Department of Human and Engineered Environmental Studies
Graduate School of Frontier Sciences
The University of Tokyo

2020

Master’s Thesis

A System Dynamics Approach towards Design and Management for Software Development Projects

Submitted July 30, 2020

Adviser: Associate Professor Kazuo HIEKATA (seal)

Student ID Number 47186847

Mst. Taskia KHATUN

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background

Project management is a technique for assimilating available resources-time, cost, people- against business project aims-early completion date, and the final date. Products of software projects have some features such as invisibility, complexity, and flexibility which makes software projects different from projects of other disciplines. According to statistical analysis, the most frequent challenges to the software project manager is coping with keeping the deadlines(Lyneis and Ford 2007).

Many software projects are not successful due to the lack of time to complete them. Usually four control actions are considered when schedule slippages happens. These are hiring workforces, overtime working, extending project completion date as well as reducing the requirements, and each of them should be taken into consideration under certain situation. Among these actions, the most common are working overtime and hiring workforces to keep the deadline fixed. When considering project management, either software project or any other project, it has to follow a series of activities of deciding the whole project scope and certain requirements and utilizing the resources. Again, if problems occur during the development period of the project, main focus goes to the above-mentioned control action, although these actions can result in various dynamic feedback responses. For example, hiring workforce to a development team is a time-consuming process. And after hiring, an employee needs to be trained to understand the working environment as well as project status. Instead, if the project manager focuses on working overtime, it can be easily achieved and also may cost less. However, working overtime may achieve the goal, it generates many dynamic feedbacks throughout the development period.

1.2 General Description of Project Management

When talking about project management, one of the important constraint of project management is scope. Scope represents the features and functions of the product. More features and functions increase project scope which in turn increases the number of tasks of the project. The scope of a project can be changed during the development period or after development. When this happens, it becomes difficult to manage all other resources following the previous way before changing scope. It might affect the project schedule, workforces' working-hour, and their productivity. Therefore, to avoid contention, the project scope should be decided as fixed before starting to develop the project. This will help the project manager to allocate the resources more accurately.

1.3 Introduction to Software Development Project

A software development project is a complex state proceeding by a team within the boundaries of time, budget, and specific scope to enhance or produce new computer source code which adds useful value to the existing or a new process. A project consists of a series of activities directed to accomplish the desired goal. The major features of a project are scope, cost, and time. Along with these, several features are included in a project as following(Chetan D. Vajre, 2003).

✓ Completing specific objective with specified and tangible details

✓ Start and End dates

✓ Budget/Cost

✓ Resources (People/Equipment).

A software project is said to be successful, when it meets the customer's desired requirements, enlightens a high quality, and is delivered within time and budget.

1.4 Need for Simulation

Simulation is a very common word applied to a very wide field and industrial applications. It represents a wide range of methods and applications that reproduce the behavior of real systems using a computer along with the appropriate software. Many good reasons can be ascertained for the need of computer simulation (Chetan D. Vajre, 2003).

- Using theoretical derivations, a few limited distributions could be calculated to get results
- System involving queues may not follow the standard rules
- The system behaves dynamic
- To know the state of a system, it may be required to go on calculating for long time intervals, which can not be feasible and practical
- To compare a system with different scenarios, everything needs to be reworked with the new parameters

All these work great when resource utilization, queue size are known to the modeling of systems, however, in certain systems, it might not be relevant. When we are considering a software industry, there is no queue and becomes difficult to measure resource utilization. To model and study the behavior of such a system, many factors might not seem to affect the system at first or maybe there are several intangible factors that add quality of the product (Chetan D. Vajre, 2003)(Armindokht, 2012). These factors do not add value to the product but would affect the expectation if neglected. A well-known example is the prediction of deadline for new software. The absence of an engineer does not add any value to the product, but it affects the deadline. Since deadlines and quality are becoming major factors in the success of the software industry, more and more importance were being given to start studying the system behavior.

1.5 Organization of the Thesis

This chapter dealt with the background of software development project management and the reason for modeling software development projects. The problem statement, research objective, and approach have been described in chapter 2. After that, a literature review and the basic concept of the System Dynamics method have been explained in chapter 3. Then, the proposed methodology has been discussed in chapter 4. Next, model validation and a case study have been explained in detail through chapters 5 and 6 respectively. After that, a sensitivity analysis has been discussed in chapter 7. Finally, chapter 8 and 9 have been used for discussion, conclusion, and future research information.

Chapter 2 Problem Statement

2.1 Issues with Human Resource Management

Human factors in project management have gained importance as it is recognized as the core of effective software project management. Several problems often happen in software development and implementation due to shortage in human resources. Software project managers find themselves paying more attention to human resource issues since hiring decisions involving timing and experience level of the engineers which are the key to a project's success (Chetan D. Vajre, 2003). Hiring the desired number of workforces for project completion is also difficult due to the uncertainty of rework generation which causes fluctuation of man-days.

2.2 Problems Regarding Dynamic and Complex Project Behavior

The proper planning and management of a project often becomes difficult. Project management has to follow a series of activities to decide the whole project scope and the requirements, the project schedule and utilization of the budget and workforces as well (Lyneis and Ford, 2007)(Tarek A. Hamid, 1984). Hence, if problems happen during the project, the common action chosen by project managers is to work overtime to control the project progress. However, this action may result in various dynamic feedback responses which affect project progress in various way. The application of overtime is not always good as high overtime adversely affects productivity. Although overtime is a very common policy, very few papers have discussed its impact but not clearly in which way it can be effective (Jianguo Jia et al, 2007).

2.3 Previous work on Software Development Project Management

Several System Dynamics models have been developed to manage the performance of software development projects.

(Tarek A. Hamid, 1984), (Armindokht, 2012) explained using interval-based overtime through System Dynamics modeling. They discussed the consequences of interval-based overtime. However, using continuous overtime was not in consideration.

(Tarek A. Hamid ,1989) proposed a System Dynamics approach regarding project staffing. In this work, the author focused on the dynamic behavior of software project staffing for the software development lifecycles. The issue is that the author mainly shows the dynamic attitude of adding project staffing, but when considering real-world project development, most of them focus on static behavior for adding new workforces if needed.

(Chetan D. Vajre, 2003) developed a System Dynamics model for managing project performance focusing on human resource management. The author provided several combinations of expert workforces and new workforces considering hiring and explains the required time for project completion. Nevertheless, the author did not consider the impact of a learning although considered hiring new workforces.

(Jianguo, Xia et al, 2007) designed a System Dynamics model for overtime management strategy of the software projects. In their work, they focused on the application of overtime and the consequences of dynamic behavior and its impact on project management although they did not explain the range and types of overtime for use. They simply showed the consequences of using overtime.

Considering the difficulties, the outcomes of those research work and the limitations, we are going to address the usage of overtime and hiring for project planning and its impact on project performance.

2.4 System Dynamics Approach

Before defining the objective of this research, a brief description of System Dynamics has given since this approach has been chosen to design our method.

System Dynamics is a tool for analyzing the behavior of complex systems and problems through computer simulation software. Dr. J. W. Forrester designed this methodology in 1960 at MIT. He became interested in the complexity of business management and the causes attributed to its success and failures. He thought that people did not analyze a complex system very well and neglected many factors that indirectly but significantly affect the running of a business (Chetan D. Vajre, 2003). The basic mistake committed is attributing the success and failures of different factors and the result is the policy for achieving a goal turns out to be very simple and instead of the system behaving as per the expected pattern, it behaves maybe in reverse way that leading to failure (Raymond, 1984) (Chetan D. Vajre, 2003).

Systems Dynamics models are more devoted with capturing the structure and policies of the system, the mode of behavior of the whole system rather than accurate prediction. After developing a System Dynamics model and specifying the initial conditions, a computer can simulate the behavior of different model variables over time.

2.5 Research Objective and Approach

The objective of this research is to develop a method for supporting decisions for software development project management against uncertain rework and fluctuating productivity of engineers. This method would support for prediction and evaluation of process improvement of project planning, controlling through model-based decision making to manage performance which would enrich and enhance our understanding of and making prediction about model-based decision making. The developed decision support method would also help to overcome the difficulty of designing a feasible plan for software projects through methods that coordinate project activity.

To develop a simulation method as a decision support for software development project management, two popular methods, Multi-Agent System(MAS) and System Dynamics(SD) are used for sensing the process behavior and dynamic interaction.

Multi-Agent System is a disaggregated bottom-up approach and defines the inner behavior of different elements and the relations and the behavior emerge. It seeks to forecast trends based on simple rules between individuals and specific network shape (Zhikun Ding et al, 2018).

System Dynamics is defined as an aggregated top-down approach that represent the relations among different key elements of the system and can illustrate the structure which produces the behavior of the system and clarify the leverage points for policymaking (Zhikun Ding et al, 2018). System Dynamics modeling has the capability to provide insights by observing virtually any aspect of the software process. It can be used to evaluate and compare different life-cycle processes, defect detection techniques, business cases, interactions between interdisciplinary process activities (Raymond, 2007).

Since our objective is to develop a method for decision support which would help us to understand the system behavior and policymaking, we have chosen the System Dynamics(SD) method to develop our simulator.

To obtain the defined objective, we have done the following procedures:

- Developed a simulation model using System Dynamics(SD)
- While developing the model, two major actions have been taken under consideration- hiring and overtime management plan. Different scenarios have been explored, through which decisions can be made to design a proper plan for managing project performance.

The first purpose of developing this model is that it would be used as a tool to get a rough estimation of cost and duration of developing software projects and can be used to observe the current progress of the project. The parameter values can be changed to observe project performance behavior since several variables affect the software development process. Besides, many of them are interrelated to each other.

The second purpose of our model is to make predictions about the general process by which software projects are developed. The model would use as a framework for analyzing of managerial policies and procedures required for hiring and overtime management through which proper planning can be done for development.

One major focus of the model building is the consideration of the rework cycle. The rework cycle is identified as a canonical structure of System Dynamics modeling (Lyneis and Ford, 2007). The use of this structure helps us to identify error-based tasks that needed to be resolved to make the project successful.

Well, one key limitation of our analysis, after hiring new workforces and getting trained, we considered the same productivity for all of the workforces. Another limitation is that while considering the rework cycle, we did not distribute the manpower separately for task development and rework cycle.

Chapter 3 Literature Review

3.1 Project Management and System Dynamics

(Reichelt and Lyneis, 1999) worked on the dynamics of project performance. Often projects fail to achieve the designed cost and schedule. They claimed that despite an increase in effort to improve project management, very often the management team failed because the model-tools are not considering the complexity and dynamic nature of projects in which they are working.

(Jogeklar and Ford, 2003) focused on resource allocation. They stated that minimizing a project's duration is adverse to the project development process. Resource allocation to development activities can strongly influence the project duration since it is represented as principle operator of progress and effective management. They provided a resource allocation policy matrix to describe resource allocation policies in various dynamic systems.

(Lyneis and Ford, 2007) defined that, the most dominated and successful application area of System Dynamics is project management. They explained about the measurement of new System Dynamics theory, model structures, applications among other indicators to underline their theory.

3.2 Software Project Management and System Dynamics

(Tarek A. Hamid, 1989) designed a System dynamics approach regarding project staffing. In this work, the author worked focused on the dynamic behavior of software project staffing for the software development lifecycles. The model has been used as a decision support system to test the degree of interchangeability of men and months on particular software projects.

(Rus et al 1999) explained the usage of a process simulator to support software project, mainly in planning and management. They defined that the model can be used in project planning as a predictor for the effect of management and reliability as a part of the decision support system.

(Kahen et al, 2001) talked about the Feedback, Evolution, and Software Technology (FEAST) research, focusing on the consequences of long-term evolution on software production. They presented elements from previous works, and then combined with new approaches to analyze the consequences for decision making in software projects.

(Jianguo Jia et al, 2007) designed a System Dynamics model for the overtime management strategy of a software project. In their work, they focus on the application of overtime and the consequences of dynamic behavior and its impact on project management.

3.3 Building Blocks of Lyneis's Model for Project Management

(Lyneis and Ford, 2007) explained that to design a System Dynamics model focusing project management, the model structure should be categorized into four groups-Project Features, Rework Cycle, Project Control, and Ripple and Knock-on Effects.

1. **The Project Features:** System Dynamics focuses on modeling features found in the real system. These features include task development processes, resources, managerial mental models, and decision-making. Adding more features to the actual project increases the efficiency to simulate realistic project dynamics.

2. **Rework Cycle:** Rework cycle is defined as the most important feature of the System Dynamics project model(Suinan Li, 2008). The basic structure of rework cycle is shown in figure 3.1.

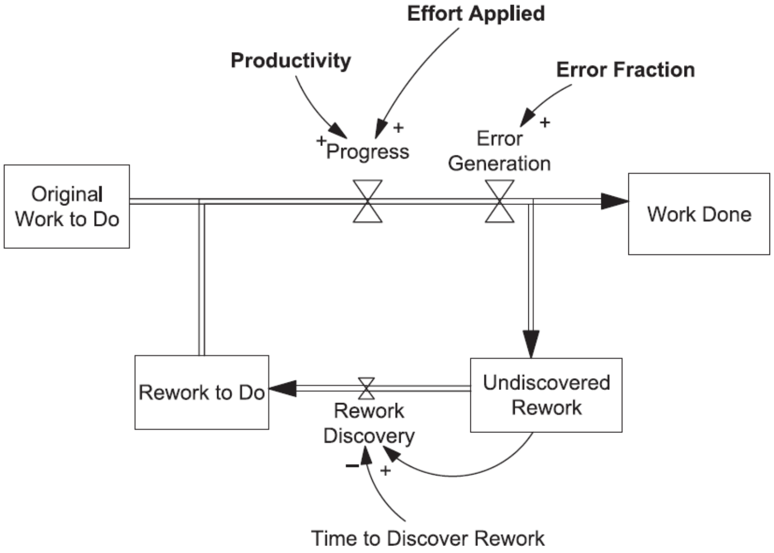


Fig. 3.1 Rework Cycle for Project Management (Lyneis and Ford, 2007)

3. **Project Control:** Managerial actions taken to control project performance are modeled as to close the gap between target and actual performance. Two common methods for project control are increasing work intensity with working overtime, or slipping the deadline. Three common actions can be taken by project managers when there is a probability of missing the deadline: hire additional workforce, work overtime, and work faster.

4. **Ripple and Knock-on Effects:** Several Actions that are taken to minimize the gap between project performance and targets have unintended side effects result policy resistance. Such effect is called "ripple effects". These ripple effects are defined as the primary side effects of project control on rework and productivity and the "Knock-on effects" is called the secondary effects of project control efforts.

3.4 Building Blocks for Abdel-Hamid's Model of Software Project Management

(Abdel-Hamid and Madnick, 1991) suggested that to get a better understanding of the software development process, a fully integrated model is needed that would take other variables into account on the software development subsystem as shown in figure 3.2(Timothy, 2010).

The *Human Resource* subsystem includes the functions hiring, training, and assimilation. These functions operate as endogenous variables that both affect and are affected by the other subsystems. The *Workforce available* influences the allocation of human resources in the *Software Production* subsystem and both the *Controlling* subsystem and *Planning* subsystem have a direct effect on the *Human Resource* subsystem, and therefore the *Workforce available.*

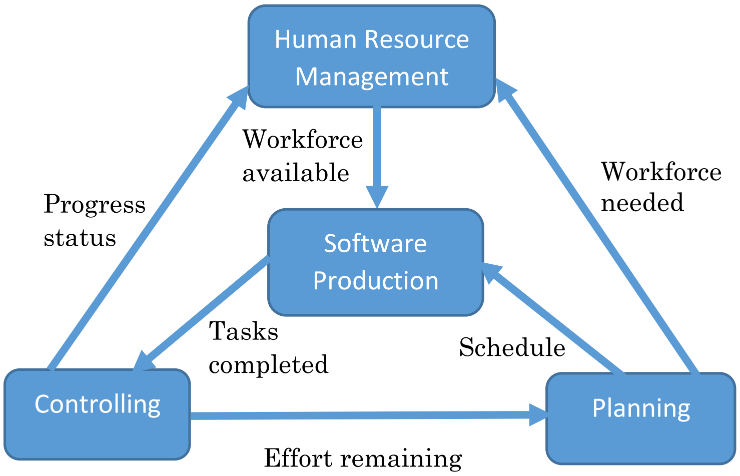


Fig. 3.2 Software Development Project Subsystems (adopted from Abdel-Hamid and Madnick, 1989).

The *Planning* subsystem defines project estimation activities and directly affect both the project schedule and the workforce are required to develop the software. Estimations are initially happened and then updated gradually throughout the life-cycle of the project. These estimations create the environment to manage interventions, such as adding more engineers to the project or adjusting the schedule, and thus can lead to some of the unintended project behaviors.

The *Software Production* subsystem includes four activities: development, quality assurance, rework, and testing.

The *Controlling subsystem* represents the activities through which project managers can track progress status in development activities.

3.5 Software Project Management and Uncertainty

Software development projects are notorious for their high failure rate (Julie et al, 2010). One of the main reasons for this failure is uncertainty which includes requirements uncertainty, rework, and also interpersonal conflict.

Requirements uncertainty is a process-oriented feature caused by lack of process control. Uncertainty often changes the nature of environment which ultimately results unstable requirements and this instability crates conflict among stakeholders and project managers and thus results a negative impact on decision -making.

Rework is another principle cause of uncertainty in project performance. Rework represents the error-based tasks of a project that needs extra effort to accomplish. Generation of rework is very uncertain and it can result several factors such as error, omission, schedule and cost overrun, project failure, insufficient communication and coordination. Rework could causes disruption delays on project performance, adversely affect productivity and overall project development environment(Lin Wang et al, 2017). Hence proper management of rework is very necessary for project success.

3.6 System Dynamics (SD) Method

The general explanation of System Dynamics approach has stated below(Sterman, 2000).

Software development management has proven as a challenge over many years due to the complex development and production environment. To improve our understanding of the mechanisms and provided policies of the environment, a method is needed to be deployed which allows to model and understand the behavior of the system. The method must be able to define complex systems and simulate the relationship between variables against time.

The System Dynamic approach is a method that focuses on the above-mentioned necessity. The approach provides an understanding of complex dynamic systems over time. This is achieved through the internal feed-back loops and time-delays that will influence behavior in the system as a whole. Hence, the system dynamics approach allow to build and test policies and assumptions to improve understanding of system behavior or to change the observed behavior (Sterman, 2000).

Three building blocks in System Dynamics are essential for modeling behavior and providing policies - feedback and causal loops, stock and flows, and delays.

3.6.1 Feedback and Causal Loops

Feedback is essential in System Dynamics. All dynamics arises from the interaction of feedback. Two kinds of feedback loops used in System Dynamics: positive and negative feedback loops.

The positive feedback loops reinforce the processes that are obvious in the system. The negative feedback loops are self-correcting systems that counteract the change. It is very important to understand the interaction among the feedbacks. The complexity of a system is not derived from the number of variables or complexity of structures, but from the feedback relationships between components and variables of the system (Sterman, 2000). The track and concept of the feedback loop are designed by "*Causal loop Diagrams" (CLDs).* CLDs are excellent for designing hypotheses about causes of dynamics. A causal diagram consists of variables connected by arrows which shows the causal influences and inert-relation among variables. The important feedback loops are also identified in such a diagram, and variables are related by causal links. The pictures below represents the different components of a causal loops diagram (CLD).

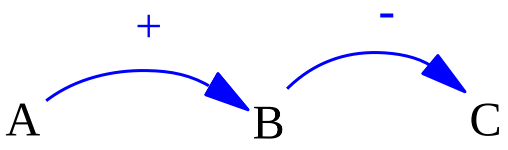


Fig. 3.3 Positive and Negative Relation

In the figure 3.3, we can see that two kinds of arrows used in a causal loop diagram to define the relationship between two variables. The polarity is given by the + or – sign at the arrowhead. When there is a positive relationship between two variables, means an increase in one variable increases in the other. If there is a negative relationship between the variables, it defines that an increase in one variable decreases in the other. The little CLD provided below shows a causal loop between two variables.

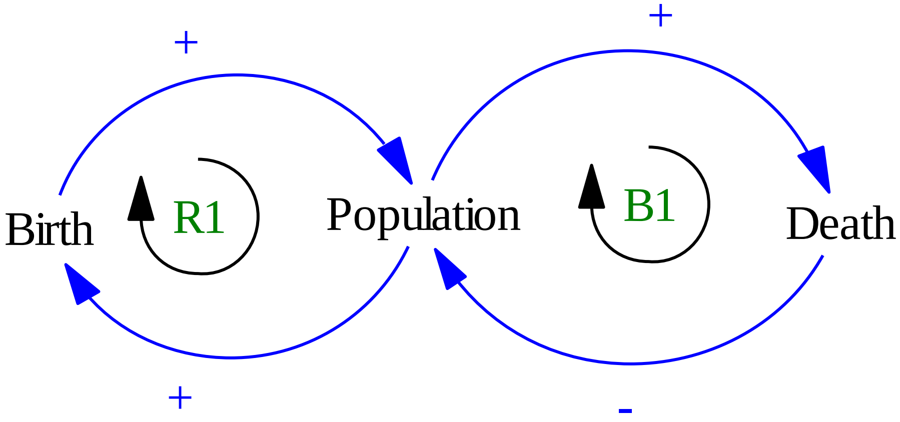


Fig. 3.4 CLD with Positive and Negative Feedback Loops

The figure shows the relationship among the population and births and deaths of population. Increase in population increases the number of fertile humans and thus more births will occur. With more births the population will also increase. When population increases, more humans reach old age and will die. So, increased population leads to an increased death-rate, and the overall population will decrease. This behavior provides the reinforcing and balancing aspects of a CLD, and shows how two loops are causing the behavior in the population-stock. In the diagram, R1 represents reinforcing and B1 represents the balancing loop.

3.6.2 Stocks and Flows

The second core concept of System Dynamics modeling is stock and flow. Causal loop diagrams are useful to understand the causal relation among variables. However, this relation is explained in a qualitative way. Hence, to define a system in a quantitative way, stocks and flows are required. Stocks are represented as accumulations, and define the state of the system. Stocks generate information based upon time. Stocks provide a system with memory as well as inertia and are the source for the delay. Stocks provide this delay by assembling the difference between inflow and outflow to a process(Sterman, 2000).

The following figure 3.5 shows a simple stock-flow diagram. Stocks are depicted as squares, while the inflow and outflow are depicted as pipes entering or leaving the system.

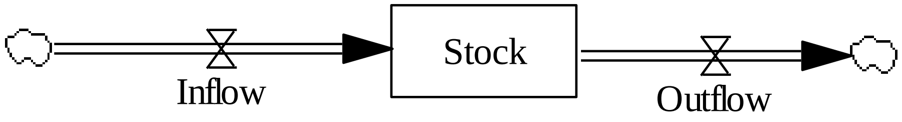


Fig. 3.5 Simple Stock and Flow diagram

The stock of the system is pictured in the square, and the pipes with valves control the inflow and outflow. The clouds at the beginning and end of inflow/outflow illustrate variables or processes outside the model boundary.

In a System Dynamics model, there are three kinds of variables that are utilized when a system is built and simulated.

1) A constant is a variable that is initialized at the first time-step of a model and kept constant during the simulation run.

2) The stocks of a system are initialized at the beginning of a simulation. Stock changes its value with time over the model run. The operating inflows and outflows will handle this value.
3) Auxiliary variable is the another variable in a stock and flow model. These variables calculate information and forward it through the system. The auxiliary variables are calculated by each time-step through the model run. Auxiliaries are used to control the flow-rates of a stock.

3.6.3 Delays

Delay is an important source of dynamics in every system. Delay is a process whose output remains behind of its input. One good example to understand is the process a letter is being sent from the sender to the recipient. The time takes between the letter has been put in the letterbox to arrives in the box of the recipient illustrates how letters in transit constitute a delay. Measuring and reporting the result takes time, decision making takes time, and it takes time before decisions enter into effect in the system. Therefore it is important to understand how delays behave and how they can be represented(Sterman, 2000).

Chapter 4 Proposed Methodology

In this chapter, at first, a basic description of project planning has been described that how alternatives can be obtained through the proposed methodology. After that the System Dynamics model development for software development project management has been explained. As described in the literature review, while developing a System Dynamics model, major features of software development and project dynamics must be considered to get the dynamic feedbacks through which decisions can be made to manage project performance.

4.1 Project Planning Model

After getting a project, the project manager undertakes project planning. Project planning is undertaken and completed before starting the development activity which consists of several functions(Software Project Planning, module 11).

- ➢ Estimating the attributes of the project
  - ✓ Project size
  - ✓ Cost
  - ✓ Duration
  - ✓ Effort
- ➢ Scheduling manpower and other resources
- ➢ Staffing plans

In principle, to measure the cost and duration of the project, a project manager has to focused on project size and based on the project size how much effort is needed, how to distribute the resources and plan for staffing and other control actions.

Through our proposed method, we have shown designing a project plan through measurement of cost and duration based on staffing planning and overtime planning by analyzing several alternatives. Among the alternatives, one feasible option would be chosen for planning and developing the project. The simulation flow is as follows in figure 4.1 .

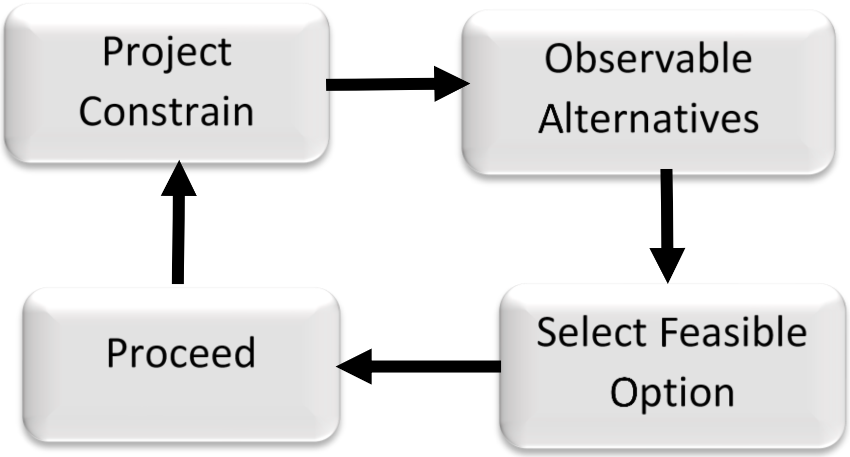


Fig. 4.1 Flow of Project Planning

Based on the project constrains, a set of alternatives would be designed and measured the project completion time and cost. Among the alternatives, a feasible option can be chosen for planning the resources of the designed project and proceeded for development of the project.

4.2 Overview of Proposed Methodology

The following figure 4.2 represents an overview of the proposed methodology.

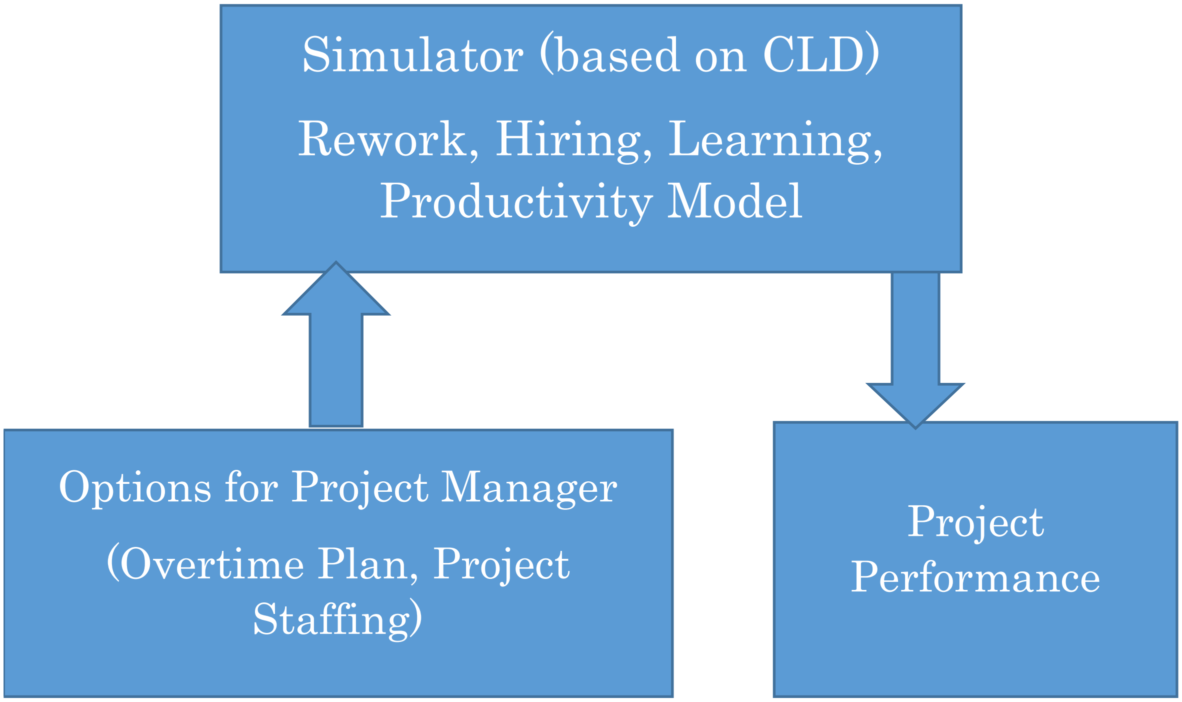

Simulator (based on CLD)
Rework, Hiring, Learning, Productivity Model
Options for Project Manager
(Overtime Plan, Project Staffing)
Project Performance

Fig. 4.2 Overview of the Proposed Methodology

This diagram represents the aggregation of input, output, and basic process to obtain the desired objective. The detailed description of each part of the proposed methodology has given below.

4.3 Options for Project Manager

To complete a project in time, several options can be chosen by project manager during the development phase if the resources are not enough. And the common options are hiring and working overtime.

Hiring defines adding new workforces to the development team. If there is a shortage in human resources to complete a project within given resources, and the project manager intends to add new members in the team, then they can request for new workforces. But hiring workforces is a time consuming process since it involves many employment procedure. On the other hand, if the project manager doesn't want to add new workforces and continue with the remain workforces, then, no hiring will happen. In this case, if the resources are not enough, then deadline slippage may occur.

The other option to get into the deadline is working overtime. Overtime defines the amount of additional time a workforce perform beyond the nominal working hour. Working overtime is a very common situation all over the world. In Japan, workweeks that exceed 60 hours are no exception (Debby et al, 2004). Working overtime typically depends on the decision of the project manager. This overtime can be categorized in two ways - interval-based overtime and continuous overtime.

- ✓ Interval-based overtime represents that workforces will work overtime for a certain period. After that they will work with nominal working hour and then again do overtime if necessary.

- ✓ Continuous overtime represents that workforces will do overtime in a continuous way based on the necessity until the project is finished.

4.4 Design of Simulator

4.4.1 Overview of Project Manager's options as Causal Loop Diagram (CLD)

In a software development project, several factors and parameters are interrelated to each other. The major considerations for analyzing project performance are human resource, controlling, and planning. Human resource management involves the hiring policy of new full-time equivalent workforces, controlling and planning to define the schedule maintenance with overtime or changing completion date. Considering these factors, the basic causal loop diagram (CLD) has been designed as showing in the following figure 4.3.

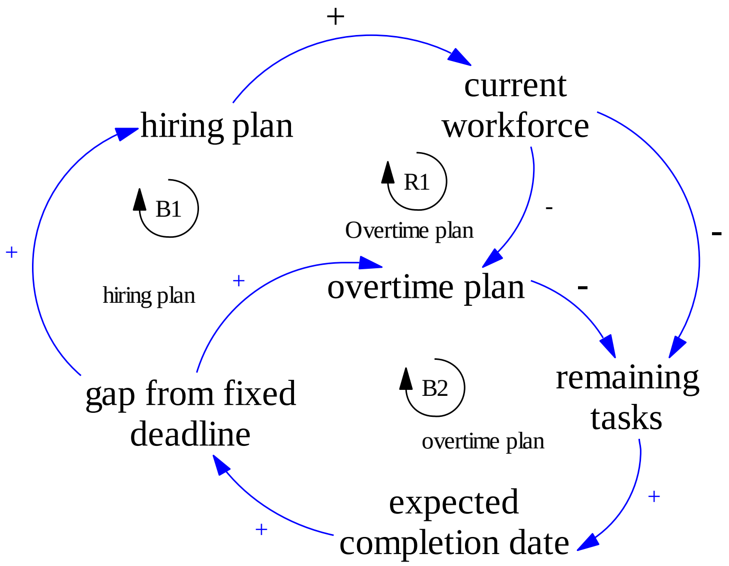


Fig. 4.3 CLD for Hiring and Overtime Management

This figure 4.3 represents the very simple and basic causal loop diagram for hiring and overtime. The feedback loop B1 represents the basic hiring model. More tasks require more time to accomplish which increases the expected completion date. An increase in expected completion time also increases the gap form the deadline. And to keep the deadline fixed and complete all tasks in time, hiring happens which in turn increases the total number of workforces.

Other feedback loops B2 and R1 represent the method of considering overtime policy depending on the expected completion date and current full-time equivalent workforce.

4.4.2 Uncertainty by Rework

The rework cycle is the most important feature of System Dynamics project model and defined as a canonical structure and defined as. The basic structure of the rework cycle, adopted from (Lyneis and Ford, 2007), is shown in figure 4.4.

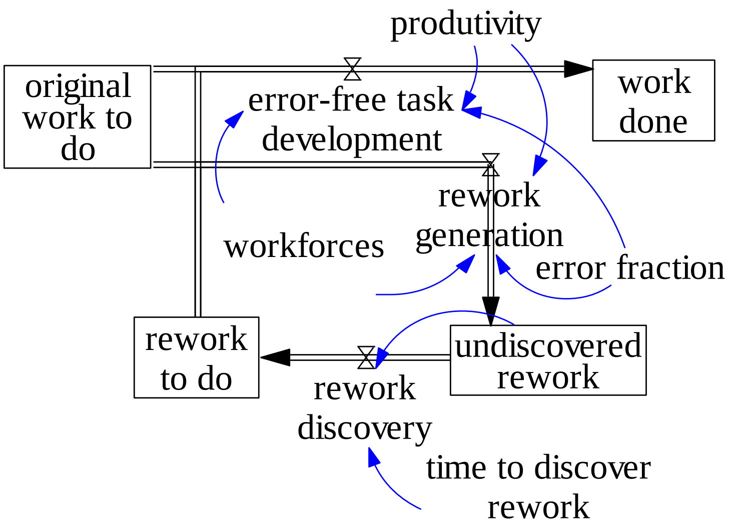


Fig. 4.4 Basic Rework Cycle(modified based on Lyneis and Ford, 2007)

The rework cycle includes four stock variables: *original work to do, undiscovered rework, rework to do, and work done*. Each project that is to be modeled, identified as a series of tasks that have a specific and predefined objective and are assumed to be consist of a certain number of tasks referred as *original work to do*. While developing the tasks, they can either be completed correctly or may contain an error. An error fraction measure that could vary within the range of 0 to 1, carries more or less of the works being done into the rework cycle. The tasks are developed correctly and without error, generated by the flow *error-free tasks development*, go to the stock *work done*. And the error based tasks go to another stock *undiscovered rework*, generated through the flow *rework generation* until it is discovered. After discovering through the flow *rework discovery*, they become *undiscovered rework* to *rework to do* which needed to be developed again. The rates at which work is being completed correctly and incorrectly are affected by the value of the parameters' original work to do and workforces, productivity, and error fraction.

The auxiliary variables productivity, error fraction are affected by several other variables such as schedule pressure, hiring new workforces, exhaustion level due to overtime, schedule pressure, and so on.

General equations for rework cycle variables have given in the following table 4.1.

Table 4.1 General Equations for Rework Cycle Variables

Variable type	Variable name	Equation	Unit
Stock	$Original\ work\ to$	$\int(-errorfree\ task\ development$ $rework\ generation)dt$	task
	$Undiscovered\ rework$	$\int(rework\ generation -$ $rework\ discovery)dt$	task
	$Rework\ to\ do$	$\int(rework\ discovery -$ $errorfree\ task\ development)dt$	task
	$Work\ done$	$\int(errorfree\ tasks\ development$	task
flow	$errorfree\ tasks\ develop$	$productivity * workforces *$ $(1 - error\ fraction))$	task/time unit
	$rework\ generation$	$(productivity *$ $workforces *$ $error\ fraction)$	task/time unit
	$rework\ discovery$	$((undiscovered\ rework)/$ $(time\ to\ discover\ reowrk))$	task/time unit
Auxiliary	$productivity$	$nominal\ productivity *$ $impact\ of\ overtime *$ $impact\ of\ mix\ workforces *$ $impact\ of\ communication\ overh$	task/time/person
	$error\ fraction$	$nominal\ \ error\ fraction *$ $impact\ of\ overtime *$ $impact\ of\ mix\ workforces *$ $impact\ of\ schedule\ pressure *$ $impact\ of\ error\ density$	Dmnl

4.4.2.a Error Fraction and Rework Generation

Error fraction represents the percentage of accomplished work that contains an error. As mentioned before, tasks developed in the first attempt are not all accomplished accurately. Error fraction represents the fraction of error based tasks and based on this fraction, rework generation happens that are determined by *rework generation* flow in the above figure 4.4. And finally, the error must be discovered before they can be corrected and this is modeled with the rework discovery rate.

Several factors affect error fractions such as organizational factors, project factors, although these factors are different from project to project but remain invariant during the life of a single project (Albert DS, 1976). The combined effects of all such factors has represented as a single nominal variable in the model, *Nominal Error Fraction*. And since this nominal variable represents different types of errors in the model, the value is not constant but changes over the project development and decreases gradually depending on tasks development and comes to the point zero when all tasks are developed(Suinan Li, 2008). The value of the nominal error fraction is a function of project accomplishment and the highest value could be 25% (Tarek A. Hamid, 1984), (Raymond, 2007)(Suinan Li, 2008).

Based on this consideration, we have assumed the nominal error fraction as shown in figure 4.5. In the figure, the amount of nominal error fraction has been shown through vertical axis with respect to time in horizontal axis.

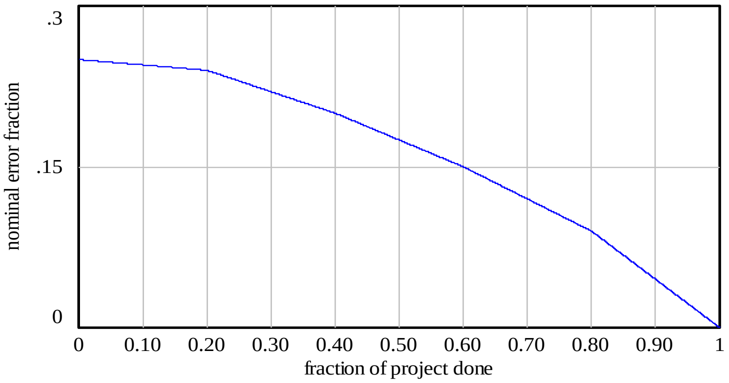


Fig. 4.5 Nominal Error Fraction

4.4.2.b Undiscovered Rework and Rework to do

Through the rework generation flow, error-based tasks are generated and stored in the stock *undiscovered rework*. *Undiscovered rework* gives non-linear behavior. The value of this stock increases gradually to a peak value and decreases when most of the tasks are developed. Depending on the time required for discovering rework, the rework is discovered and transform from *undiscovered rework* to *rework to do*. *Rework to do* ascertains the error-based tasks that are ready to be developed and requires extra effort.

4.4.3 Hiring Model

For hiring new Full-Time Equivalent Workforces, a request is being proceeded based on the labor resource deficit. Labor resource deficit is calculated through the current Full-Time Equivalent Workforce and desired Full-Time Equivalent Workforce required depending on estimated completion time. As mentioned before, hiring new workforce is a time consuming process and this time is defined as hiring adjustment time or hiring delay. The value for this hiring adjustment time has been set to 60 days in the simulator, however this value is can be changeable.

This value is formulated as

$$hiring\ adjustment\ time = 60\ days$$

And to execute the hiring process, the following consideration has been embedded.

- Based on the deadline (fixed), expected completion date, and the current Full-Time Equivalent Workforce, the labor resource deficit is calculated. According to this labor resource deficit with including hiring adjustment time, new Full-Time Equivalent Workforces are hired.

Based on the project status, hiring options can be considered in two ways, no hiring and hiring. If the labor resource deficit is low or the project manager wants to apply other control actions such as working longer periods instead of hiring, then the hiring process will be ignored. Otherwise, workforces will be hired based on necessity. The following Figure 4.6 shows the options for hiring.

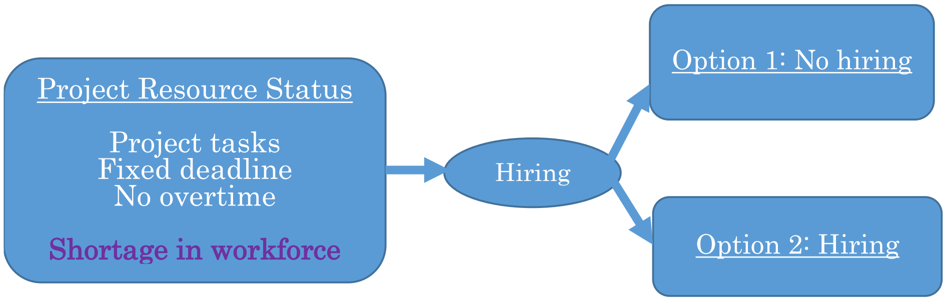


Fig. 4.6 Options Consideration for Hiring New Workforces

4.4.4 Non-Linear Behavior of Learning

Hiring new workforces increases the total number of developers in a team but at the same time decrease productivity. The productivity of new Full-Time Equivalent Workforces is half the productivity of expert Full-Time Equivalent Workforces [16]. To obtain the same productivity, new Full-Time Equivalent Workforces need to be trained and the duration for training is referred to as a learning point for new Full-Time Equivalent Workforces. Figure 4.7 represents the typical learning curve for new Full-Time Equivalent Workforces based on the time series data. In the graph, the vertical axis represents the amount of obtaining productivity with respect to time given in horizontal axis. New workforces joining in a development team, their productivity is lower comparing to the workforces who are already in the development team. Hence, they need to be learnt and trained to increase their productivity.

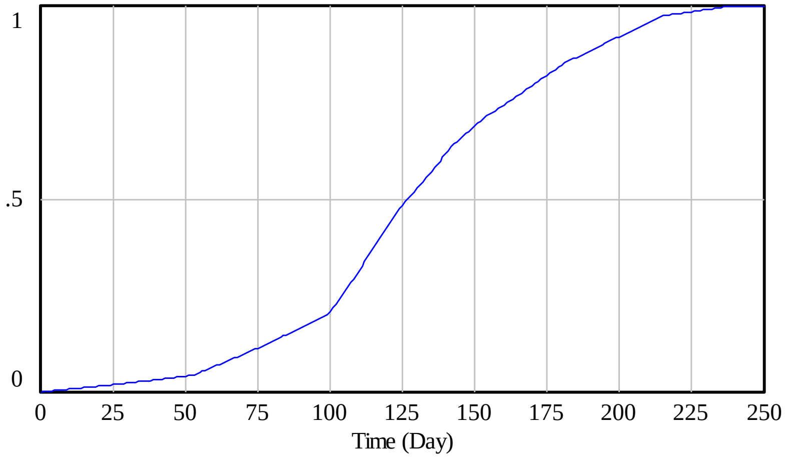


Fig. 4.7 Learning Curve for Full-Time Equivalent Workforces

The learning curve for new workforces can be calculated as follows to obtain high productivity.

$$learning\ rate = hiring\ rate * time * (ovarall\ prodcutivity - prodcutivity\ of\ new\ workforces)/(assimilation\ delay)$$

The following figure 4.8 is used as a reference for defining the behavior of the learning curve.

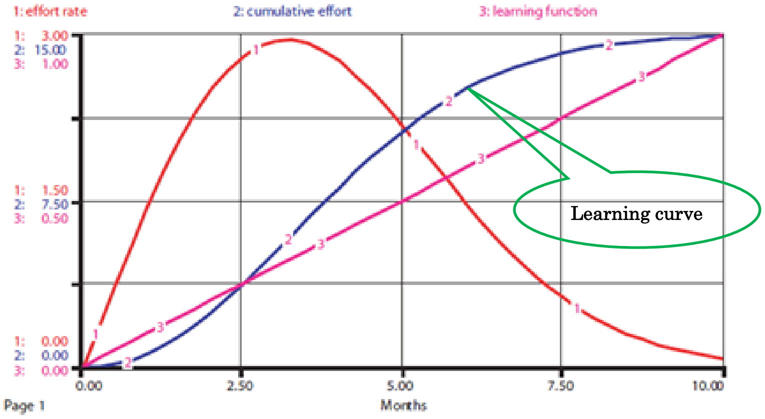


Fig. 4.8 Reference Graph for the Learning Curve[11)]

4.4.5 Overtime Planning

While modeling for overtime management plan, several parameters have to be considered, both impacting and impacted. The process of overtime plan has been defined as follows:

- ✓ Based on the deadline(fixed), expected completion time, current Full-Time Equivalent Workforces, the shortage in man-days has been calculated. After that, based on this shortage amount of man-days and how much shortage can-handled in 1 day, the work-rate is boosted and increases the overtime plan. The following figure 4.9 gives an overview of planning overtime.

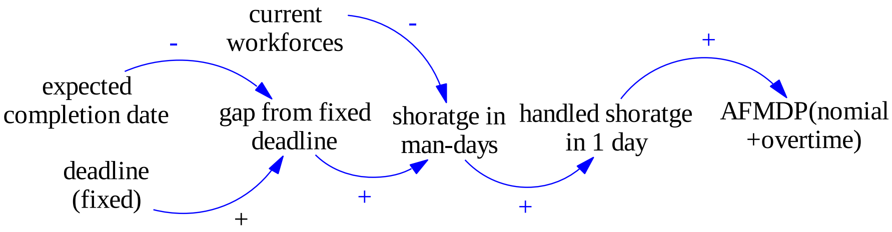


Fig. 4.9 Planning Overtime

Although usage of overtime depends on organization policy and project manager, two different overtime plans and its consequences on project performance has been explored, (i) using interval-based overtime (ii) using continuous overtime.

4.4.5.a Overtime plan #1: Using Interval-based Overtime

Interval-based overtime interprets working overtime for some period and taking an interval from working overtime and work with a nominal working hour and then again working overtime when the workforces will be out of fatigue and also depending on the necessity for project completion.

While considering the interval-based overtime plan, several authors have talked about the range of choosing overtime. The average overtime work per week should be 3 - 4.5 hours and overtime week should be between 8 – 12 weeks (Debby et al.

2004)(Armindokht, 2012). After this period, engineers will work with the nominal working hour for a certain period and do overtime if necessary.

Keeping in mind these options, for this overtime plan #1, followings are assumed, (i) workdays with overtime should be shorter than 50 days, (ii) after workdays with overtime, an average of 30 workdays without overtime is required. The nominal fraction of one workday (AFMDP) is 1.0 and 1.35 in average for workdays with overtime. However, this value represents the average and can be changed if the workforces need to work more for project completion. The project manager pushes the Full-Time Equivalent Workforce to do overtime if there is a gap between the expected completion date and the fixed deadline. Figure 4.10 gives an overview of understanding interval-based overtime.

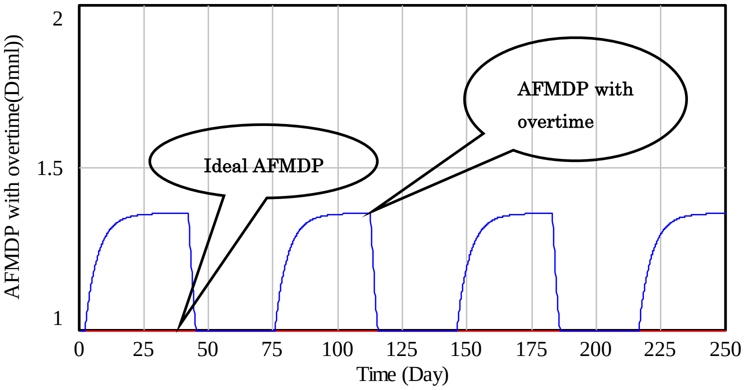


Fig. 4.10 The AFMDP with Overtime Considering Interval-based Overtime

4.4.5.b Overtime Plan #2: Continuous Overtime

Continuous overtime specifies to work overtime continuously based on necessity until the project is finished.

For this overtime plan #2 followings are assumed, (i) Workdays with overtime can continue without limits until the project finished. The nominal fraction of one workday is 1.0 and 1.2 in average for workdays with overtime. The project manager pushes the Full-Time Equivalent Workforce to do overtime if there is a gap between the expected completion date and the fixed deadline. The following figure 4.11 illustrates the picture of applying continuous overtime.

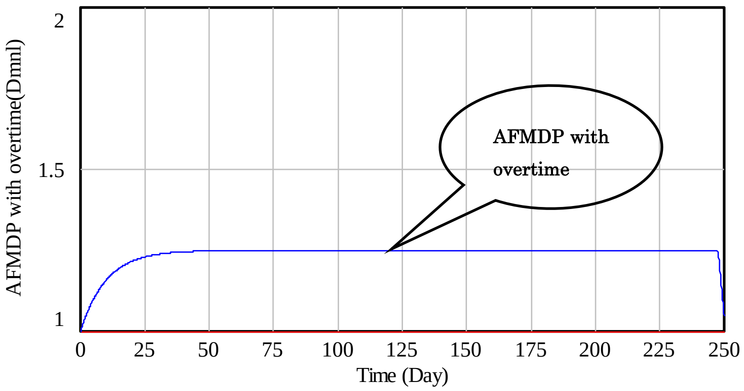


Fig. 4.11 The AFMDP with Overtime Considering Continuous Overtime

4.4.6 Non-Linear Behavior of Overtime and Productivity Model

Productivity is usually a function of three factors namely fatigue which is an outcome of overtime, the ratio of new to experienced personnel, and communication congestions among the workforces. As described in hiring model that productivity is affected when new workforces are joined in a team. Again, while applying overtime in a project, it increases the progress rate since extra man-hour is added in the development time. but at the same time, it decreases productivity gradually when the range for overtime increases and when they get exhausted. Also when the hiring process happens, due to the fraction of experience, productivity decreases. As a result, the actual productivity becomes different from the nominal productivity. When productivity decreases, it affects project performance. The following figure 4.12 is the assumption for overtime and productivity relationship. As the overtime happens, productivity decreases in accordance. The vertical axis of figure 4.12 defines how productive is affected (decreasing) while working overtime. The assumption is a simplified version of figure 4.13 (Chul-Ki Chang, 2017).

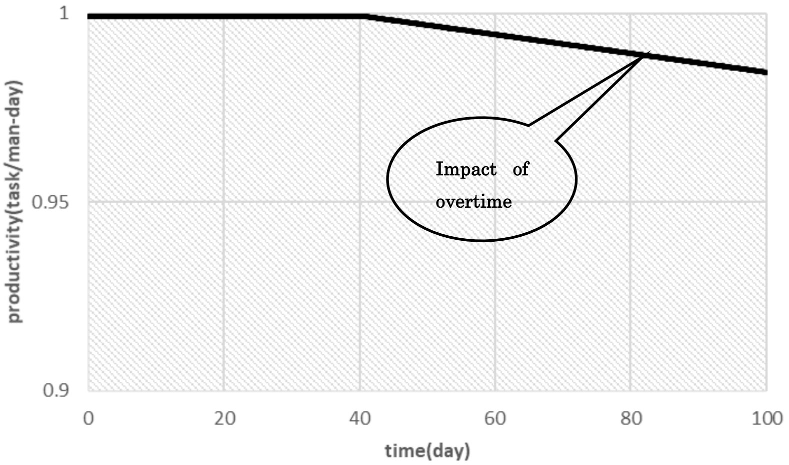


Fig. 4.12 Impact of Overtime on Productivity

This productivity can be formulated as

$$productivity = nominal\ productivity$$
$$* productivity\ loss\ due\ to\ fatigue(impact\ of\ overtime)$$

The following Fig. 4.13 is used as a reference for the behavior of productivity while using overtime.

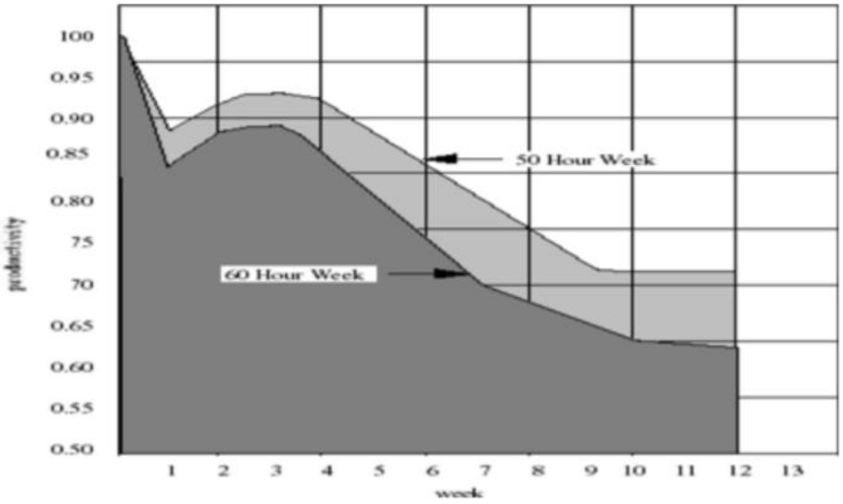


Fig. 4.13 Reference Graph for Overtime Affecting Productivity[23)]

4.4.7 Actual Fraction of Man-day in a Project (AFMDP)

A man-day is regarded as the number of working hours one person use for productive work in a day.

The ideal fraction of Man-day in a project(IFMDP) is defined as 1 which represents the nominal working hour per day. When overtime is considered, the value of overtime added to AFMDP and increases the value of AFMDP.

- When the AFMDP is 1, there is no overtime. If it crosses 1, this will count as overtime.
- The average overtime work per week is 3 – 4.5 hours, that is when the nominal AFMDP is 1, with overtime its average value would be 1. 35 – 1.56 (Debby et al, 2004)
- The maximum AFMDP maybe double of NFMDP, which is considered as moderate (Tarek A. Hamid, 1984)
- Once increased AFMDP goes beyond 1, people are supposed to do overworking. In this working overtime period, the increasing value of AFMDP affects Effective WF

To catch up with the schedule, employees boost the work rate by working overtime. And when the work overtime, the Actual Fraction of Man Day on Project (AFMDP) increases from its nominal value.

But when people work in the beyond nominal working hours, they get exhausted. This exhaustion level negatively affects productivity that has shown in Fig. 4.12.

4.5 Project Performance

Several factors are involved to prescribe project performance. The major factors that we have considered during our work are given in the following table 4.2.

Table 4.2 Project Performance Variables

Factors	Measures	Measuring Unit	Benefit
Productivity	Working capability	Number of task per time	Represent the completion of a project
Project Schedule	Time required for project completion	Planned and actual completion	Define the time of using software

1. **Projects' Overall Productivity:** One of the key factors for measuring project performance is productivity since productivity describes various measures of efficiency. While applying control actions like overtime, or hiring new workforces, affects productivity and thus affect the overall performance of the project. So, this variable plays a significant role in measuring project performance.

2. **Project Schedule (gap from fixed deadline):** Schedule is another important measurement of project performance. Completing projects in time is the goal for every project manager. To obtain better project performance, a project should be completed within the deadline. Since productivity fluctuates due to hiring, and working overtime, it affects the progress rate as well. As a result, the project schedule also varies from the actual plan. To close the performance gap, we have to manage the control actions properly, so that the deadline cannot be slipped. Figure 4.14 pictures the sample of project performance behavior in terms of task accomplishment on time.

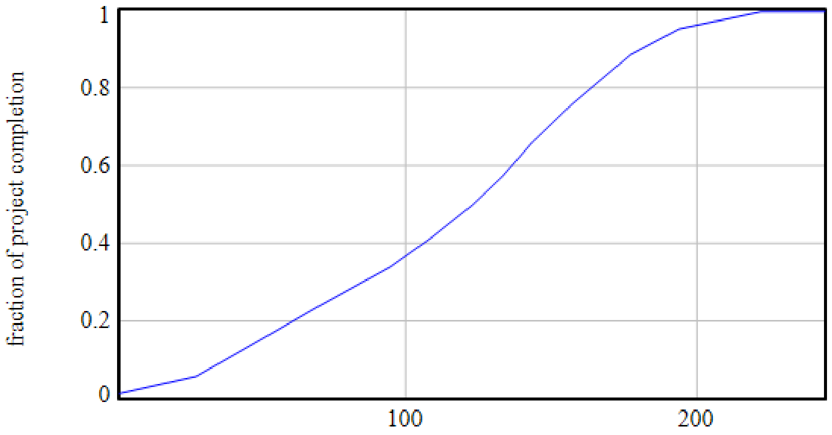


Fig. 4.14 Project Completion Following the Scheduled Time

Chapter 5 Model Validation

5.1 Validation Process

Validation is the most important part after developing a simulation model. It is a way to define the usefulness of a model. Although it is said that there is no model that can be verified and validated because many assumptions are made in the model (Sterman, 2002). In System Dynamics, validation requires not only the model is following the known "physical laws" of the system, it also offers usefulness and provides confidence to the end-user. The validation process of a model gives the usefulness of the model in terms of informing decision-making and to help to address the management of model risk (North Amarican CRO Council, 2012)

For model validation, a variety of tests can be performed (Raymond, 2007). A list of a validation test is given below.

Table 5.1 A List of Model Validation Tests

Test	Focus	Criteria
Dimensional Consistency	Structure	The measuring units of model variables must be consistent and logical with real-world meaning
Integration Error	Behavior	Model behavior sensitive by choosing different integration methods and time steps
Boundary Adequacy	Structure	The model structure contains variables and feedback effects for study
Parameter Sensitivity	Behavior	Model behaviors are sensitive to reasonable variations in parameters
Structural Sensitivity	Behavior	Model behavior sensitive to reasonable alternative
Statistical Test	Behavior	Model output behaves statistically with real system data
Face Validity	Structure	The model structure resembles a real system to persons familiar with the system

Among these test categories, *Dimensional Consistency* and *Integration Error* tests are collectively called a *Technical Validation* test, and the others are represented as *Model Behavior* tests.

5.1.1 Technical Validation

The purpose of this test is to define the sensitivity of the results in terms of choosing the time and numerical integration method. Technical validation includes dimensional consistency and integration error test.

The *Dimensional Consistency* test is one of the most basic tests for System Dynamics model validation. It focuses on the equations in the model, and the units that are derived from the calculations. The measurement unit for each variable included in the model must be logical with real-world meaning. Hence, every equation must be dimensionally consistent without the inclusion of arbitrary scaling that has no real-world meaning (Armindkkht, 2012). In our model, the dimension of the variables for each equation agrees with the computation. Simulation time units are days, the unit for project size are tasks and effort are in man-days. A detailed explanation of the units is given in appendix A.

The *Integration Error* test focuses on the usage of time steps and integration types during the simulation run. The model is identified as faultless while using different time steps and different integration methods give similar behavior.

5.1.2 Validation of Model Behavior

The usefulness of a model depends on its behavior. Several options for model behavior tests have been shown in table 5.1. And to perform those validation tests, different activities such as expert intuition, real system measurements, and theoretical result analysis can be used (Sterman, 2002)(North Amarican CRO Council, 2012). Among these actions, real system measurements is a comparison with a real system or assumption of a real system and have been defined as the most reliable and preferred way to validate a model.

5.2 Validation Overview of Proposed Model

For proceeding Technical Validation, we have done both dimensional consistency test and integration error test and concluded that the model is dynamically consistent and valid with the integration error test.

As for the validation of model behavior test, the real system measurement approach has been chosen. The purpose of this validation process is to reproduce the dynamic behavior of the system. And to perform this procedure, the dataset based on simple assumptions for software deployment projects has been considered. After that, validation has been done both for hiring and overtime management scheme separately with different datasets to provide a better understanding of several option sets for project managers following the project requirements and resources. In each scheme, the usage of the rework cycle has been considered along. The use of the rework cycle highly influences the time required for project completion due to the uncertain generation of rework. Through this validation procedure, the insights of applying hiring and overtime for several scenarios have been provided which would give us a clear understanding of model-based decision making.

For the model behavior test, the following two different validation has been performed.

i. Model Validation with Hiring and without Overtime

ii. Model Validation with Overtime and without Hiring

Through the validation process, different scenarios have been analyzed to get an idea of designing a proper project plan by estimating the duration and cost. Based on this estimation, a project manager could get understanding about the resource allocation and use of control actions that are needed during the development of the project.

5.3 Model Validation With Hiring and Without Overtime

In this validation process, we have considered the hiring plan for project completion based on the given resources, then analyzed and compared different cases to obtain a tangible option to meet the project goal.

5.3.1 Project Description and Scenario Settings

While considering hiring, we have chosen 3 different cases based on the number of initial full-time equivalent workforces. And each case has two different phases, one is with hiring and another is without hiring. The purpose of using a different number in initial workforces is to obtain the dynamic behavior of performance.

Hiring new workforces is a time-consuming process since several formalities are included. So after requesting for new workforces by project managers, they have to wait for a certain period and after that period, new workforces get involved in the project. As mentioned in section 4.3.3, the hiring adjustment time has set to 60 days. After 60 days, additional workforces are got hired and start to work on the project. The input parameters along with the cases for this validation process has given in table 5.2.

Table 5.2 Input Parameters for Model Validation with Hiring

	Input values					
	Case 1a	**Case 1b**	**Case 2a**	**Case 2a**	**Case 3a**	**Case 3b**
Project size(tasks)	1000	1000	1000	1000	1000	1000
Full-time equivalent workforces (initial) (person)	3	3	4	4	5	5
Deadline(fixed) (days)	250	250	250	250	250	250
Nominal productivity (task/man-day)	1	1	1	1	1	1
Nominal error fraction(Dmnl)	.25	.25	.25	.25	.25	.25
Nominal time for rework discovery (days)	20	20	20	20	20	20
Hiring	No	Yes	No	Yes	No	Yes
Overtime	No	No	No	No	No	No

For each case, the scope of the project is considered as fixed, which means the number of tasks is fixed till the project is finished.

5.3.2 Ideal Calculation with Input Parameters

If we do the simple mental or ideal calculation with the given input data, the time required for each case with the initial workforces is as follows.

With 3 Full-Time Equivalent Workforces:

Time required =(1000tasks)/(1 task/man-day*3 person)

= 333.33 days

With 4 Full-Time Equivalent Workforces:

Time required =(1000tasks)/(1 task/man-day*4 person)

= 250 days

With 5 Full-Time Equivalent Workforces:

Time required =(1000tasks)/(1 task/man-day*5 person)

= 200 days

With 5 full-time equivalent workforces, the project requires less time to be accomplished, and with 4 full-time workforces, it requires the exact time as the deadline. But these calculations are without considering rework and steady productivity. When we are considering rework, the generation of an uncertain rework often increases project schedule beyond the deadline and negatively affects project performance. Also, the value productivity is affected by several factors as we mentioned in section 4.4.2. Since the intend is to keep the deadline fixed and working without overtime, we have focused on hiring new Full-Time Equivalent Workforce and analyzed the performance behavior.

5.3.3 Result Analysis of Validation Process with Hiring and without Overtime

To obtain and state the project performance we have analyzed several parameters that are considered as the most important factor for analyzing performance behavior. In section 4.5, we have defined the parameters used to measure project performance. Along with those parameters, we have chosen different parameters for the detailed explanation of simulation results. The parameters include the actual fraction of man-day in a project(AFMDP), productivity, error-free tasks development rate, rework generation, undiscovered rework, rework to do, and the time required for the accomplishment of total tasks. The result for each of these parameters has been explained gradually.

5.3.3.a Actual Fraction of Man-day in Project (AFMDP)

Usually, when the workforces work according to the nominal working hour assigned by the organization, AFMDP is 1. But when the workforces work overtime, an increase in man-hour per day increases the AFMDP. Since in this validation process, we are not considering overtime, so for each case, AFMDP is always 1 as shown in figure 5.1. The vertical axis of the figure represents the AFMDP with respect to time.

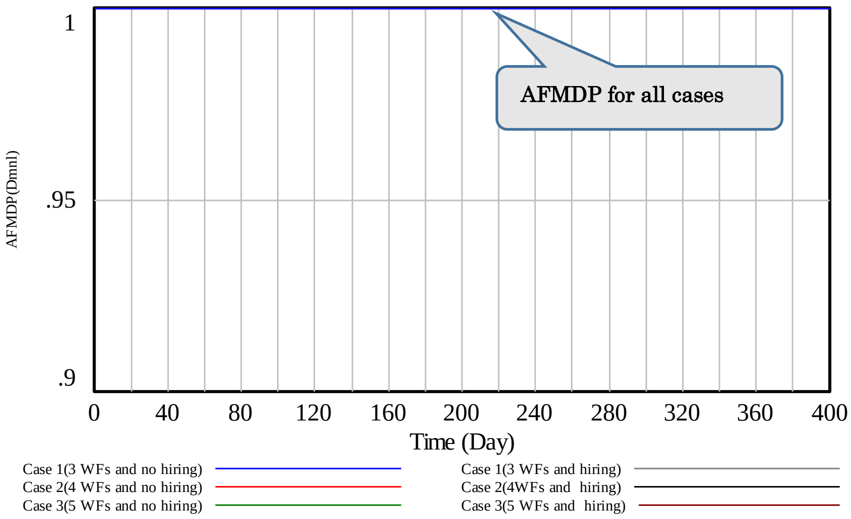


Fig. 5.1 Actual Fraction of Man-Day in a Project (AFMDP)

5.3.3.b Expected Project Completion Time

With the initial number workforces, we showed the expected project completion through ideal calculation in section 5.3.2. But that analysis was done without considering rework. When we are considering rework, the expected completion time for each case has obtained as follows in the figure 5.2. In the figure, the vertical axis represents the expected completion time for project.

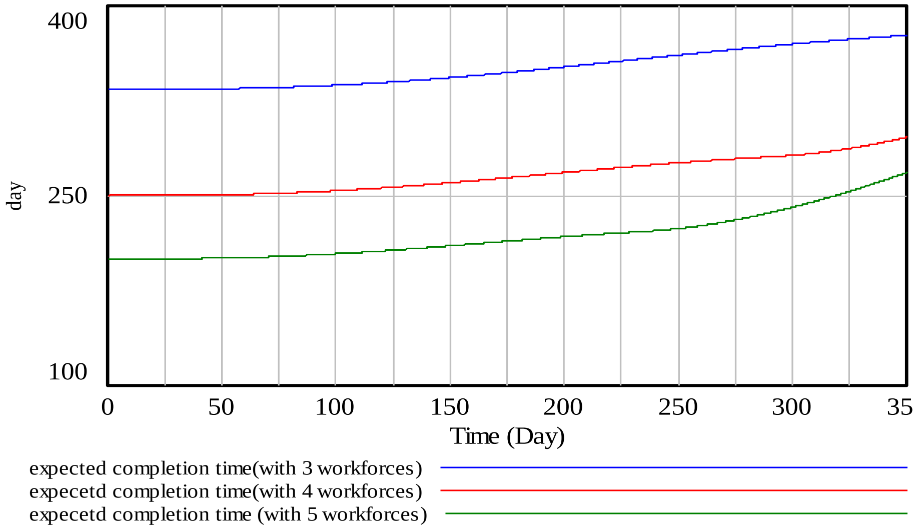


Fig. 5.2 Expected Project Completion Time Without Hiring

Through this figure, we can see that, considering rework and without hiring, the expected completion time remains within deadline when the number of workforces is high that is with *5 workforces*. In case of with *3 workforces* and *4 workforces*, the expected project completion time goes beyond the deadline at the very beginning. So in order to keep the completion time within deadline, hiring is required.

5.3.3.c Full-time Equivalent Workforce

In each case, for one phase we have considered hiring and ignored for another phase. So, when hiring has been activated, several workforces are hired for each phase following the desired workforces required for project completion as shown in figure 5.3. This figure represents how many workforces are hired for each phase when hiring is activated. The vertical axis counts the number of workforces and horizontal axis represents the time.

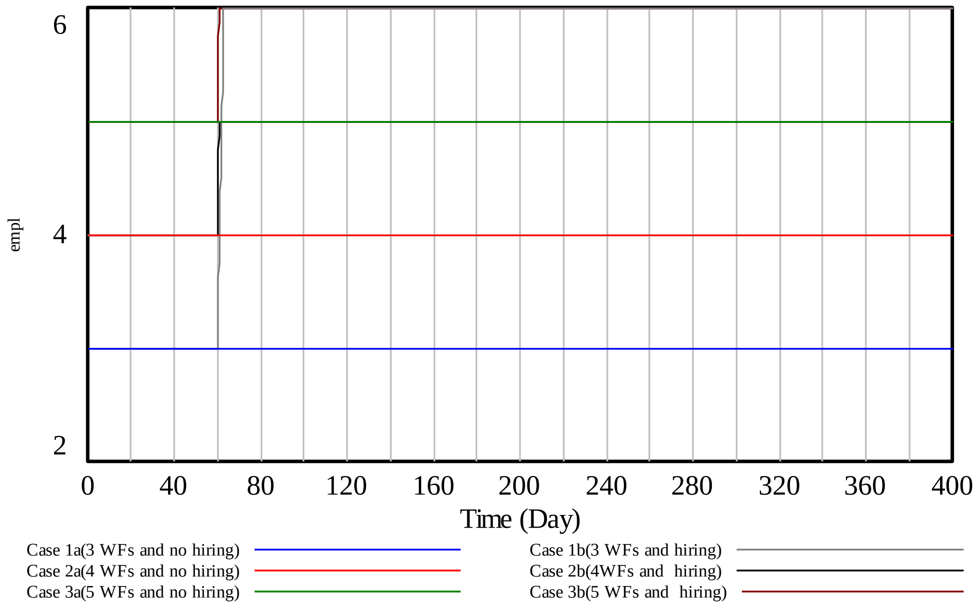


Fig. 5.3 Full-Time Equivalent Workforce After Hiring

In this figure we can see that, when the project starts with 3 initial workforces, since the number of staffs are comparatively low to meet the desire performance, so, 3 more workforces have been hired to complete the project. On the other hand, the project starts with 4 workforces and 5 workforces, since the number of human resources is average according to the project requirements, 1 more workforce is hired for each phase.

The total number of full-time equivalent workforces after hiring is given in table 5.3.

Table 5.3 Total Workforces After Hiring of Each Case

	Initial workforces (persons)	Hired workforces (persons)	Total workforces after hiring (persons)
Case 1b (hiring)	3	3	6
Case 2b (hiring)	4	1	5
Case 3b (hiring)	5	1	5

5.3.3.d Productivity and Task Development

Productivity defines the measure of efficiency. In project development, it represents how much tasks a person can perform in a certain period. During our simulation analysis, the nominal productivity has been assumed 1 task/man-day, which means 1 task can be developed each day by one person. When the workforces are constant from the beginning of the project, the overall productivity remains constant. But adding new workforces after starting to develop the project, affects productivity. Since they are new and don't have much idea of that project, their productivity remains low. So after hiring, they need to be trained to know about the project. And after hiring to before getting trained, their productivity remains low and reduces the overall productivity as shown in figure 5.4. In this figure productivity is represented by Y-axis and X-axis represents the time through which the impact of hiring has been shown in productivity.

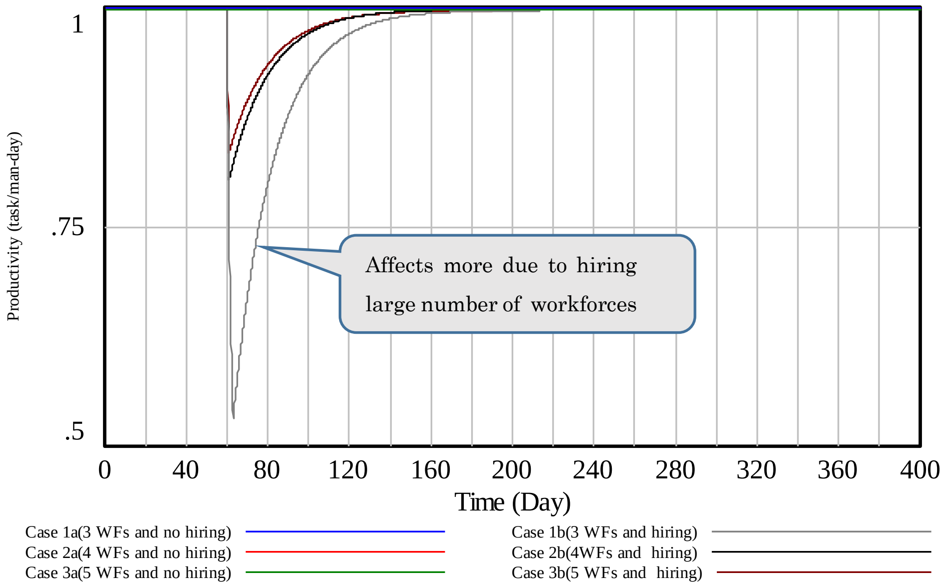


Fig. 5.4 Impact of Hiring on Productivity

This figure represents the overall productivity in a time series data. As we can see productivity tends to decrease when newly hired workforces are added to the project and start to increase again when they are getting trained, and after getting trained the productivity remains high again. For *case 1b (with 3 workforces and hiring)*, productivity gets lower comparing to 2 other phases due to hiring more workforces.

Following the productivity, workforces develop the tasks of the project. While developing the task, due to the rework cycle, error-based tasks are generated along with error-free tasks. That's why at the beginning, the rate of error-free task development is lower as compared to the mental calculation. But as time goes, most of the tasks are developed and it increases the tasks development rate as shown in figure 5.5. In the figure, the vertical axis represent how many error-free tasks can be completed in each day and the horizontal axis represents that time. The amount of developed tasks per day also depends on the number of workforces who are actively working on the project.

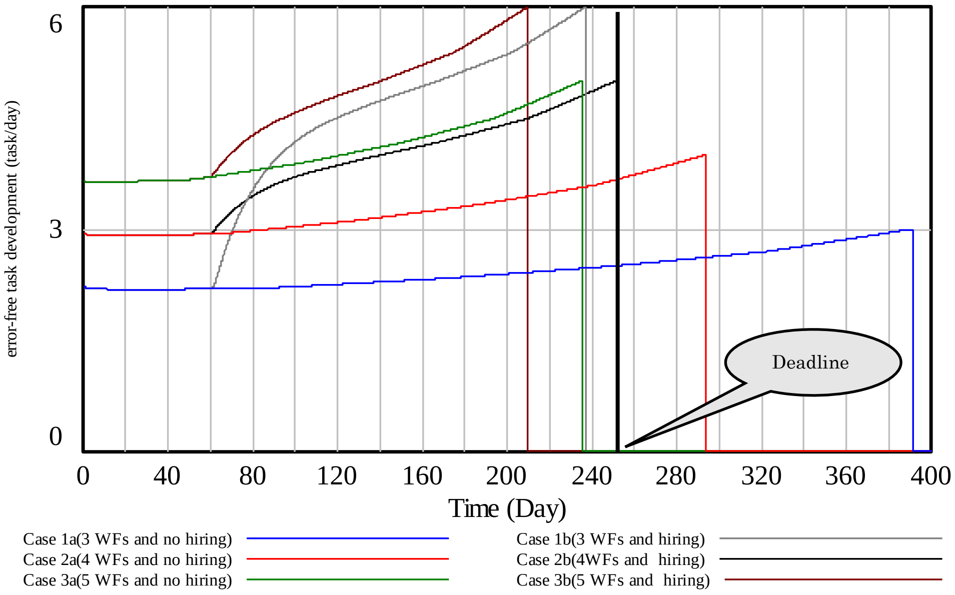


Fig. 5.5 Error-Free Tasks Development

More workforces develop more tasks and increase the number of error-free developed tasks in time. From the above figure 5.5, we can see that while having a large number of workforces, the rate of error-free task development increases, and it reaches the highest point when all the tasks are accomplished correctly. And after developing all tasks, it comes to the point zero to indicate project completion. Among the different phases of the scenarios, the result of four phases gives better performance since they are accomplished within deadline. These are – *case 1b (3 WFs and hiring), case 2b (4 WFs and hiring), case 3a (5 WFs and no hiring), case 3b (5 WFs and hiring).*

So through the analysis of figure 5.5, the project managers would get an obvious idea about hiring and its impact. Since this figure represent the amount of task development in each day based on the workforces and their productivity, the project manager would get an estimation of how many workforces are needed to be hired according to necessity if existing workforces are not enough for the project development. They can also interpret which scenario options can be applicable based on the given resources and can be completed within the deadline, not very early of the deadline, or exceed the deadline. And this will help to design a more accurate and feasible plan.

5.3.3.e Rework Generation

Rework generation causes error based tasks and is defined as rework. This is one of the major criteria of the rework cycle. Rework generation initially depends on the error fraction. As mentioned before, developed tasks are not 100% error-free at the beginning of the project. It may contain an error and based on this error rework generation happens and creates error based tasks. Developing more tasks may create more error-based tasks but after a certain period, it starts decreasing and goes to zero at the last when all of the tasks are developed as error-free. This behavior can be obtained from figure 5.6. The vertical axis represents the amount of rework generation with time while horizontal axis represents time.

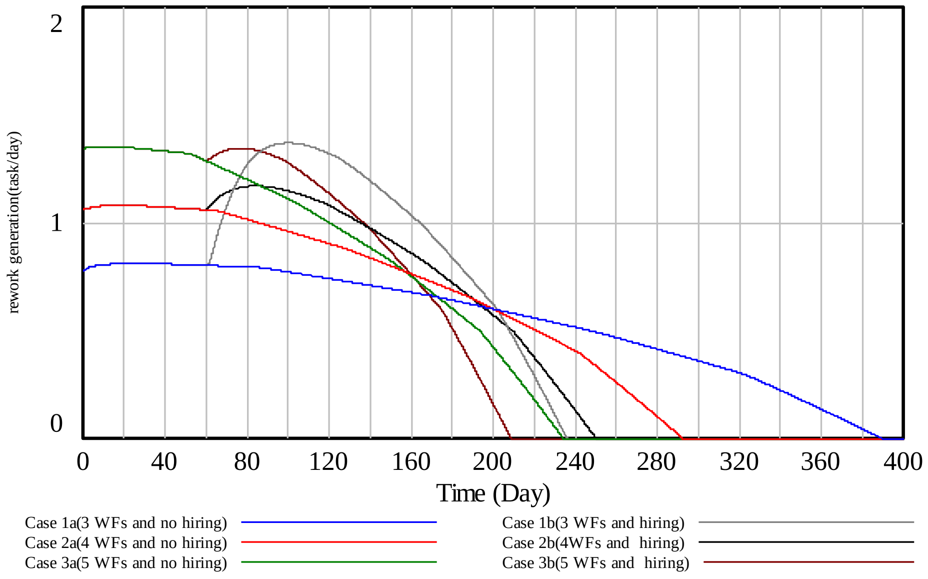


Fig. 5.6. Rework Generation

5.3.3.f Undiscovered Rework and Rework To do

Through the rework generation when the error based tasks are generated, they are stored as undiscovered rework stack initially since we don't know in which of the tasks we have to do rework. In Figure 5.7, the amount of undiscovered rework for each phase of all cases has been shown through vertical axis with respect to time in horizontal axis. Undiscovered rework is defined as the root of the propagation of the project schedule through the project.

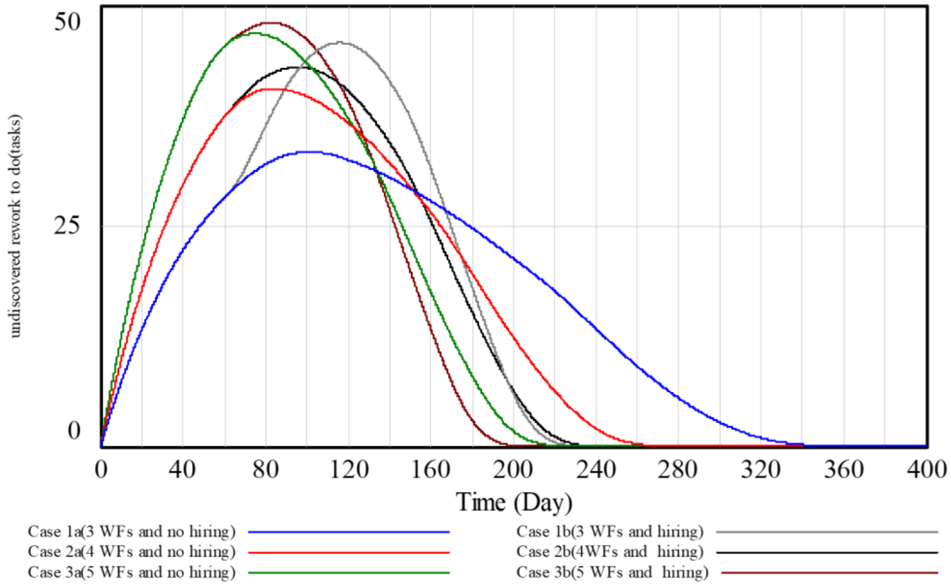


Fig. 5.7 Undiscovered Rework

Rework generation starts from the beginning of the project and the number of undiscovered increases gradually as shown in the above figure 5.7. After reaching the peak value the value of undiscovered rework starts decreasing due to discover the rework and identify them as rework to do as shown in figure 5.8. In the figure, we can see that rework to do happens after a certain period. This is because discovering rework requires time which is defined as *time to discover rework* as shown in the figure 4.4.

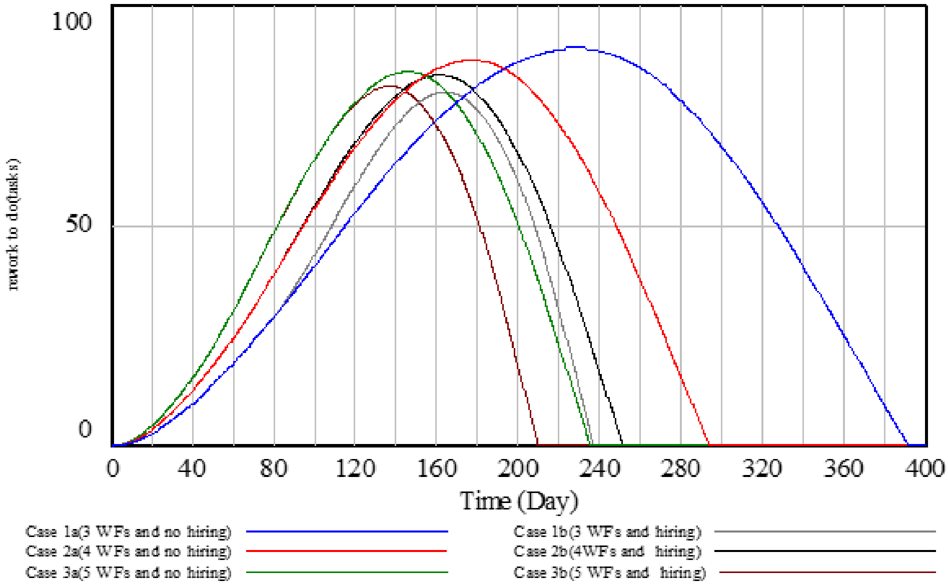


Fig. 5.8 Rework to Do

In this figure 5.8, the amount of rework to do for each phase has been presented though the vertical axis and horizontal axis represents the time. After discovering the rework, rework to do happens and are added to the total amount of work to do that is needed to be developed to complete the project. Rework to do increase the number of unaccomplished tasks and requires more time than predicted to complete the project. The amount of rework is changeable as it is not possible to measure the exact amount. However, through this analysis, at least an estimation can be obtained since it also affect project schedule. Based on the estimation of rework to do, and the error-free tasks development explanation, it would be easier to design a feasible plan for project development where the deadline is fixed.

Rework cycle is one of the most important features since it generates error-based tasks as rework to do which need to be developed again and increases the total amount of tasks. Hence this impact of the rework cycle directly goes to the project schedule and human resources and increases the uncertainty to the project completion time. So, while making a feasible plan for project development, we should keep in mind the consequences of the rework cycle and its impact on project performance, especially on the project schedule. The rework cycle should be managed thoroughly so that its impact does not adversely affect project performance.

5.3.3.g Tasks Accomplishment for Project Completion

With the error-free tasks development rate, generating rework, and discovering the rework as rework to do when all the tasks are developed of the project, then it can be said that the project is complete. The following figure 5.9 shows the accomplishment of the size of the total tasks of the project. The project size is 1000 tasks and we must complete all 1000 tasks for project completion. The vertical axis shows the total amount of tasks accomplishment and horizontal axis represents the time through which the task accomplishment happens and provides how much time is required to complete all of the tasks.

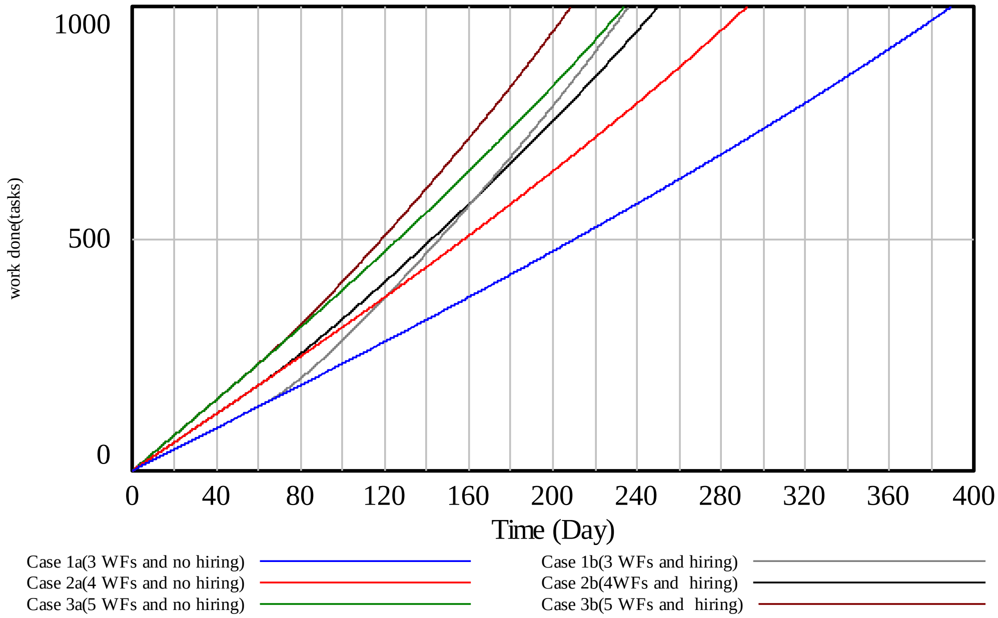


Fig. 5.9 Total Tasks Accomplishment

By analyzing all of the phases, we got the total tasks accomplishment for project completion as shown in figure 5.9. Among these we can see that from *case 3b (with 5 WFs and hiring)*, the project finishes very early from the deadline. Two other phases, *case 1a (3 WFs and no hiring)*, and *case 2a (4 WFs and no hiring)* both exceed the deadline which we have also shown in figure 5.2. The other three phases- *case 1b (3 WFs and hiring), case 2b (4 WFs and hiring), and case 3a* (*5 WFs and no hiring*)– project completion time is near to the deadline.

By this analysis, we can easily understand that, to complete the project on time, which criteria can be chosen based on the given resources by keeping in mind the generation of uncertain rework. This analysis gives a better understanding of hiring when necessary, when we need hiring and how many workforces should need to hire based on the desire.

Figure 5.9_a naturally represents the number of tasks in each time step. The project size is 1000 tasks, so the summation of work to do, undiscovered rework, rework to do, and work done must be 1000 at every time. Figure 5.9_a represents these criteria where Y-axis gives the total amount of tasks and X-axis represents time. Form this figure, it is seen that the amount of rework is not very high. If the amount of rework is very high, then more time will be required for project completion.

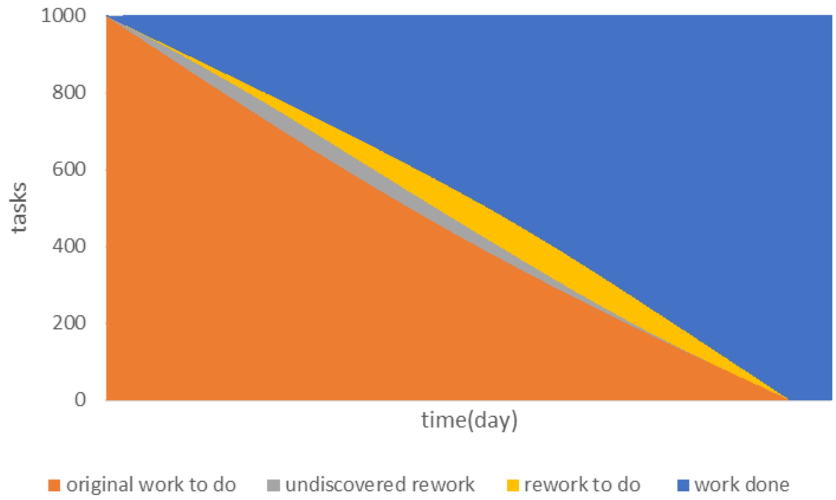


Fig. 5.9_a Total Tasks Accomplishment(case 1)

The summary of the results for each case is given in table 5.4.

Table 5.4 Summary Results of Model Validation with Hiring and without Overtime

	Workforces after hiring (person)	Productivity (task/man-day)	Project completion time (from the simulated result)(days)	Cost (Man-days)
Case 1a (3 WFs and no hiring)	3 (no hiring)	1	389 (exceed deadline)	1167
Case 1b (3 WFs and hiring)	3(initial)+3(hiring)	0.97(impact of hiring)	236	1236
Case 2a (4 WFs and no hiring)	4 (no hiring)	1	292 (exceed deadline)	1168
Case 2b (4 WFs and hiring)	4(initial)+1(hiring)	0.98(impact of hiring)	249	1185
Case 3a (5 WFs and no hiring)	5 (no hiring)	1	234	1170
Case 3b (5 WFs and hiring)	5(initial)+1(hiring)	0.98(impact of hiring)	208	1188

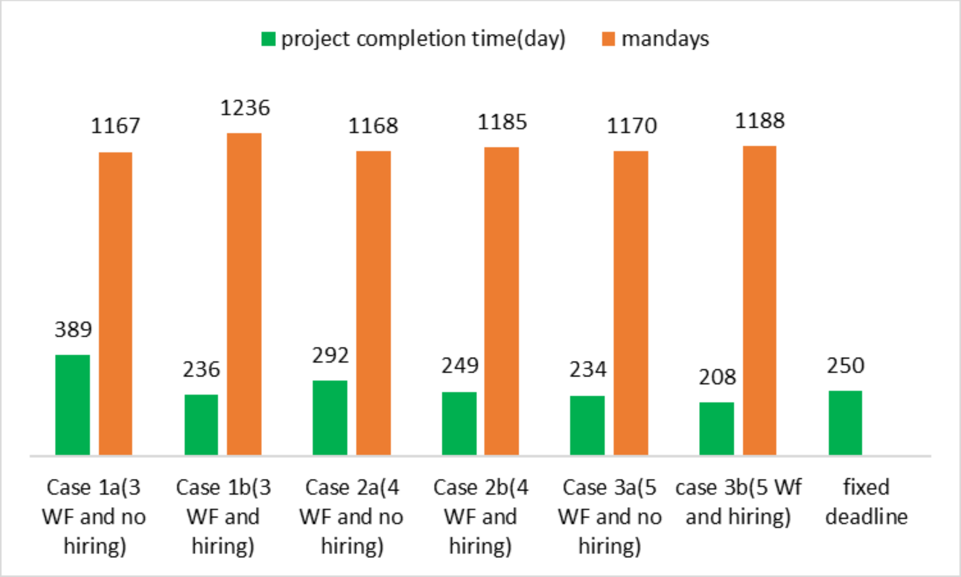


Fig. 5.10 Project Completion Time with Man-Days

5.3.3.h Altering Feasible Options

Based on the result analysis shown in table 5.4 and figure 5.10, it is now easier to understand to obtain a feasible option for the designed project. This circumstance also can easily be understood through the following figure 5.11. This figure represents the required cost for each scenario along with the project completion time.

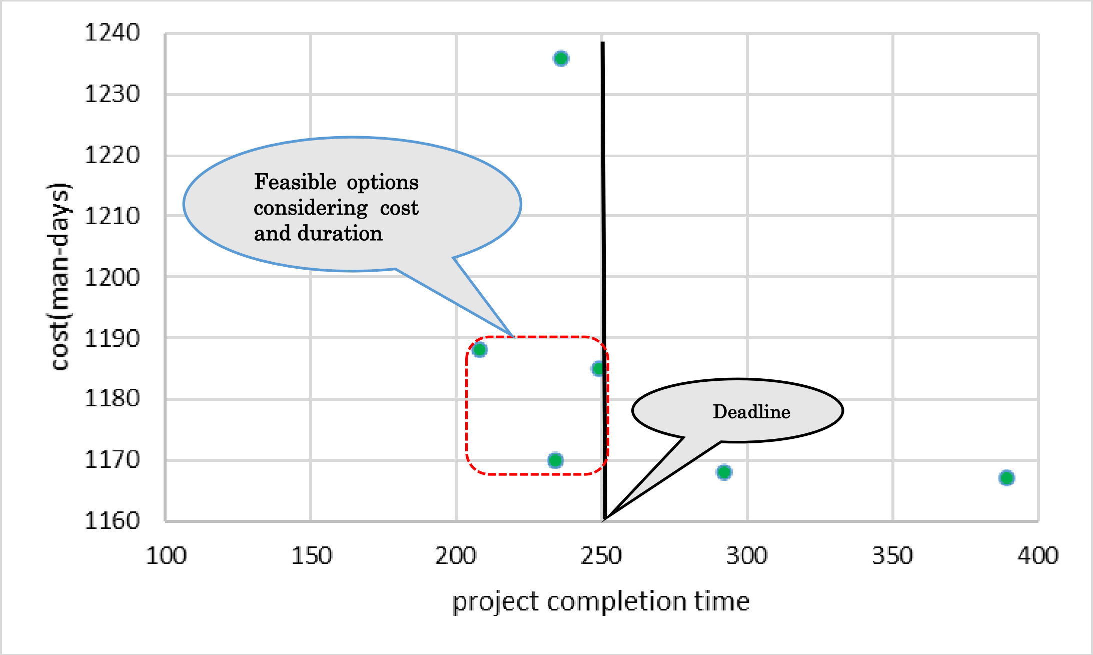


Fig. 5.11 Scenarios with Completion Time and Cost

Keeping in mind the fixed deadline, the above figure gives a visualization of choosing a feasible option for this project development in terms of cost and completion time. And based on these options, planning and decision can be made that what amount of resources is required and how to distribute them for this project development.

The project used for this validation process is an exemplification of actual project, so this method would be applied to actual software projects for designing and planning.

5.4 Model Validation With Overtime and Without Hiring

The second validation has been conducted considering overtime and ignoring hiring. And to perform this validation process, we have considered 3 different cases the same as for validation with hiring, however, in this case, we have considered only applying overtime. In each case, there are three different phases- no overtime, interval-based overtime, and continuous overtime. For this project, the scope is also defined as fixed.

5.4.1 Project Description and Scenario Settings

The input parameters along with the cases for this validation process with overtime and without hiring has given in table 5.5.

Table 5.5_a Input Parameters for *Case 1* for Validation with Overtime and without Hiring

	Input Values		
	Case 1x	**Case 1y**	**Case 1z**
Project size(tasks)	750	750	750
Full-time equivalent workforces (initial) (person)	3	3	3
Deadline(fixed) (days)	200	200	200
Nominal productivity (task/man-day)	1	1	1
Nominal error fraction (Dmnl)	.25	.25	.25
Nominal time for rework discovery (days)	20	20	20
Hiring	No	No	No
Overtime	No	Interval-based	Continuous

Table 5.5_b Input Parameters for ***Case 2*** for Validation with Overtime and without Hiring

	Input Values		
	Case 2x	**Case 2y**	**Case 2z**
Project size(tasks)	750	750	750
Full-time equivalent workforces (initial) (person)	4	4	4
Deadline(fixed) (days)	200	200	200
Nominal productivity (task/man-day)	1	1	1
Nominal error fraction (Dmnl)	.25	.25	.25
Nominal time for rework discovery (days)	20	20	20
Hiring	No	No	No
Overtime	No	Interval-based	Continuous

Table 5.5_c Input Parameters for ***Case 3*** for Validation with Overtime and without Hiring

	Input Values		
	Case 3x	**Case 3y**	**Case 3z**
Project size(tasks)	750	750	750
Full-time equivalent workforces (initial) (person)	5	5	5
Deadline(fixed) (days)	200	200	200
Nominal productivity (task/man-day)	1	1	1
Nominal error fraction (Dmnl)	.25	.25	.25
Nominal time for rework discovery (days)	20	20	20
Hiring	No	No	No
Overtime	No	Interval-based	Continuous

5.4.2 Ideal Calculation with Input Parameters

With 3 Full-Time Equivalent Workforces:

Time required =(750tasks)/(1 task/man-day*3 man)

= 250 days

With 4 Full-Time Equivalent Workforces:

Time required =(750tasks)/(1 task/man-day*4 man)

= 187 days

With 5 Full-Time Equivalent Workforces:

Time required =(750tasks)/(1 task/man-day*5 man)

= 150 days

From this calculation, we can see that with 3 Full-Time Equivalent Workforces, it is not possible to complete the project within the deadline. while considering with 4 full-time workforces and 5 full-time workforces, the project can be finished within deadline, but these calculations are without considering rework. When rework is added to the project, it increase the project completion time. In order to finish a project in time, the usage of overtime has been explored in this validation process. But while considering 5 workforces, the accomplishment time without rework is very lower than the deadline. Hence, maybe it is possible to complete the project without taking any control actions. But still, we have shown the application of overtime and its consequences on project completion time.

5.4.3 Result Analysis of Validation with Overtime and without Hiring

To obtain and state the project performance, we have analyzed the same parameters as we discussed for the validation #1 process with hiring. The parameters include the actual fraction of man-day in a project(AFMDP), productivity, progress rate, rework generation, undiscovered rework, rework to do, and the time required for the accomplishment of total tasks. The result for each of these factors for validation has been explained sequentially.

5.4.3.a Expected Project Completion Time

In section 5.4.2 we showed the ideal Calculation for project completion, and that calculation was without rework. When rework is happening and, it takes more time to compete the project. Based on the rework and with initial workforces, the expected project completion time has obtained for each case as shown in figure 5.12. The vertical axis represents the expected project completion date for each case which changes with time represented in horizontal axis.

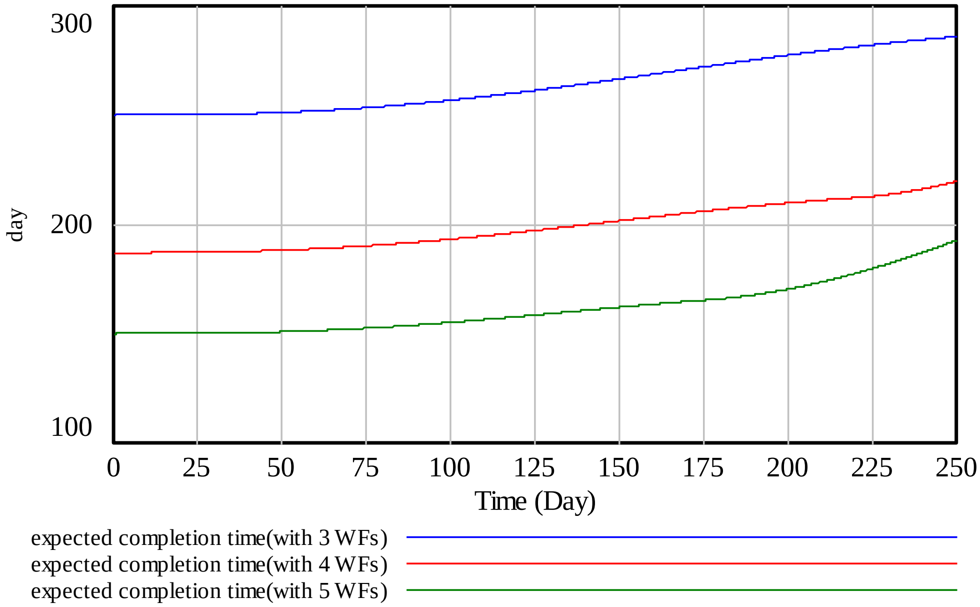


Fig. 5.12 Expected Project Completion time Without Overtime

In here we can see that while working with 5 workforces and considering rework, the project can be finished within the deadline that is 200 days. But for the other cases, with 3 workforces and with 4 workforces, the expected completion time is beyond the deadline. So, to keep the project completion time within deadline, the workforces need to work overtime since hiring option has not considered in this validation process.

5.4.3.b Actual Fraction of Man-day in Project (AFMDP)

When the workforces work overtime, an increase in man-hour per day increases the AFMDP. The value of AFMDP with overtime is shown in figure 5.13. The figure represents the value of AFMDP with overtime through vertical axis along with time series data.

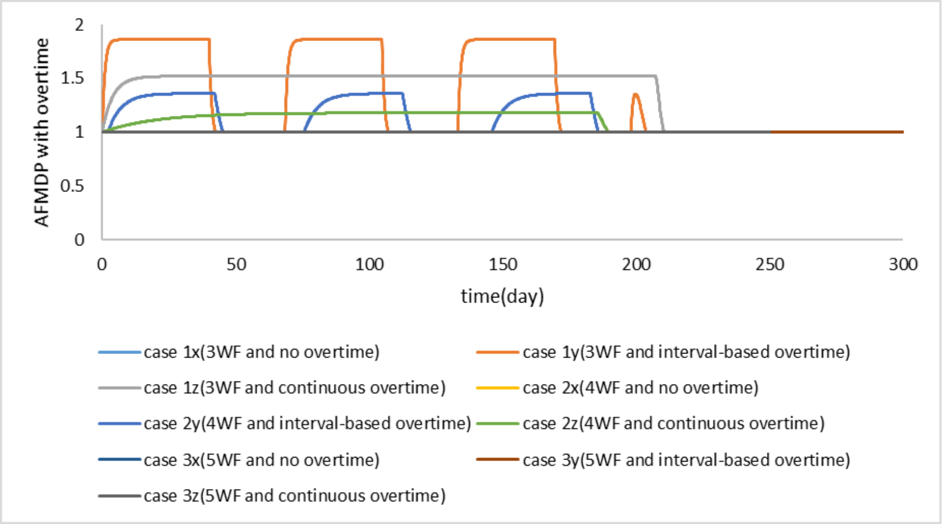


Fig. 5.13 The AFMDP with Overtime

In this figure, we see that when applying interval-based overtime, AFMDP has saturated several times. Working with overtime, AFMDP goes high, otherwise remains in nominal value. The behavior of increasing AFMDP is a representation of the dynamic behavior of System Dynamics.

When applying continuous overtime, the workforces tend to work overtime continuously until the project is finished. Since overtime happens continuously, the range for overtime should be lower than interval-based overtime.

For *case 2 (with 4 WFs),* the usage of either interval-based overtime or continuous overtime, starts gradually after a certain period with a less amount of overtime unlike for *case 1 (with 3 WFs)* where overtime happens at the very beginning of the development period. This happens due to the differences in the number of initial workforces. working with less number of workforces, it requires more man-hour to finish within deadline. Also while working overtime, the average AFMDP value with overtime for *case 2* is lower than *case 1* and the termination of working overtime for *case 2* happens earlier.

When considering for interval-based overtime, the average value for AFMDP is 1.35 and for continuous overtime, it is 1.2. But in case of *Case 1x* (3 WFs and interval-based overtime) and *Case 1z* (3 WFs and continuous overtime), the average value of AFMPD is obtained higher than the designed value. This is because, at first, the number of human resources is lower, hence to complete the project within deadline, the workforces need to work high overtime.

Again in case of *Case 3*, the project can be finished within the deadline without overtime. That's why the value of AFMPD for *case 3y* and *case 3z* are same as *case 3x* (5 WF and no overtime).

5.4.3.c Full-time Equivalent Workforce

Since in this validation process, the hiring option has been ignored, the number of workforces remains constant during development for all scenarios as in the figure 5.14. The number of workforces are shown through vertical axis in a time-base by horizontal axis.

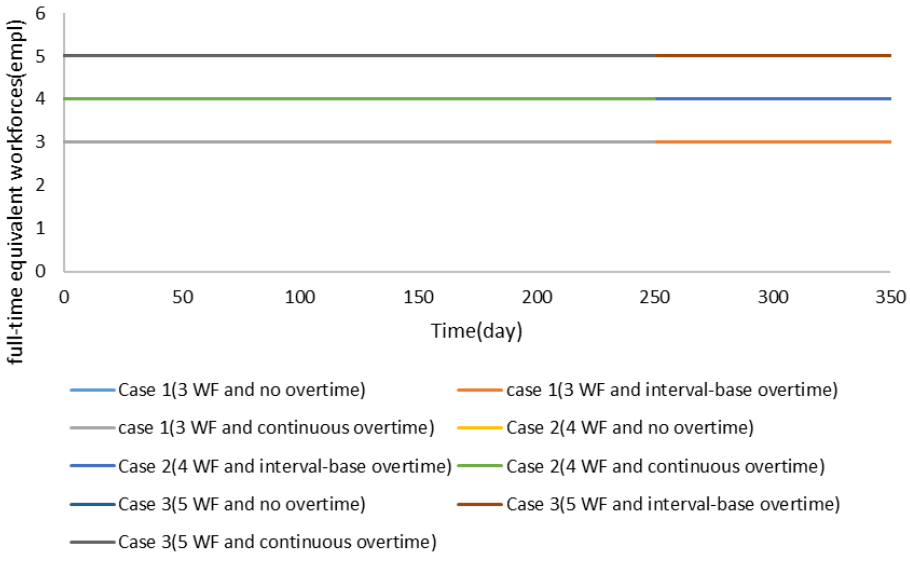


Fig. 5.14 Full-Time Equivalent Workforces

5.4.3.d Productivity and Task Development

When workforces work with nominal AFMDP, the productivity remains steady. When they work with overtime, it increases the man-hour per day, and thus increases the progress rate. But working overtime has an adverse effect on productivity as we have discussed in chapter 4. Initially, when the overtime is low, productivity remains the same for a period, but after that when overtime increases, it tends to decrease the productivity because, working with long overtime, workforces get exhausted and affect productivity. Figure 5.15 represents the impact of overtime on productivity. In the case of interval-based overtime, when workforces work overtime, their productivity decreases but when there is no overtime, productivity remains high. That's why productivity is saturated several times depending on the overtime period. But in case of continuous overtime, as the workforces are continuously working overtime, its impact on productivity is also resulting continuously. In this case, in the beginning, productivity remains the same as the potential productivity, but as time goes, productivity tends to decrease and it decreases continuously till workforces work overtime continuously. The following figure 5.15 represents this behavior.

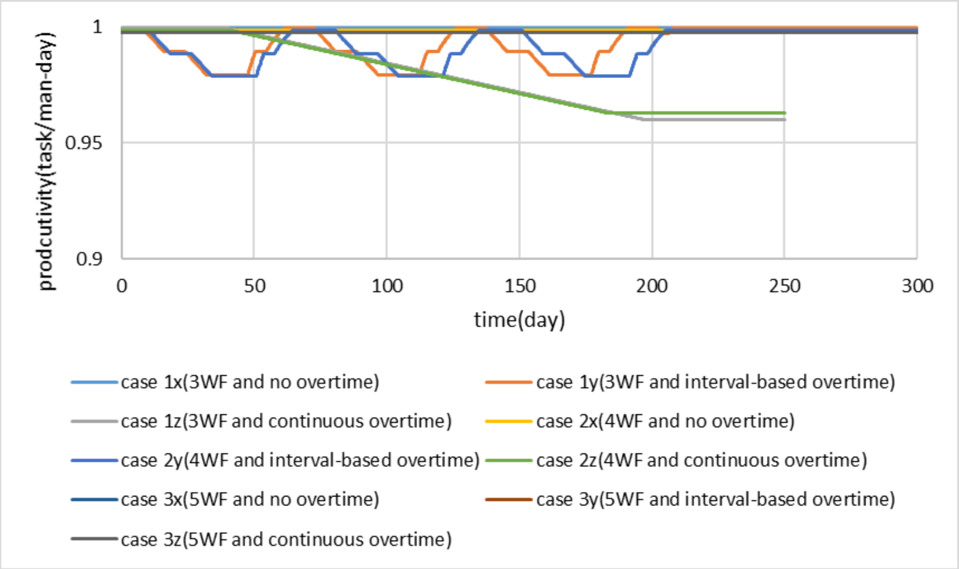


Fig. 5.15 Impact of Overtime on Productivity

In figure 5.15, productivity is represented by Y-axis and X-axis represents the time. Along with this time productivity is changing due to overtime work.

Based on the productivity and effort applied to the project, error-free task development has shown in figure 5.16. As mentioned before working overtime increases the progress rate. Workforces working with overtime, task development rate increases. In all phases, the error-free tasks development rate goes high at the point when all tasks are developed correctly to complete the project. Through the vertical axis, the maximum number of error-free task developed per day has shown and the horizontal axis represents the day.

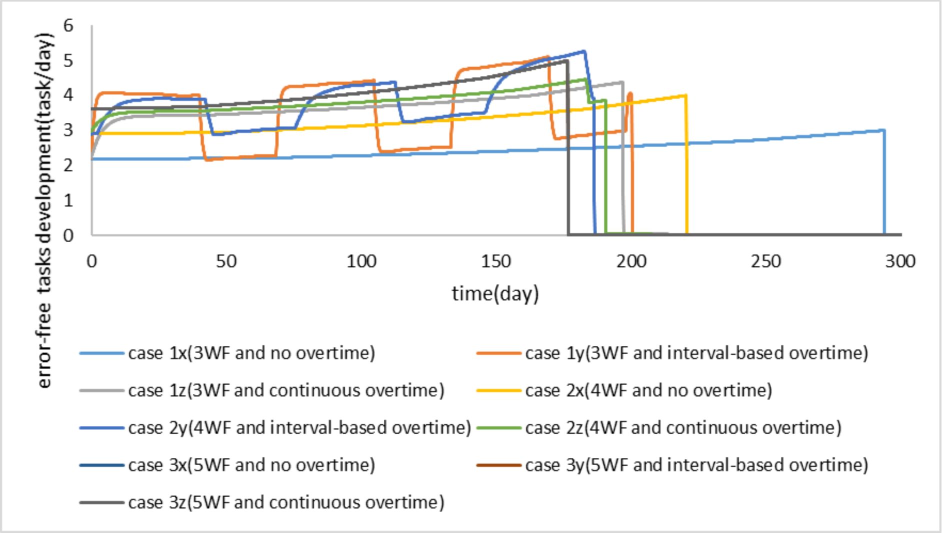


Fig. 5.16 Error-free Tasks Development

From this figure 5.16, the project manager would get an better understanding of applying overtime and the consequences of the overtime on performance. This figure represents how many tasks are developed each day. So the project manager can calculate and observe the behavior of task development with overtime. And based on this observation, they can choose one of the overtime plan depending on resources and necessity.

5.4.3.e Rework Generation

Along with error-free tasks development, rework generation happens based on error fraction. Working overtime also affects rework generation in several ways. Firstly, when workforces are working overtime, more tasks are developed and the proportion of rework generation also increases. Again working overtime, workforces get exhausted and increase error fraction which ultimately increases rework generation. However, as time goes and all tasks are developed correctly, rework generation becomes zero at the end of the project shown in the following figure 5.17. Through the vertical axis, the maximum number error-based task developed per day has shown and the horizontal axis represents the day.

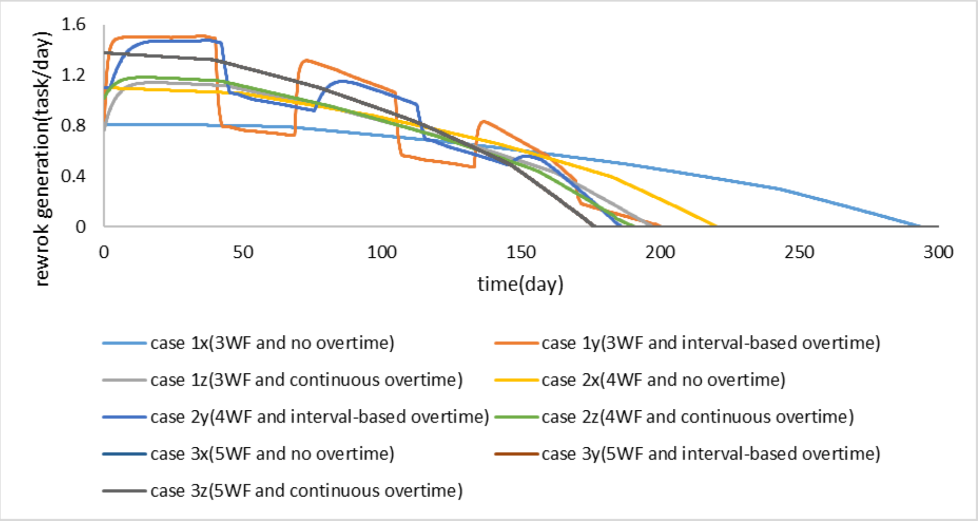


Fig. 5.17 Rework Generation

5.4.3.f Undiscovered Rework and Rework to Do

Based on the rework generation, error based tasks are generated and stored as undiscovered rework. As mentioned before, the rework generation starts from the beginning of the project, and the number of undiscovered rework increases gradually as represented in figure 5.18. The amount of undiscovered rework are represented by Y-axis and X-axis represents the time by which value of undiscovered rework changes. After reaching its peak value, undiscovered rework begins decreasing due to rework discovery, and through this discovery, the task is represented as rework to do shown in figure 5.19.

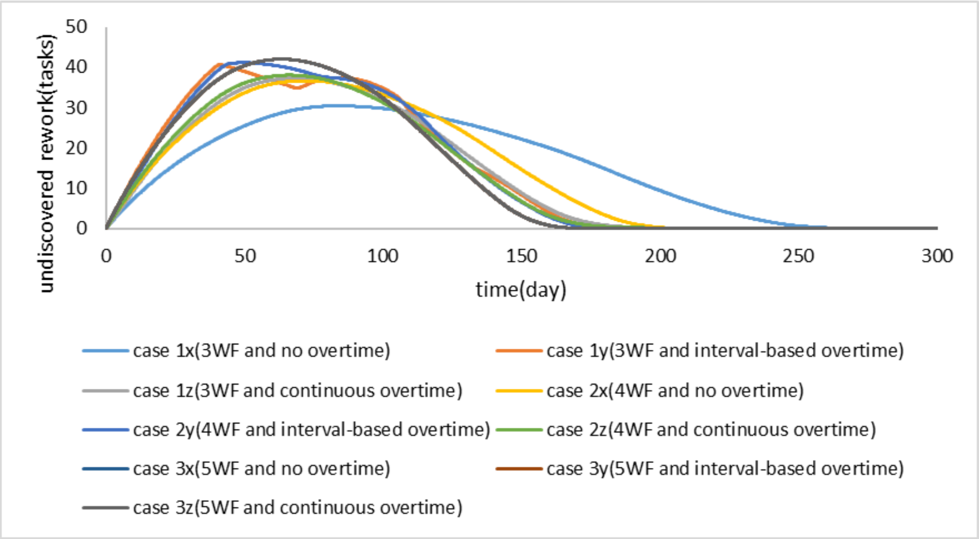


Fig. 5.18 Undiscovered Rework

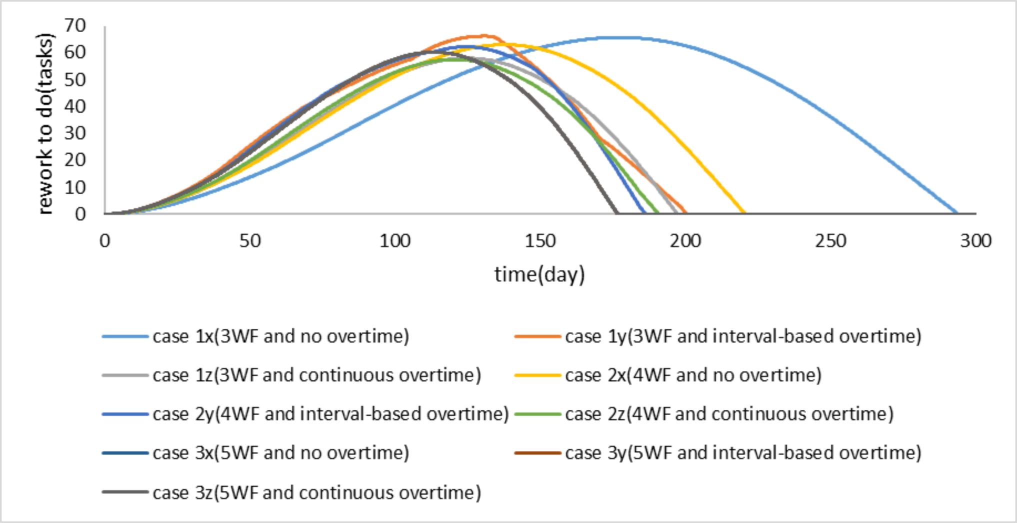


Fig. 5.19 Rework to Do

In figure 5.19, the amount of rework to do is represented for each phase of each case through the vertical axis generated by each time. After discovering the rework, rework to do are added to the total amount of work to do that is needed to be developed to complete the project. The amount of rework cannot me measured in an exact way as it depends on the experience of the workforces and way of task development, but according to the error fraction, we can get an assumption and this would help to understand that how it affects project schedule.

5.4.3.g Tasks Accomplishment

With tasks development rate, rework generation, and discovering the rework as rework to do, when all the tasks are developed of the project, it asserts project completion. The following figure 5.20 shows the accomplishment of the total tasks of the project. The project size is 750 tasks and we must complete all 750 tasks for making project deliverable. The vertical axis shows the total amount of tasks accomplishment and horizontal axis represents the time through which the task accomplishment happens and provides how much time is required to complete all of the tasks.

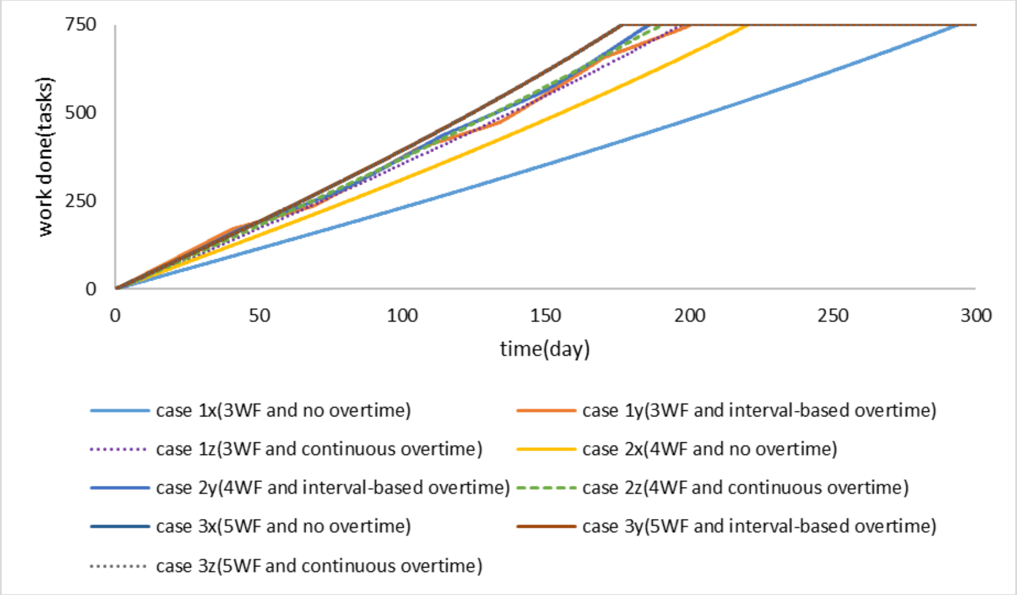


Fig. 5.20 Total Tasks Accomplishment

Form this graph, we can see that for *case 3 with 5 workforces*, since it is possible to complete the project without working overtime, so project completion time for all *case 3x, case 3y* and *case 3z* are same. As for *case 1x (3 WFs and no overtime)* and *case 2x (4 WFs and no overtime)*, project is completed beyond the deadline.

The following table 5.6 includes the summary result of the validation with overtime.

Table 5.6 Result Summary of the Validation Process with Overtime

	AFMDP (average overtime impact)	**Productivity (task/man-day)**	**Project completion time (from the simulated result)**	**Cost (Man-days)**
Case 1x (3 WFs and no overtime)	1	1	294 (exceed deadline)	882
Case 1y (3 WFs and interval-based overtime)	1.4	.97	200	851
Case 1z (3 WFs and continuous overtime)	1.29	.98	198	826
Case 2x (4 WFs and no overtime)	1	1	220 (exceed deadline)	880
Case 2y (4 WFs and interval-based overtime)	1.25	0.98	187	874
Case 2z (4 WFs and continuous overtime)	1.1	0.98	195	938
Case 3x (5 WFs and no overtime)	1	1	176	880
Case 3y (5 WFs and interval-based overtime)	1	1	176	880
Case 3z (5 WFs and continuous overtime)	1	1	176	880

As mentioned before, from the analysis, we can see that for *case 3 with 5 workforces*, project can be finished within the deadline without applying overtime since the number of workforces are high. Since in this case no overtime is required, so, for *case 3y (5 WFs and interval-based overtime)* and *case 3z(5 WFs and continuous overtime)*, no overtime is applied. For all these phases, the value of AFMDP remains 1.

For *case 2x(*4 WFs and no overtime), project completion time exceeds the deadline. Hence, in this scenario, both interval-based and continuous overtime have been applied, after applying the overtime, it is possible to complete the project within deadline as shown in table 5.6. From this scenario, with working interval-based overtime, the project requires less completion time and less man-days comparing to applying interval-based overtime.

For *case 1x (3 WFs and no* overtime), the completion time is very high comparing to the deadline. As a result, overtime is required to complete the project and is applied through *case 1y* and *case 1z*. However, the average value of AFMDP for this case is very high because less number of workforces are working in the project. With this high overtime, project can be finished within time, but it may adversely affects the mental condition of the workforces.

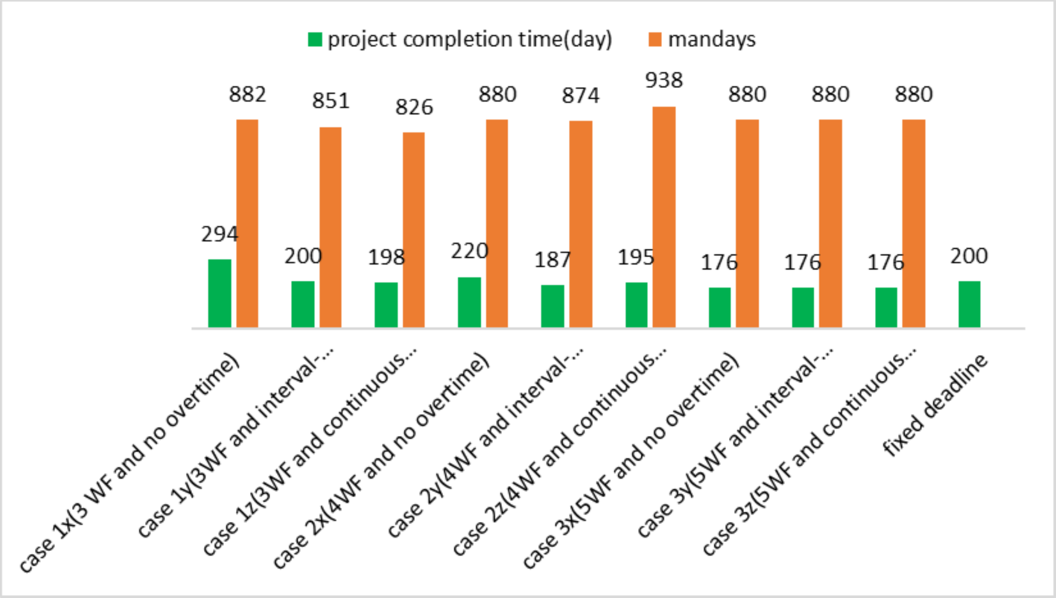


Fig. 5.21 Project Completion Time with Man-Days

Figure 5.21 and 5.22 both represent the project completion time and cost for each scenario through which a very clear idea could be obtained to choose and design a feasible plan for this project.

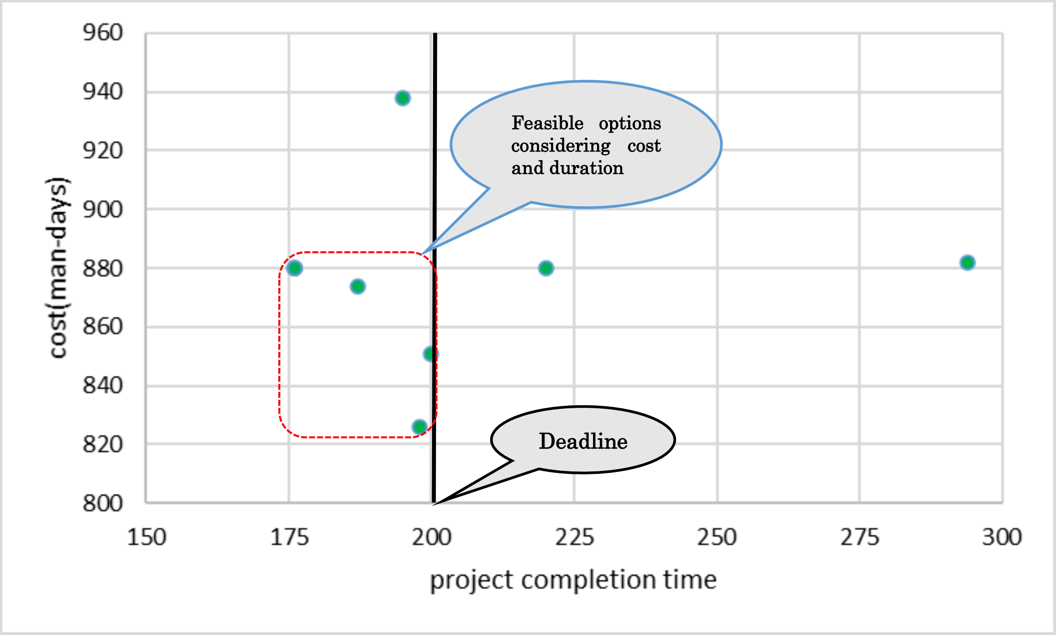


Fig. 5.22 Scenarios with Completion Time and Cost

Form this analysis, we can see that when considering to work overtime, the following scenarios are feasible to make a plan - *Case 2y*(4 WF and interval-based overtime), *Case 2z*(4 WF and continuous overtime) and *Case 3x*(5 WF and no overtime). The other two scenarios - *Case 1y*(3 WF and interval-based overtime), *Case 1z*(3 WF and continuous overtime), the project can be finished within deadline but the amount of overtime is very high. Hence, if the project manager intends to work with average amount of overtime, then the first 3 scenarios are more feasible.

This analysis gives a better understanding of using different types of overtime and their impact on project performance. When the given resources of a project are not enough and the project manager does not have the intention to hire new people, then the principle option is to work overtime. And the amount of overtime depends on in which way they are interested to work overtime. If they do overtime continuously until the project is finished, then the range should be smaller otherwise they will get exhausted very fast and may provide a very adverse effect project development. On the other hand, if the workforces work overtime with an interval, then the range could be high to finish the project on time.

Through the study, we have explained the usage and impact of overtime on project performance to get an idea about model based decision-making for a better project planning. Although in this project to meet the deadline, overtime is not necessary if we have more human resources, but in case of a large project and with less human resources, this explanation will give a better idea of applying overtime for software development projects when the intent is to keep the resources fixed and without hiring new workforces.

5.5 Summary of Model Validation

The purpose of the validation process is to obtain the usefulness of the model and to observe and understand the dynamic behavior of the parameters that are involved in project performance. From the result analysis for validation, the following criteria can be attained.

i. **Pragmatic:** This criterion represents that the conceptual model should have some degree of logical consistency in the system. From the behavior of the result obtained, it can be stated that both hiring and overtime plans can be applicable to some extent.

ii. **Predictive:** A predictive conceptual model would provide certain conditions, the corresponding phenomenon that provides the validity of the model. The plan and options used for both cases in our analysis, all are logical and considered in many software industries to meet the schedule. Based on the analysis, it can be said that the options for both hiring and overtime management are valid in various dimensions.

iii. **Feasibility:** The explanation of the simulation results through the graphical analysis gives an understanding of the feasible concept through which the right decisions can be made when necessary.

Chapter 6 Case Study

In this chapter, we have represented a case study. After the validation process, we have conducted a case to examine the model's behavior in detail considering the large project.

To design a feasible plan, the implementation of hiring and overtime have been analyzed separately in the validation process. So, during this study, both hiring and overtime have been considered simultaneously to analyze the performance behavior.

To gain an understanding of the managerial policies and their implications, we need to analyze the detailed performance behavior of the variables. And to perform this, the case study has been applied.

6.1 Project Description

To achieve the objective of this study, the input parameters have been used from an Example project (Suinan Li, 2008) which was run and analyzed to obtain the dynamic behavior based on actual project. The initial values of the input parameters have been given below.

Table 6.1 Input Parameters for Case Study

Parameters	**Input values**
Project Size(fixed)	1200 tasks
Full-Time Equivalent Workforce	15 person
Deadline (fixed)	1800 days
Potential Productivity	0.048 task/man-day

The simple mental calculation for hiring with the input data given in table 6.1 is as follows.

With 15 Full-Time Equivalent Workforces:

Time required =(1200tasks)/(.048 task/man-day*15 person)

= 1666.67 days **(without rework)**

This calculation is simple and done without considering rework. However, it involves rework and would require more time to complete the project.

6.2 Scenario List for Case Study

For this case study, we have considered six different scenarios as a combination of hiring and overtime management plan. The following table 6.2 represents the scenario list.

Table 6.2 A List of Scenarios for the Case Study.

	Hiring	**Overtime**
Scenario #1	Yes	Interval
Scenario #2	No	Continuous
Scenario #3	Yes	No
Scenario #4	No	Interval
Scenario #5	Yes	Continuous
Scenario #6	No	No

6.3 Result Analysis of Case Study

To examine the performance behavior through the case study, we have analyzed the same parameters as the validation process since those parameters are considered as most important and influenced parameters for project performance. However, in this case, we have analyzed the consequences of hiring and applying overtime together.

6.3.1 Expected Project Completion time

In section 6.1, we showed the ideal calculation for project completion time without rework. However, when rework is happening , it takes more time to compete the project. Based on the rework generation and with initial workforces, the expected project completion time has obtained in figure 6.1 where vertical axis gives the expected completion date with respect to time in horizontal axis.

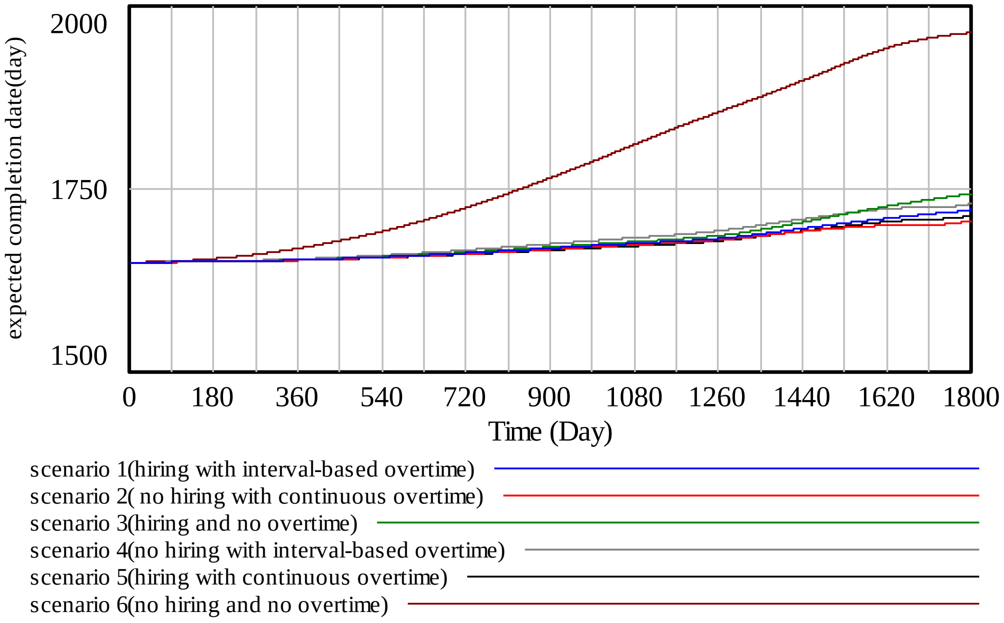


Fig. 6.1 Expected Project Completion Time for All Scenarios

The vertical axis represents the expected completion time with respect to time. The given deadline is 1800 days. In this figure, we can see that working with given resources and without hiring and without working overtime, the project completion time exceeds the deadline which is represented through scenario 6.

At the beginning the expected completion date is very same for all scenarios, because the initial number of workforces are same as shown in figure 6.1_a.

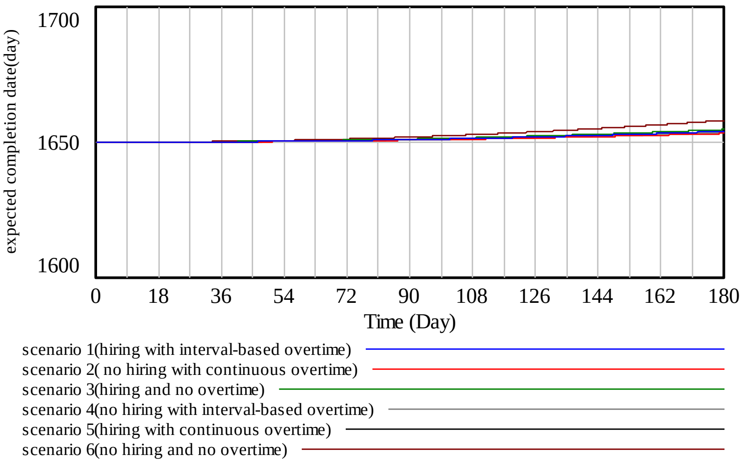


Fig. 6.1_ a Expected Completion Date Behavior Between Time 0-180 days

After a while when the control actions that is overtime and hiring happens, the expected completion date changes and are different from each other based on the control actions are taken. Again, when hiring happens, it affects the expected completion date that the expected completion date will be reduced but since at the same time rework generation also happens, which ultimately increases the completion time. Therefore, the expected completion time remains continues to increase naturally.

6.3.2 Actual Fraction of Man-Day in Project (AFMDP)

Figure 6.2 represents the AFMDP for all scenario. In this figure, the Y-axis represents the AFMDP with overtime for each scenario corresponding to time in X-axis. When overtime is not applied, AFMDP remains at level 1. And applying overtime either interval-based or continuous increases AFMDP. For scenarios 3 and 6, no overtime has been considered, so the value of AFMDP is 1. In the case of scenarios 1 and 4, interval-based overtime is considered and for scenarios 2 and 4, continuous overtime has been practiced.

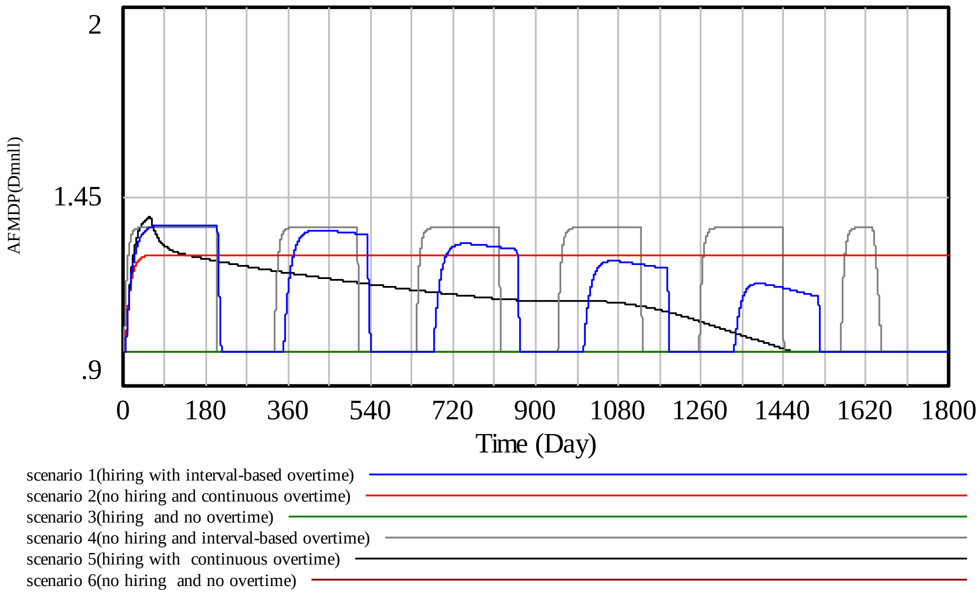


Fig. 6.2 The AFMDP with Overtime

6.3.3 Full-Time Equivalent Workforce

The next important parameter for project development is the human resource that is the workforce. The scenarios that have been considered for the case study, among them for scenarios 1, 3, and 5, we have considered hiring additional workforces along with the existing workforces. Since hiring and overtime are applying simultaneously, a few numbers of workforces are hired. In the following figure 6.3, the number of total workforces for all scenarios has been represents through the vertical axis and the horizontal axis gives the time representation.

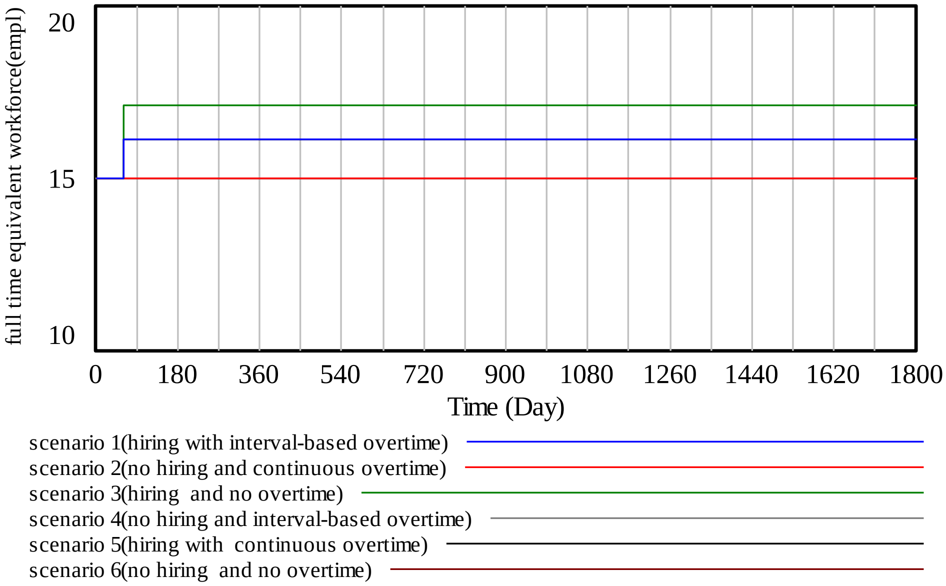


Fig. 6.3 Hiring New Workforces

For scenarios 2, 4, and 6 – no hiring happens, so the workforces are constant that is 15 persons. In the case of scenario 1 and 5, hiring happens along with overtime, so in both cases, only 1 additional workforce has been hired based on the desired requirement. For scenario 3, since no overtime is applying, so to complete the project within time more workforces are needed to be hired. That's why 2 additional workforces have been hired while analyzing this scenario. The total number of full-time equivalent workforces are 17 persons in this scenario.

Table 6.3 Total Full-Time-Equivalent Workforces After Hiring

	Initial workforces(persons)	Hired workforces(persons)	Total workforces(persons)
Scenario #1	15	1	16
Scenario #2	15	No hiring	15
Scenario #3	15	2	17
Scenario #4	15	No hiring	15
Scenario #5	15	1	16
Scenario #6	15	No hiring	15

6.3.4 Productivity and Task Development

The following figure 6.4 shows the productivity which is affected by both hiring and overtime. With respect to the time in horizontal axis, the vertical axis gives the overall productivity. In this figure, we can see that, since both hiring and overtime has been considered, productivity is affected by both of these actions.

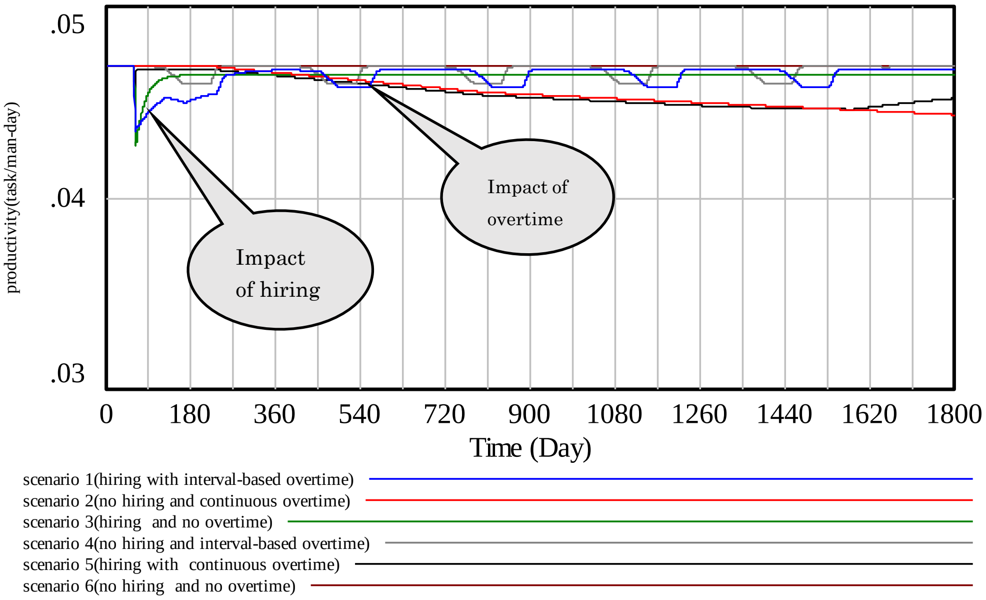


Fig. 6.4 Impact of Hiring and Overtime on Productivity

Task development rate is influenced by the size of the team , productivity and working hours. A team with a large number of workforces developed more tasks per day comparing to the team with fewer workforces with the same productivity. Additionally, when they work overtime, the task development rate also increases.

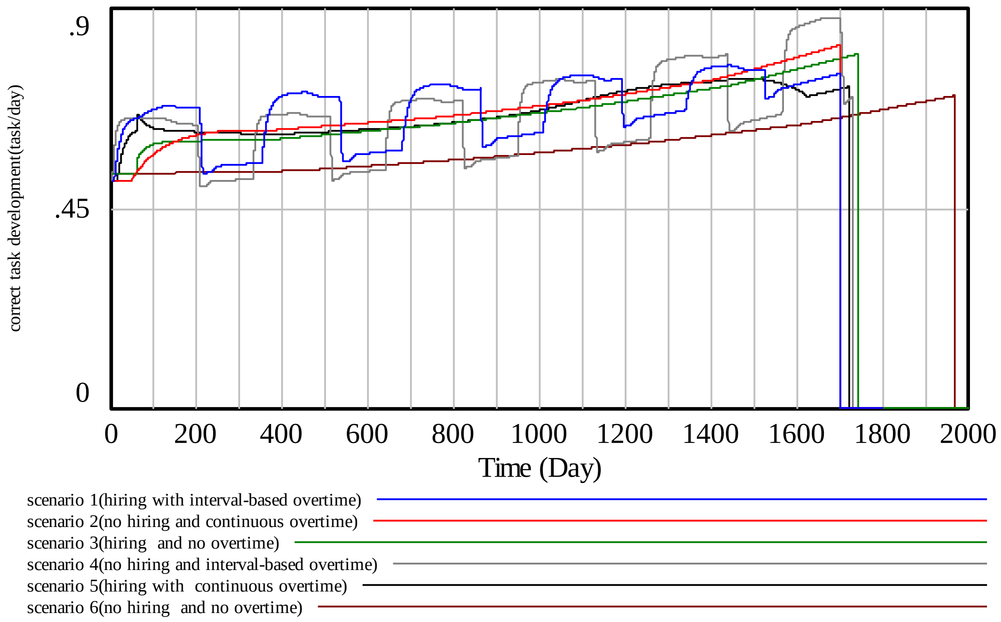


Fig. 6.5 Error-free Tasks Development

Figure 6.5 shows the error-free task development rate for each scenario. The vertical axis represents how many tasks can be developed in each day. From this figure, it is understandable that working overtime either interval-based or continuous and adding more workforces to the development team increase the task development rate and it goes high when all the tasks are correctly developed. For scenario 6, since no hiring and overtime are applied, tasks development rate increases slowly and takes a longer time to complete all the tasks.

6.3.5 Rework Generation

Along with developing error-free tasks, rework generation happens due to error fraction, and the rate increases when people get exhausted due to overtime and the new workforces work to the project. At the beginning of a project, the rate of rework generation is high and decreases slowly when the rate of error-free tasks development increases and gradually tends to zero when all the tasks are correctly developed to complete the project. Figure 6.6 ascertains this criterion. In this figure, the amount of rework generation has been shown through vertical axis which is obtained with respect to time at horizontal axis.

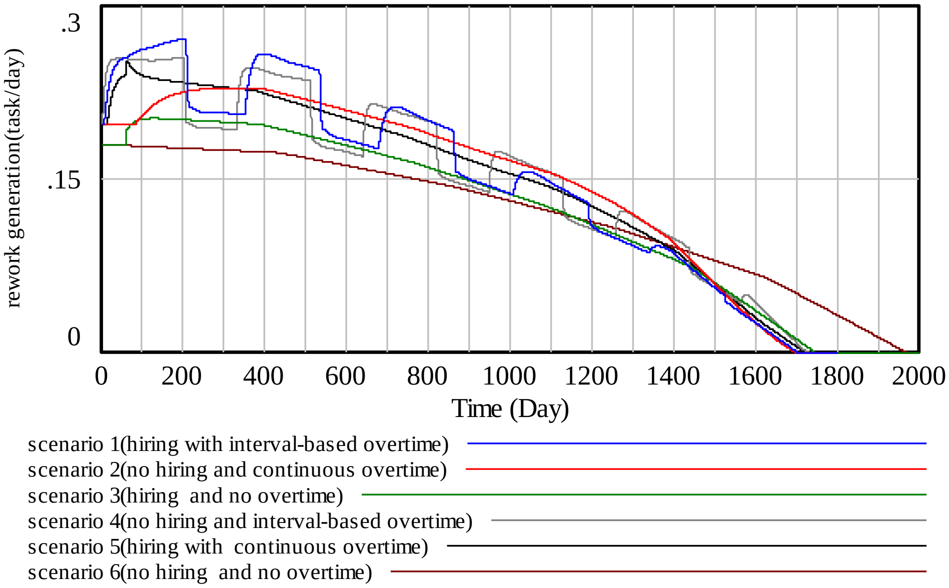


Fig. 6.6 Rework Generation

6.3.6 Undiscovered Rework and Rework to Do

Depending on the rework generation, error-based tasks are created and stored as undiscovered rework shown in figure 6.7. The amount of undiscovered rework is obtained in accordance to time and this amount is shown through the vertical axis in figure 6.7.

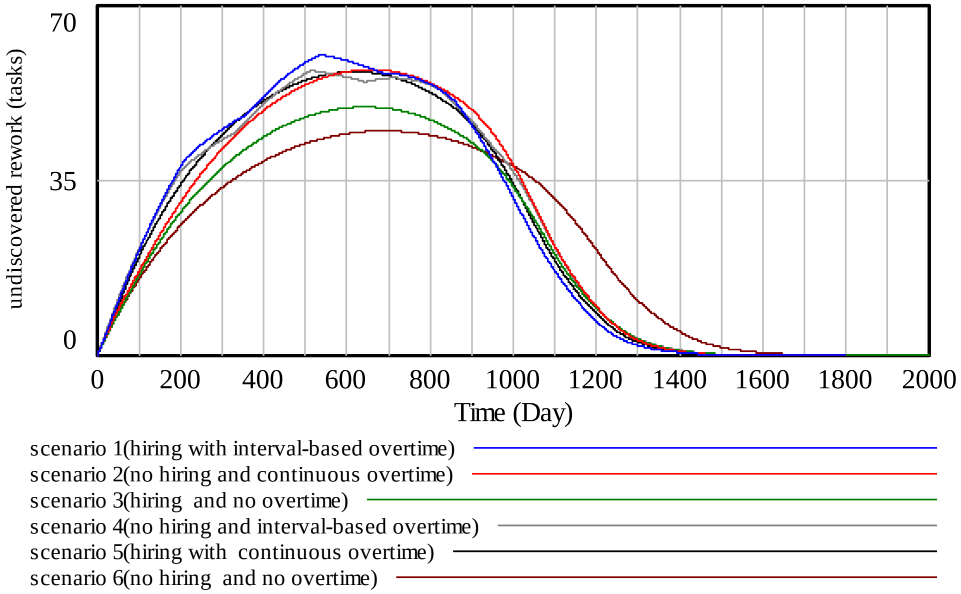


Fig. 6.7 Undiscovered Rework

After generating the undiscovered rework, workforces have to spend several times to discover the error of the task, and after discovering, those tasks are stored as rework to do. These rework to do are added to the total amount of the tasks of the project and have do develop again. Because of these undiscovered rework and rework to do, project completion time increases. In figure 6.8, rework to do is obtained by the Y-axis and this amount is generated with respect to time.

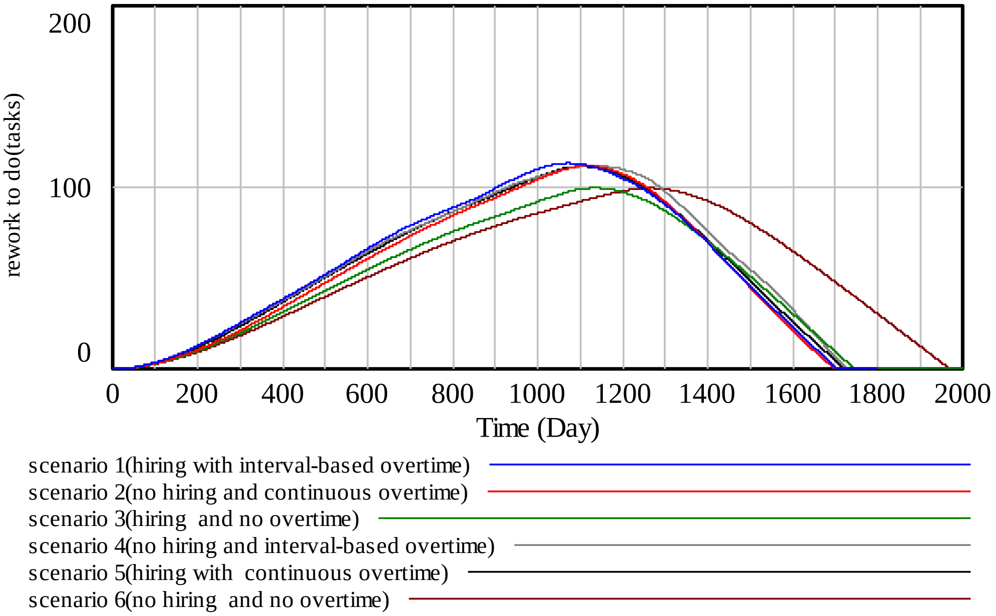


Fig. 6.8 Rework to Do

6.3.7 Tasks Accomplishment

Following the error-free task development rate, generating and discovering rework, and then developing those rework again, when all the tasks are developed correctly, we can assert project completion. And the performance of project completion depends on how much time is required to complete the project. The following figure 6.8 shows the total amount of tasks accomplishment for each scenario. The vertical axis shows the total amount of tasks accomplishment and horizontal axis represents the time through which the task accomplishment happens and provides how much time is required to complete all of the tasks.

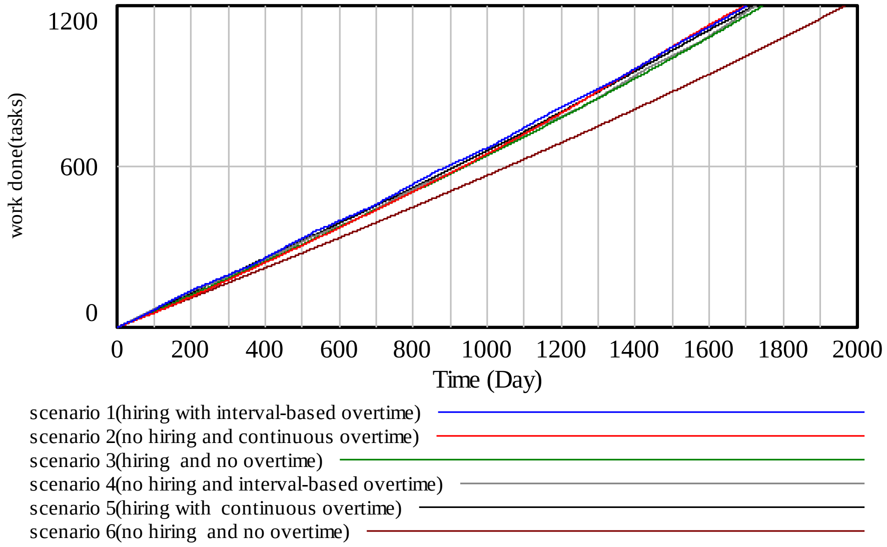


Fig. 6.9 Total Tasks Accomplishment

This figure is the representation of project completion for each scenario. From scenario 1 to scenario 5, there is no big difference among project completion time. But in case scenario 6, since no hiring and no overtime is applied, it takes a longer time compared to other scenarios to complete the project which exceeds the deadline.

6.4 Summary of Case Study

The following table 6.4 gives the summary results of the case study.

Table 6.4 Summary Results of the Case Study

	AFMDP (average overtime impact)	Workforce (persons)	Overall productivity(hiring and overtime impact) (task/man-day)	Project Completion time [days] (from the simulated results)	Cost (Man-days)
Scenario #1	1.28	15(initial)+ 1(hiring)	.046	1703	31146
Scenario #2	1.2	15	.046	1697	33919
Scenario #3	1	15(initial)+ 2(hiring)	.047	1745	31715
Scenario #4	1.35	15	.046	1731	32864
Scenario #5	1.12	15(initial)+ 1(hiring)	.046	1721	33393
Scenario #6	1	15	.047	1969(exceed deadline)	29535

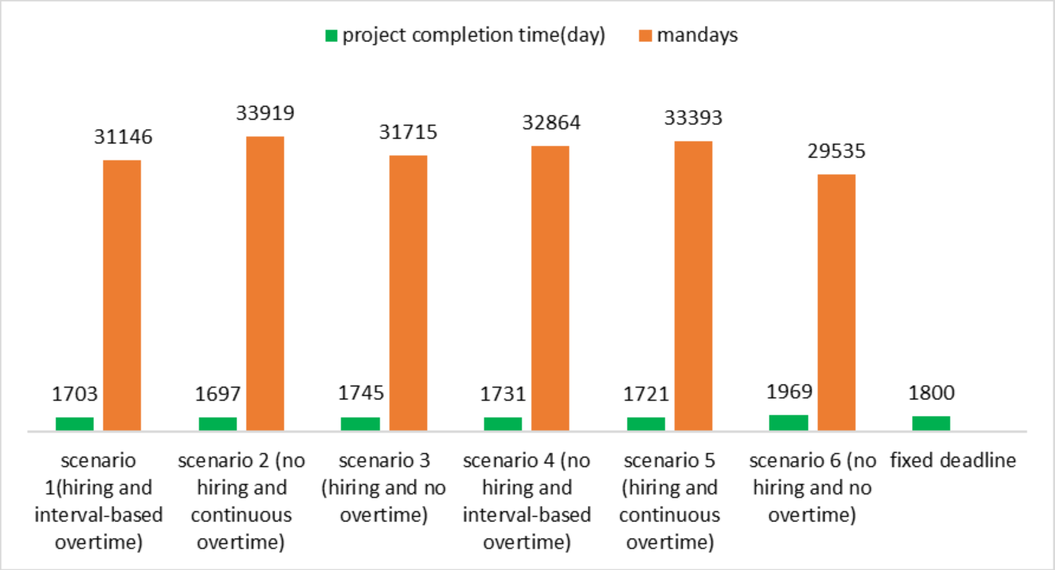


Fig. 6.10 Project Completion Time with Man-Days

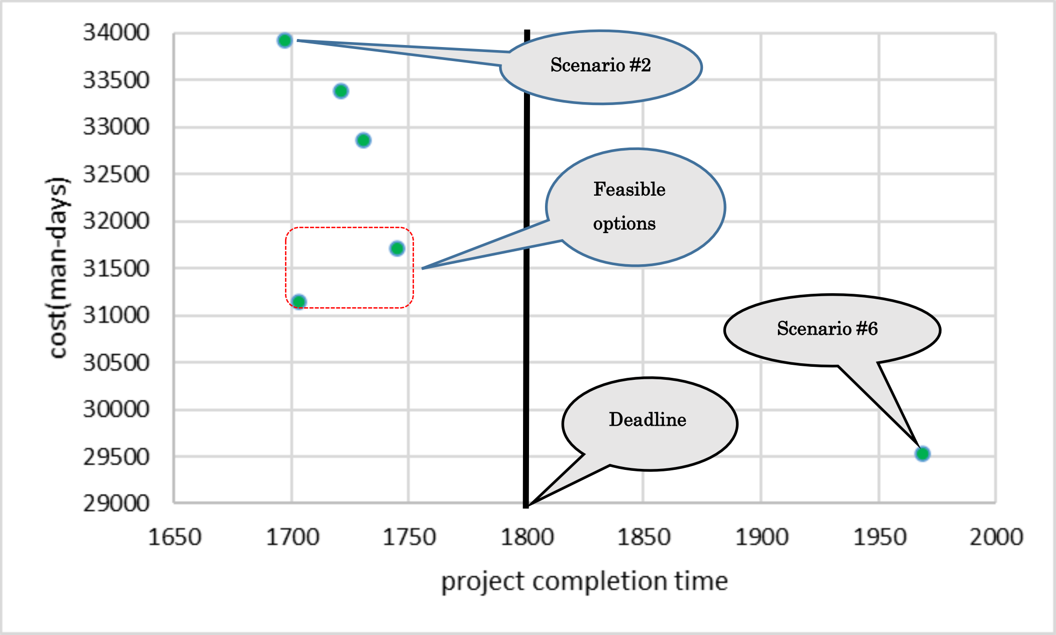


Fig. 6.11 Scenarios with Project Completion Time and Cost

In the analysis of man-days, it is seen that *scenario 6* which exceeds deadline but requires less man-days. As for *scenario 2*, with no hiring and continuous overtime, project completion time is less but costs very high. While comparing between interval-based and continuous overtime, in case of interval-based overtime, sometimes it costs less comparing to continuous overtime. This is because, during the interval-based overtime, workforces do overtime in a interval way unlike in a continuous way. That's why the man-days obtained with interval-based overtime is less than with continuous overtime.

Developing a project with given resources and without taking any control actions, in reality, often very difficult to complete on time. *Scenario 6* reflects this aspect. In this scenario, the project exceeds the deadline but costs less. However, it is common for software industry to hire new workforces and working overtime based on the necessity. So, how many workforces are needed to hire or how much overtime can be worked to obtain project features, is very important to know for designing a proper plan before starting the project or at the beginning of the project.

Both hiring and overtime increase man-days since hiring includes additional workforces and overtime increases man-hour to the nominal working hour. Analyzed scenarios from 1 to 5 in the case study provides a way of understanding the usage of overtime along with hiring and also associate cost which would help us to understand and make prediction about model-based decision making through which a feasible plan can be made to manage project performance.

Chapter 7 Sensitivity Analysis

In this chapter, we have discussed the sensitivity test. The sensitivity test is the process of alternating the assumptions of constant's value in the model and examining the resulting output.

7.1 Sensitivity Analysis Process

The value of the constant parameters of System Dynamics models are subject to change, so sensitivity analysis is an important task for assuring the reliability of simulation results. Since System Dynamics is a behavior-oriented simulation method, the sensitivity of behavior pattern measures should be evaluated to explore the effects of parameter uncertainty on the behavior patterns (Mustafa, 2010). The result of sensitivity analysis may allow determining that which of the model parameters are more important for the simulation output. The parameters to which model is more sensitive, it requires more data analysis to reduce the uncertainty in the parameter value.

Manual sensitivity testing includes changing the value of a constant or several constant and simulating, and repeating this action several times to get an extension of output values (User's Guide, Vensim, 2007).

But for System Dynamics modeling, one of the benefits is the capability of being able to perform a sensitivity analysis of the various variables within the model, and, Vensim, the software package used has the capability to do repeated simulation through Monte-Carlo simulation (User Guide's, Vensim, 2007). Sensitivity test would be very helpful to understanding the behavioral boundaries of a model and examining the robustness of model-based policies.

7.2 Project Description for Sensitivity Analysis

For the sensitivity test, we have considered the simulation result analysis of model validation with hiring and without overtime as described in section 5.3. The initial input parameters that we have taken into consideration are given in the following table 7.1.

Table 7.1 Input Parameter List

Parameters	Input values
Project Size	1000 tasks
Full-Time Equivalent Workforce (initial)	5 person
Deadline(fixed)	250 days
Nominal Productivity	1 task/man-day

Using these inputs, the project has been developed and we have shown the different scenarios of the result in chapter 5. In this chapter, we will perform sensitivity analysis considering these inputs and with initial workforces 5 persons.

The sensitivity analysis was conducted for our simulation each of which was examined with respect to their impact on total tasks accomplishment, undiscovered, and task development rate. The variables varied were nominal productivity, hiring delay. After that, we have also done sensitivity for productivity with respect to overtime usage. And, as for the number of workforces, during the model validation, we have analyzed with a different number of workforces which reflects the sensitivity test, so in this chapter, we did not include this factor.

Table 7.2 Variable List Used for Sensitivity Test

	Variables	Model value	Ranges for sensitivity test	Distribution
Sensitivity run #1	Productivity	1 task/ man-day	.8 – 1.2 task/man-day	Uniform
Sensitivity run #2	Hiring delay	60 days	30-90 days	Uniform

7.3 Result Analysis for Sensitivity Test

7.3.1 Impact on tasks accomplishment

The first analysis of the sensitivity test we have done for total task accomplishment. In order to compare the sensitivity test, at first, we have shown the initial result analysis. Figure 7.1 represents task accomplishment based on the input parameters from table 7.1 where he Y-axis shows the amount of tasks accomplishment with respect to time in X-axis.

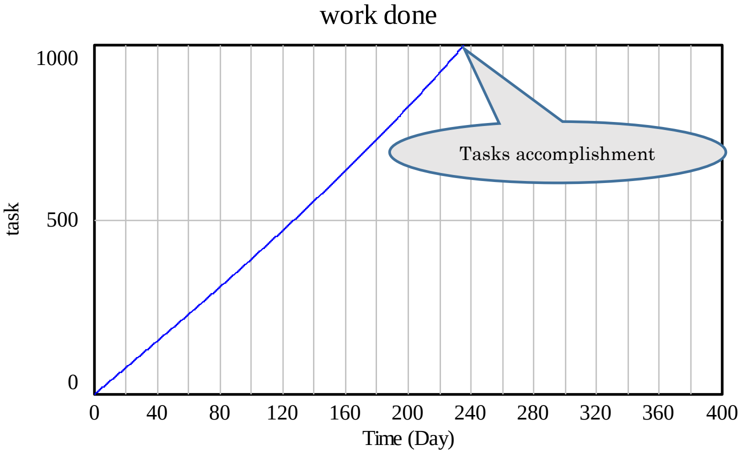


Fig. 7.1 Total Tasks Accomplishment

After that, we did the sensitivity test by varying the value of nominal productivity and hiring delay time. The results for the sensitivity test on task accomplishment has shown through figure 7.2 and 7.3.

While comparing with figure 7.1 and 7.2, we can see that the variation of productivity has a significant impact on task accomplishment. As we know that productivity defines the task development rate which is directly related to total tasks accomplishment, so if the productivity is low, it requires more time to accomplish the task and with high productivity requires less time for accomplishment. During our sensitivity test, we have varied the value of 20%. So within this variation, we can observe the differences in task accomplishment in figure 7.2 where number of accomplished tasks are shown by Y-axis.

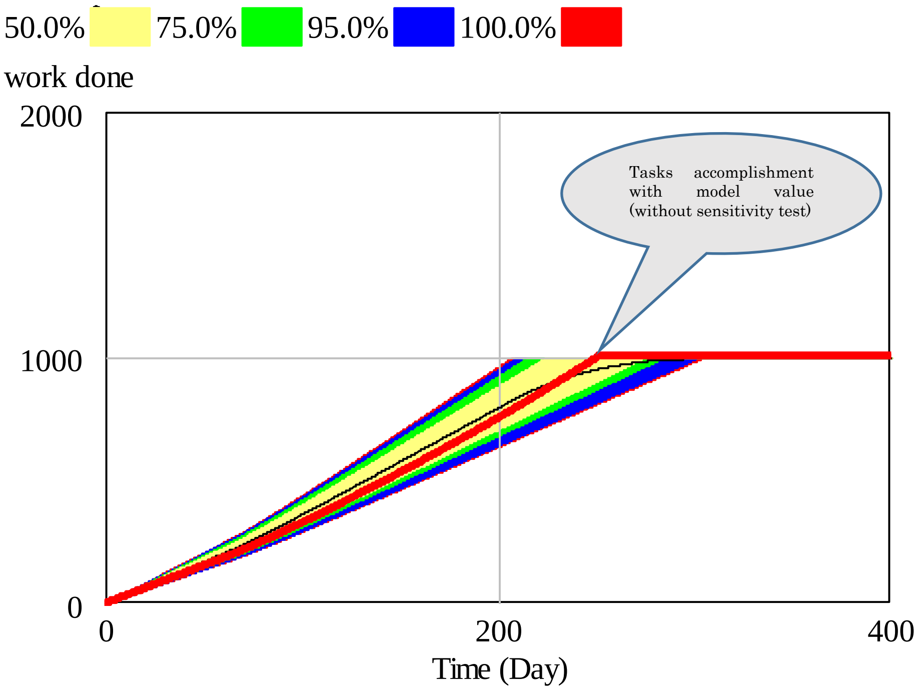


Fig. 7.2 Sensitivity run #1 for Tasks Accomplishment

As for the hiring delay, the time is considered for adjusting hiring, and after this period, new workforces are joined to the development team. In our model, we have assumed the value for hiring delay 60 days. So during the sensitivity test, we have varied this value in between 40 to 90 days to observe the differences. The uncertainty of task accomplishment is not very high in this run since this is a secondary influence to task accomplishment through the number of total workforces.

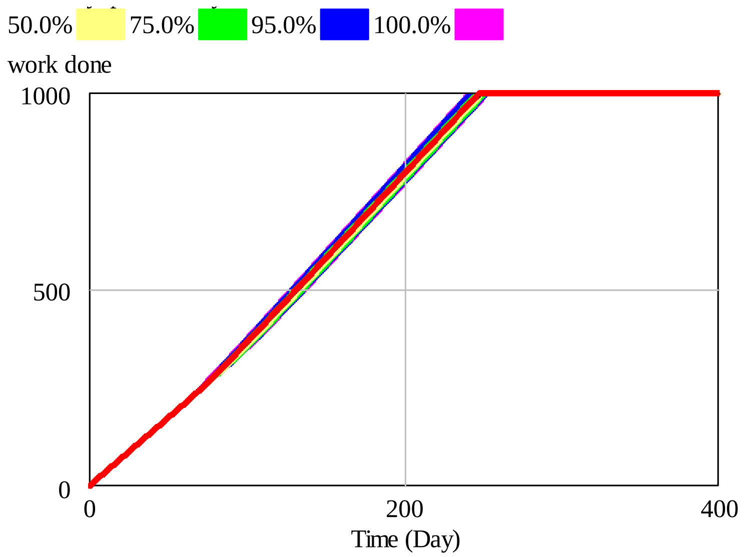


Fig. 7.3 Sensitivity run #2 for Tasks Accomplishment

7.3.2 Impact on Undiscovered Rework

In the area of undiscovered rework, productivity has a significant impact as we can see from figure 7.5. We can see that the amount of undiscovered goes high when the productivity varied since productivity is directly related to undiscovered through rework generation. And following this, hiring delay have the least impact on productivity as shown in figure 7.6. In the figures, Y-axis represents the amount of undiscovered rework that are generated depending upon the time in X-axis.

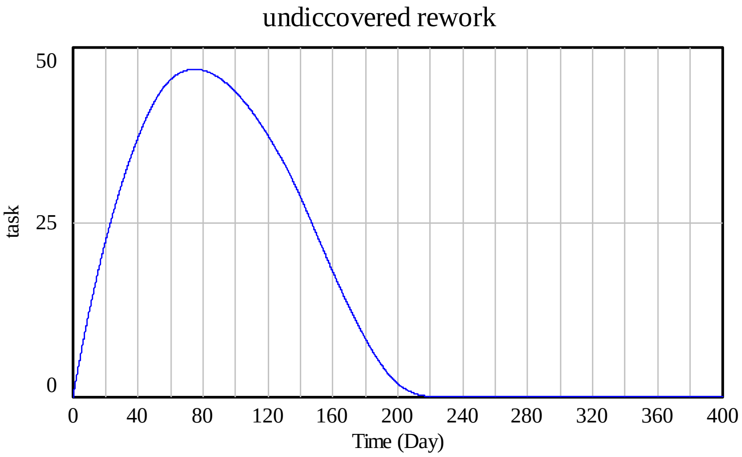


Fig. 7.4 Undiscovered Rework

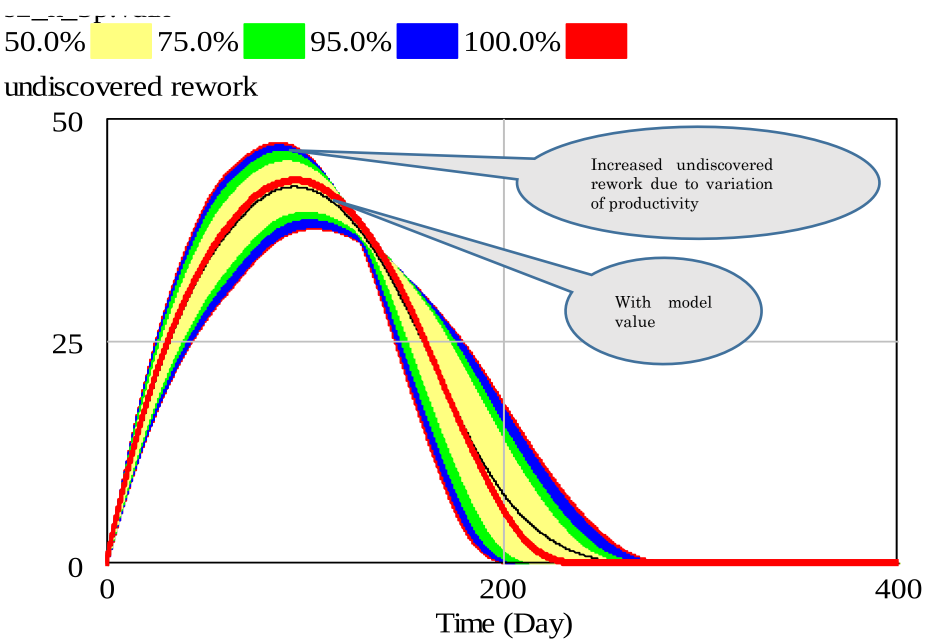


Fig. 7.5 Sensitivity run #1 for Undiscovered Rework

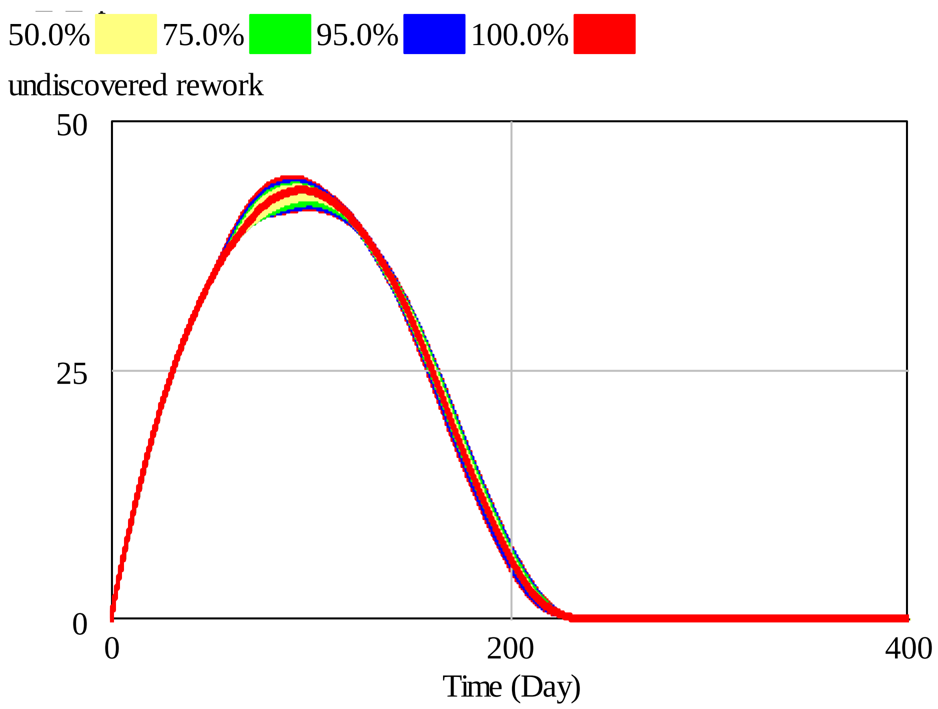


Fig. 7.6 Sensitivity run #2 for Undiscovered Rework

7.3.3 Impact on tasks development rate

Productivity is directly related to task development. So, the variation of productivity affects task development (increased) as we can see from figure 7.8. And the hiring delay has minimum impact on task accomplishment rate this is because , this variable act as a secondary impacting factor to task development. Through the figures 7.7, 7.8, 7.9, the amount of error-free tasks developed in each day has shown by vertical axis where the horizontal axis represents that time.

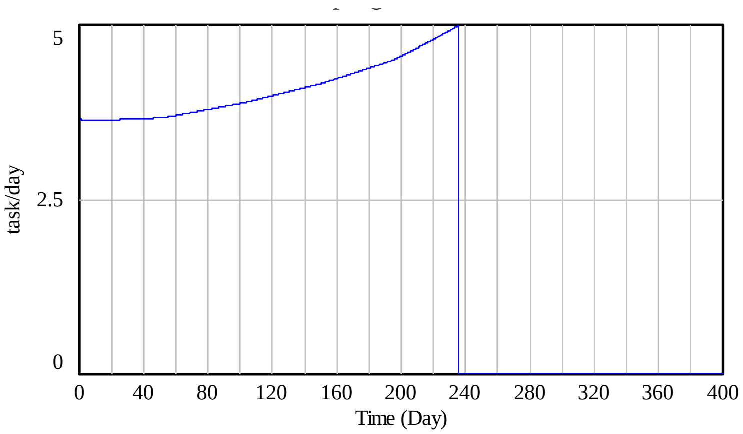


Fig. 7.7 Tasks Development

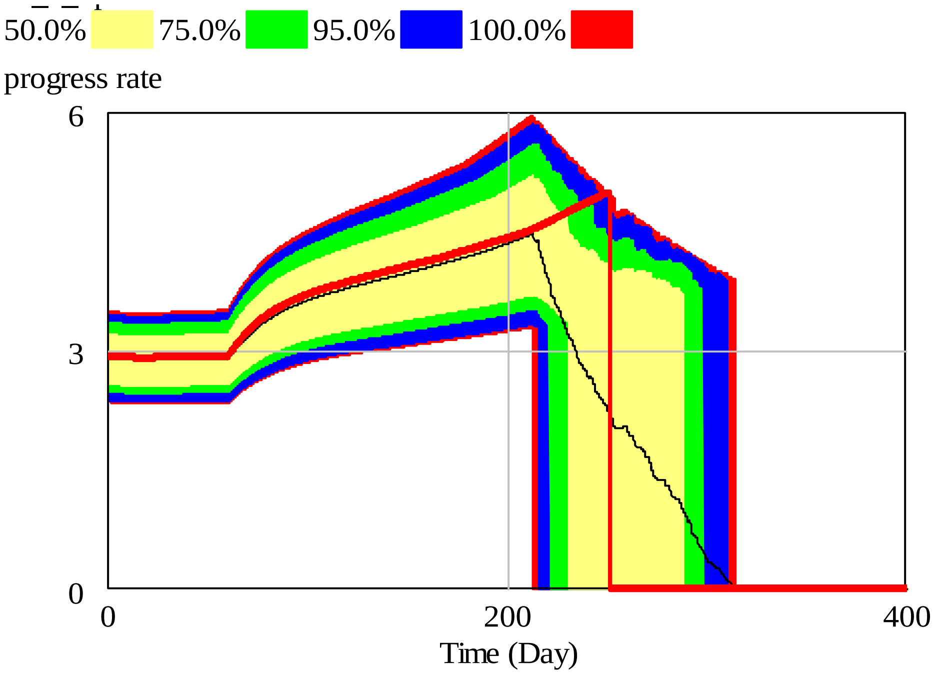


Fig. 7.8 Sensitivity run #1 for Task Development

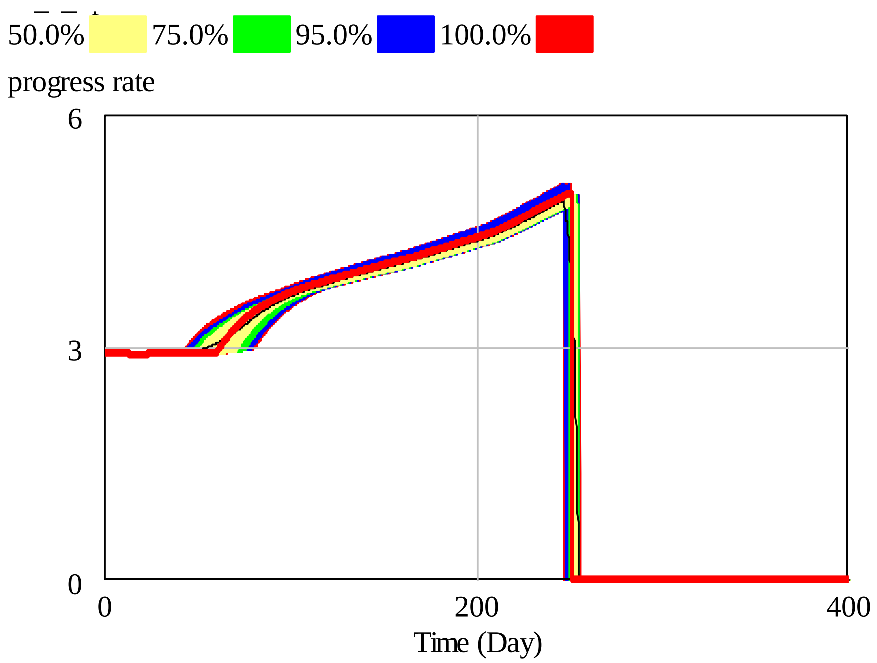


Fig. 7.9 Sensitivity run #2 for Task Development

7.3.4 Simulated Change on Perception

Table 7.3 Simulated Change to Performance Variable Due to Changing Productivity

	Project completion time	Undiscovered rework	Task development
Productivity [0.8 - 1.2] (task/man-day)	Decrease by 12%	Increase by 12.5%	Increase by 20%
	Increase by 12%	Increase by 12.5%	Decrease by 20%

Table 7.4 Simulated Change to Performance Variable Due to Changing Hiring Time

	Project completion time	Undiscovered rework	Task development
Hiring delay [30-90](days)	increase by 1.7%	decrease by 1.3%	Decrease by 1.15%
	decrease by 1.7%	increase by 1.3%	increase by 1.15%

7.3.5 Impact of overtime on Productivity

Another sensitivity test we have done for productivity considering overtime. As we have seen in the validation and case study result analysis that overtime has a major impact on productivity. When workforces work with overtime for a long time, it decreases productivity. Moreover, if the amount of overtime per day is high, it affects more. Considering these situations, we have done the sensitivity test for productivity by choosing the different amounts of overtime and obtained the productivity result as following in figure 7.10.

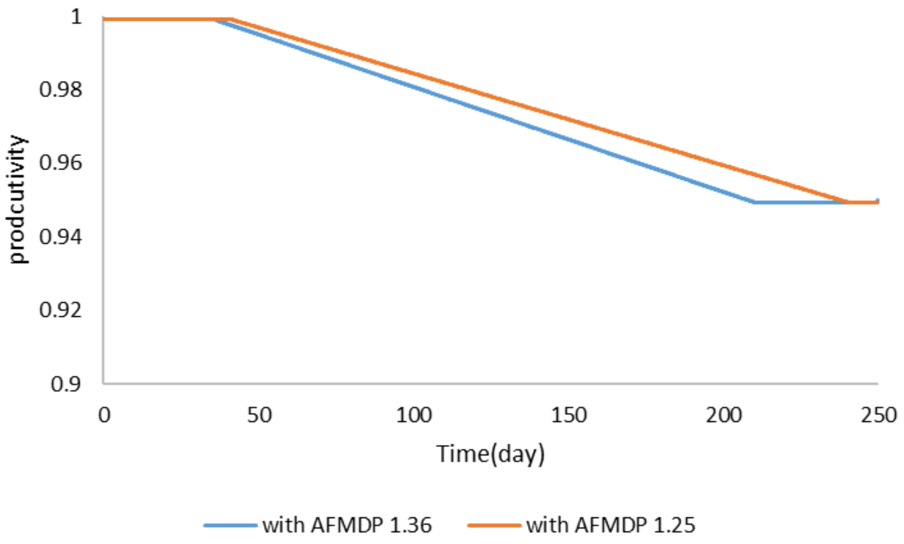


Fig. 7.10 Sensitivity Analysis for Productivity Considering Overtime

Form this figure we can see that when the overtime amount is high, AFMDP is 1.35 on average, productivity decreases more. But the amount of decrease is not high comparing to the nominal value, around 5% of the nominal value.

So considering either interval-based or continuous overtime, any of the options can be chosen for project development.

7.4 Summary

The sensitivity analysis generates awareness of what variables are sensitive to certain other variables while changing within the system boundaries. Once these sensitivities are understood and characterized during the project cycle, the potential effects will be known and the user can dynamically analyze the impact in the specific area.

During model validation and case study, we have used several datasets and analyzed the results. But the consequences of the influenced variables are the same for all cases. That's why in the sensitivity analysis, we have shown the impact of changing the value range for only one case.

Through the sensitivity analysis, we have seen that productivity has a substantial impact on the majority of the variables that are related to measuring project performance. This is because productivity defines the measure of efficiency and are directly related to project performance. As for the other parameters-overtime, hiring delay also generates uncertainty in the project performance. Therefore, a project manager trying to control the progress of the project may use these parameters as a meddling point of the system.

Chapter 8 Discussion

In this chapter, at first, we will discuss about Policy Design and then explained the insights from the research that have been obtained.

8.1 Policy Design for Project Management

Through the analysis, firstly, we have explained about model validation process and performed validation with different scenarios. Since the validation process has been completed, the model is said to be a base of theoretical ground. According to the analysis of several essential parameters involved to project performance both in the validation and case study process, we can come to a final point that it is possible to do the prediction and can make decisions about design and management following the given resources to obtain a better performance. Based on the simulation results analysis, a list of several policies would be illustrated.

- **Designing a Feasible Project Plan Considering Hiring:** A project plan indicates the success or failure of a project. A project that is both overestimated and underestimated, may cost more at the end. Thus to make a proper project plan before starting the project, is very essential. Also when considering rework, it affects the project schedule as we have seen through the analysis. The purpose of analyzing different scenarios through the validation and case study is to set up the policies for making a decision through which a realistic plan can be made based on the requirements.

For example, if the project manager focuses on hiring additional workforces to meet the project criteria, then from the figure 8.1, we can interpret which scenario options can be applicable based on the given resources and can be completed within the deadline, not very early of the deadline, or exceed the deadline. So making a proper and feasible plan is very important.

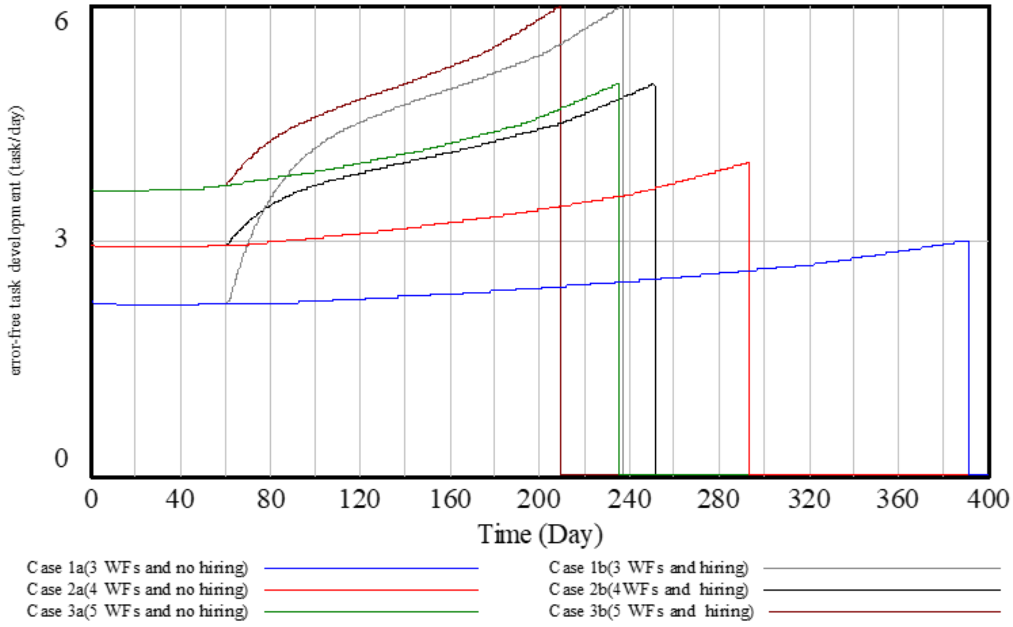


Fig. 8.1 Tasks Development Rate Considering Hiring

This figure represents the amount of tasks development per day. Hence, when the project managers are making plan, through this figure, it would be easier for them to get an idea about how many workforces are needed to be hired based on the resources and what will be the expected completion time for the project.

And along with project completion time, if the project manager wants to get an estimation of average cost, then from the following figure 8.2, this criteria can be easily obtained for a feasible planning.

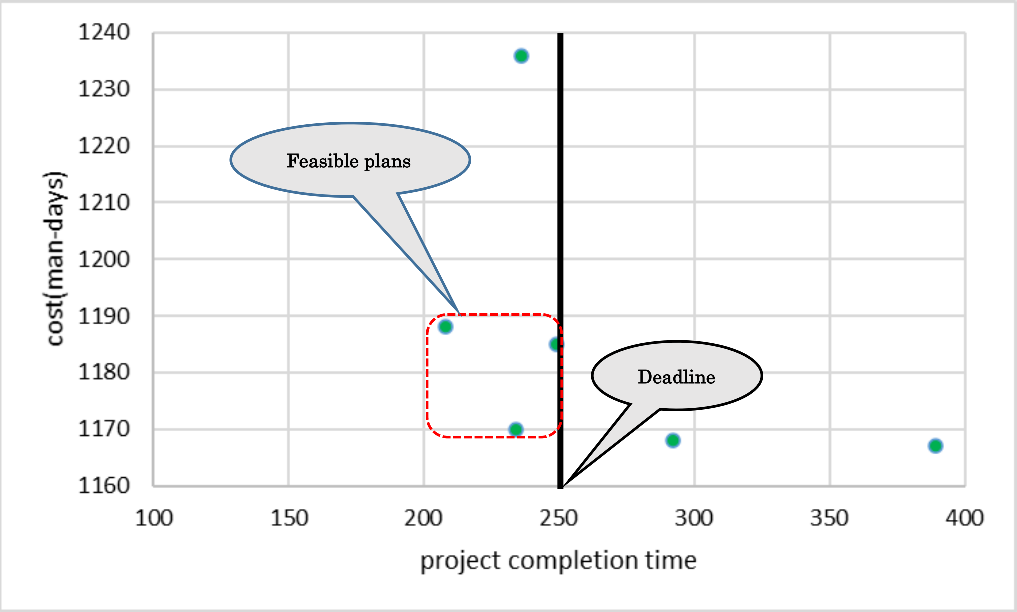


Fig. 8.2 Tasks Accomplishment Time with Required Cost

From this figure, a project manager can easily understand that how much time is required and how much it would cost for each scenario to complete the project. And based on this assumption, a better project planning could be made before starting the development of the project.

- **Designing Feasible Project Plan Considering Overtime:** Again, if the project manager doesn't want to hire new workforces, instead attempt to work overtime, if necessary for project completion, then from the following figure 8.3, a sensible option can be chosen that, in which way and how much overtime is needed to complete the project on time by keeping all the resources fixed.

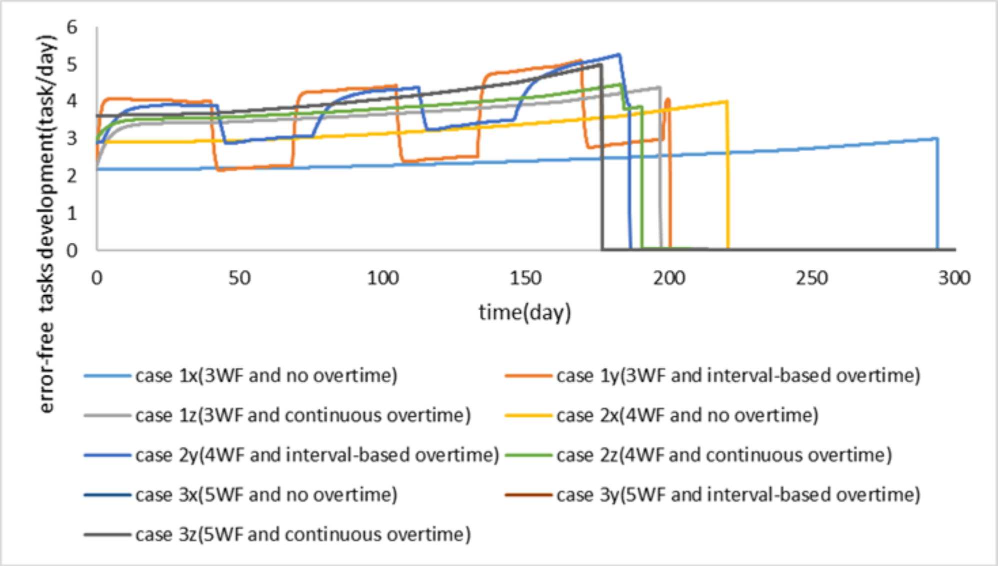


Fig. 8.3 Tasks Development Rate Considering Overtime

From this figure 8.3, the project mangers would get an clear idea about consequences of applying overtime. They would understand the behavior of task development per day with overtime and without overtime. For overtime, both interval-based and continuous overtime have been considered. So, from this analysis, a project manager could get concept task development whether they need to consider overtime or not and if yes, then which option will be better for their project. This would help them to make decision of applying overtime. So, it would be easier for a project manager to choose a feasible plan considering both completion time and cost while intention is to apply overtime. Choosing a feasible plan considering overtime would be clearer from the following figure 8.4 through observing both cost and duration. A project manager can easily understand and decide an option with the type of overtime.

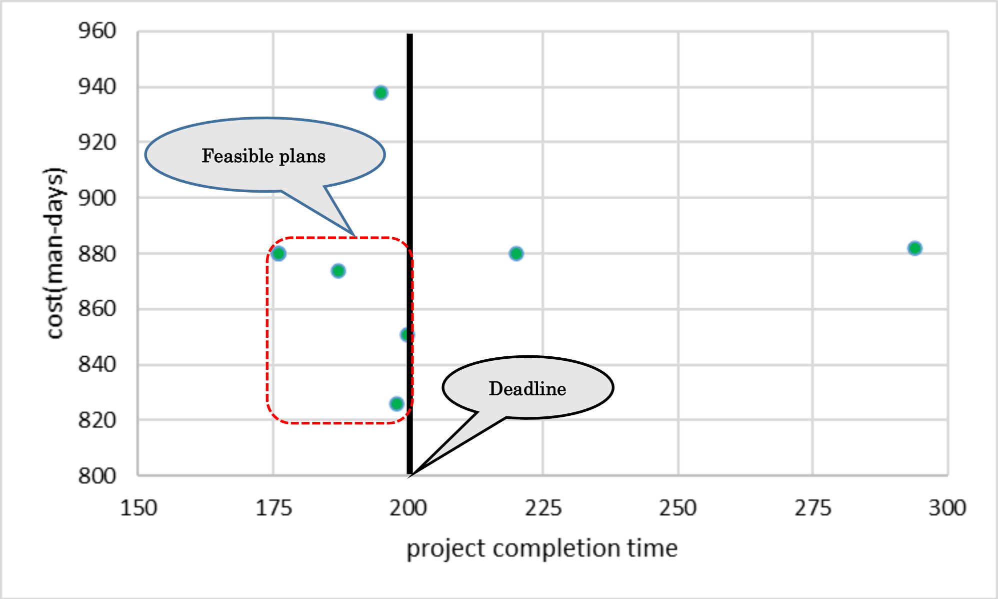


Fig. 8.4 Tasks Accomplishment Time with Required Cost

8.2 Contribution

In this thesis, we have explored the following hypothesis:

"Modeling software development projects considering rework to enhance our understanding of and make a prediction about model-based decision making through which planning, controlling can be done to improve the performance of software projects".

We have emphasized on two points through this hypothesis.

1. **Enhance Understanding About Model-Based Decision Making:** For enhancing our understanding of project behavior, the proposed model describes causal loop diagram to represent a key dynamic, then reproduce detailed behavior and dynamics of a project by System Dynamics simulation model. Overtime and hiring are considered as input and the simulator provides decrease of productivity, rework amount, accomplished task, and other detailed parameters of a project as time series data as we shown through our analysis. These detailed outputs of the proposed simulator

enhance and embody the project expectations of project members and stakeholders of the project and would decrease the ambiguity of their expectations. While considering overtime, two different types of overtime usage - continuous overtime and interval-based overtime has been explained. And for the detailed output analysis, several options has been explored in order to get better understanding of decision-making for project planning .

2. **Making Prediction About Model-Based Decision Making** : As for the contributions to model-based decision making in project management, project managers can make decisions referring to the results of the simulator. The results include fully described details in time series, the project manager can explore their options and the outcomes of the options selected by the simulator. In the simple cases in Chapter 5, along with figures 8.2 and 8.4, the project manager has several options which are relevant for the given project context. Hiring and overtime are common options and usually they decide subjectively and intuitively. The proposed approach provide at least traceability of the decisions and is expected to improve the quality of the decisions. This framework for software development project management would be designed to better management of projects in real-time and support for behavioral understanding, prediction, and evaluation of process improvement, project planning across a range of alternative processes.

8.3 Limitations

One key limitation of our analysis, after hiring new workforces and getting trained, we considered the same productivity for all of the workforces. Another limitation is that while considering the rework cycle, we did not distribute the manpower separately for task development and rework cycle. Also during our model analysis, we have considered the project scope as fixed. In this case, if scope change happens during project development, the consequences have not been defined.

Chapter 9 Conclusion and Future Research

9.1 Conclusion

A method for supporting decisions for software development project management against uncertain rework and fluctuating productivity of engineers has been developed in this study.

This decision support method gives a clear understanding of uncertain generation of rework and its impact on performance through a detailed explanation of different steps that have shown across the simulation results analysis. The model successfully demonstrated the behavior of a project and can predict the relative change in effort, productivity, through hiring and overtime . Based on understanding of these changeable behavior, it would be spontaneous to design a feasible project plan. The method would support software development project management to embrace both static and dynamic elements of the existing, and take comprehensible actions based upon the quantitative results of the analysis.

Another purpose of designing this method is to enhance our understanding of and make prediction about model-based decision making through which a proper project planning can be obtained before starting the development phase. Through the analysis of validation process and case study, it would be said that, our proposed method gives an understandable basis of designing and making decision of choosing a feasible plan among several options based on the given resources, mainly focusing human resources and managerial policies. And it can be stated that this method would support behavioral understanding, prediction, and evaluation of process improvement, project planning, and controlling based on overtime plan and human resources.

The summary of the results that we have obtained through our analysis has given below. This analysis would give an idea of model-based decision making.

9.1.1 Summary of the Results

The purpose of this work is to design a decision support method for project management which would also enhance our understanding of and gain insights into model-based decision making through which software development is managed. And to achieve this objective, we have accomplished the following tasks

i. Developed a System Dynamics model of software development project management.
   - ➢ Integrates our knowledge of software development projects such as productivity, planning, controlling. The model identifies the feedback mechanisms and uses them to structure the relationships in software project management.

ii. Performed model validation with two different datasets for hiring and overtime separately and showed a comparison of different scenarios to get an understanding of making a feasible plan for project development.
   - ➢ The variety of scenarios helps us to understand to make a project plan feasibly if we need to consider hiring or applying overtime based on the given resources. It also helps to understand the complexity and uncertainty of the project based on rework.

iii. Performed a case study considering both hiring and overtime management simultaneously with different scenarios and explained the consequences of each scenario for a better understanding.
   - ➢ Helps to understand the dynamic behavior and feedback mechanisms in a more detailed way when the project size is large and need to consider hiring and overtime management both at the same time during the development phase of the project.

iv. Used the model to study the dynamic implications of managerial policies and procedures.

- ➢ Through the analysis of different scenarios, it assists us to make managerial policies which would be helpful for the project manager to make a more accurate plan.

9.2 Future Research

If we try to model every possible factor that affects the system, the model would become extremely complex and difficult to understand the performance behavior clearly. Taking into consideration the scope of the research, we have developed our simulation model with a certain usage of factors and assumptions. The further activities based on this research will be as follows

i. **Scope Change Management**: The scope of a software project defines the features of the software. This scope can be changed during the development period or after development. When the scope change happens, it affects the development environment difficultly. In our current work, we have only focused on fixed scope, after planning scope won't be changed. So, in the next phase, we will focus on scope change management.

ii. **Testing Schemes**: In our current work we have focused on the rework cycle for managing the error-based tasks and did not consider separate testing phase which would be in the consideration of or next phase.

iii. **Multi-Project Situation:** In our model, we did not consider multi-project situations and therefore did not assume the transfer of the engineers between projects. An improvement over our model could be to consider multi projects being done concurrently and look into the performance behavior when transfer of workforces happen between projects.

References

[1] James M. Lyneis[a]* and David N. Ford[b] “System dynamics applied to project management: a survey, assessment, and directions for future research ”, Syst. Dyn. Rev. 23, 157–189, 2007

[2] “A Guide to the PROJECT MANAGEMENT BOBY OF KNOWLEDGE”, (PMBOK® GUIDE), sixth edition

[3] Chetan D. Vajre, “Modeling dynamic interactions in a software Development project”, University of South Florida, Graduate Theses and Dissertations, 2003

[4] Armindokht Nasirikaljahi, “ The dynamic of modern software development project management and the software crisis of quality An integrated system dynamics approach towards software quality improvement”, Thesis paper on Master of Philosophy in System Dynamics, UNIVERSITY OF BERGEN, 2012

[5] Tarek K. Abdel-Hamid, “The Dynamics of Software Development Project Management: An Integrative System Dynamics Perspective”, Massachusetts Institute of Technology January 1984

[6] Jianguo Jia, Xia Fan, Yu Lu, “System Dynamics Modeling for Overtime Management Strategy of Software Project”, January 2007

[7] Tarek Abdel Hamid, “The Dynamics of Software Project Staffing: A System Dynamics Based Simulation Approach ”, IEEE Transaction on Software Engineering, Vol. 15, No. 2, 1989

[8] Raymond Joseph Madachy, "A Software Project Dynamics Model For Process Cost, Schedule And Risk Assessment", University Of Southern California, 1984

[9] Kahen, G., Lehman, M.M., Ramil, J.F. & Wernick, P., "System Dynamics Modeling of Software Evolution Processes for Policy Investigation: Approach and Example", *The Journal of Systems and Software*, 59: 271-281, 2001

[10] Zhikun Ding, Wenyan Gong, Shenghan Li and Zezhou Wu *, "System Dynamics versus Agent-Based Modeling: A Review of Complexity Simulation in Construction Waste Management", Sustainability 2018, 10, 2484, 2018

[11] Raymond J. Madachy, "Software Process Dynamcis", 2007

[12] Jogeklar, N. & Ford, D., "Product Development Resource Allocation with Foresight, European Journal of Operational Research", 160(1): 72-87,2003

[13] Reichelt, K.S. & Lyneis, J.M., "The Dynamics of Project Performance: Benchmarking the Drivers of Cost and Schedule Overrun", *European Management Journal* 17(2): 135-150,1999

[14] Rus, I., Collofello, J. & Lakey, P., "Software Process Simulation for Reliability Management, *The Journal of Systems and Software"*, 46: 173-182,1999

[15] Suinan Li, "A generic model of project management with Vensim", master's thesis, Faculty of Engineering and Science, Agder University, 2008

[16] Cooper KG., The rework cycle (a series of 3 articles): why projects are mismanaged; how it really works . . . and reworks . . . ; benchmarks for the project 1993

[17] Abdel-Hamid, Tarek K., and Stuart E. Madnick, "Software project dynamics: An integrated approach", New Jersey: Prentice-Hall, Inc., 1991

[18] Timothy Charles Delobe, "Project dynamics: An analysis of the purpose and value of system dynamics applied to information technology project management ", Masters Theses, James Madison University, 2010

[19] Sterman, J.D., " Business Dynamics: Systems Thinking and Modeling for a Complex World", Irwin/McGraw-Hill, Chicago, USA, 2000

[20] Debby G. J. Beckers, Dimitri van der Linden, Peter G.W. Smulders, Michiel A. J. Kompier, Marc J. P. M. van Veldhoven, Nico W. van Yperen, "Working Overtime Hours: Relations with Fatigue, Work Motivation, and the Quality of Work", JOEM • Volume 46, Number 12, December 2004

[21] Albert DS, "The Economics of Software Quality Assurance", National Computer Conference, 1976.

[22] Nicholas D. Kefalas, "Project Management Utilizing System Dynamics and Design Structure Matrices in Conjunction with the Earned Value System", Master of Science in Engineering and Management, Massachusetts Institute of Technology, 2000

[23] Chul-Ki Chang* and Sungkwon Woo**, " Critical Review of Previous Studies on Labor Productivity Loss due to Overtime", KSCE Journal of Civil Engineering, 21(7), 2017

[24] John D. Sterman*, "All models are wrong: reflections on becoming a systems scientist", Syst. Dyn. Rev. 18, 501–531, 2002

[25] "Model Validation Principles Applied Risk and Capital Models in the Insurance Industry", North American CRO Council, 2012

[26] Mustafa Hekimoglu, Yaman Barlas, "Sensitivity Analysis of System Dynamics Model by Behavior Pattern Measures", 2010

[27] "Vensim®,Ventana®, Simulation Environment", User's Guide, Copyright © 1998-2007 Ventana Systems, Inc. Revision Date: July 4, 2007

[28] Deutsch, M.S. "Verification and Validation," in Software Engineering, R.W. Jensen and C.C. Tonies (eds), Prentice-Hall, Inc., Englewood Cliffs, NJ,1979.

[29] Tim Taylor[a] and David N. Ford[b]*, "Tipping point failure and robustness in single development projects", Syst. Dyn. Rev. **22**, 51–71, 2006

[30] James M. Lyneis,[a]* Kenneth G. Cooper[a] and Sharon A. Els[a], "Strategic management of complex projects: a case study using system dynamics", *Syst. Dyn. Rev.* **17**, 237–260, 2001

[31] Hazhir Rahmandad[a]* and Kun Hu[a], "Modeling the rework cycle: capturing multiple defects per task", *Syst. Dyn. Rev.* **26**, 291–315 2010

[32] Hazhir Rahmandad[a]* and David M. Weiss[b], "Dynamics of concurrent software development", *Syst. Dyn. Rev.* **25**, 224–249, 2009

[33] Morvin Savio Martis, "Validation of Simulation Based Models: A Theoretical Outlook", The Electronic Journal of Business Research Methods, Volume 4, Pp 39-46, 2006

[34] C P Ogbu[1] and N. Olatunde[2], "Relationship Between Organizational Effectiveness and Project Performance of SME Contractors: a Developing Country Perspective", Journal of Construction Business and Management, JCBM, 3(2). 1-16, 2009

[35] Brandon D. Owens,[a]* Nancy G. Leveson[b] and Jeffrey A. Hoffman[c], "Procedure rework: a dynamic process with implications for the "rework cycle" and "disaster dynamics"", Syst. Dyn. Rev. 27, 244–269 2011

[36] Antony Lee Powell, "Right on Time: Measuring, Modelling and Managing Time-Constrained Software Development", Submitted for the degree of Doctor of Philosophy, University of York, Department of Computer Science, 2001

[37] James Hannan, “Acquiring Entry-Level Programmers,” Computer Programming Management, 1982.

[38] Yujia Ge[2]*, Bin Xu[1], “Dynamic Staffing and Rescheduling in Software Project Management: A Hybrid Approach”, PLOS ONE, June 10, 2016

[39] LAN CAO, BALASUBRAMANIAM RAMESH, TAREK ABDEL-HAMID, “Modeling Dynamics in Agile Software Development”, ACM Transactions on Management, Information Systems, Vol. 1, No. 1,Article 5, Publication date: December 2010

[40] David G Chernoguz, “The system dynamics of Brooks’ Law in team production”, Simulation: Transactions of the Society for Modeling and Simulation International 0(00) 1–29, 2010

[41] Moonseo Parka* and Feniosky Peña-Morab, “Dynamic change management for construction: introducing the change cycle into model-based project management”, *Syst. Dyn. Rev.* **19**, 213–242, 2003

[42] Julie Yu-Chih Liu[a,1], Hun-Gee Chen[b,2], Charlie C. Chen[c,*], Tsong Shin Sheu[d,3], “Relationships among interpersonal conflict, requirements uncertainty, and software project performance”, International Journal of Project Management 29, 547–556, 2011

[43] Lin Wang[a,*], Martin Kunc[b], Si-jun Bai[c], “Realizing value from project implementation under uncertainty: An exploratory study using system dynamics☆”,International Journal of Project Management 35, 341–352, 2017

Acknowledge

I would like to show my sincerest gratitude to the people who have helped and supported me during my Master's study at The University of Tokyo. Foremost, I would like to give my sincere thankfulness to my academic supervisor, Associate Professor Kazuo Hiekata, for his precious instructions and uncountable enlightenments on my academic study. Without him, I could get lost facing various obstacles of my research life during the study. I also want to express my super appreciations to Associate Professor Bryan Moser for his help and kind supports sharing time with my research at any time despite being so busy and made an opportunity to talk with Professor James M. Lyneis. A lot thanks to Professor Yutaka Takahashi, for his instruction on developing my model.

Great appreciations should go to Dr. Marc-Andry-Chevy-Macdonald, who helped me a lot to understand System Dynamics and to design a model with System Dynamics.

Thanks to Ms. Kazuko Yamamoto, Ms. Yoko Ohmori for their kind supports of administrative affairs. A great thanks to Mr. Masakazu Enomoto for technical support.

I appreciate supports from my lab members. To my seniors who have already graduated from my lab: Dr. Shinnosuke Wanaka, who gave me many valuable advice. Kensuke Uno, who was very kind. Great appreciation to Rujia Wang, who helped me a lot to settle in Japan; without her help, it was impossible for me to deal with regular activities. I am very grateful to Kohei Matsuo, who helped me by sharing his brilliant advice regarding my research; Tatsuya Kasahara who always helped me whenever I face any problem shares many joys and also about Japanese culture.

Many Thanks to Yuji Horii and Aitaro Chaya who were very kind and sincere. Thanks to Yushorino Okubo, with whom I talked a lot about my research work and many joyful things; Yuta Mizono who taught me to learn calligraphy.

Many thanks to my current lab members. Ira Winder, who is very kind always helps whenever is needed. Zhinan Zaho, who always helps me whenever I face a problem regarding the Japanese language during the course work. We did many assignments of course work together which was in Japanese and he helped me to understand. We participated together as a team Hackathon program. Both of us are hoping to be graduated this semester. I wish him the best of luck.

Saul Trujillo Castillo, another international student, is always very helpful and I use to talk to him many times and share a lot of joy and culture. Special thanks to Misa Makino, who always helps me and come with me whenever I need help regarding Japanese, I am very grateful to her. Thanks to Yuta Shirono for his kind and helpful attitude. Thanks to Mingrui Wang, with whom I talk a lot about my research work as well as many other things.

Saul Trujillo Castillo, Misa Makino, and Yuta Shirono have to attend their interim review of their thesis. I wish them the best of luck.

Many thanks to Dylan Delaporte, a special auditing student, for taking part in an experiment related to teamwork. Thanks to UTSIP student, Prince Agarwal, for taking part in an experiment on IoT.

The new members of our laboratory- Shio Bo, Nayu Yuyama, Michio Hayashi, and Katsuya Torii, I could not meet them fact to face due to COVID-19 pandemic. But we talk through online meetings always. Many thanks to them for their kind attitude and for helping me by giving many precious ideas regarding my research.

I have learned and experienced a lot during my master's study. I have got many new ideas, learned many new things which will be beneficial for me in the future.

My greatest appreciation to my friends who have been always there for me. Whenever I need help, I always get them either they are in Japan or Bangladesh.

And my furthermost gratitude to my family who always has given me the biggest support and strength for my pursuit of the oversea study.

2020/07/30

Mst. Taskia KHATUN

Appendix A

Model Implementation

A.1 Vensim Introduction

For developing the System Dynamics model, we have used Vensim software.

Vensim is a visual modeling tool that allows you to conceptualize, document, simulate, analyze, and optimize models of dynamic systems. Vensim provides a simple and flexible way of building simulation models from a causal loop or stock and flow diagrams(User's Guide, Vensim, 2007).

By connecting words with arrows, relationships among system variables are entered and recorded as causal connections. This information is used by the Equation Editor to help you form a complete simulation model. You can analyze your model throughout the building process, looking at the causes and uses of a variable, and also at the loops involving the variable. When you have built a model that can be simulated, Vensim lets you thoroughly explore the behavior of the model.

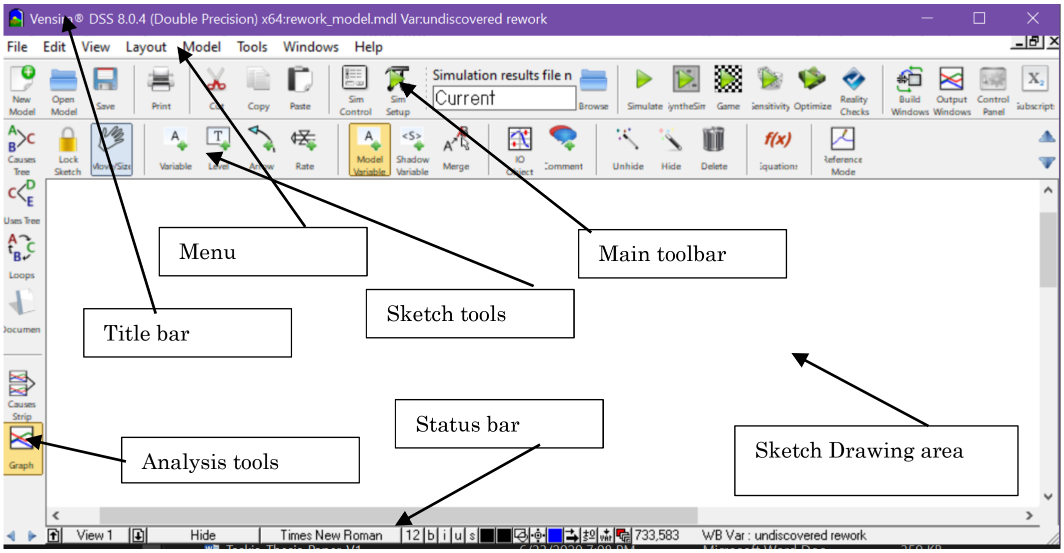
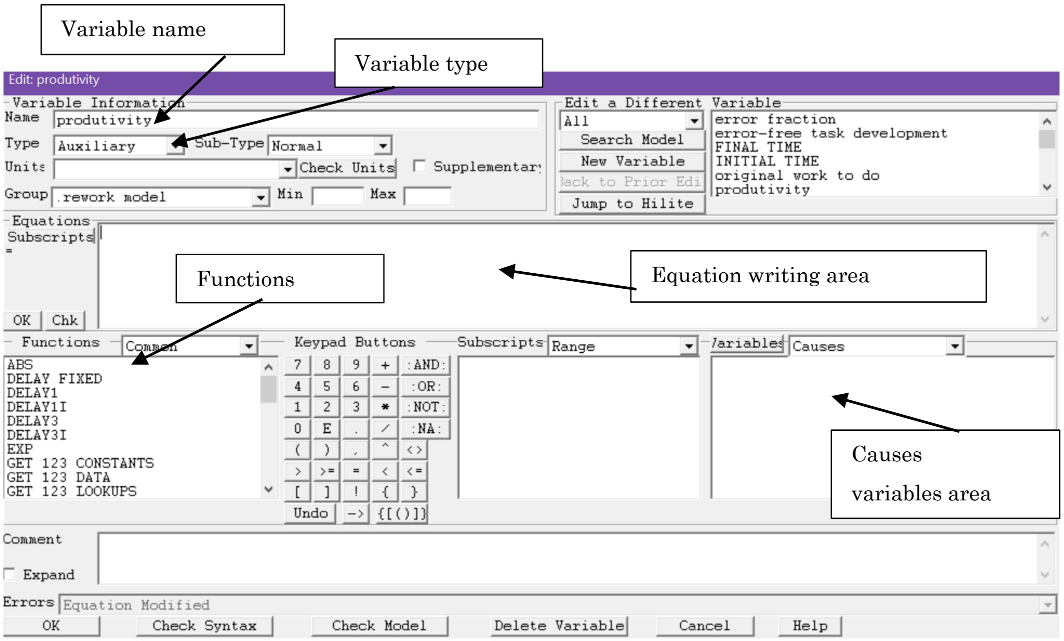


Fig. A.1 The Window of Vensim for Model Sketching



Edit: produtivity

Variable Information
Name: produtivity
Type: Auxiliary
Sub-Type: Normal
Units:
Group: .rework model
Min:
Max:

Edit a Different Variable: All
error fraction
error-free task development
FINAL TIME
INITIAL TIME
original work to do
produtivity

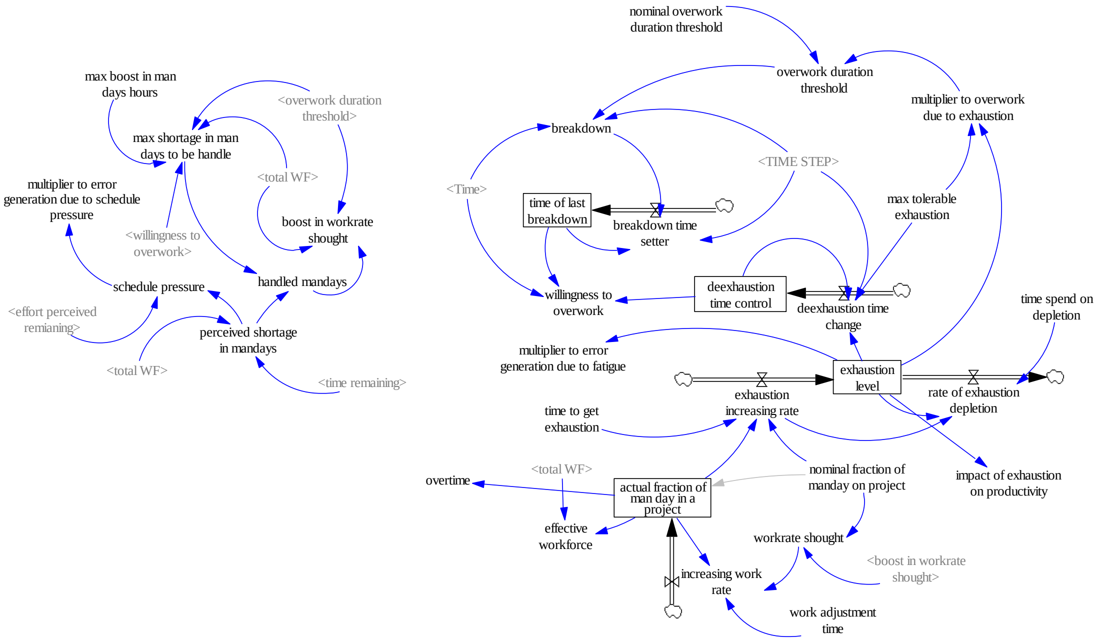
Functions: Common
ABS
DELAY FIXED
DELAY1
DELAY1I
DELAY3
DELAY3I
EXP
GET 123 CONSTANTS
GET 123 DATA
GET 123 LOOKUPS

Subscripts: Range
Variables: Causes

Errors: Equation Modified

Fig. A.2 Equation Editor of Vensim

A.2 System Dynamics Model



nominal overwork duration threshold
overwork duration threshold
multiplier to overwork due to exhaustion
max boost in man days hours
<overwork duration threshold>
max shortage in man days to be handle
<total WF>
boost in workrate shought
multiplier to error generation due to schedule pressure
<willingness to overwork>
handled mandays
schedule pressure
<effort perceived remaining>
perceived shortage in mandays
<total WF>
<time remaining>
breakdown
<TIME STEP>
<Time>
time of last breakdown
breakdown time setter
max tolerable exhaustion
willingness to overwork
deexhaustion time control
deexhaustion time change
time spend on depletion
multiplier to error generation due to fatigue
exhaustion level
exhaustion increasing rate
rate of exhaustion depletion
time to get exhaustion
impact of exhaustion on productivity
overtime
<total WF>
actual fraction of man day in a project
nominal fraction of manday on project
effective workforce
workrate shought
<boost in workrate shought>
increasing work rate
work adjustment time

Fig. A.3 Sub-System for Overtime Management

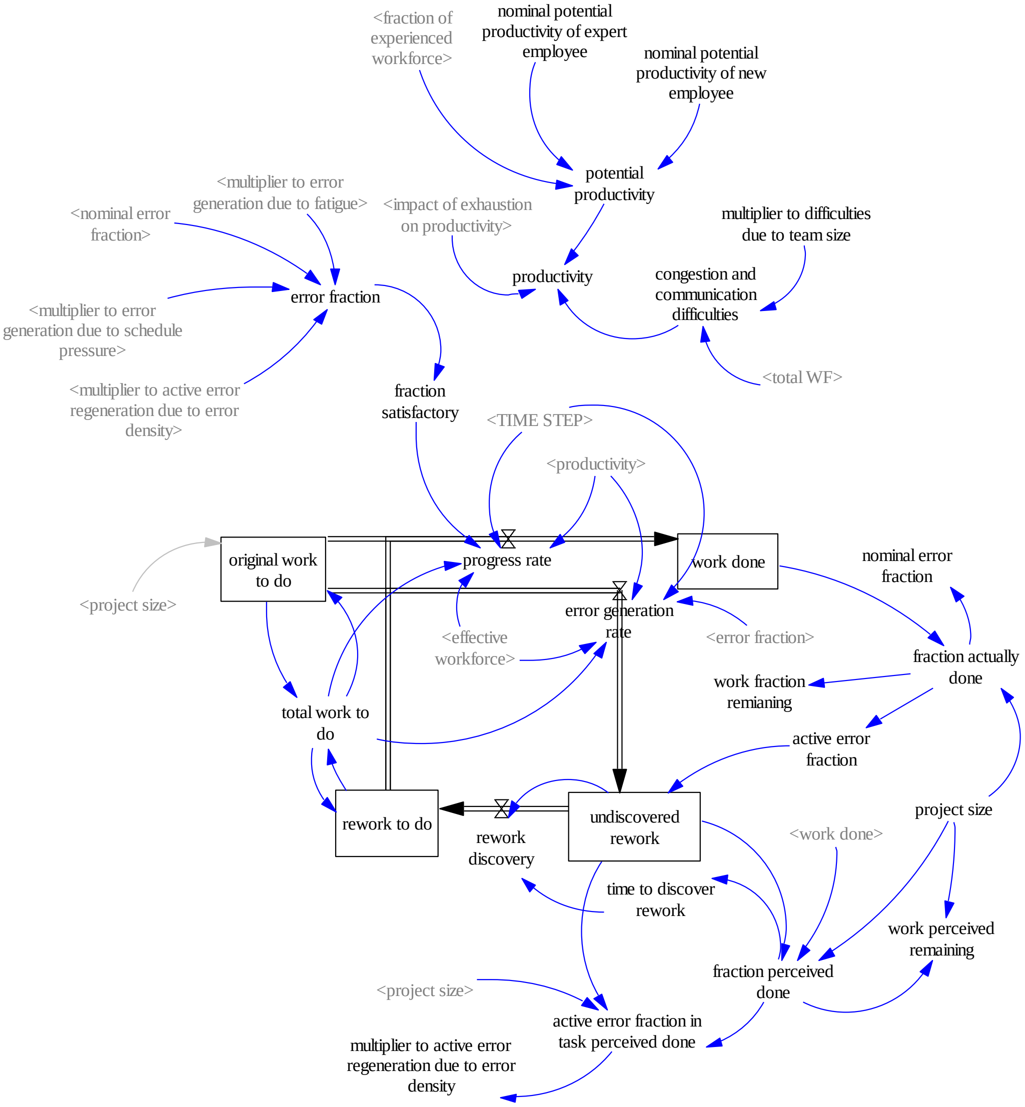

<fraction of experienced workforce>
nominal potential productivity of expert employee
nominal potential productivity of new employee
potential productivity
<multiplier to error generation due to fatigue>
<nominal error fraction>
<impact of exhaustion on productivity>
multiplier to difficulties due to team size
productivity
congestion and communication difficulties
error fraction
<multiplier to error generation due to schedule pressure>
<total WF>
<multiplier to active error regeneration due to error density>
fraction satisfactory
<TIME STEP>
<productivity>
original work to do
progress rate
work done
nominal error fraction
error generation rate
<project size>
<effective workforce>
<error fraction>
fraction actually done
work fraction remianing
total work to do
active error fraction
rework to do
rework discovery
undiscovered rework
project size
<work done>
time to discover rework
work perceived remaining
fraction perceived done
<project size>
active error fraction in task perceived done
multiplier to active error regeneration due to error density

Fig. A.4 Sub-System for Task Accomplishment with Rework Cycle

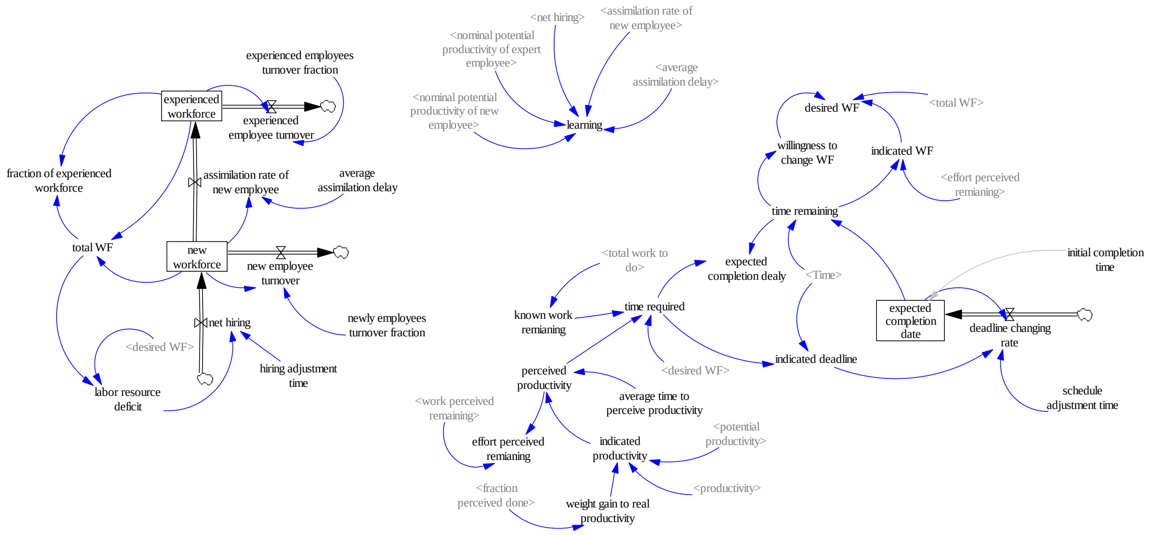


Fig. A.5 Sub-System for Hiring and Schedule Planning

A.3 Model Source Code

Stocks

```
original work to do= INTEG (-error generation rate*original work to do/total work to do-
progress rate*original work to do/total work to do, project size)
    ~       task
    ~               |
undiscovered rework= INTEG ((error generation rate*active error fraction-rework
discovery),
            0)
    ~       task
    ~               |
work done= INTEG (progress rate, 0)
    ~       task
    ~               |
rework to do= INTEG (rework discovery-(error generation rate*rework to do/total work
to do)-(progress rate*rework to do/total work to do),0)
    ~       task
    ~               |
experienced workforce= INTEG (assimilation rate of new employee-experienced
employee turnover,3)
    ~       empl
```

~ |

new workforce= INTEG (net hiring-assimilation rate of new employee-new employee turnover,0)
~ empl
~ |
actual fraction of man day in a project= INTEG (increasing work rate, nominal fraction of man-day on project)
~ 1
~ |
time of last breakdown= INTEG (breakdown time setter,-1)
~ Day
~ |
De-exhaustion time control= INTEG (de-exhaustion time change,0)
~ Day
~ |
exhaustion level= INTEG ((exhaustion increasing rate-rate of exhaustion depletion),0)
~ Dmnl
~ |
scheduled completion date= INTEG (deadline changing rate, estimated completion time)
~ Day
~ |

<u>Flows</u>

progress rate= MIN (productivity*effective workforce*fraction satisfactory, total work to do*fraction satisfactory /TIME STEP)
~ task/Day
~ |
error generation rate= MIN(productivity*error fraction*effective workforce, total work to do*error fraction/TIME STEP)
~ task/Day
~ |
rework discovery= undiscovered rework/time to discover rework
~ task/Day
~ |
experienced employee turnover= experienced employees turnover fraction*experienced workforce
~ empl/Day
~ |
new employee turnover= new workforce*newly employees turnover fraction
~ empl/Day
~ |
assimilation rate of new employee= new workforce/average assimilation delay
~ empl/Day
~ |
net hiring= DELAY FIXED(labor resource deficit, hiring adjustment time , 0)
~ empl/Day

```
    ~        |
  breakdown time setter= (MAX(time of last breakdown, breakdown)-time of last
breakdown)/TIME STEP
    ~    Dmnl
    ~        |
  De-exhaustion time change= IF THEN ELSE( exhaustion level/max tolerable
exhaustion>=0.1,1,-deexhaustion time control      /TIME STEP )
    ~    Dmnl
    ~        |
  rate of exhaustion depletion= IF THEN ELSE( exhaustion increasing
rate<=0,exhaustion level/time spend on depletion,0 )
    ~    1/Day
    ~        |
  increasing work rate= (workrate sought-actual fraction of man day in a project)/work
adjustment time
    ~    1/Day
    ~        |
  deadline changing rate= (indicated deadline-scheduled completion date)/schedule
adjustment time
    ~    Dmnl
    ~        |
  exhaustion increasing rate= WITH LOOKUP (((1-actual fraction of man day in a
project)/(1.2-nominal fraction of man-day on project+0.0001))/time to get exhaustion,
          ([(-0.5,0)-(2.5,2.5)],(-0.5,2.5),(-0.4,2.2),(-0.3,1.9),(-0.2,1.6),(-0.1,1.3),(0,1),\
          (0.1,0.9),(0.2,0.8),(0.3,0.7),(0.4,0.6),(0.5,0.5),(0.6,0.4),(0.7,0.3),(0.8,0.2),(0.9\
          ,0),(1,0) ))
    ~    1/Day
    ~        |
```

Auxiliary Variables

```
  error fraction= nominal error fraction*multiplier to error generation due to
fatigue*multiplier to error generation due to schedule pressure*multiplier to active error
regeneration due to error density
    ~    Dmnl
    ~        |
  learning= (net hiring*(nominal potential productivity of expert employee-nominal
potential productivity of new employee)*assimilation rate of new employee/average
assimilation delay)
    ~    Dmnl
    ~        |
  fraction perceived done= MIN((undiscovered rework+ work done)/project size,1)
    ~    Dmnl
    ~        |
  workrate shought= (1+boost in workrate shought)*nominal fraction of manday on project
```

```
	~	1
	~		|
impact of exhaustion on productivity= WITH LOOKUP ( exhaustion level,
		([(0,0)-(80,1)],(0,0),(5,0),(10,0),(20,0.01),(30,0.01),(40,0.02),(50,0.02) ))
	~	
	~		|
productivity= potential productivity*(1-congestion and communication difficulties)*(1-
impact of exhaustion on productivity)
	~	task/(Day*empl)
	~		|
perceived shortage in man-days= MAX(total WF*time remaining,0)
	~	Day*empl
	~		|
work fraction remaining= 1-fraction actually done
	~	Dmnl
	~				|
fraction actually done= work done/project size
	~	1
	~		|
max shortage in man days to be handle= max boost in man days hours*overwork
duration threshold*total WF*willingness to overwork
	~	Day*empl
	~		|
known work remaining= total work to do
	~	task
	~		|
effective workforce= IF THEN ELSE( actual fraction of man day in a project>1,actual
fraction of man day in a project*total WF, total WF )
	~	empl
	~		|
active error fraction= WITH LOOKUP ( fraction actually done,
		([(0,0)-(1,1)],(0.1,1),(0.2,1),(0.3,1),(0.4,0.95),(0.5,0.85),(0.6,0.5),(0.7,0.2),(0.8\
		,0.075),(0.9,0),(1,0) ))
	~	Dmnl
	~		|
active error fraction in task perceived done= undiscovered rework/(project size*fraction
perceived done+0.001)
	~	Dmnl
	~		|
multiplier to overwork due to exhaustion= WITH LOOKUP ( exhaustion level/max
tolerable                    exhaustion,                    ([(0,0)-
(2,2)],(0,1),(0.1,0.9),(0.2,0.8),(0.3,0.7),(0.4,0.6),(0.5,0.5),(0.6,0.4),(0.7\
		,0.3),(0.8,0.2),(1,0),(1.1,0),(1.2,0),(1.3,0),(1.4,0),(1.5,0) ))
	~	Dmnl
	~		|
boost in workrate shought=handled mandays/(total WF*(overwork duration
threshold+0.0001))
	~	Dmnl
	~		|
  nominal error fraction= WITH LOOKUP ( fraction actually done, ([(0,0)-
(1,0.25)],(0,0.25),(0.2,0.24),(0.4,0.2),(0.6,0.15),(0.8,0.09),(1,0) ))
     ~        Dmnl
     ~               |
  overwork duration threshold= nominal overwork duration threshold*multiplier to
overwork due to exhaustion
     ~        Day
     ~               |
  perceived productivity= SMOOTHI(indicated productivity, average time to perceive
productivity,1 )
     ~        task/(Day*empl)
     ~               |
  multiplier to error generation due to fatigue= WITH LOOKUP ( exhaustion level, ([(0,1)-
(100,2)],(0,1),(10,1.01),(20,1.01),(30,1.01),(40,1.02),(50,1.03),(100,1.05) ))
     ~        Dmnl
     ~               |
  schedule pressure= perceived shortage in man-days/(effort perceived remianing+0.001)
     ~        Dmnl
     ~               |
  indicated productivity= productivity*weight gain to real productivity+ potential
productivity*(1-weight gain to real productivity)
     ~        task/(Day*empl)
     ~               |
  time required= known work remaining/perceived productivity/desired WF
     ~        Day
     ~               |
  time to discover rework= WITH LOOKUP ( fraction perceived done, ([(0,0)-
(1,50)],(0,50),(0.1,50),(0.2,50),(0.3,40),(0.4,35),(0.5,30),(0.6,25),(0.7,20),(0.8,10),(0.9,5),(1,5)
))
     ~        Day
     ~               |
  labor resource deficit=  MAX(desired WF-total WF,0)
     ~        empl
     ~               |
  total WF= experienced workforce+ new workforce
     ~        empl
     ~               |
  total work to do= rework to do+  original work to do
     ~        task
     ~               |
  multiplier to error generation due to schedule pressure= WITH LOOKUP ( schedule
pressure,([(-0.4,0.9)-(1,2)],
(0.4,0.9),(-.2,0.94),(0,1),(0.2,1.01),(0.4,1.02),(0.6,1.06),(0.8,1.08),(1,1.1) ))
     ~        Dmnl
     ~               |

  multiplier to active error regeneration due to error density= WITH LOOKUP
(SMOOTH(active error fraction in task perceived done, 90),
[(0,0)m(1,10)],(0,1),(0.1,1.1),(0.2,1.2),(0.3,1.25),(0.4,1.45),(0.5,1.65),(0.6,1.95),
(0.7,2.5),(0.8,3.2),(0.9,4.1),(1,5.5) ))
    ~       Dmnl
    ~               |
  willingness to change WF= WITH LOOKUP (time remaining, ([(0,0)-
(2000,1)],(0,0),(90,0),(180,0),(270,0.1),(360,0.3),(450,0.7),(540,0.9),(630,1),(720,1),(810,1),(9
00,1),(990,1),(1080,1),(1170,1),(1260,1),(1350,1) ))
    ~       Dmnl
    ~               |
  willingness to overwork=
    IF THEN ELSE( Time>=time of last breakdown+ de-exhaustion time control,1,0)
    ~       Dmnl
    ~               |
  multiplier to error generation due to WF mix= WITH LOOKUP (fraction of experienced
workforce, ([(0,1)-(1,2)],(0,2),(0.2,1.8),(0.4,1.6),(0.6,1.4),(0.8,1.2),(1,1) ))
    ~       Dmnl
    ~               |
  weight gain to real productivity= WITH LOOKUP (fraction perceived done, ([(0,0)-
(1,2)],(0,0),(0.2,0.1),(0.4,0.25),(0.6,0.5),(0.8,0.9),(1,1) ))
    ~       Dmnl
    ~               |

  potential productivity=fraction of experienced workforce*nominal potential productivity
of expert employee+(1-fraction of experienced workforce)*nominal potential productivity of
new employee
    ~       task/(empl*Day)
    ~               |
  time remaining=MAX(scheduled completion date-Time,0)
    ~       Day
    ~               |
  work perceived remaining= project size*(1-fraction perceived done)
    ~       task
    ~               |
  breakdown= IF THEN ELSE( overwork duration threshold=0,Time+TIME STEP,0 )
    ~       Day
    ~               |
  congestion and communication difficulties= multiplier to difficulties due to team
size*total WF*total WF
    ~       Dmnl
    ~               |
  desired WF= indicated WF*willingness to change WF+total WF*(1-willingness to change
WF)
    ~       empl
    ~               |
  effort perceived remianing= work perceived remaining/perceived productivity
    ~       Day*empl
    ~               |
  exhaustion increasing rate= WITH LOOKUP (((1-actual fraction of man day in a
project)/(1.2-nominal fraction of man-day on project+0.0001))/time to get exhaustion,([(-
0.5,0)-(2.5,2.5)],(-0.5,2.5),(-0.4,2.2),(-0.3,1.9),(-0.2,1.6),(-0.1,1.3),(0,1),
(0.1,0.9),(0.2,0.8),(0.3,0.7),(0.4,0.6),(0.5,0.5),(0.6,0.4),(0.7,0.3),(0.8,0.2),(0.9,0),(1,0) ))
    ~        1/Day
    ~              |
  expected completion delay=time required-time remaining
    ~        Day
    ~
    |
  fraction of experienced workforce=experienced workforce/total WF
    ~        Dmnl
    ~              |
  fraction satisfactory=1-error fraction
    ~        Dmnl
    ~              |
  handled man-days= MIN( max shortage in man days to be handle ,perceived shortage in
man-days)
    ~        Day*empl
    ~              |
  indicated deadline=
    Time+ time required
    ~        Day
    ~              |
  indicated WF=
    effort perceived remaining/time remaining
    ~        empl
    ~              |
```

Constant variables

```
  project size= 1000
    ~        task
    ~              |
  average assimilation delay= 20
    ~        Day
    ~              |
  hiring adjustment time= 60
    ~        Day
    ~              |
  nominal fraction of man=day on project= 1
    ~        Dmnl
    ~              |
  nominal overwork duration threshold= 50
    ~        Day
    ~              |
  nominal potential productivity of expert employee= 1
    ~        task/(Day*empl)
    ~                   |
  nominal potential productivity of new employee= 0.5
    ~         task/(Day*empl)
    ~                   |
  average time to perceive productivity= 30
    ~         Day
    ~                   |
  max boost in man days hours= 1
    ~         Dmnl
    ~                   |
  multiplier to difficulties due to team size= 0.0001
    ~         Dmnl
    ~                   |
  experienced employees turnover fraction= 1e-05
    ~         1/Day
    ~                   |
  schedule adjustment time= 20
    ~         Day
    ~                   |
  estimated completion time=200
    ~         Day
    ~                   |

  max tolerable exhaustion= 50
    ~         Dmnl
    ~                   |
  newly employees turnover fraction= 0.0002
    ~         1/Day
    ~                   |
  time spend on depletion=20
    ~         Day
    ~                   |
  time to get exhaustion=1
    ~         Day
    ~                   |
  work adjustment time=3
    ~         Day
    ~                   |


  ********************************************************
    .Control
  ********************************************************
              Simulation Control Parameters
    |
  FINAL TIME  = 400
    ~         Day
    ~         The final time for the simulation.
    |
  INITIAL TIME  = 0
     ~         Day
     ~         The initial time for the simulation.
     |
  SAVEPER  =
       TIME STEP
     ~         Day
     ~         The frequency with which output is stored.
     |
  TIME STEP  = 0.125
     ~         Day
     ~         The time step for the simulation.
     |
```