Transdisciplinary Engineering for Complex Socio-technical Systems
K. Hiekata et al. (Eds.)

doi:10.3233/ATDE190177

System Dynamics Modeling to Manage Performance Based on Scope Change for Software Development Projects

Kazuo HIEKATA[a,1], Mst. Taskia KHATUN[a] and Marc-Andre Chavy-MACDONALD[b]
[a]*Graduate School of Frontier Sciences, The University of Tokyo*
[b]*Graduate School of Engineering, The University of Tokyo*

Abstract: The success of a project can be described based on the output performance of the project. Output Performance defines successful completion of a project with given constraints. To keep the project's output performance high, it is very necessary to complete the project with defined a set of resources. One of the important resources of a project is scope. Scope defines the set of tasks need to be completed for a project. Scope change is a very common issue in software development project and this change has a major impact on productivity as well as the overall performance of the project. In this paper, we have proposed a simulation model based on rework cycle using System Dynamics for managing the project's performance considering scope change. We have tried to compare the scenarios both for baseline project and project with scope change. In this case we have shown how moderate use of schedule pressure and overtime gives positive impact on productivity. And this increase of productivity also increases the performance in developing the project. The proposed model helps for decision making process that will increase the productivity and keep the project's output performance high when the scope is changed.

Keywords: Software Development, System Dynamics, Project Management, Scope, Output Performance

Introduction

Project management is one of the most important field in management. The primary challenge of project management is to achieve the project's goal within the given constraints. The main constraints of the projects are scope, time, quality and budget. One of the application field of project management is in the IT sector for software development. Software development is a very dynamic and complicated process because it includes uncertainty, random and dynamic behavior and feedback mechanisms. Different physical aspects, policies and processes can be used to explain the dynamic behavior [2]. One of the easy ways of understanding this dynamic behavior is modeling because simulation models allow understanding the process behavior and dynamic interactions. Two popular methods, System Dynamics (SD) and Multi-Agent System (MAS) can be used for modeling. System Dynamics is an aggregated top-down approach that defines the relations between different key elements of the system and can illuminate the structure which produce the behavior of the system and clarify the leverage points for policy making. On the other hand, Multi-Agent System is a disaggregated bottom-up approach

[1]Corresponding Author, Email: hiekata@edu.k.u-tokyo.ac.jp.

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 [13]. Since our purpose is to focus on policy making for a system and to observe the behavior of the system, we have chosen System Dynamics for modeling. It is an effective methodology to explain the conditions of project status and to provide insights for best practice in project management. Another important module, rework cycle, is considered as the heart of modeling projects and one of the major researches and application are of this in System Dynamics [8]. Several authors have used SD and rework cycle for managing projects. For example, Lyneis et. al. [8] used rework cycle with System Dynamics to identify and manage risk, and to assess the benefit of several process and organization changes which were implemented on the project. Ownes et al. [9] linked rework cycle and disaster dynamics to define the system operations and to update different procedures used in the system. Rahmandad & Kun [12] used System Dynamics modeling with rework cycle to capture multiple defect per task and analyzed the project's finish time and delivered quality across different models.

From these literatures review we can see that System Dynamics modeling with rework cycle is very popular and effective for managing different types of projects. Based on these reviews, we have chosen SD modeling with rework cycle for managing project's output performance. Output performance is one of the important ways of identifying project status. And in this case rework cycle provides a considerable impact on project's performance. In this paper we have focused on this fundamental aspect of project's success by managing the resources smoothly throughout the project. We have considered one of the important resources of project, scope. Since software development process is very dynamic, scope change during the development period is very natural. Specifically, we have examined the productivity and the overall output performance considering scope change by applying optimal level of schedule pressure and overtime. We have shown how optimal use of schedule pressure and overtime can increase both productivity and performance without changing the availability of other resources.

As for the structure of this paper, in the introduction part, we introduced the background and claimed the overall aim and objectives. A background of System Dynamics and rework cycle is described in section 1. The methodology is presented in section 2. The analysis of model is described in section 3. After that, simulation results have been explained in section 4. At last, discussion and conclusion are described in section 5 and 6 respectively.

1. System Dynamics (SD) Background

The System Dynamics is introduced by J.W. Forrester. It applies the engineering principles of feedback and control to social systems. In System Dynamics, a system is defined as a collection of elements that continually interact with each other and the elements from outside over time to form unified whole. The fundamental principle of system dynamics is based on the premise that the behavior of the system is caused by its underlying structure [1][3].

System Dynamics is a combination of stock-flow and a set of variables. Stock variable is an entity whose value increases over time from inflows and decreases from outflows. Stocks are changed only by flows into the system and out of the system. Stocks normally have a certain value at each moment of time. Figure 1 shows an example of system dynamics model of software development.

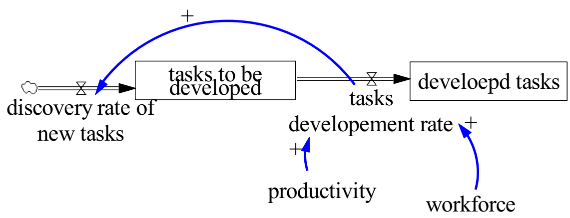


Figure 1. Basic stock flow diagram for software development.

1.1 Rework Cycle

The rework cycle is considered as the most important feature of System Dynamics project models. Its recursive nature generates rework that generates more rework and often creates challenges for project management. The basic structure of rework cycle (adopted form Lyneis 2007) is given in Figure 2. Here the stocks are Original work to do, Undiscovered rework, rework to do and Work done. The flows are work done correctly, error generation, rework generation and rework discovery and rework done. The other variables are represented as auxiliary variable that may depend on other variables or have constant value.

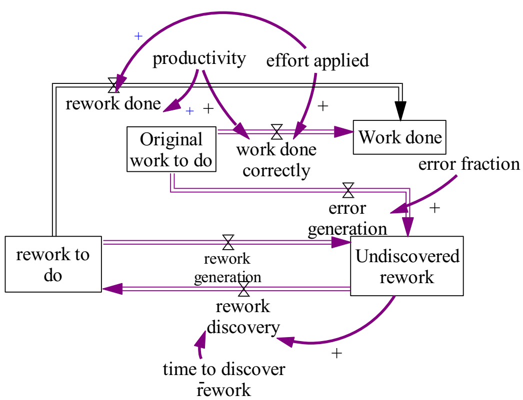


Figure 2. Basic rework cycle (adopted from [3]).

2. Methodology

A full SD model is developed using causal-loop diagram and stock-flow diagram. To develop a project successfully, managing productivity properly is essential. There are several factors that have both positive and negative impact on productivity. The causal-loop diagram for productivity is described in section 2.1.

We have focused on developing our model based on requirements discovery and separating them as hard and soft requirements because the productivity depends on requirements discovery. This requirements discovery and successful completion of requirements based on rework cycle has been shown as stock-flow diagram in section 2.2.

2.1 Causal Loop Diagram for Productivity

Causal-Loop Diagrams (CLDs) illustrate cause-effect relationships between different variables. In CLDs, arrows are used to indicate the relationships and polarity is used to define the status of the relationship. The causal loop diagram for productivity is shown in figure 3. In this figure, the burn out loop defines that due to overtime working, the employee becomes exhausted and thus decreases the productivity and also increases error. Reduced productivity keeps the amount of work to do greater and thus more completion time is needed. This also increases the workforce hiring and causes skill dilution. On the other hand, increasing human resources also increases communication overhead which decreases the productivity and increases error fraction. In case of hiring new members, it is a lengthy process and the delay mark between hiring and workforce defines this situation.

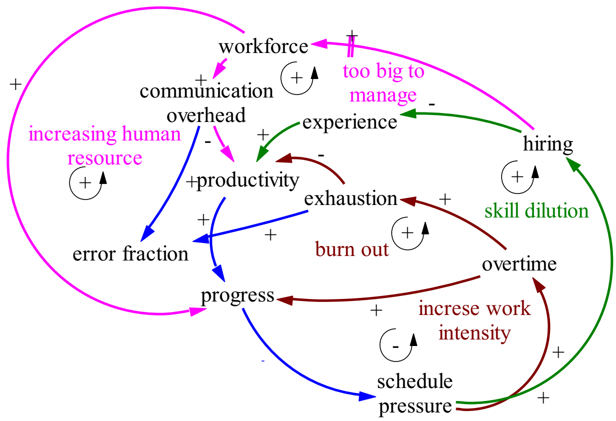


Figure 3. Causal Loop Diagram for Productivity.

2.2 Stock-Flow Diagram

Figure 4 shows the one portion of stock-flow diagram for requirements discovery and output performance.

We designed the stock-flow diagram based on rework cycle. In the diagram, three parts labeled in different color shows the stock-flow for requirements discovery, rework cycle and requirements completion. In the upper part of the diagram, stocks for requirement discovery have been used. Initially we assumed that most of the requirements are undiscovered and after discovering them we categorized as soft requirements and hard requirements. The rates requirements hardening and requirements softening categorize the requirements as hard requirements and soft requirements. The rates requesting features and requirements removal defines scope change.

After discovering the requirements, all of them go to the stock work to do in the rework cycle that is shown at the bottom part of the diagram.

In the middle part of the diagram, requirements completion is shown using the stocks hard requirements fulfilled and soft requirements fulfilled. The rate overflow between these two stocks defines the soft requirements completion. And the overall performance of a project depends on the completion of hard and soft requirements.

Using these stock-flow diagram and the causal-loop diagram, we have developed our full SD model.

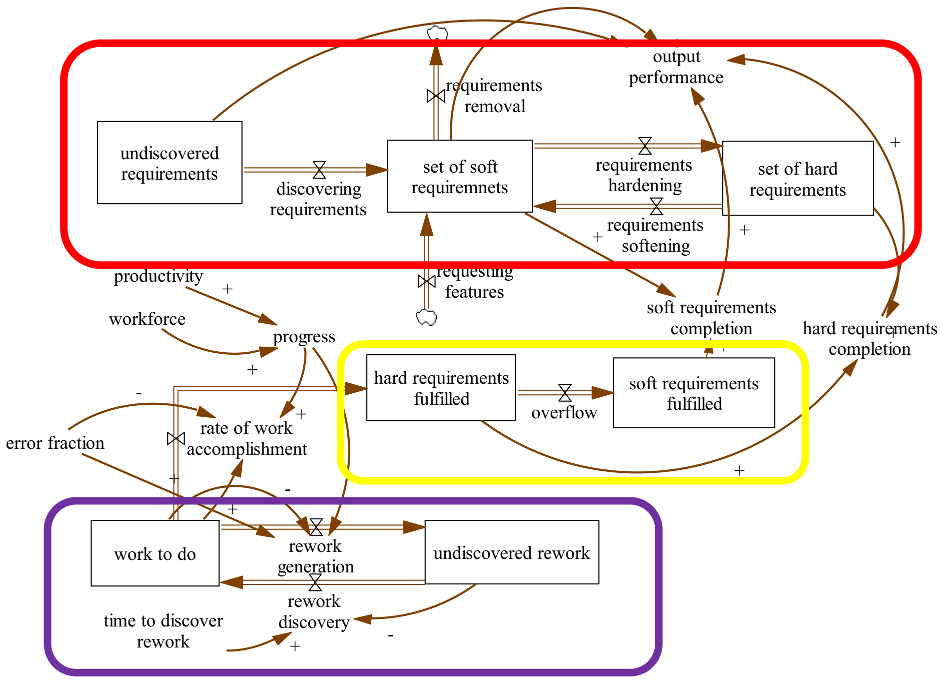


Figure 4. Stock-Flow Diagram for requirements discovery and output performance.

2.2.1 Scope Change

Scope change is a very sensitive issue because it affects in every aspects of the other resources such as budget, time and also the overall performance. In software development project, scope change is very common. There are several causes of scope change. These causes can be either internal or external. Internal causes include rework, training rate for new employee and managerial decision. External causes include customer involvement, financial and technical issues and so on. In figure 4 scope change is defined by the flows requesting features and requirements removal.

3. Model Analysis

3.1 Baseline Project

For analyzing the model, we have considered two types of projects, one is baseline project and the another one is project with scope change. Baseline project is defined as the project with fixed resources. In this case it is assumed that all the resources remain fixed till the project is finished.

3.2 Project with Scope Change

In this case projects have been considered with scope change. We have considered that the scope change is happened during the development of the project. There are several reasons for scope change described in section 2.2.1. Rework is a very important feature for completing the tasks more accurately. But often large amount of rework causes the delay for project completion and thus causes scope creep. Another important factor for scope change is customer involvement. The customers provide their feedback and satisfaction level and thus often causes scope creep.

3.3 Policies

In order to manage a project smoothly, different policies can be used depending on the type and condition of the projects. Smith et al. [18] defines several policies that can be applied for managing software development projects shown in Table 1.

Table1. List of policies.

Policies \ Used in the model	Yes	No
Implementing a project with a fixed number of staffs from the beginning	X	
Using moderate overtime instead of sustained overtime since it doesn't increase productivity in the long run	X	
Completion of project within budget		X
The use of 'experts' can significantly increase the project performance	X	
Focusing on client and consumers		X
A moderate amount of schedule pressure is optimal	X	

3.4 Output Performance

WE have defined the project status with output performance. Output performance defines the amount of hard and soft requirements completion. The range of output performance is measured based on hard and soft requirements completion. To make a project success, the hard requirements must be completed 100%(=1) and the soft requirements shall be completed more than 90%(>.9). The completion of software requirements is followed by the completion of hard requirements.

4. Simulation Result Analysis

After developing the full SD model, at first, we did simulation run of the model using several past data. For this purpose, we extracted different data from previously validated simulation model and used those data to observe the behavior of our model. We measured the performance in terms of time and amount tasks completion and each case the model gives a better performance. After that we have analyzed the model for baseline project and project with scope change.

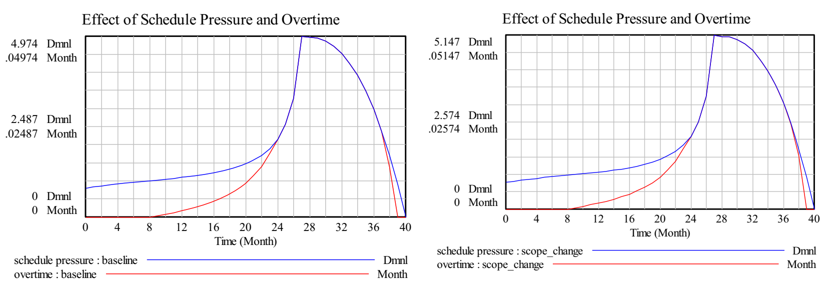
Developing a project within given constraints and without changing the availability of the resources is often difficult, especially for software project because of its uncertain and dynamic behavior. But using compatible policies and control actions, it is possible to make a project success. During simulation run, we have kept the resources fixed and applied the policies mentioned above to see the behavior of the performance of completing the tasks.

Table 2 shows the amount of some constant input parameters that are used for the projects.

Table 2. Initial data declaration for the projects.

Parameters	Amount of data		Units
	Baseline project	Project with Scope change	
Workforce	10	10	person
Potential productivity	10	10	tasks/person/month
Number of tasks	1200	1375	tasks
deadline	40	40	month

When a team consists of both expert and new members, the productivity of them will be different. Diversification of productivity has a considerable impact on the overall performance. Very often it becomes difficult to complete the project within deadline. In this case, one of the solutions is to apply moderate schedule pressure and overtime instead of changing other resources such as scheduled completion time and increasing more workforce. Because moderate schedule pressure and overtime often increase productivity and helps to complete a project within time. According to T. A. Hamid [1], optimal schedule pressure increases the performance and relaxes the quality assurance activities. Park et. al. [2] defines the average value of normal schedule pressure is 4 dmnl and for high schedule pressure is more than 5 and projects can be completed by applying this amount of schedule pressure. The amount of overtime depends on level of schedule pressure.



(a) (b)

Figure 5. Effect of schedule pressure and overtime for (a) baseline project (b) project with scope change.

Figure 5 show the impacts of schedule pressure and overtime for both projects. At the beginning when the schedule pressure is low, the overtime is zero and at a certain point the overtime increases gradually with the increase of schedule pressure and after a period of time both the schedule pressure and overtime decrease. The optimal value of schedule pressure for baseline project is 4.974 and for project with scope change 5.147 is used to complete all tasks. Based on the schedule pressure, overtime increases in a moderate way and this moderate increase of overtime gives a positive impact on productivity shown in Figure 6. This increase of productivity also increases the project's output performance. In case of baseline project, the productivity is higher than in case of project with scope change.

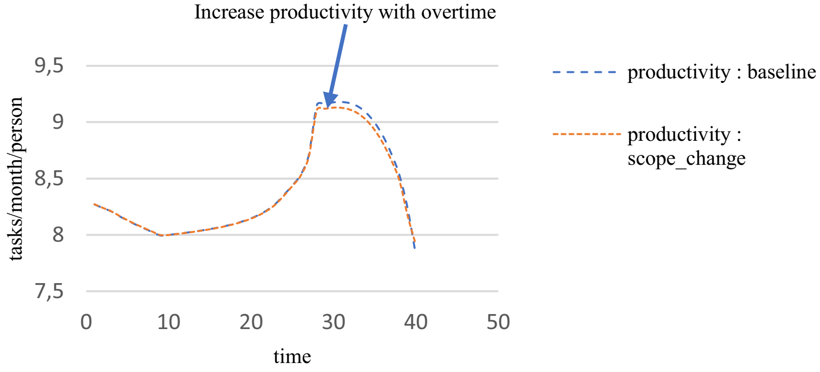


Figure 6. Productivity for both projects.

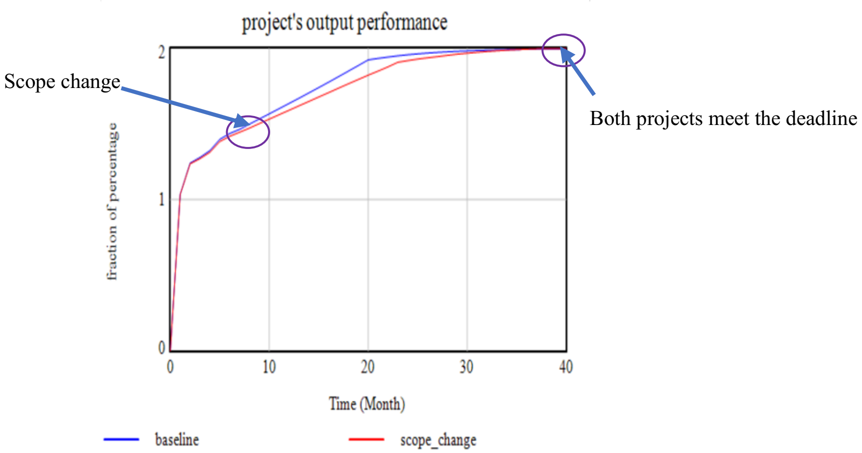


Figure 7. Output Performance.

In Figure 7, the output perfromance is shown in term of hard and soft requirements completion. The label in y-axis show the fraction of percentage of requirements completion. The first portion(1[100%]) shows hard requirements completion and the second portion(2[another 100%]) shows soft requirements completion. During the completion of requirements, the gap between baseline project and project with scope change defines the scope change. In case of scope change, since it happens during the project, so more time is needed to analyze and design the tasks befor implementation. That's why the performance in case of scope change becomes low. Although the perfromance becomes low, applying the moderate schedule pressure and working with optimal overtime, both projects can meet the deadline as shown in the figure.

4.1 Effect of Fatigue on Human Behavior

Another important factor that has an impact on project performance is fatigue. Fatigue is described as objective change in physical performance. Acording to Phillips[16], fatigue in performance decrement can be defined as a diminished capacity for work and possibly decrements in attention, perception, decision making and skill performance. Fatigue at work is normal in everyday experience. But excess level of fatigue affects the person's productivity. The natural behavior of fatigue is described by Beurskens et al. [17] based on several questionaires referred as checklist individuals strength (CIS) questionaires in a function of time.

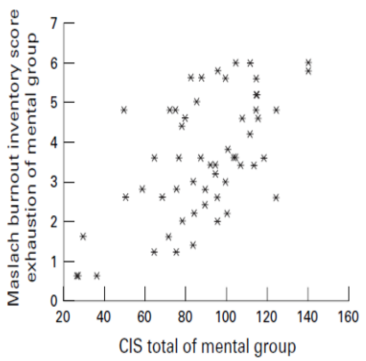


Figure 8. Behavior of fatigue with time according to CIS[17].

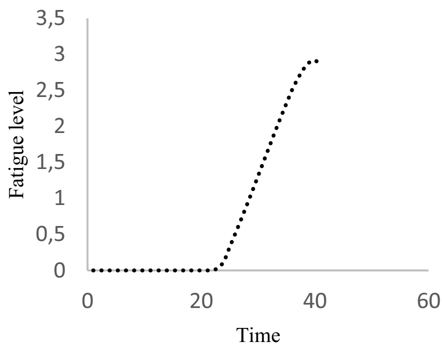


Figure 9. Behavior of fatigue as a function of time.

According to Figure 8 it is mentioned that when the number of people in mental groups increases with time, the exhaustion level also increases. It defines the natural behavior of exhaustion depending on time. Comparing to this graph, Figure 9 also shows the same behavior of exhaustion level in a function of time that incraeses with the increase of schedule pressure and overtime which defines the correlation of the model with human behavior.

5. Discussion

In this paper, we have defined the methodology of keeping the projects output performance high considering a small portion of scope creep. We have analyzed the influence of schedule pressure and overtime on productivity as well as on the performance and drew the conclusion that without changing the schedule and workforce, it is possible to complete a project successfully even when there will be scope creep.

6. Conclusion

Since software development is a complex process and System Dynamics provides an easy way to understand the dynamic behavior of a system, we have tried to a develop a SD model for both measuring and increasing the output performance based on rework cycle and hard and soft requirements completion. We have shown how productivity and output performance can be kept high by applying moderate schedule pressure and overtime without changing the other resources and complete the projects successfully in time. Although high schedule pressure often puts negative impacts on performance but optimal amount of schedule pressure depending on situation can increase the performance which we have obtained from our model.

References

[1] T. K. Abdel-Hamid, *The Dynamics of Software Development Project Management: An Integrative System Dynamics Perspective*, PhD thesis, Massachusetts Institute of Technology, 1984.
[2] M. Park, W. Kim, Y. Yoon, and M.P. Nepal, Scheduling Decisions and their Dynamic Consequences on Construction Performance, *KSCE Journal of Civil Engineering*, 2010, 14(3):251-259.
[3] J.M. Lyneis and D.N. Ford, System dynamics applied to project management: a survey, assessment, and directions for future research, *System Dynamics Review* , 2007, Vol. 23, pp. 157–189.
[4] D.N. Ford, J.D. Stermen, Dynamic Modeling of Product Development Processes, *System Dynamics Review*, 1998, Vol. 14, Issue 1, pp. 31-68.
[5] S. Li, *A generic model of project management with Vensim*, Master's thesis, Faculty of Engineering and Science, Agder University, 2008.
[6] C.D. Vajre, *Modeling dynamic interactions in a software development project*, Master's thesis, University of South Florida, 2003.
[7] N.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.
[8] J.M. Lyneis, K.G. Cooper and Sharon A. Elsa , Strategic management of complex projects:a case study using system dynamics, *System Dynamic. Review,* 2001, Vol. 17, pp. 237–260,
[9] B.D. Owens, N.G. Leveson and J.A. Hoffman, Procedure rework: a dynamic process with implications for the "rework cycle" and "disaster dynamics", *System Dynamics Review*, 2011, Vol. 27, No. 3, pp. 244–269.
[10] A. Nasirikaljahi, *The dynamic of modern software development project management and the software crisis of quality, An integrated system dynamics approach towards software quality improvement*, Master thesis, University of Bergen, Norway, 2012.
[11] T.K. Abdel-Hamid and S. Madnick, Software productivity: potential, actual, and perceived, *System Dynamics Review*, Vol. 5, No. 2, pp. 95-113.
[12] H. Rahmandad and K. Hu, Modeling the rework cycle: capturing multiple defects per task, *System Dynamics Review*, 2010, Vol. 26, pp. 291–315.
[13] T. Taylor and D.N. Ford, Tipping point failure and robustness in single development project, *System Dynamics Review*, 2006, Vol. 22, pp. 51–71.
[14] J. Hillston, Model Validation and Verification, In: P.M. Swamidass (eds) *Encyclopedia of Production and Manufacturing Management*, Springer, Boston, 2000.
[15] H. Rahmandad and J. Sterman, Heterogeneity and Network Structure in the Dynamics of Diffusion: Comparing Agent-Based and Differential Equation Models, *Management Science*, Vol. 54, No. 5, pp. 998–1014.
[16] R.O. Phillips, *What is fatigue and how does it affect the safety performance of human transport operators? Fatigue in Transport Report 1*, Istitute of Transport Economics, Oslo, 2014.
[17] A.J.H.M. Beurskens, U. Bültmann, I. Kant, J.H.M.M. Vercoulen, G. Bleijenberg and G.M.H Swaen, Fatigue among working people: validity of a questionnaire measure, *Occup Environ Med* 2000, Vol. 57**,** pp. 353–357.
[18] B.J. Smith, H. Nguyen, and R.F. Vidale, Death of a software manager: How to avoid career suicide through dynamic software process modeling, *American Programmer*, 1993, Vol. 3, No. 5, pp. 10-17.