A Generic Model of Project Management with Dynaplan Smia

Abstract

Projects are hard to be managed due to the reason of complexity, non -linear relationship and high dynamic. System dynamics has been proven a powerful and efficient way to manage projects. In this research, a generic system dynamics model applying to project management has been made. Through studying the model, the system behaviours have been analyzed and policies to deal with the problems of project management have been established. In addition, an object-oriented feature has been added into the model. The attempt is new to system dynamics modelling, and the outcome is satisfying since it makes the diagram clearer and better to understand.

Key word

Project management, system dynamics, object -oriented

Table of contents

1 Introduction

Projects abound in all aspects of the social infrastructures, such as public service, finance, industry and etc. As a series of task or activities, projects are proven hard to be managed, since they have a specific scope with certain requirements to be fulfilled; they have specific start and completion dates; the cost of projects is limited; there are not infinite resources to be consumed or utilized. All the aspects make project management challenging. Project management suffers from cost overruns, schedule slippage, hiring, costly rework and etc. Cost overruns of 100 to 200% are common. Projects are often delayed by the conditions of the market since they are changed quickly from where they were designed earlier. For example, software development often suffers from Brooks' Law, which states "adding resources to a late project makes it even later". [Sterman, 1992] Customers change their designs frequently, generating costly ripple effects which create delay and disruption throughout the whole project. Project turns to go smoothly at the end, after experiencing costly rework, delay and disruption, overtime, hiring, productivity delay, schedule slippage. All the negativities make the project low profitability, loss of market and reputation, increase the turnover of management and workforce, and often cause the contractor to sue the customers over responsibility for overruns and delays.

The hardness of managing projects, especially large scale projects, are largely because project conditions and performance evolve extremely complex, consisting of multiple interdependent components; they are highly dynamic and many evolving nonlinear relationships; they go overtime as a result of multiple feedback responses. System dynamic is the origin of the current trend of 'whole systems thinking'. It deals with complex interdependent components and enables the strategic modelling of nonlinear relationships. It is a risk-free way of refining plans before implementation using computer simulation and can facilitate ideas for both specific solutions and generic rules. These advantages make system dynamics a greatly suitable way to model project management problems.

This paper introduces a generic model to single project management by system dynamics. The reason of focusing on single project management in this paper is because that using system dynamics to model multiple projects is beyond the scope of this paper. But even models of single projects vary widely depending on their structures or applications in detail. Therefore

in this paper, only most important and general model structures in conceptual form are focused, and references to additional details are provided.

The generic project management model is built up based on the rework cycle , which is the one of the most canonical structures for project models. The rework cycle means that, in a project model, most of the original tasks are done correctly and n eed not to rework, while a fraction of tasks being done with errors needs to rework and the rework will generate more rework. The rework cycle causes performance gap, such as delay on deadline and deficit of labour resource. So the first step we took in the model is controlling feedbacks, including slipping the deadline and adding resources. However, these actions cause unintended side effects that generate policy resistance. For example, adding new people to a project dilutes experience as workers with less skill or less familiarity with the project are brought on. There will be a delay on productivity between new workers and experienced workers. These components make up of the generic model in this paper. Further, knock-on effects generated by the ripple effects, such as fatigue caused by working overtime, will be added into the model.

Dynaplan smia is the modelling tool that is used in this project. It contains an object-oriented feature which is different from the traditional features of system dynamic tools, such as Vensim and Powersim studio. Since the traditional way is to map all the problems out in single instance, adding object-oriented features to a system dynamic model becomes a new attempt on system dynamic modelling in this paper. More details are specified in the following chapters.

1.1 Outline of the paper

The chapter 2 introduces the methodology of system dynamics modelling. Chapter 3 describes the problem area and relevant literatures. In chapter 4, the processes and approaches to build the project management model have been documented. Chapter 5 is the verification and validation of the model. Chapter 6 discusses with the policies that applying to the problem. Chapter 7 summarize the work that is done in this paper. The last two parts are reference and appendix.

2 Background

System dynamics has repeatedly been proved an effective analytical tool in a wide variety of situations, both academic and practical, and is being used broadly by numbers of corporations all over the world. In this chapter, the basic concept of system dynamics modelling is introduced. Then the problem area of this paper and the sub-problems are also discussed in this chapter.

2.1 What is system dynamics?

System dynamics is the method of bringing whole system thinking to life in a rigorous, testable way. The purpose of system dynamics is to help people make better decisions when confronted with complex, dynamic systems. The field provides a philosophy and tools to model and analyze dynamic systems. Its basis is learning to see the patterns of behaviour in organizations and their operational processes and policies. The modelling language is intuitive. It uses software to map processes and policies at a strategic level, populate the map with data and simulate the evolution of the processes under transpare nt assumptions, policies and scenarios.

2.2 Modeling tools of system dynamics

System dynamics is a strategic rather than an operational tool. It can be used to integrate processes and policies across organizations where feedback is important, and analyze the variations after integration is done. The significant elements of system dynamics are shown below.

2.2.1 Feedback loops

Feedback refers to the situation of X affecting Y (figure 1.) and Y in turn affecting X perhaps through a chain of causes and effec ts, for example C between X and Y. We can not study the link between X and Y or, independently, the link between Y and X and predict how the system will behave. Only the study of the whole system as a feedback system will lead us to correct results.

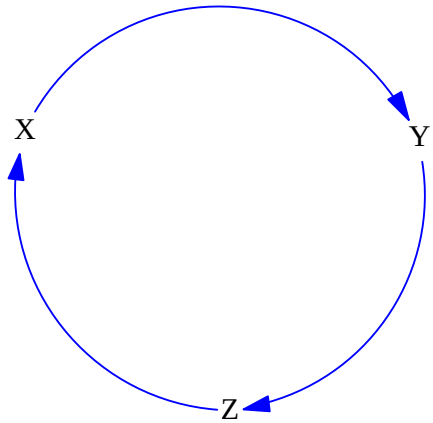

Figure 1

2.2.2 Causal-loop diagrams

Causal-loop diagrams identify the principal feedback loops without distinguishing the nature of the interdependent variables, which means levels, rates, auxiliaries and constants in am organization are all treated the same [Sterman, 2000]. Causal-loop diagrams serve as preliminary sketches of causal hypotheses when we build up a system dynamics mode. It makes the illustration of a model simple for both the modellers and observers, allowing them to quickly communicate with the structural assumptions underlying the model.

Feedback loops represent in two ways in the causal -loop diagramming---reinforcing loops and balancing loops. The chance that two types of feedback appear in causal -loop diagram varies in different system. It depends on the real situation that a case encounters. However, the system behaviour can be predicted once the causal relationships of the case are given. The research indicates that all system behaviour involving feedback is made up of four behaviour patterns. The examples for the behaviour patterns are as follows.

Reinforcing feedback

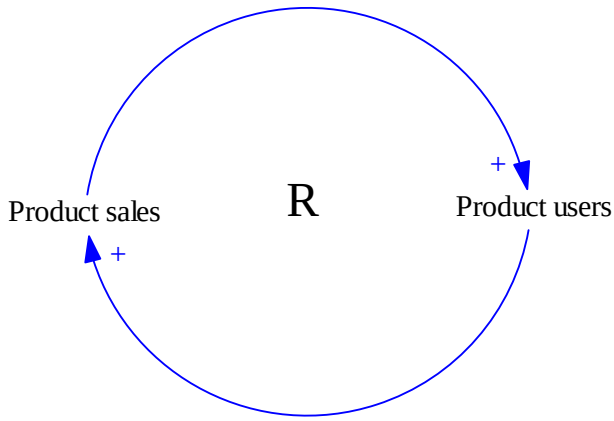

Figure 2-Reinforcing feedback

The figure.2 shows a reinforcement relationship in a simple reality system. 'Product sales' represents the scale that a product has bee n sold. It has a positive influence on 'Product users' which means an increase on 'Product sales' results in an increase on 'Product users'. 'Product users' shows the number of the people who are using the product. It has a positive influence on 'Product sales' in the opposite direction. These make up of a reinforcing feedback loop. The single reinforcing feedback results in exponential growth behaviour of the system (figure 3).

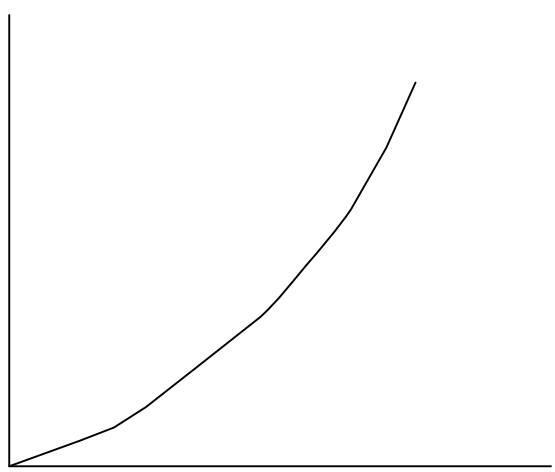

Figure3- Exponential Growth

Balancing feedback

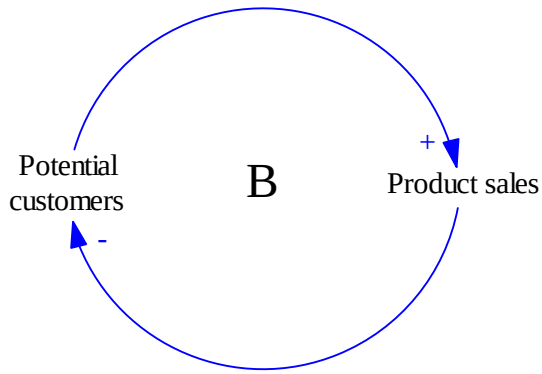

Figure 4-Balancing feedback

The figure 4 shows a balancing feedback loop that consists of 'Potential customers' and 'Product sales'. The increase of Potential customers results in an increase in 'Product sales'. While the increase of 'Product sales' indicates that a fraction of 'Potential customers' becomes customers, the number of 'Potential customers' actually decreases. The balancing feedback results in a Goal-seeking behaviour (figure 5).

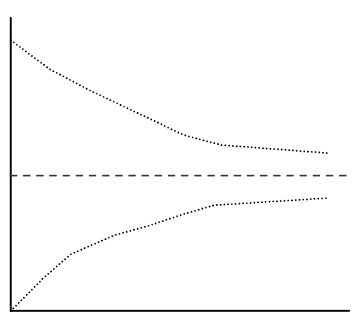

Figure 5-Goal seeking

Reinforcing and balancing feedback

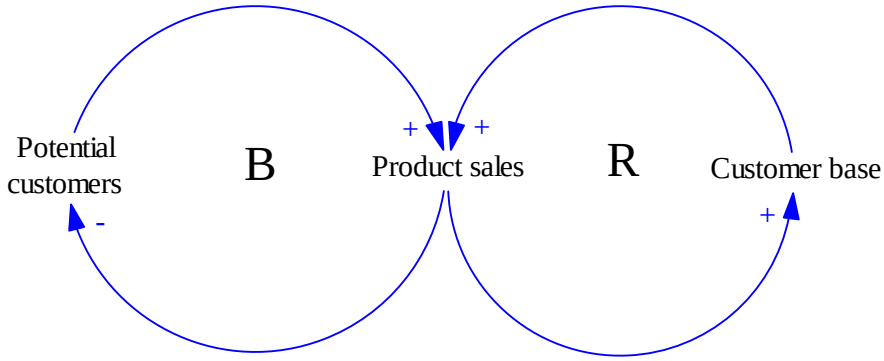

Figure 6-Reinforcing and balancing feedback

Figure 6 shows a combination of reinforcing and balancing feedback loops. The loop in the left is a balancing feedback loop as same as the one figure 3 shows, while the right part is a reinforcing feedback loop. Each of the 'Product sales' and 'Customer base' has a positive influence on the others. The combination of reinforcing and balancing feedbacks results in an S-shaped behaviour of the system (figure 7).

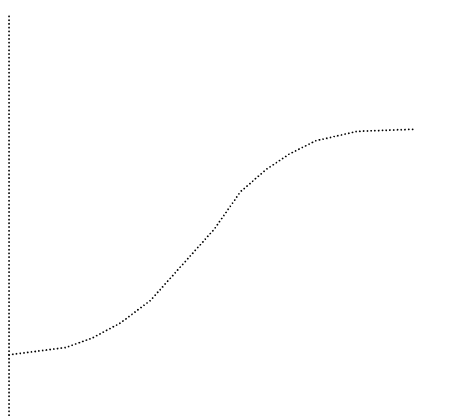

Figure 7 S-shaped

The fourth type of behaviour is oscillations (figure 8). It often happens in the complex dynamic systems that have numbers of reinforcing and balancing feedback loops. Here we are do not go into detail about the models that cause oscillations behaviour due to the limit scope of the thesis.

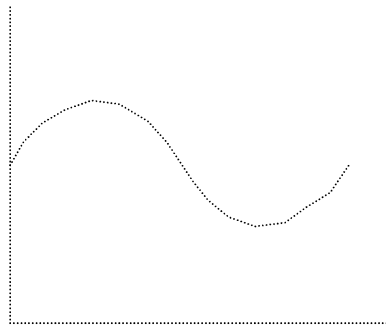

Figure 8-Oscillations

2.2.3 System archetypes

System archetypes are the classification of structures that respond for generic patterns of counter-intuitive behaviour over time. These generic structures represent the actions and unintended system reactions, and recognize delays in reaction times. Archetypes are used to generate understanding in new application domains, which makes itself a very powerful tool for understanding dynamic complexity. A lot of system archetypes were proposed in the past. Peter M. Senge has identified numbers of archetypes such as 'limits to growth', 'shifting the burden', 'eroding goals', 'escalation-arm race', 'success to the successful', 'fixes that fail', 'tragedy of the commons' and 'growth and underinvestment' [Senge, 1990]

The recent research by Eric Wolstenholme suggests that all archetypes can be expressed as one of a total of four types consisting of two feedback loops which are r einforcing and balancing feedbacks [Wolstenholme, 2003]. In addition, each of the problem archetypes has a solution archetype.
The four archetypes are as follow:

Underachievement

The underachievement archetype consists of an intended outcome which is a reinforcing loop and an unintended system reaction which is a balancing loop.

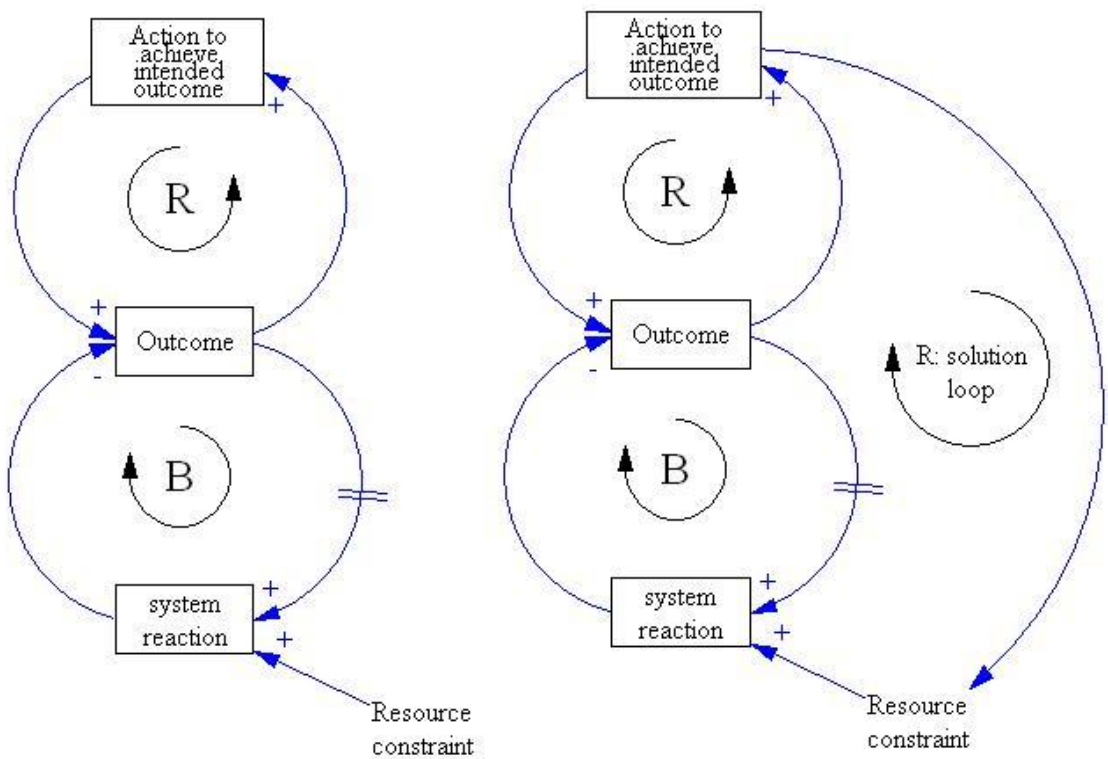

Figure 9 Underachievement problem and solution archetype

In figure 9, to the left is the underachievement problem archetype, it contains a reinforcing loop to get the intended outcome. However it produces an unintended consequence loop which compromises the outcome. And it is often caused by the constraint of the resource. The solution archetype for underachievement is in figure 9 to the right. It adds a solution link which unblocks the resource constraint in order to minimize the negative system reaction.

Out of control

The out of control archetype consists of an intended outcome which is a balancing loop and an untended system reaction which is a balancing loop.

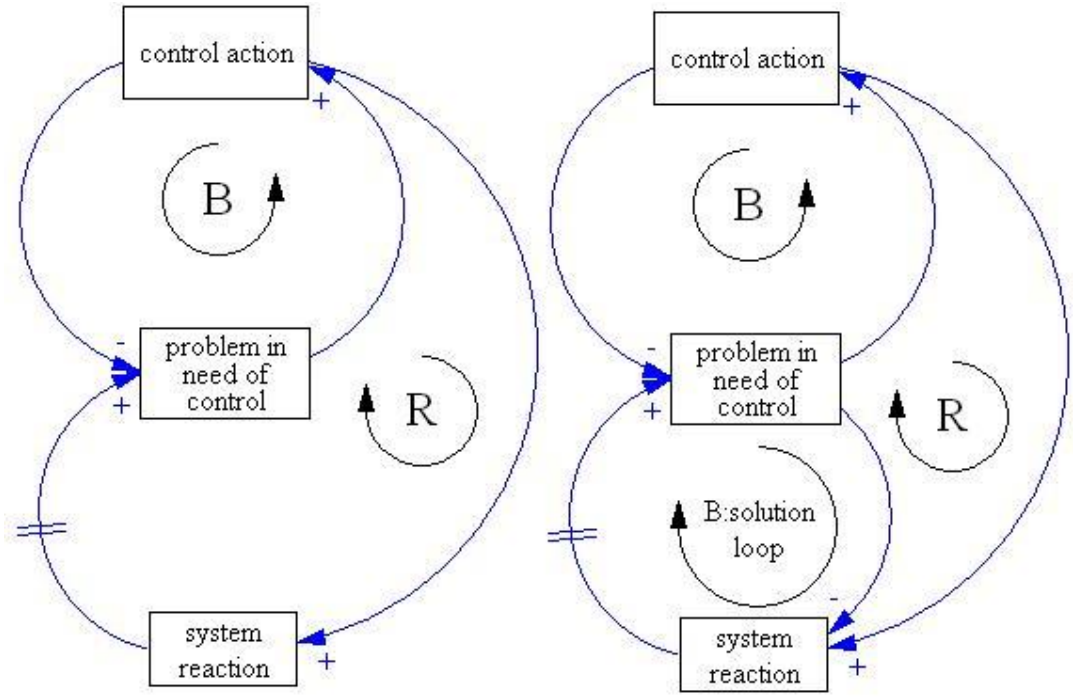

Figure 10 out of control problem and solution archetype

Figure 10 shows the problem and solution archetype refer to out of control problem. The problem archetype consists of a balancing loop which intends to control the problem, while it also generates an unintended system reaction that reinforces the problem. The solution to this type of problem is to set up a direct link between the problem and the system reaction.

Relative achievement

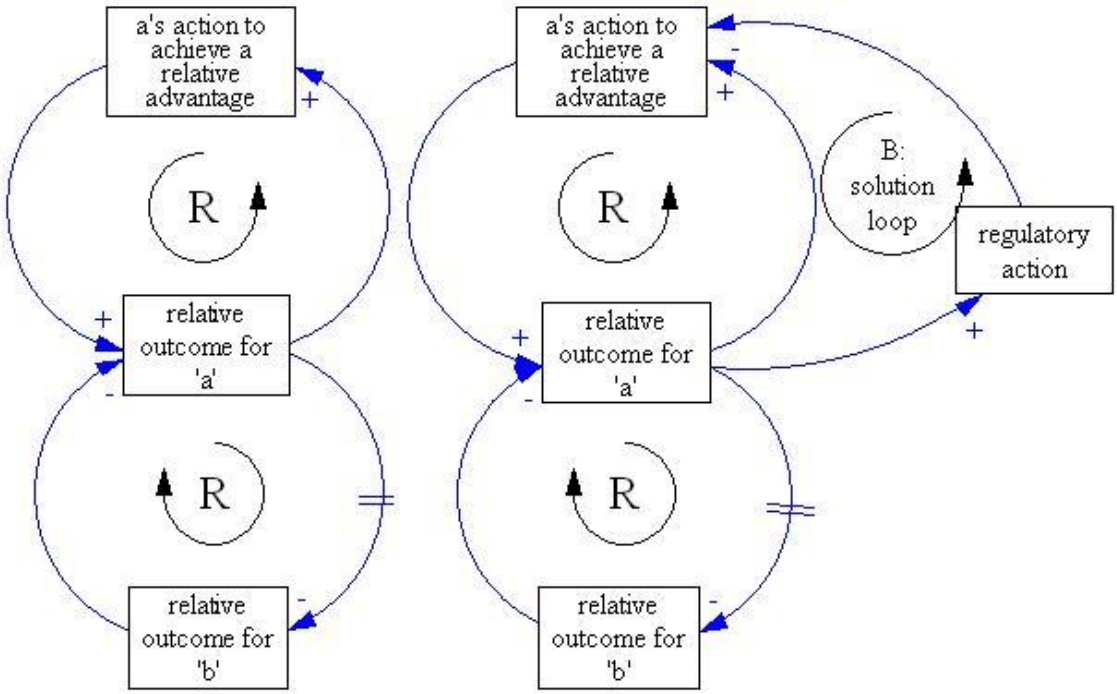

Figure 11 relative achievement problem and solution arch etype

The relative achievement contains two reinforcing loops. The intended loop is to achieve an improvement for a. The unintended loop magnifies the outcome of 'a', but it does so at the expense of 'b'. So the net result is a zero-sum game. The solution to this type of problem is to add a solution loop to the model, which introduces a regulatory action in order to balance the action of 'a' and leads it to a new state.

Relative control

The action of 'a' is to achieve a relative outcome. However the relative outcome induces an unintended reaction of 'a' which compromises the relative outcome. The solution to the problem is to define an absolute target and a new balancing loop to stabilize the outcome.

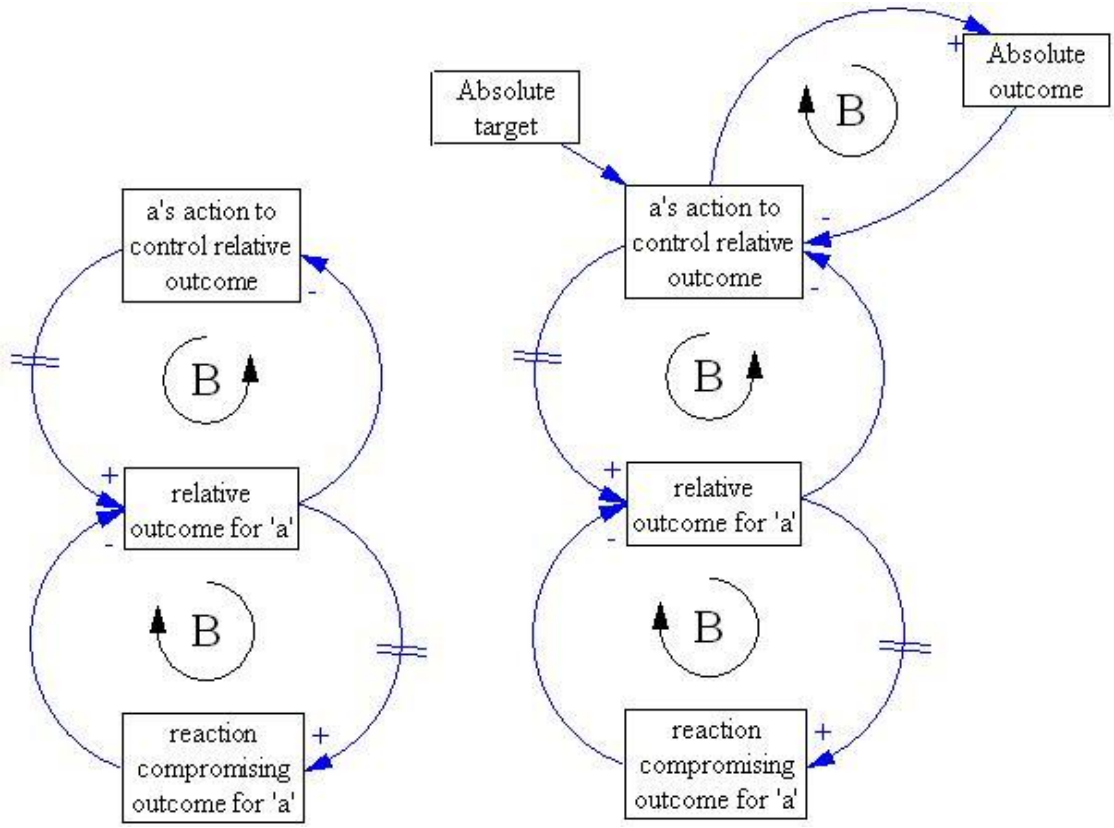

Figure 12 Relative control problem and solution archet ype

System archetypes are used to help people identify the complex problems in complex systems and establish the solutions in a fast and clear way. When dealing with a complex system dynamic model, we need to firstly identify the outcome that we intend to achieve and what kind of feedback loop the intended consequence loop is , then find out the unintended system reaction, knowing the types of the unintended feedback loops. After figuring out these, we could identify what kind of archetype the problem is and carry out the suitable solution for the problem precisely and quickly.

2.2.4 Stocks and flows

A model with causal-loop diagram can not be simulated by the computers. It only helps people to know the system better. To have a system dynamics quantitative si mulation, Stocks and flows are required. Stocks and flows provide the fundament for quan titative simulation [Sterman, 2000]. Stocks (also called levels) are used in system dynamics for representing states, which are measurable quantities of any resource in the system at any point of time [Gonzalez]. The resource can be any quantity of interest in a managed system, such as

material, goods, time, money, people or even knowledge. The basic process in natural or managed system is that of converting resources between states.

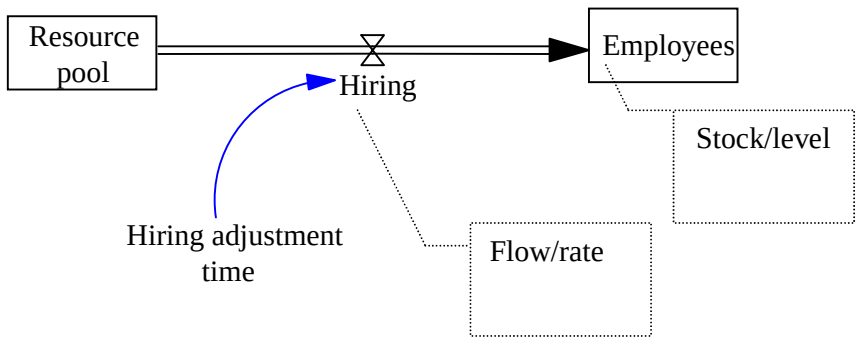

Figure 13-stock and flows

The example in figure 13 is a simple stock and flows diagram. 'Resource pool' and 'Employees' are the resources and are also two stock variables. 'Resource pool' keeps converting into 'Employees' over time. The rate by which resources are converted between states is represented by the flow variable---'Hiring'.

To create a stock-and-flow diagram, we need firstly to identify the resources that exist in the problem. Then we need to figure out what states the resources are. Considering all possible conversions is the last step of making a stock-and-flow diagram and linking them together with the right causal relationships.

2.3 How to solve the problems in real world

Complexity and high dynamics make the problems in real world hard to solve. Having general knowledge about system dynamic modelling is not enough. It requires several steps to address the problems.

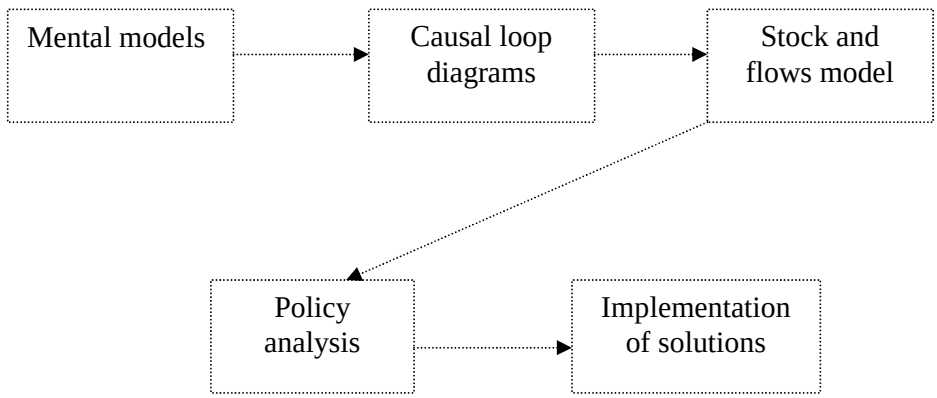

Figure 14-modeling problems in real world

Figure 14 shows the primary process of solving a real world problem. To start, we need to identify the problem, figuring out causal relations that involved in the problem and building up a mental model in mind. As soon as we have the mental model in mind, we develop a dynamic hypothesis explaining the cause of the problem. The approach to develop the hypothesis is to create a causal-loop diagram which identifies all the factors in the problem, mapping out their causal relationships. Creating stock and flows diagram tha t can be simulated by computer is the next step. During making stock and flows diagram, we need to carefully identify the roles that each variable plays in the model. To distinguish between the stock and flow variables is fundamental. The purpose of building stock and flows diagram is to test the model and observe the behaviour of the problem. Alternative policies will be devised and tested after knowing the system behaviour, in order to alleviate the problem. The final step is to implement the solutions that have been proven successful in the previous tests.

2.4 Dynaplan smia

Among the software tools for system dynamics, Dynaplan smia is relatively new comparing to Powersim studio, Vensim and Stella. Dynaplan smia is easy to use, and at same time it is a powerful tool, including object-oriented features, dynamic simulation, risk and uncertainty analysis,, optimization, and sensitivity analysis. The main difference from the traditional system dynamics tools is that Smia offers a n object-oriented feature that allows people to

model a system dynamics problem in an object-oriented way, which is also relatively new in system dynamics modelling.

3 Problem description and relevant literature

Most large projects are mismanaged: Cost and deadline overruns, missing the project scope and insufficient quality are typical in all kind of projects – whether software development, product development, plants, buildings, etc. Costs of offshore projects can be several billion NOK more expensive than budgeted and delays of one or more years are typical. Some projects are never finished, typically software projects.

System dynamics has proven to be an effective methodology to explain the reasons of project failure and to provide insights for best practice in project management [Lyneis, *2007*]. In more than 100 projects where system dynamics has been applied, the cost-benefit ratio is 1:200. For litigations cases, where the contractor has sued the contracting company for having caused project costs and deadline overrun owing to seemingly "minor" changes in specification, system dynamics models have proven extremely useful. In USA and UK, where more than 50 such cases have dealt with in court, parties using system dynamics models to prove their case have always won and they have on the average recovered 50% more than comparable court cases using other methodologies.

Accordingly, one would expect that system dynamics model are commonly used to improve project management. However, this is not the case. The reason is that system dynamics modelling is still a rare ability – there are far more cases where system dynamics could and should be applied than there are modellers available.

There is a strong need for a generic system dynamics model for raising awareness. In this chapter, the problems within building a generic system dynamics model are described, as well as relevant literature that has been researched.

3.1 Sub problems

The sub problems indicate the problems encountered during modeling. To solve these problems is to move stably towards our original target. The problems are carried out based on the understanding of project management and assumptions when modeling the problems. For each sub problems we have a solution, however, which is concep tual idea that needs to be practiced.

3.1.1 How to build a model that can be simulated based on the causal loop diagrams

Problem: A system dynamic model is built based on the causal loop diagrams, though a well structured and functional model is much more compli cated that the original causal loop diagrams. There are much more factors that affect the system dynamic procedures in reality and may not be included in causal loop diagrams, and each of them functions as a part of the system dynamic model. Our task here is to find out as many as possible the variables in project management in order to build a model of project management that can be simulated well.

Solution: It is generally about to find as many as the factors that could function in the project management problem, and to include them in the system dynamic model. Further, these is a strong need to find out what type of the factors are in the model, for example stocks, flows or auxiliary variables, and at which state the variable is. After identifying all the variables and their corresponding states, we build up the stock and flows diagram.

3.1.2 How to decide initial values and equations to the variables

Problem: The values and equations corresponding to each variable in the model are critical for simulating the model. Improper values or equations result in a fail outcome of simulation. However, the task to find out the suitable values and equations is challenging.

Solution: The model is supposed to be simulated after he whole structure is done. So the initial values and equations referring to the corresponding variables are required . There is a need to go deeply into this generic model to analyze and consider the right relations between those variables in order to get the right equation. For the initial valu es, besides some real clear

ones, some assumptions can be used here and they will be adjusted while the whole structure is being test until we got right values which fit the model . Since it is a generic model, it is necessary to make it as simple as possible and also make it be widely acceptable in much industrial sectors.

3.1.3 How to add object-oriented features to the model

Problem: Object-oriented function is provided in the new tool Dynaplan smia. It would be interesting to add object-oriented feature in the system dynamic model, since a system dynamic process is more like a process-oriented. The problem is to find where and how the object-oriented feature can be added into the model and works well in the model as a whole.

Solution: Object-oriented programming is largely for software development. However, it still contains some of the features in the programming process that are in common with system dynamics modeling. For example, the class diagram in object -oriented programming is similar with causal loop diagram in system dynamics modeling. To add object-oriented features to system dynamics model, we need to identify the properties of the variables, figuring out which of them can be packed up and placed in the same instance. And it requires them to function as one class when communicating with exterior environment.

3.1.4 How to carry out the policy analysis

Problem: For almost every system dynamics model, policy analysis is a must since it provides the solution to the problem. After studying the syst em behaviors by the result of the simulation, we need to adjust current policies of the system or apply new policies to the system. The problem is what policies can deal with the problem effectively, and how it could be implemented to the system.

Solution: Having an in-depth understanding of the system behaviour is a must for making policy analysis. It requires us to know precisely which variable of the model influences the intended outcome most, or which variable of the model cause the unintended reaction . As

soon as we master the variables or consequence loops, policy analysis can be implemented smoothly

3.2 Literature research

3.2.1 Rework cycle

Project is consisted a serial of tasks that to be done over time. The traditional way of understanding how a project is done lies in a flow that represents the rate of work done over time, which goes out from the pool of 'work to be done' and goes into the pool of 'work done' (figure15).

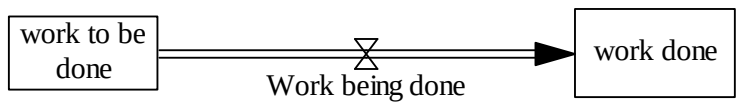

Figure 15 flow of work

And the rate of work being done is controlled by the people and productivity of the people.

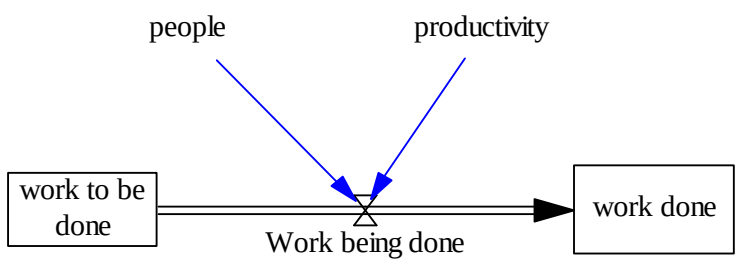

Figure 16 people and productivity affects the work flow

This is the traditional thinking about how a project is progressing. While the projects in real world are more complicated and may not progress as above. Kenneth G. Cooper suggested a rework cycle (figure 17) that applies to generic project management in his thesis [Cooper, 1993]

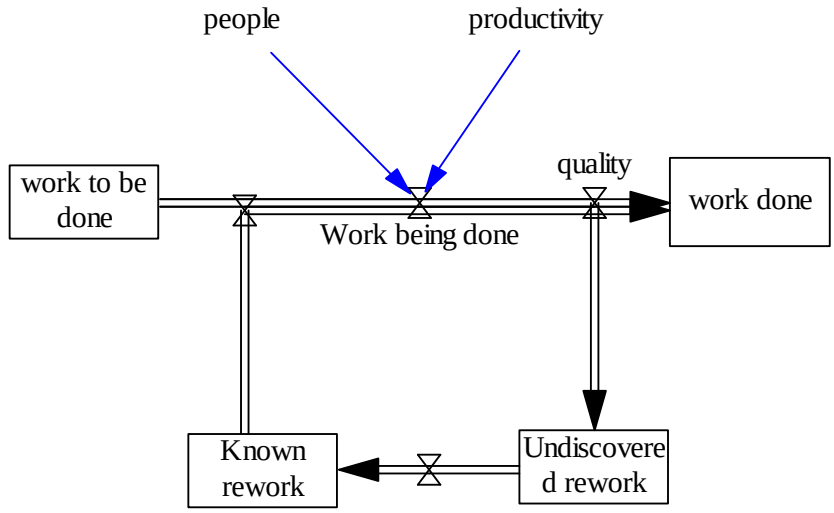

Figure17 the rework cycle

Figure 13 shows the structure of rework cycle. There are 3 new variables which are flow 'quality', stock 'undiscovered rework' and 'known rework'. The 'quality' indicates the quality of the work. Those Qualified works will go into 'work done' while those unqualified work will go into 'Undiscovered rework' and wait to be discovered. And the works in 'known rework' need to be done again. The 'quality' is measured In between the range of 0 to 1.0 diverts more or less of the work been done into the rework cycle. As long as the measure of 'quality' is less than 1.0, some work that has been done, even rework its elf, will continue to move into and through rework cycle.

3.2.2 A causal loop diagram applied to project management

Lyneis and Ford have suggested a relatively complete causal loop model for project management (figure 18). It contains the generic project features, a rework cycle, ripple effects and knock-on effects. The principal feature of system dynamic project model is the representation of development tasks or work packages as they flow through a project. The ripple effect is caused by rework cycle, wh ich is also the unintended reaction of the model. The consequence of the rework cycle often shows like schedule slipping, hiring more workforce and so on. Knock-on effect is often generated by ripple effect. Since the schedule is slipped due to the increasing amount of reworks, it generates schedule pressure and force people the work over time. Working over time causes fatigue and reduce the actual effort that people applied to the tasks. Hiring more people requires experienced people to train the new

people instead of doing the original works, causing delays to the productivity and in long run, will hurt the morale of the whole workforce, causing more delays and diluting the productivity.

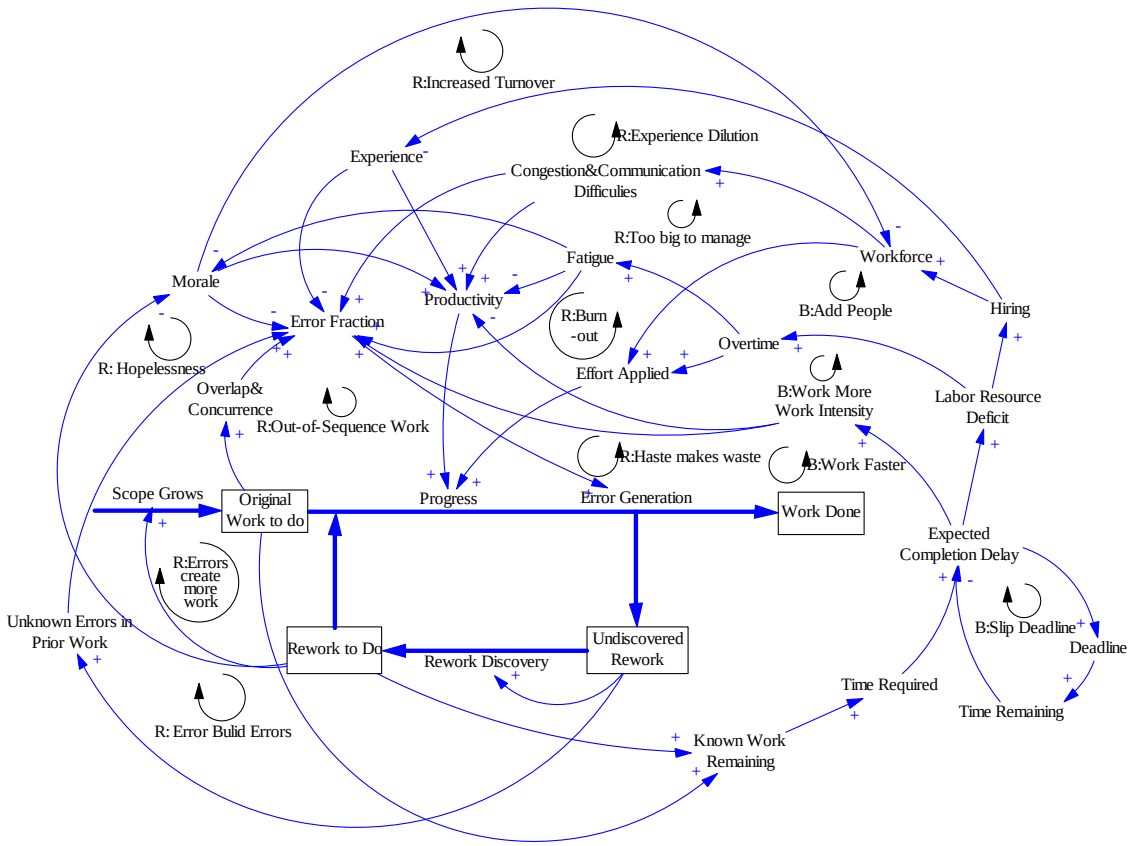

Figure 18 a generic causal loop model for project management

4 Modeling the problem

In this chapter, the approach to create the system dynamic model is illustrated. It follows a certain order just like solving a project management problem in reality.

4.1 Basic model

The basic model refers to the traditional way of modelling, which means to map all the factors out in a single instance, object-oriented features and the ripple effect---working overtime are not included in the basic model.

4.1.1 Task accomplishment

Every project starts from a number of tasks to be done, and ends up the tasks all get done. We start with two stocks and one flow between.

Figure 19 work flow

This is a completed model and is perhaps the simplest model that could be simulated. We set the initial value for 'original work to do' at 1200, the unit is task. At the start there is not work has been done, so the initial value for 'work done' is 0, the unit is task. The rate of 'progress' is decided by two factors (which will be introduced later) --- 'Perceived Progress' and 'Fraction Satisfactory'. Since we are building a model that can be simulated, we need to set up the start and stop time, and the time step.

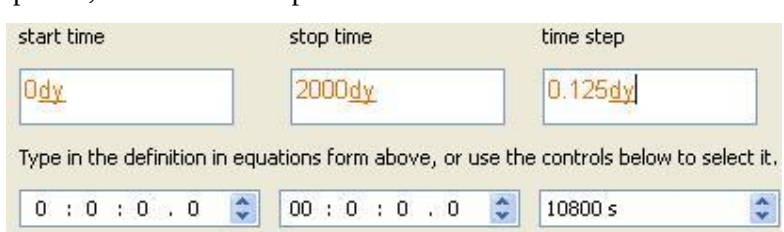

Figure 20 time condition for the model

As we can see in figure 20, we set the stop to 2000days, which does not mean the model will end simulation when it reaches 2000 days. While the model will end the simulation when

there is no more work or rework to do. The time step we set it at 0.625, which is 1/8 of a day, this should be enough to track down the changes over time. The equati ons are:

Original work to do = stock 1200tasks outflow progress
Unit: tasks
Work done = stock 0tasks inflow progress
Unit: tasks
Progress = flow 'Perceived Progress'*'Fraction Satisfactory' (1)

4.1.2 Rework cycle

Now we start to add the rework cycle to t he model. The rework cycle is caused by the quality of task done. Those unqualified tasks will be worked again. Here the variable to control the rework rate is called 'Fraction Satisfactory'. And there is a flow between 'undiscovered rework' and 'rework to do'; it is controlled by the variable 'time to discover rework'.

(1) In Dynaplan smia, we do not have to set up units for the variable that is not a stock or constant.

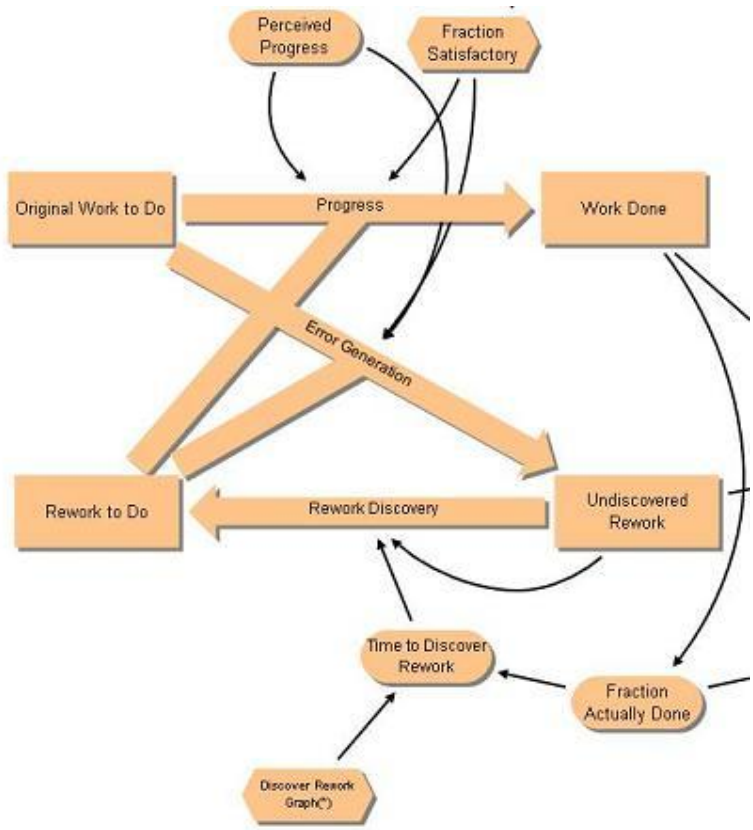

Figure 21 the rework cycle

The work that is done and flows out of 'Original work to do' is divided into two part; one goes into 'work done' through flow 'progress', and the other goes into 'undiscovered rework' through flow 'error generation'. Both of the flows are controlled by 'fraction satisfactory' and 'perceived progresses'. The initial values for 'rework to do' and 'undiscovered rework' are 0. The unit is tasks. To find the work needed to rework requires time. And the time for discovering the rework is not constant. The variable 'time to discover rework' has a look up function using 'fraction actually done' as factor. The chart for the look up function is as follows:

Discover Rework Graph	
0.00 %	300.00 dys
10.00 %	300.00 dys
20.00 %	300.00 dys
30.00 %	300.00 dys
40.00 %	285.00 dys
50.00 %	240.00 dys
60.00 %	105.00 dys
70.00 %	60.00 dys
80.00 %	45.00 dys
90.00 %	30.00 dys
100.00 %	30.00 dys

Table 1 Discover rework graph

The data in the table1 shows that at the start of the project, the time to be aware of the undiscovered rework is very slow. As the project close to complete, the discovering time is shorter and shorter. The equations for the variables are:

Original work to do= stock 1200tasks outflow Progress outflow 'Error Generation'

Unit: tasks

Work done= stock 0tasks inflow Progress

Unit: tasks

Undiscovered rework= stock 0tasks outflow 'Rework Discovery' inflow 'Error Generation'

Unit: tasks

Rework to do= stock 0tasks inflow 'Rework Discovery' outflow Progress outflow 'Error Generation'

Unit: tasks

Progress= flow 'Perceived Progress'*'Fraction Satisfactory'

Error generation= flow 'Perceived Progress'*(1-'Fraction Satisfactory')

Rework discovery= flow 'Undiscovered Rework'/'Time to Discover Rework'

Perceived Progress= Productivity*Q (1)

Fraction Satisfactory=0.7

Fraction Actually Done=’work done’/(1200tasks)

Time to Discover Rework= 'lookup linear' ('Fraction Actually Done', 'Discover Rework Graph')

Unit: days

(1)Q is the return value of effort applied,

4.1.3 Work perceived done and effort perceived remaining

Work perceived done does not equal to work actually done. Since there is a time delay for people to find out the 'undiscovered rework', people tends to believe they have done more than the actual outcome. 'Effort perceived remaining' is decided by the 'work perceived done' and 'perceived productivity', it equals to the perceived work remaining (1200tasks-'Work perceived done') divided by 'perceived productivity'. The result indicates that how many days it requires to finish the tasks due to the current workforce.

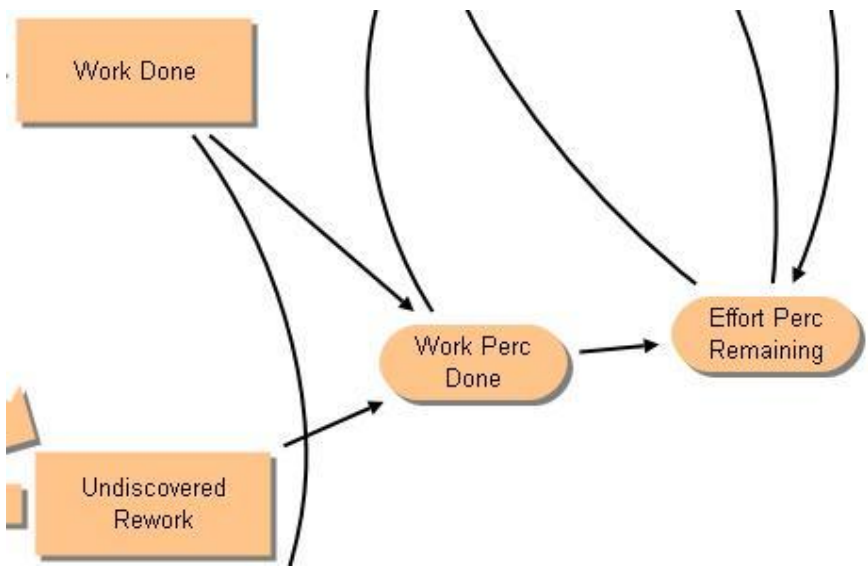

Figure 22 'Work Perc Done' and 'Effort Perc Remaining'

The equations for these two variables are:

Work Perc Done= 'Work Done'+ 'Undiscovered Rework'
Effort Perc Remaining= (1200tasks-'Work Perc Done')/'Perceived Productivity'

4.1.4 Effort remaining

In this section, the purpose is mainly to calculate the result of 'perceived productivity', 'Perceived productivity' is decided by the 'indicated productivity' and 'the average time to perceived productivity'. Since there is a time delay before people can perceive the productivity, a delay function called 'smooth' is used here to help calculating the 'perceived productivity'. The delay time is set to 180 days. 'Real productivity' is decided by 'productivity' and 'Fraction Satisfactory', the outcome of it is to decide the 'indicate

productivity'. The other variable which also decides 'indicate productivity' with 'real productivity' is 'weight given to real productivity'; it decides how much weight the 'real productivity' converts into 'indicate productivity'. 'Weight given to real productivity' uses a lookup function with the fraction of work perceived done as a factor. The equation for the fraction of work perceived done is 'Work Perc Done'/1200tasks. The table for the look up function is:

Weight Given Graph	
0.00 %	0.00
20.00 %	0.10
40.00 %	0.25
60.00 %	0.50
80.00 %	0.90
100.00 %	1.00

Table 2 the Weight given by 'real productivity'

As we can see from table 2, the closer the distance from completing the project is, the more weight of 'real productivity' converts into 'indicate productivity'. The causal relationship between these variables is as follows:

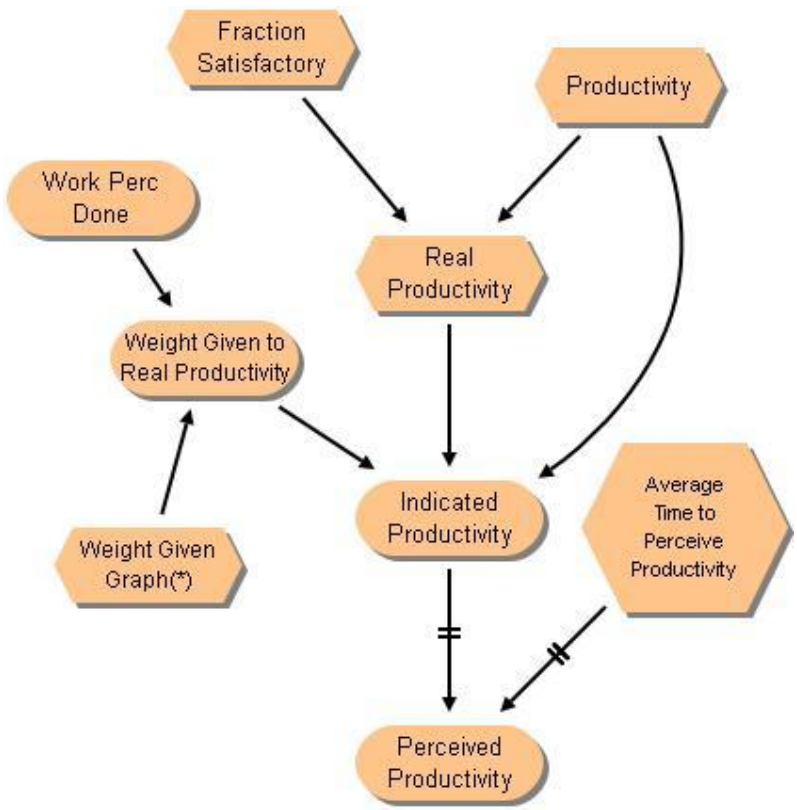

Figure 23 effort remaining

The equations for these variables are:

Real Productivity= Productivity*'Fraction Satisfactory'

Weight Given to Real Productivity= 'lookup linear' (('Work Perc Done'/1200tasks),'Weight Given Graph')

Indicated Productivity= 'Real Productivity' * 'Weight Given to Real Productivity'+ Productivity * (1-'Weight Given to Real Productivity')

Average Time to Perceive Productivity=180days

Perceived Productivity= smooth ('Indicated Productivity', 'Average Time to Perceive Productivity')

4.1.5 Time condition

In this section, we have modelled the slipping deadline problem and produce a critical variable that used to control workforce. 'Effort Perc Remaining' is here to determine the perceived required time together with 'desired workforce'. As soon as we have 'time prec remaining', we can not get the 'indicated deadline' by simply calculating the sum of current time and 'time prec remaining'. The 'indicated deadline' is used to adjust the 'deadline'. But it takes time to actually carry out. The time for adjusting deadline is set to 180days. Adjusting deadline is implemented by having a flow 'Deadline Adjustment' go into stock 'deadline'. The flow equals to ('Indicated Deadline'-Deadline) divided by 'deadline adj time'. The initial value of 'deadline' is set to 1200days. Finally, we have figured out the real time remaining by using 'deadline' minus current time.

The causal relationships are as follows:

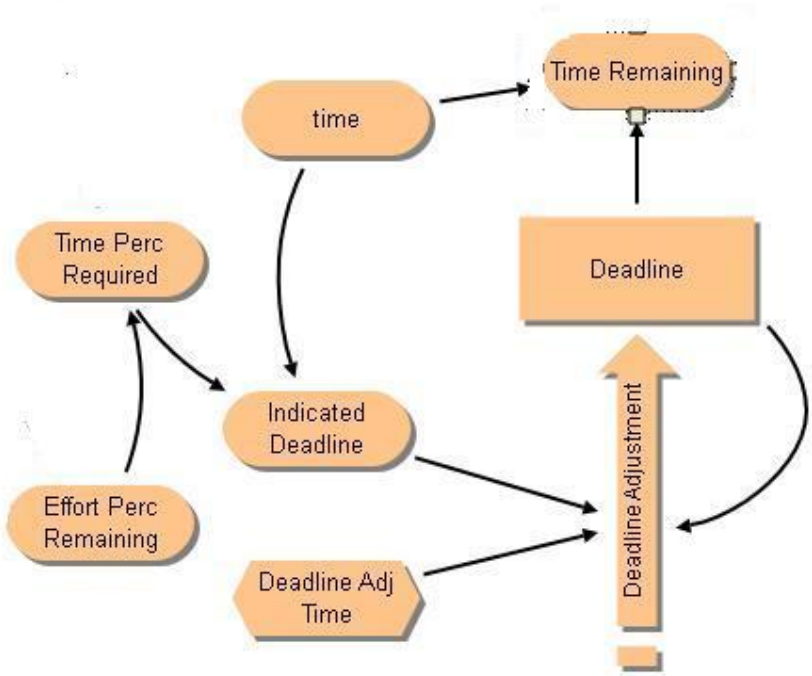

Figure 24 time condition

The equations for the variables are:

Deadline= stock 1200dys inflow 'Deadline Adjustment'

Unit: days

Deadline Adjustment= flow ('Indicated Deadline'-Deadline)/'Deadline Adj Time'

Time Remaining= Deadline-time

Time&t= time (horizon)

Time Perc Required= 'Effort Perc Remaining'/'Desired WF'

Indicated Deadline= time+'Time Perc Required'

Deadline Adj Time= 180dys

Unit: days

4.1.6 Desired workforce

The ‘desired workforce’ is needed in the model to calculate the deficit of the labor resource, and is controlled by the ‘indicated WF’ and ‘willingness to change WF’, the ‘indicated workforce’ is decided by ‘effort prec remaining’ and ‘time remaining’. The ‘willingness to change WF’ uses a look up function with the ‘time remaining’ as a factor. The table for the look up function is as follows:

Willingness Graph	
0.00 dys	0.00
90.00 dys	0.00
180.00 dys	0.00
270.00 dys	0.10
360.00 dys	0.30
450.00 dys	0.70
540.00 dys	0.90
630.00 dys	1.00
720.00 dys	1.00
810.00 dys	1.00
900.00 dys	1.00
990.00 dys	1.00
1,080.00 dys	1.00
1,170.00 dys	1.00
1,260.00 dys	1.00
1,350.00 dys	1.00

Table 3 willingness to change workforce

We learn from the table that: at start the willing to change workforce is high. As time goes by, the willingness falls down and the more close to complete the project, the less willingness to change workforce.

The causal relationship is:

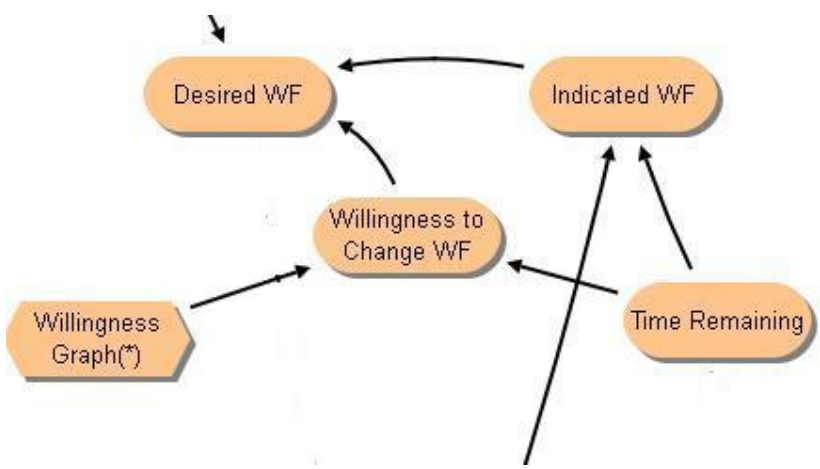

Figure 25 Desired workforce

The equations for the variables are:

Desired WF= 'Indicated WF'*'Willingness to Change WF'+ 'Work Force'*(1-'Willingness to Change WF')

Indicated WF= 'Effort Perc Remaining'/'Time Remaining'

Willingness to Change WF= 'lookup linear'('Time Remaining', 'Willingness Graph')

4.1.7 Effort applied

'Effort applied' is to determine the average effort of an employee applying to the work. It is set to define by the 'workforce' and 'normal effect'. Here we set the value of 'normal effect' to 100%, while it is not the case in reality; more details will be discussed in the following sections. The stock 'workforce' has an inflow 'hiring' goes into it. Normally it has a time delay for the new employees to function as the workforce. The delay time has been set to 90days. The deficit of labor resource is the causation to adjust workforce, and is decided by 'workforce' and 'desired workforce'.

The causal relationship is as follows:

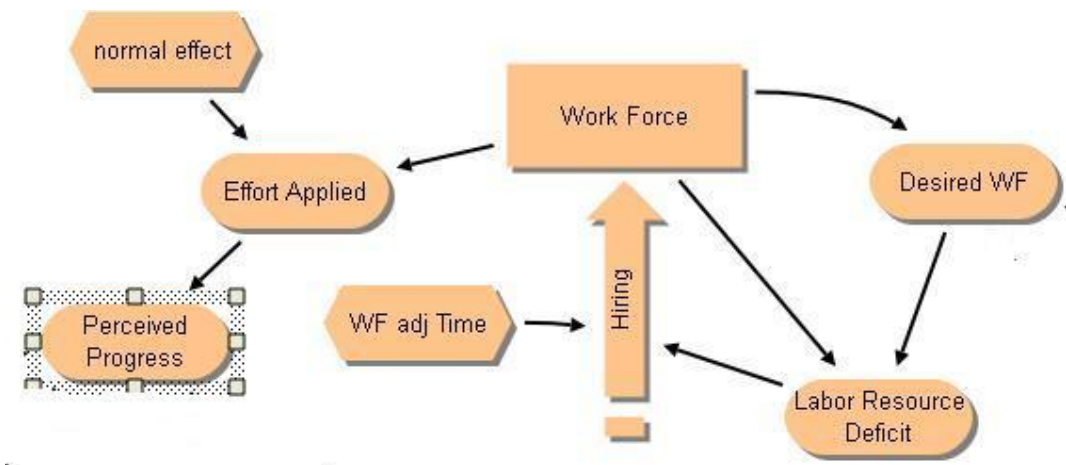

Figure 26 Effort applied

The equations are:

Work Force= stock 2 employees inflow Hiring

Hiring= flow 'Labor Resource Deficit'/'WF adj Time'

Labor Resource Deficit= 'Desired WF'-'Work Force'

WF adj Time= 90dys

Unit: day

Effort Applied= 'normal effect'*'Work Force'

Normal effect= 100%

Now all the variables have been linked up and the basic system dynami cs model for project management has been completed. The view of the whole model is as follows"

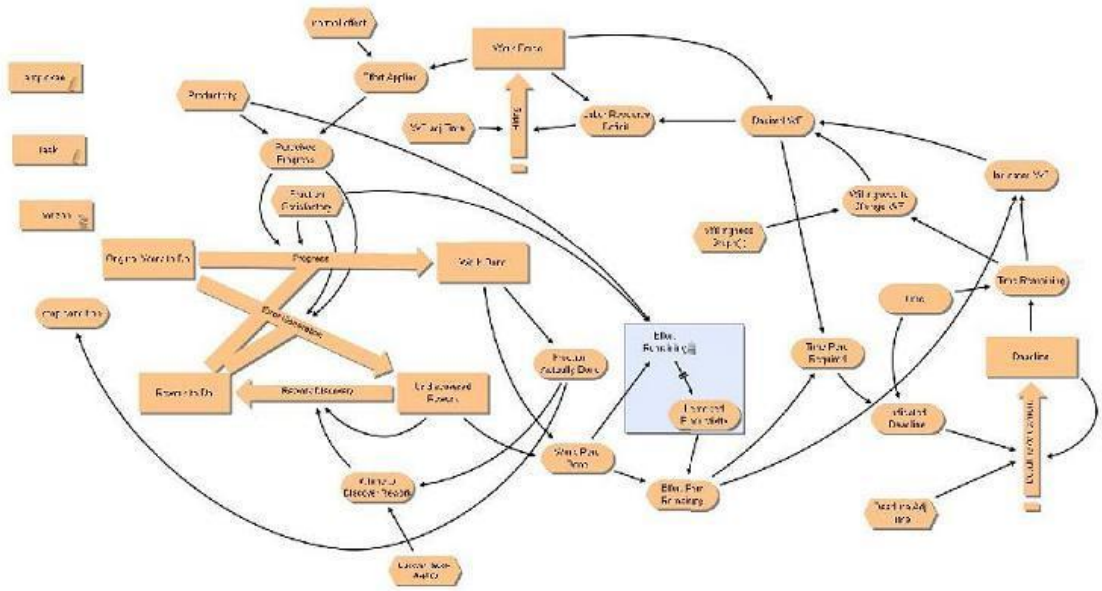

Figure 27 basic model for project management

4.2 Object-oriented modelling and ripple effect

In this section, the ripple effect generated by the system gap i s introduced to the model. It mainly deals with the working overtime problems, which generates unintended system reaction---fatigue. Employees get exhausted when they keep working overtime. And it reaches the threshold of overwork, their effort applying to the work quickly goes down, and the whole effort applied by the workforce goes down.

Object-oriented features are also added into the model. It is implemented by classifying the variables as objects, creating individual instance and put the objects into corresponding instances (classes), in order to achieve class diagrams and package diagrams.

4.2.1 Object-oriented modelling

Adding object-oriented features in Dynaplan Smia requires us to use the term —original to create a component. It can either be a cl ass or an instance. The second step is to classify the variables into the categories which variables have the similar attribute such as ‘indicated deadline’ and ‘deadline’, or else the variables have closed causal relationship such as ‘workforce’ and ‘effort applied’.

In this model, five classes (or instances) have been created. Each of them has a return value and several input values. The objects inside the instance function the same as they did in non - object oriented models. The differences are that there can be only one output value in an class (or instance) once we have established the return value for the instance. Four out of five classes (or instance) have been made to the surface of the application. They are class 'effort remaining', 'Time condition', 'Desired workforce' and 'Effort applied'. Each of them is made up of several objects that are causally linked.

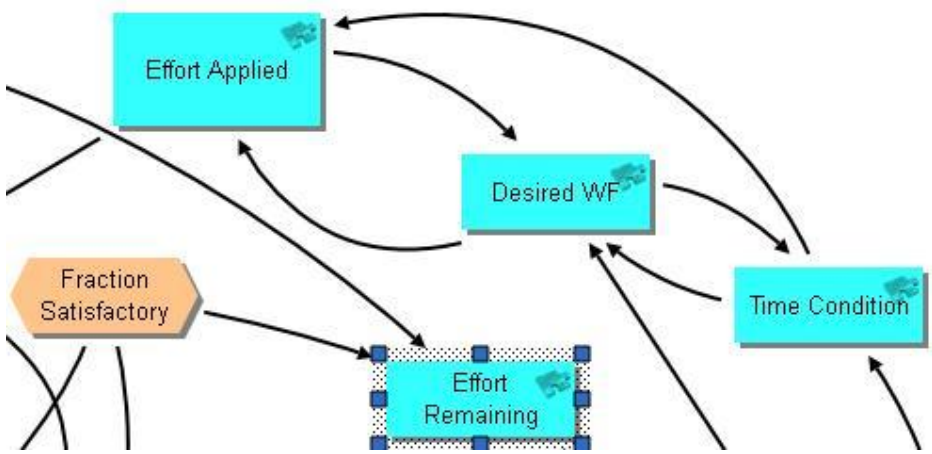

Figure 28 the four classes

Effort remaining

The equation for class 'effort remaining' is:

Effort remaining= Original

And the contents in class 'effort remaining' are:

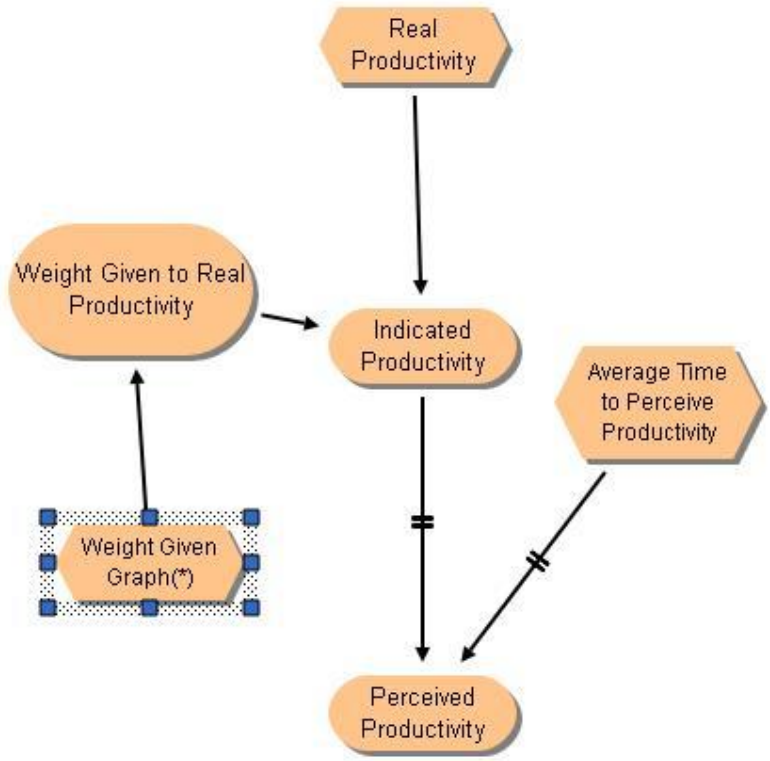

Figure 29 Class 'effort remaining'

Most of the variables are defined the same as they were in the basic model. Only difference is that 'perceived productivity' is defined as the return value of class 'effort remaining'. The equation for it is:

Perceived Productivity= return smooth ('Indicated Productivity', 'Average Time to Perceive Productivity')

Time condition

The equation for class 'time condition' is:

Time condition= original

The contents in 'time condition' are:

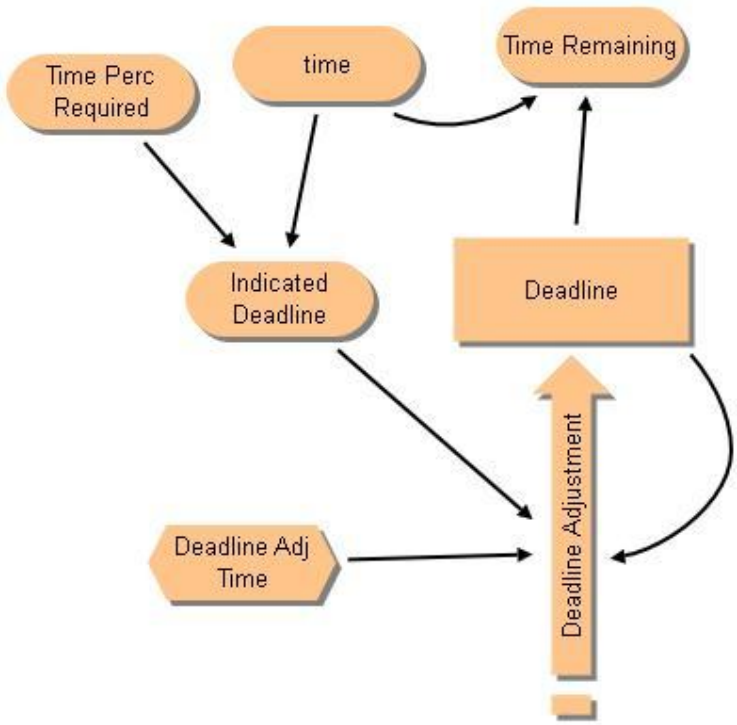

Figure 30 class 'Time condition'

The variable 'time remaining' is set to be the return value of the class. The equation for 'time remaining' is:

Time remaining= return Deadline-time

Desired workforce

The equation for class 'desired WF' is:

Desired WF= original

The contents in 'desired WF' are:

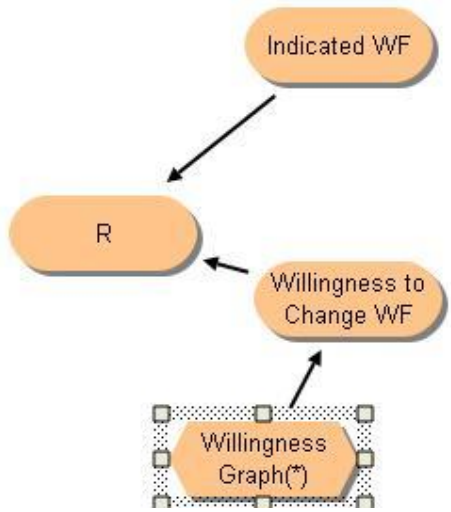

Figure 31 class 'desired WF'

The R variable is set to the return value of class 'desired WF'. The equation for R is:

R= return 'Indicated WF'*'Willingness to Change WF'+ 'Work Force'*(1-'Willingness to Change WF')

Effort applied

The equation for 'effort applied' is"

Effort applied= original

The contents in class 'effort applied' are

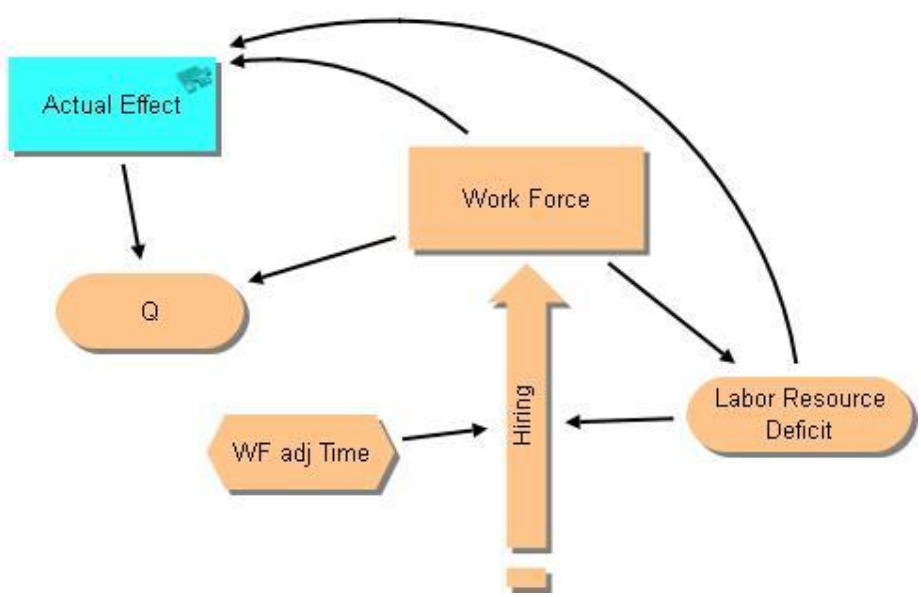

Figure 32 class 'Effort applied'

Q is the return value of class ‘effort applied’, the equation is:

Q= ‘Actual Effect’ * 'Work Force'

As we could see in figure 28, there is another class ‘Actual Effect’ here. Instead of ‘normal effect’ in basic model, ‘actual effect’ is here to calculate the return value of class ‘effort applied’. More details about ‘actual effect’ will be specified in the following section.

By adding five class components into the model, modelling system dynamics problems with object-oriented features has been achieved. It makes the map clear and helps the modeller to identify the causal relations better and faster. Here is the view of whole model after adding object-oriented features.

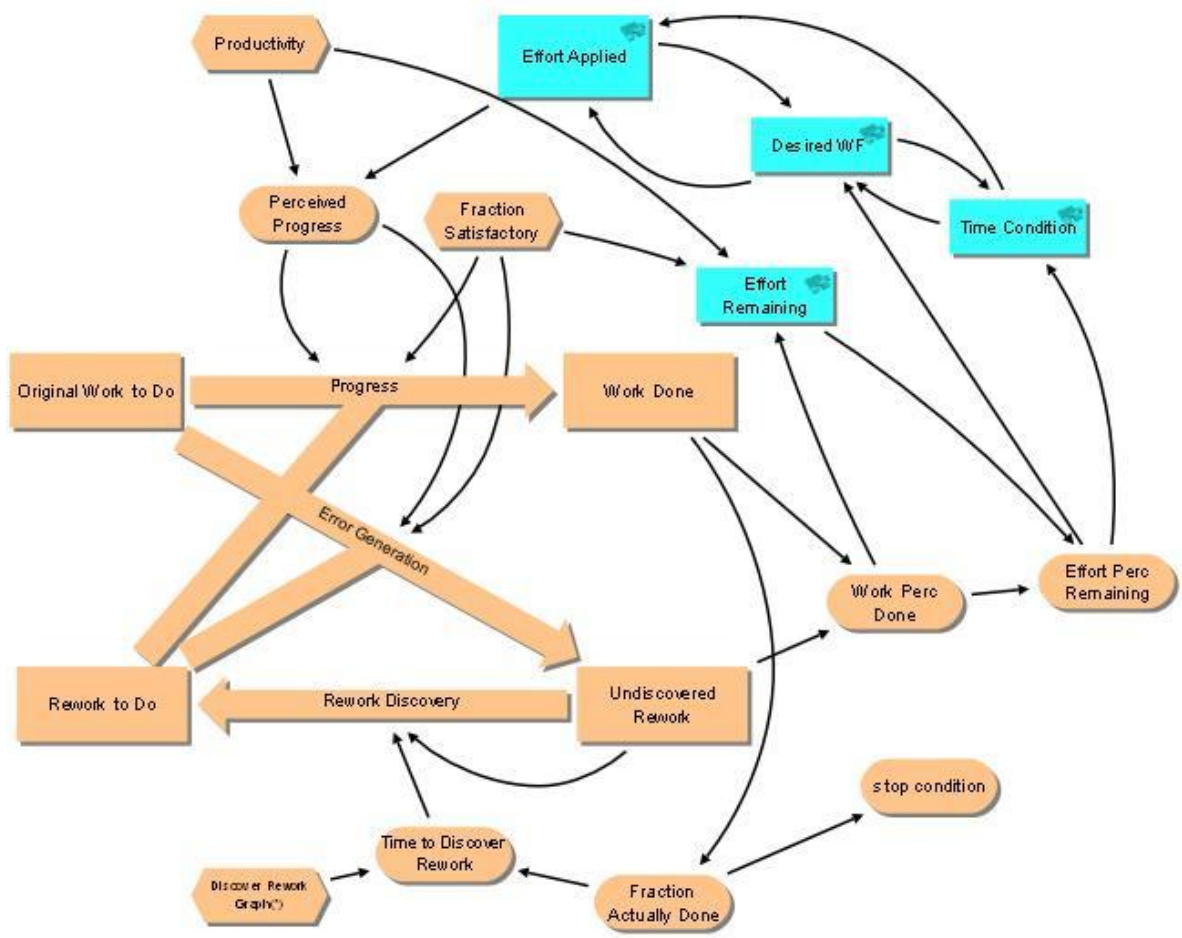

Figure 33 model with object-oriented features

4.2.2 Ripple effect

The ripple effect is initially caused by the rework cycle. Since a fraction of work done has to be reworked, the project scale has increased, resulting in the employees have to work overtime to catch up the project schedule. Keeping working overtime makes the employees exhausted. Consequently, it will lead to a reduction of effort applied by the employees.

In the basic model, we assume that the 'normal effect' that an employee's efficiency at work is 100%. However, the 'normal effect' in real world is less than 100%. It is suggested that the 'normal effect' should be at 60% [Abdel-Hamid, 1991]. The efficiency of an employee at work is decided by 'actual effect'. In class 'actual effect', Y is set to be the return value.

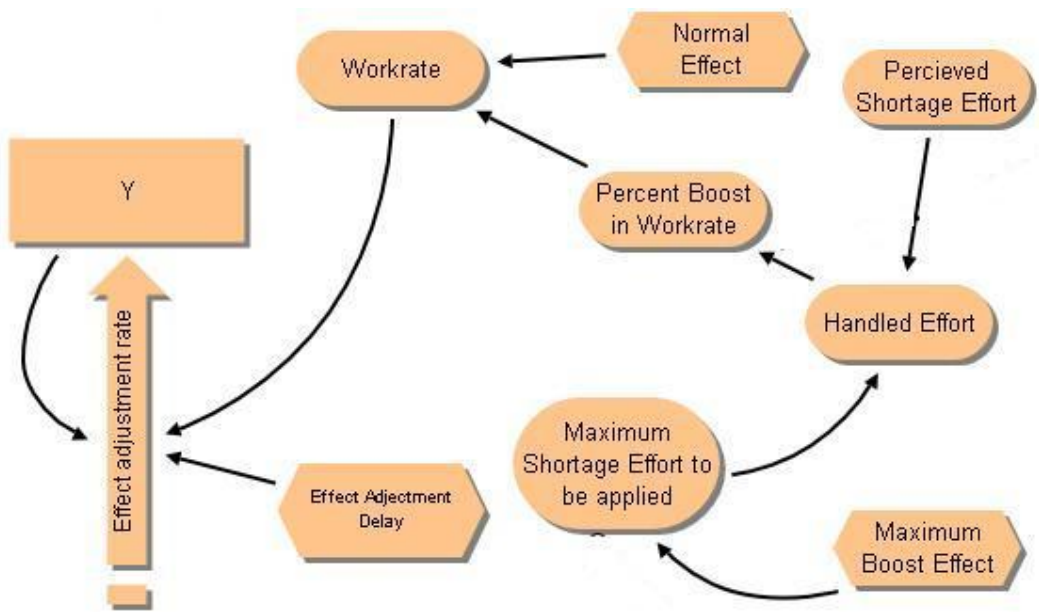

Figure 34 calculate for actual effect

Figure 34 shows the causal relations that determine the value of 'actual effect'. The 'percent boost in workrate' defines a work rate goal in terms of effort allocated to the project. The goal is not achieved immediately, since employees take time to adjust their work habits. So a delay is placed between Y and workrate. The delay time is set in the model at 14 days. When the project is perceived behind schedule, indicating a shortage effort ---'perceived shortage effort', the employees seek to boost their workrate to what they perceive is necessary to handle either all the 'perceived shortage effort' or the 'Maximum shortage effort to be applied', whichever is smaller. The 'percent boost in workrate' to handle the shortage effort equals the value of 'handled effort' divided by the product of 'workforce' and 'Overwork Duration Threshold'.

The equations for the variables in figure 34 are:

Y= stock 0.6 inflow 'Effect adjustment rate'

Effect adjustment rate= flow (Workrate-Y)/'Effect Adjustment Delay'

Effect Adjustment Delay= 14

Unit: days

Workrate= (1+'Percent Boost in Workrate')*'Normal Effect'

Normal effect= 0.6

Percent Boost in Workrate= 'Handled Effort'/('Work Force'*('Overwork Duration Threshold'+0.0001dy))

Handled Effort= min ('Maximum Shortage Effort to be applied', 'Perceived Shortage Effort')

Percieved Shortage Effort= ('Time Perc Required'-'Time Remaining')*'Labor Resource Deficit'

Maximum Shortage Effort to be applied= ('Overwork Duration Threshold'*'Work Force'*'Maximum Boost Effect')*'Willingness to Overwork'

Maximum Boost Effect= 100%

As the employees keep working at rate that is higher than normal effect, they will get exhausted.

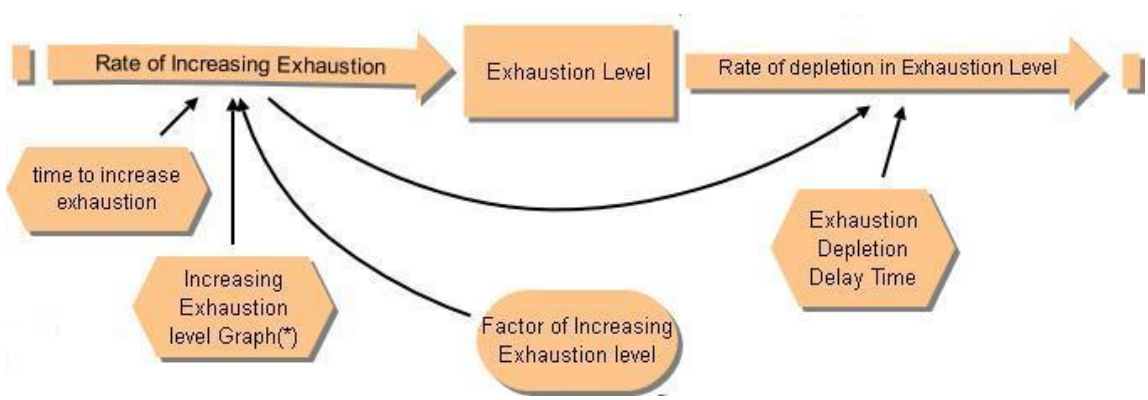

Figure 35 Exhaustion level

The flow goes into ‘exhaustion level’ uses a look up function to control the rate flows into ‘exhaustion level’, the graphic for the look up function is as follows:

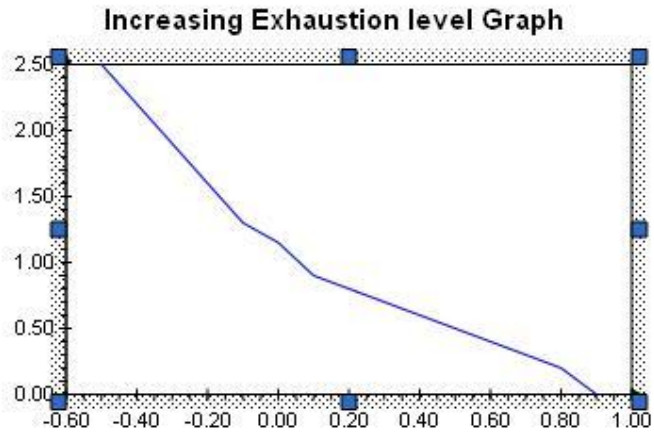

Figure 36 increasing exhaustion level graph

It shows the graph that the rate of increasing exhaustion increases as the actual effect increases.

The equations for variables in figure 35 are:

Exhaustion Level= stock 0.0 inflow 'Rate of Increasing Exhaustion' outflow 'Rate of depletion in Exhaustion Level'

Rate of Increasing Exhaustion= flow lookup ('Factor of Increasing Exhaustion level', 'Increasing Exhaustion level Graph')/'time to increase exhaustion'

Time to increase exhaustion= 1

Unit: day

Factor of Increasing Exhaustion level= (1-Y)/ (1-'Normal Effect')

Rate of depletion in Exhaustion Level= flow if('Rate of Increasing Exhaustion'$\leqslant$0 , 'Exhaustion Level'/'Exhaustion Depletion Delay Time',0)

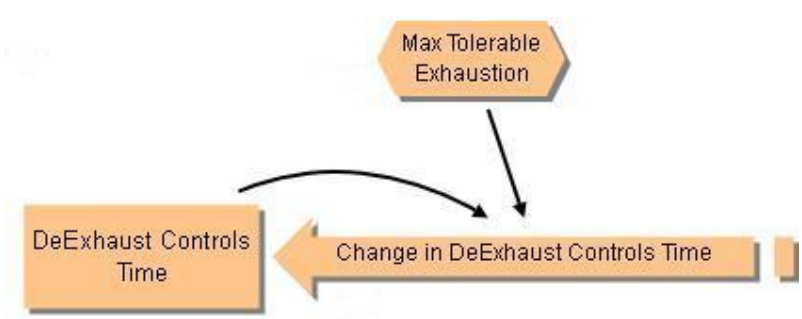

Figure 37 DeExhaust controls time

'DeExhaust controls time' is one of the factor that determines the 'willingness to overwork'

The equations for variables in figure 37 are:

DeExhaust Controls Time= stock 0dys inflow 'Change in DeExhaust Controls Time'

Change in DeExhaust Controls Time = flow if('Exhaustion Level'/'Max Tolerable Exhaustion' ≥0.1, 1, -'DeExhaust Controls Time'/'time step')

Max Tolerable Exhaustion = 50

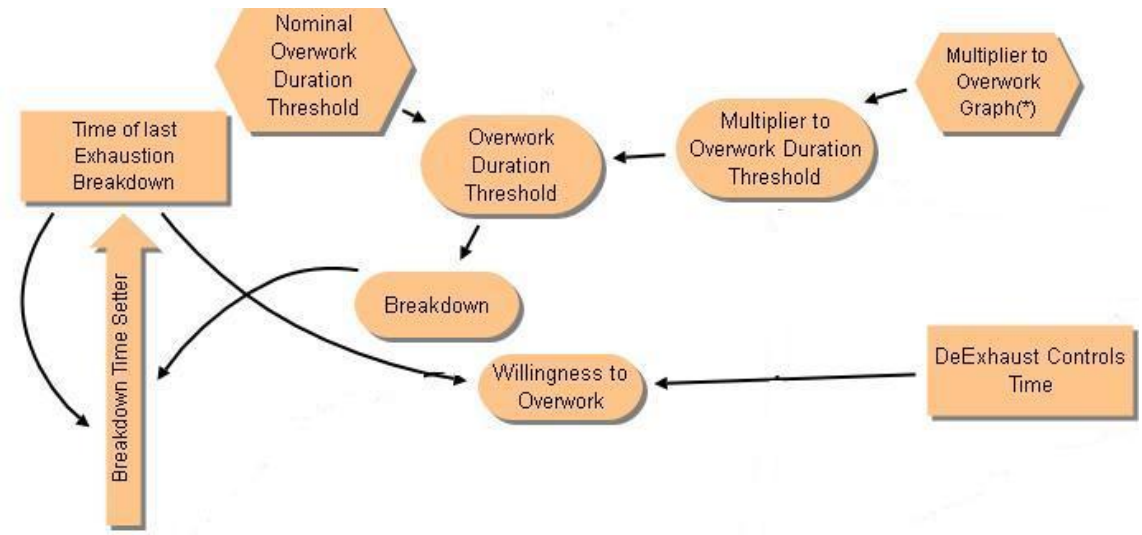

Figure 38 overwork duration threshold

The effect of exhaustion on the 'overwork duration threshold' is formulated as the 'multiplier to the overwork duration threshold'. The latter variable uses a look up function with ('Exhaustion Level'/'Max Tolerable Exhaustion') as factor. The look up graph is as follow:

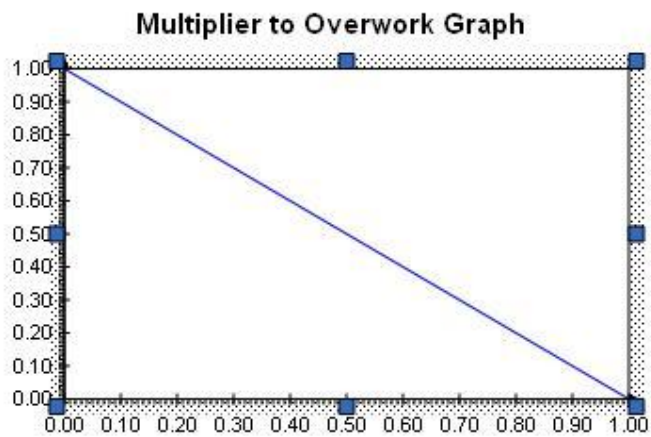

Figure 39 Multiplier to overwork

The Nominal Overwork Duration Threshold is set to be 50 days. As people starts to work at a rate above their normal rate, the threshold is cut down until possibly it reaches a va lue of zero.

The equations for variables in figure 39 are:

Time of last Exhaustion Breakdown = stock -1dy inflow 'Breakdown Time Setter'

Unit: day

Breakdown Time Setter= flow (max('Time of last Exhaustion Breakdown',

Breakdown)-('Time of last Exhaustion Breakdown'))/'time step'

Breakdown= if('Overwork Duration Threshold'=0dys, time+ 'time step',0)

Overwork Duration Threshold= Multiplier to Overwork Duration Threshold'*'Nominal Overwork Duration Threshold'

Nominal Overwork Duration Threshold= 50

Unit: day

Multiplier to Overwork Duration Threshold= 'lookup linear('Exhaustion Level'/'Max Tolerable Exhaustion', 'Multiplier to Overwork Graph')

Willingness to Overwork= if(time ≥('Time of last Exhaustion Breakdown'+ 'DeExhaust Controls Time'), 1,0)

The whole model for class ‘actual effect’ is as follow:

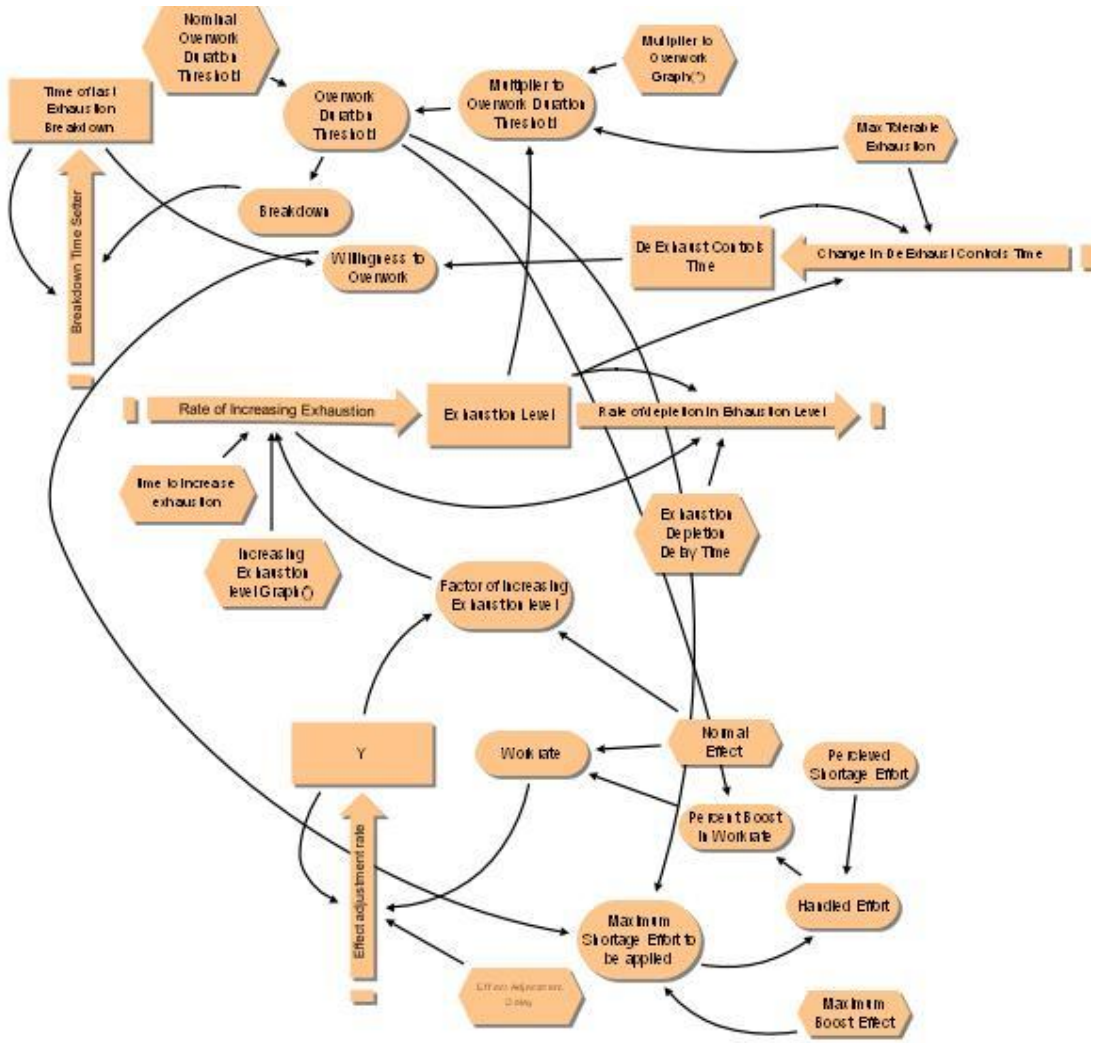

Figure 40 whole view of working overtime problem

4.3 Stop condition

The stop time for Simulation is set to 2000 days. But it does not mean that the simulation will last for so long. So the stop condition needs to be carried out. In this case, we create a variable called 'stop condition' with a link to 'Fraction Actually Done'.

The equation for 'stop condition' is:

Stop condition= 'Fraction Actually Done'>99% as [false]

So the simulation stops automatically when the 'fraction actually done' is above 99%.

5 Verification and validation

In this chapter, the system behaviours are observed through the results of simulation. The goal is to verify whether the model corresponds to the problem, and validate the model behaviour.

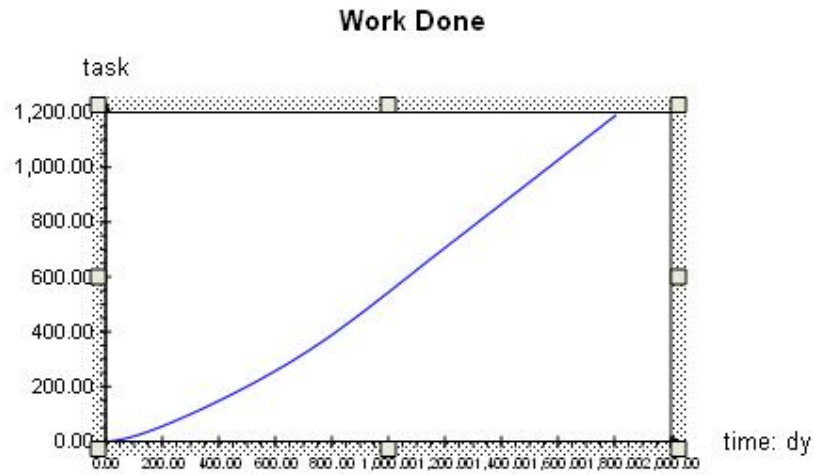

Figure 41 work done

The curve grows slowly at the beginning, because the rework affects the progress of the project . As tasks are done over time and increase of workforce, the progress goes faster as well as the work done curve.

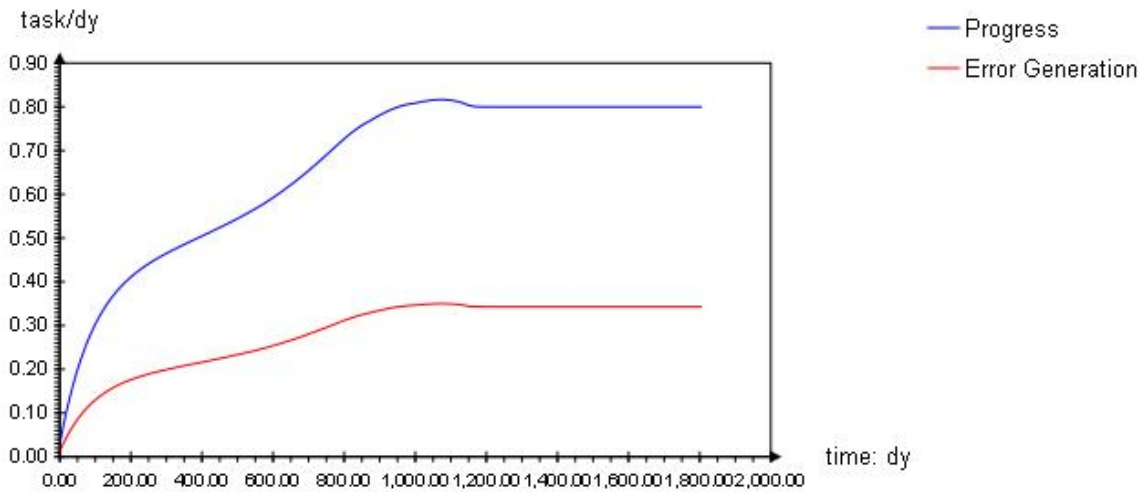

Figure 42 progress and error generation

The progress curve and error generation curve correspond to the ratio of 7:3. both of the curve grows fast in the beginning, and a bit slow in the middle, then goes fast again and finally are stable at certain levels. This behaviour is caused by multiple reasons. At the beginning, employees tend to press themselves and work overtime because of the rework. The employees get exhausted by working overtime and finally reach the breakdown, their effort applied decreases and the rate of progress slows down. After recovering from exhaustion, the rate of

progress and error generation grows fast again, and finally stabilizes in a certain level since it is close to completing the project, the schedule pressure is not significant . Employees do not need to work overtime.

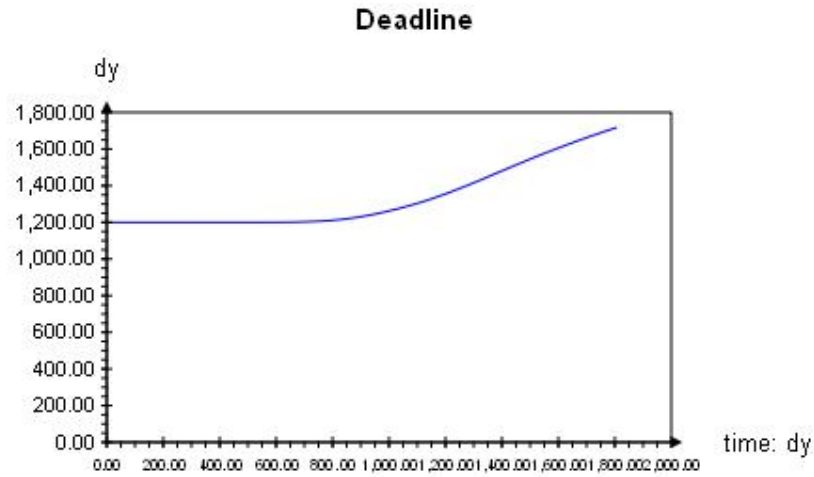

Figure 43 deadline

The deadline doesn't increase until the middle of the project. It increases slowly and stops between 1700 to 1800 days.

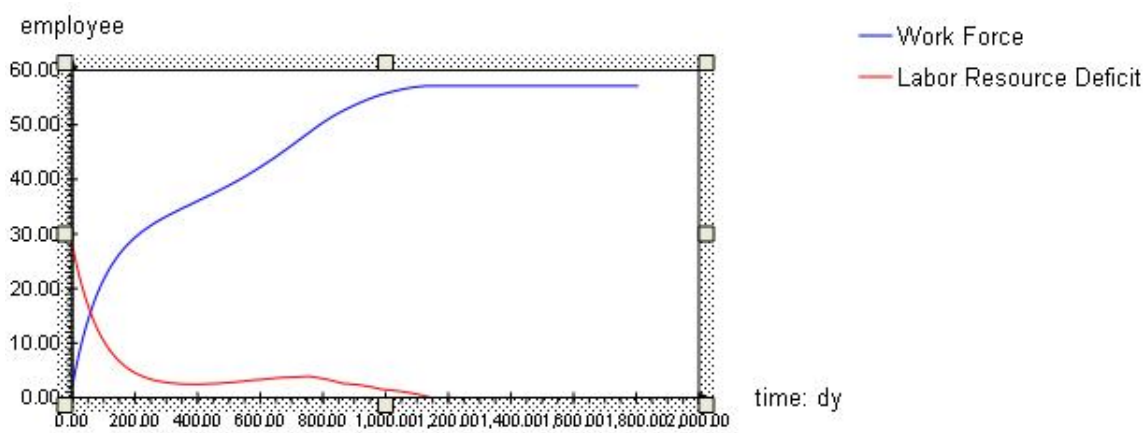

Figure 44 workforce and labor resource deficit

The workforce increases fast and the deficit of the labor resource is very high. These may due to the reason that the initial workforce is only 2 employees. The desired workforce is high at the beginning, resulting in a high deficit of workforce at the beginning.

6 Policy analysis

The policies are established to deal with the system gap. I n project management, the problem lies in schedule slipping, rework, cost overruns, hiring and so on. In this chapter approaches are taken to deal with those problems.

6.1 Shorten rework discovery time

The rework cycle is the causation of the problems in the project management area. A task that needs to be reworked is often delayed or some time till someone finds it out. So to shorten the discovery time becomes an idea to release the pain caused by the rework cycle.

The original time to discover undiscovered work is shown in table 1. Here we make it half of the original time to discover the undiscovered rework.

Here is the original graph of 'rework discovery':

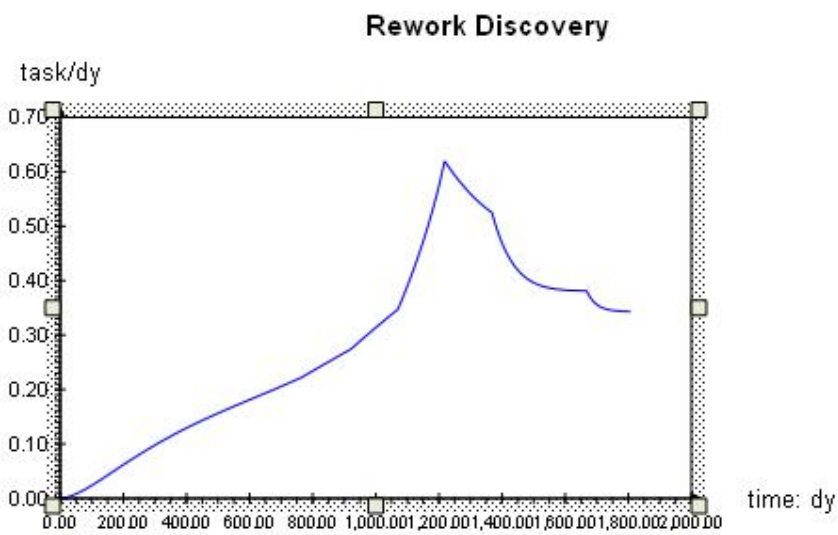

Figure 45 original graph of 'rework discovery'

Simulation stopped at 1802 days.

Then we change the data of 'Discover rework Graph':

Discover rework Graph	
0.00 %	150.00 dys
10.00 %	150.00 dys
20.00 %	150.00 dys
30.00 %	150.00 dys
40.00 %	142.00 dys
50.00 %	120.00 dys
60.00 %	52.00 dys
70.00 %	30.00 dys
80.00 %	22.00 dys
90.00 %	15.00 dys
100.00 %	15.00 dys

Table 4 new data for Discover rework graph

And the new graph of 'rework discovery' is:

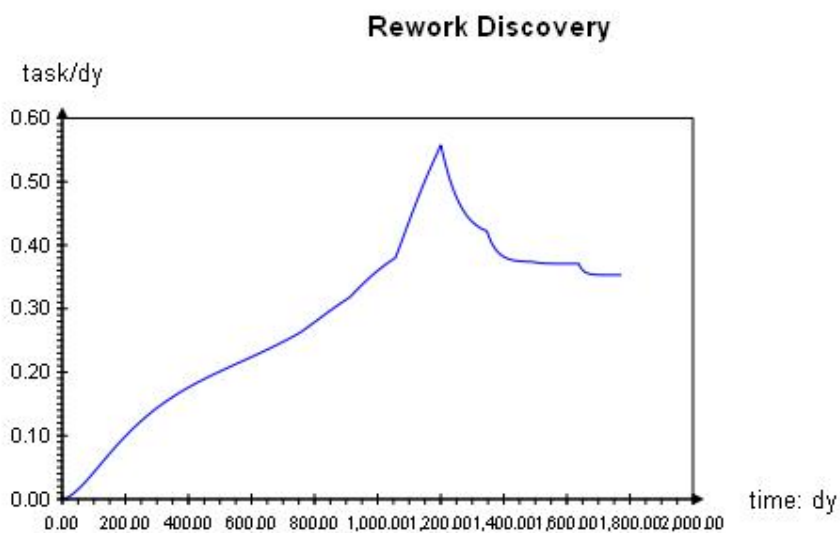

Figure 46 new result for rework discovery

The simulation stopped at 1770 days.

Turns out changing discover rework time is not efficient way to solve rework problems.

6.2 reduce time for changing workforce

The delay on changing work force leads to low productivi ty, schedule slipping, and etc. When a new employee is hired, he can not do the job well as the experienced employee does. It requires experienced employees to train the new employees till they can play a role in the project like an experienced employee does. It is essential to reduce the time the new comer becomes an experienced one. In this model, we will try in reduce the 'WF adjustment time' and see the result.

Here is the old graph for ‘progress’.

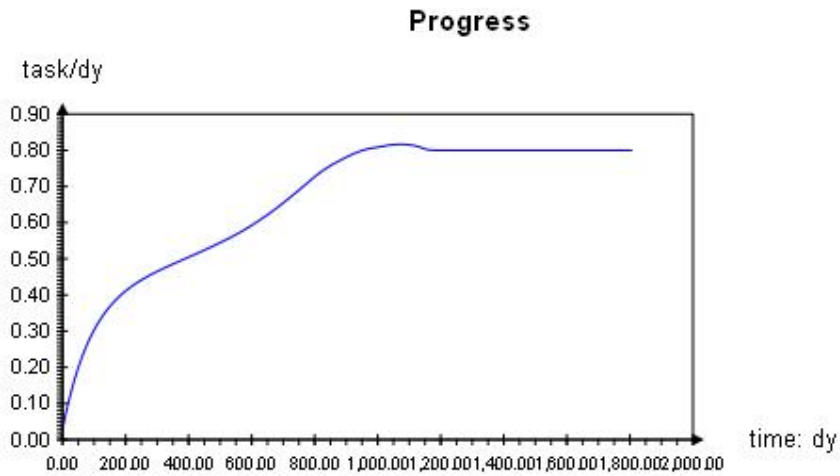

Figure 47 old data for progress

The simulation stops at 1804 days.

Now we change the ‘WF adj time’ from 90 to 30 days.

The result is:

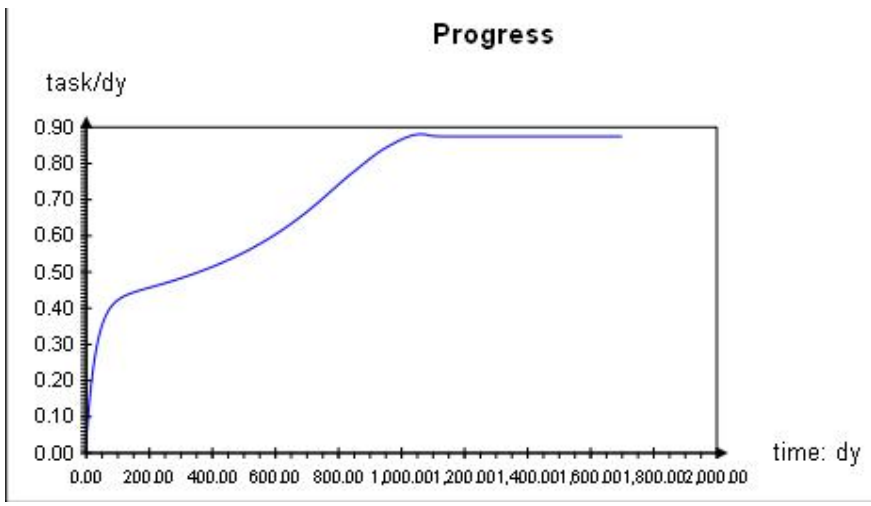

Figure 48 new data for progress

The result shows the progress rate increases significantly, and the simulation stops at 1696 days, which is around 100days leading the old approach.

This indicates that reducing ‘WF adj time’ to improve system performance works well.

There are many other policies maybe work well on solving project management problems. However we can not discuss all of them here since the scope of the thesis is limited.

7 Conclusion

In this research, a generic system dynamics model of project management has been successfully established. It contains the primary causes of project dynamics ---project features, rework cycle, project control and ripple effect. An object-oriented feature has been added into the model, which makes the model clearer and easier to understand comparing to traditional project management model.

The model correctly represents the project management problems. The verification and validation of the model are satisfying. Policies that apply to the problem have also been analyzed in this paper, which contains both futile and useful policies.

The future work lies in developing a turnover feature of workforce and keep on researching the knock-on effects that generated by the ripple effects.

Reference

John D. Sterman, System dynamics modeling for project management, 1992

John D. Sterman, Business dynamics: Ch.4-5, 2000

Peter M.Senge, The fifth discipline: The art of the learning organization , 1990

Eric Wolstenholme, Towards the definition and use of a core set of archetypal structures in system dynamics, 2003

John D. Sterman, Business dynamics: Ch.6 , 2000

Jose J. Gonzalez, Stock and flows: course material for University of Agder

James M. Lyneis and David N. Ford , System dynamics applied to project management: a survey, assessment, and directions for future research , 2007

Kenneth G. Cooper, The rework cycle: How it really works and reworks, 1993

Abdel-Hamid TK, Madnick SE. Software Project Dynamics: An Integrated Approach. Prentice-Hall: Englewood Cliffs, NJ, 1991

Alan K.Graham, Beyond PM 101: Lessons for Managing Large Development Programs, 2000

Alan K.Graham, Achieving Win-Win in a regulatory Dispute: Managing 3G Competition, 2005

Craig A. stephens, Abdel-Hamid amd James M. Lyneis, System dynamics modeling in the legal arena: meeting the challenges of expert witness admissibility, 2005

C Eden* and T Williams, The role of feedback dynamics in disruption and delay on the nature of disruption and delay (D&D) in major projects, 200 0

Terry Williams and Fran Ackermann, Learning from project failure, 2005

T.M. Williams, Safety regulation changes during projects: the use of system dynamics to quantify the effects of change, 2000

Appendix

Search table

Figure search