        if(j == 0)
            return
                ((Precedence)listOfPrecedenceActivity.get(i)).getPredAct

```

```

        ivityCode();
    if(j == 1)
        return
            ((Precedence)listOfPrecedenceActivity.get(i)).getRelatio
            n();
    if(j == 2)
        return new
            Integer(((Precedence)listOfPrecedenceActivity.get(i)).ge
            tLag());
    if(j == 3)
        return
            ((Precedence)listOfPrecedenceActivity.get(i)).getSensiti
            vity();
    else
        return "NULL";
}
private Vector precedenceHeading;
public Vector listOfPrecedenceActivity;
}

```

```

package gpms.client.tablemodels;

import gpms.util.Project;
import java.util.Vector;
import javax.swing.JTable;
import javax.swing.JTable.*;
import javax.swing.table.TableColumn;
import javax.swing.table.TableColumn.*;
import javax.swing.ListSelectionModel;
import javax.swing.DefaultCellEditor;
import javax.swing.table.TableCellRenderer;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.JScrollPane;
import javax.swing.JPanel;
import javax.swing.JFrame;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.border.*;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.*;
import javax.swing.event.ListSelectionListener;
import javax.swing.event.ListSelectionEvent;
import gpms.client.gui.RelationDialog;

public class PrimaveraProjectTableModel extends
javax.swing.table.AbstractTableModel {
    private java.util.Vector projectHeading;
    private java.util.Vector listOfProject;
public PrimaveraProjectTableModel(Vector listOfProject) {
    super();
    this.listOfProject = listOfProject;
    projectHeading = new Vector();
    projectHeading.add("Project ID");
    projectHeading.add("Project Name");

```

```

}
public int getColumnCount() {
    return projectHeading.size();
}
public String getColumnName(int col) {
    return (String)projectHeading.get(col);
}
public int getRowCount() {
    return listOfProject.size();
}
public Object getValueAt(int row, int col)
{
    if(col == 0)
    {
        return new
            Integer(((Project)listOfProject.get(row)).getId());
    }
    if(col == 1)
    {
        return ((Project)listOfProject.get(row)).getName();
    }
    return "NULL";
}
}

```



```

package gpms.client.tablemodels;

import gpms.util.Project;
import java.util.Vector;
import javax.swing.table.AbstractTableModel;

public class ProjectTableModel extends AbstractTableModel
{
    public ProjectTableModel(Vector vector)
    {
        listOfProject = vector;
        projectHeading = new Vector();
        projectHeading.add("ID");
        projectHeading.add("Name");
    }
    public int getColumnCount()
    {
        return projectHeading.size();
    }
    public String getColumnName(int i)
    {
        return (String)projectHeading.get(i);
    }
    public int getRowCount()
    {
        return listOfProject.size();
    }
    public Object getValueAt(int i, int j)
    {
        if(j == 0)
            return new
                Integer(((Project)listOfProject.get(i)).getId());
        if(j == 1)
            return ((Project)listOfProject.get(i)).getName();
        else
            return "NULL";
    }
    private Vector projectHeading;
}

```

```
private Vector listOfProject;  
}
```

```

package gpms.client.tablemodels;

import gpms.util.Version;
import java.util.Vector;
import javax.swing.table.AbstractTableModel;

public class VersionTableModel extends AbstractTableModel
{
    public VersionTableModel()
    {
    }
    public VersionTableModel(Vector vector)
    {
        listOfVersion = vector;
        versionHeading = new Vector();
        versionHeading.add("Version ID");
        versionHeading.add("Version Description");
    }
    public int getColumnCount()
    {
        return versionHeading.size();
    }
    public String getColumnName(int i)
    {
        return (String)versionHeading.get(i);
    }
    public int getRowCount()
    {
        return listOfVersion.size();
    }
    public Object getValueAt(int i, int j)
    {
        if(j == 0)
            return new
Integer(((Version)listOfVersion.get(i)).getId());
        if(j == 1)
            return ((Version)listOfVersion.get(i)).getInfo();
        else

```

```
        return "NULL";
    }
    private Vector versionHeading;
    private Vector listOfVersion;
}
```

APPENDIX IV

BIOGRAPHICAL NOTE

EDUCATION

- 2003 - Present **Ph.D. Candidate** in Information Technology and Construction Management
Massachusetts Institute of Technology, Cambridge, MA
- Dissertation: “Dynamic Planning and Control Methodology: Understanding and Managing Iterative Error and Change Cycles in Large-scale Concurrent Design and Construction”
 - Thesis Committee: Professor Feniosky Peña-Mora (Advisor, UIUC), Professor Fred Moavenzadeh (Chair, MIT), Professor Jerome Connor (MIT), Professor Massood Samii (SNHU)
- 2003 **Master of Science** in Construction Management and Information Technology
Massachusetts Institute of Technology, Cambridge, MA
- Thesis: “Dynamic Quality and Change Management for Large-scale Concurrent Design and Construction Projects” (Advisor: Professor Feniosky Peña-Mora)
- 2000 **Bachelor of Engineering** in Architectural Engineering
Dong-A University, Pusan, Korea
- Architectural Design: “Space Collision: Multi-Use Cultural Center”

AWARDS & HONORS

- 2006 Berger Fellowship, Massachusetts Institute of Technology
- 2003 - 2004 UIUC Fellowship, University of Illinois at Urbana-Champaign
- 2002 - 2005 Dong-A Fellowship, Dong-A University
- 2001 - 2005 Research Assistantship, Massachusetts Institute of Technology and University of Illinois at Urbana-Champaign
- 1999 14th Space Group Architectural Design Excellence Award
- 1999 Dong-A University Graduation Design Excellence Award
- 1998 - 1999 Academic Excellence Fellowship, Dong-A University

RESEARCH INTERESTS

Concurrent Design and Construction Management; Change Management; Proactive Buffer Management; Strategic Simulation for Construction Policy Design; Information Visualization; Information Technology-based Decision Support System

RESEARCH EXPERIENCE

- 2003 - Present Doctoral Research: Construction Management and Information Technology,
Massachusetts Institute of Technology
Research Advisor: Professor Feniosky Peña-Mora
- Identified how errors and changes dynamically affect construction performance.
 - Proposed a proactive buffering approach that aims to absorb and prevent the negative impacts of errors and changes in concurrent design and construction.
 - Developed a system dynamics-based design and construction project model to evaluate the impact of iterative error and change cycles on design and construction performance.
 - Implemented a web-based error and change management system that support heterogeneous devices.
- 2000 - 2003 Master's Research: Construction Management and Information Technology,
Massachusetts Institute of Technology
Research Advisor: Professor Feniosky Peña-Mora
- Investigated behavior of errors and changes in fast-track design and construction projects.
 - Designed a prototype collaborative web project management system for effective information sharing and distribution.

TEACHING EXPERIENCE

- 2003 (Spring) Teaching Assistant, "Distributed Development of Engineering Information Systems"
Massachusetts Institute of Technology, Cambridge, MA
- 'Distance Learning' class offered for the Collaborative Program between MIT and the Malaysia University of Science & Technology
- 2002 (Spring) Teaching Assistant, "Distributed Development of Engineering Information 2001
(Fall) *Massachusetts Institute of Technology*, Cambridge, MA
- 'Real-time Distributed' course taught between the Pontificia Universidad Católica de Chile (PUC) in Chile, Centro de Investigación Científica y Estudios Superiores de Ensenada (CICESE) in México, and the Massachusetts Institute of Technology (MIT) in the US

PROFESSIONAL EXPERIENCE

- 2002 – 2004 Voluntary Consultant, Gloucester Community Development Co., Gloucester, MA
- Feasibility study of a surimi factory in Gloucester using System Dynamics.
- 1999 - 2000 Architectural Designer, SamJung Architectural Engineering Co., Pusan, Korea

- Feasibility study and schematic design for government office projects and private residential complex projects
- 1997 - 1999 Design and Inspection Assistant, SamJung Architectural Engineering Co., Pusan, Korea
- Internship for schematic design and inspection
- 1994 - 1995 Construction Engineer, Korean Military, Korea
- Responsible for concrete pouring, masonry work, carpentry, and etc. in military projects.

PUBLICATIONS

Academic Journal Papers:

1. 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, Vol. 131, No. 8, pp. 890-902.
2. Lee, S., Peña-Mora, F., Park, M. (2006), "*Dynamic Planning and Control Methodology For Strategic and Operational Construction Project Management.*" Automation in Construction, Elsevier, Oxford, UK, January, 2006, Vol. 15, No. 1, pp. 84-97.
3. Lee, S., Peña-Mora, F., Park, M. (2006), "*Reliability and Stability Buffering Approach: Focusing on the Issues of Errors and Changes in Concurrent Design and Construction Projects.*" Journal of Construction Engineering and Management, ASCE, Reston, VA, May, 2006, Vol. 132, No.5, pp.452-464.
4. Lee, S., Peña-Mora, F., Park, M. (2006), "*Web-enabled System Dynamics Model for Error and Change Management on Concurrent Design and Construction Projects.*" Journal of Computing in Civil Engineering, ASCE, Reston, VA, July/August, 2006. (In Press)

Conference Proceedings:

5. Lee, S., Peña-Mora, F., Park, M. (2003), "*Dynamic Quality and Change Management Framework for Concurrent Design and Construction.*" ASCE Construction Research Congress, Honolulu, HA, ASCE.
6. Lee, S., Peña-Mora, F., Park, M. (2003), "*Reliability and Stability Buffering Approach in Concurrent Design and Construction Projects.*" International Group of Lean Construction 11th Annual Conference, Blacksburg, Virginia, IGLC.
7. Lee, S., Peña-Mora, F., Park, M. (2005), "*Dynamic Error and Change Management in Concurrent Design and Construction.*" ASCE Construction Research Congress, San Diego, CA, ASCE.

8. Lee, S., Peña-Mora, F. (2005), "*System Dynamics Approach for Error and Change Management in Concurrent Design and Construction.*" 2005 Winter Simulation Conference, Institute of Electrical and Electronics Engineers (IEEE), Piscataway, NJ.
9. Lee, S., Peña-Mora, F. (2006), "*Latency in Error and Change Management.*" Joint International Conference on Computing and Decision Making in Civil and Building Engineering, Montreal, Canada.
10. Lee, S., Peña-Mora, F. (2006), "*Visualization of Construction Progress Monitoring.*" Joint International Conference on Computing and Decision Making in Civil and Building Engineering, Montreal, Canada.
11. Han, S. Peña-Mora, F., Lee, S., Park, M. (2006), "*Hybrid Simulation for Strategic-Operational Construction Management.*" Joint International Conference on Computing and Decision Making in Civil and Building Engineering, Montreal, Canada.
12. Lee, S., Peña-Mora, F. (2006), "*Lessons Learned from Applying System Dynamics to Construction Simulation.*" 13th EG-ICE Workshop on Intelligent Computing in Engineering & Architecture, Monte Verità, Ascona, Switzerland.

RESEARCH IN PROGRESS

1. Dynamic Planning and Control Methodology (DPM) for Large-scale Concurrent Design and Construction Projects.
2. Proactive Buffering Approach in Design and Construction (with Prof. Park, formerly with the National University of Singapore, Singapore - currently with Seoul National University, Korea).
3. Integrated System for Change Prediction and Management (with Prof. Anumba and Dr. Motowa, Loughborough University, UK).
4. Identifying, Analyzing, and Visualizing Change Impact on Schedule Performance (with Prof. Anumba and Ms. Yeoh, Loughborough University, UK).
5. Error and Change Management of Malaysia SIRIM Project (with Prof. Gunawan and Mr. Chee, Malaysia University of Science & Technology, Malaysia).
6. System Dynamic Curriculum Development (with Prof. Park, Seoul National University, Korea).
7. Visualization of Construction Progress Monitoring.

PRESENTATIONS

1. "*Dynamic Planning and Control Methodology.*" Invited Research Seminar (2005), Department of Architecture, Seoul National University, Korea

2. *"Applying System Dynamics to Complex Problems in Infrastructure Management."* Invited Research Seminar (2005), Department of Urban Planning, ChoongBuk National University, Korea.
3. *"Dynamic Error and Change Management in Concurrent Design and Construction."* ASCE Construction Research Congress (2005), San Diego, CA, ASCE.
4. *"Comparative Study of Discrete-Event Simulation and System Dynamics for Construction Process Planning."* ASCE Construction Research Congress (2005), San Diego, CA, ASCE.
5. *"Dynamic Quality and Change Management Framework for Concurrent Design and Construction."* ASCE Construction Research Congress (2003), Honolulu, HA, ASCE.
6. *"Iterative Error and Change Cycles in Concurrent Design and Construction Projects."* Invited Research Talk (2002), Department of Building, National University of Singapore, Singapore

SOFTWARE DESIGN & IMPLEMENTATION

1. *"Web-based DPM"* – A web-based error and change management system implemented following a five-tier system architecture with ORACLE 9i as the enterprise database and Java Enterprise edition as the programming language, running on heterogeneous devices ranging from Java-enabled phones, PDAs, PC, and high-end computational workstations.

PROFESSIONAL ACTIVITIES

1. Reviewer of the Journal of Construction Engineering and Management, ASCE, 2005 to present.
2. Member of the Technical Committee, ASCE International Conference on Computing in Civil Engineering, Cancun, Mexico, 2005.
3. Member of the American Society of Civil Engineers, USA.
4. Registered Architectural Engineer, Korea, 1999 to present.

RESEARCH PROPOSAL INVOLVEMENT

1. *"Dynamic Planning and control Methodology (DPM) for Large-Scale Concurrent Design and Construction Projects."* National Science Foundation CMS-0324501, \$398,000, 2003 – 2006.

COMPUTER SKILLS

UML, Java, SQL, XML, HTML, Vensim DSS (Venapp), Stellar, Sigma, ORACLE 9i, MS SQL Server,

MS Access, Primavera, MS Project, Auto CAD, 3D MAX, MS Office.

Ph.D. COURSE WORK

1. System Dynamics for Business Policy
2. Application of System Dynamics
3. System Dynamics for Engineers
4. Control Theory
5. Simulation
6. Multi-agent Systems.
7. Applied Statistics
8. Project Management
9. Construction Finance
10. Financial Derivatives
11. Strategic Management in Construction
12. Professional Concepts
13. M. Eng. Project
14. Computer Programming
15. Data Mining: Algorithms & Applications
16. Software Engineering
17. Information Technology
18. Distributed Development of Engineering Information Systems
19. Database, Internet, and System Integration
20. Advanced Non-linear Dynamics