

Understanding Software Processes through System Dynamics Simulation: A Case Study

Carina Andersson¹, Lena Karlsson¹, Josef Nedstam¹, Martin Höst¹, Bertil I Nilsson²

¹*Department of Communications Systems
Lund University, P.O. Box 118
SE-221 00 Lund, Sweden
e-mail: (carina.andersson, lena.karlsson,
josef.nedstam, martin.host)
@telecom.lth.se*

²*Department of Industrial Management
and Logistics
Lund University, P.O. Box 118
SE-221 00 Lund, Sweden
e-mail: bertil.nilsson@iml.lth.se*

Abstract

This paper presents a study with the intent to examine the opportunities provided by creating and using simulation models of software development processes. A model of one software development project was created through means of system dynamics, with data collected from documents, interviews and observations. The model was simulated in a commercial simulation tool. The simulation runs indicate that increasing the effort spent on the requirements phase, to a certain extent, will decrease the lead-time and increase the quality in similar projects. The simulation model visualizes relations in the software process, and can be used by project managers when planning future projects. The study indicates that this type of simulation is a feasible way of modelling the process dynamically although the study calls for further investigations as to how project or process managers can benefit the most from using system dynamics simulations.

1. Introduction

This study was performed in cooperation with Ericsson Mobile Communications AB and is based on a development project carried out in 1999.

As a step in the constantly ongoing work with quality improvements at Ericsson this study was made to show if simulation can be used for visualizing how different factors affect the lead-time and product quality, i.e. number of faults. One of the most important factors that affect the lead-time of the projects and the product quality is the allocation of human resources to the different process phases. Thus, the focus of this simulation study is on resource allocation.

Simulation is commonly used in many research fields, such as engineering, social science and economics. That is,

simulation is a general research methodology that may be applied in many different areas. Software process modeling and improvement is, of course, no exception and simulation has started to gain interest also in this area. For example, in [4] a high-maturity organization is simulated with system dynamics models, and in [6] a requirements management process is simulated with a discrete event simulation model. In [8] an overview of simulation approaches is given.

There are several advantages of building and simulating models of software processes. By simulation new knowledge can be gained that can help to improve current processes. Simulation can also be used for training and to enforce motivation for changes.

The objectives of the study that is presented here are to investigate if it is possible to develop a simulation model that can be used to visualize the behaviour of selected parts of a software process, and to evaluate the usefulness of this type of models in this area. The objective of the model is to identify relationships and mechanisms within a project. The study is focused on the tendencies of the simulation results and not the quantitative aspects.

The outline of the paper is as follows: In Section 2 the method used in this study is described. Section 3 describes the execution of the simulation study. Section 4 presents the results of the simulation and Section 5 discusses and summarizes the results of the study.

2. Method

This project was designed as a case study. Case studies are most suitable when data is collected for a specific purpose and when a subgoal of the study is establishing relationships between different attributes. A main activity in case studies is observational efforts.

With support from existing results in literature [3, 16], the research approach was created in three consecutive

steps: problem definition, simulation planning and simulation operation. This methodology is based on the process chain concept, but due to lack of enough available, reliable data, the process in practice went into an interactive pattern.

In the first phase, problem definition, the problem was mapped. Then through deeper definition and delimitation, an agreement was created around the study's purpose.

The main part in the second phase of the study, simulation planning, was to identify factors influencing the product quality. The assigner of this study wished to test the idea of using simulation models and this was governing in the details of the study. This was natural as most of the ideas to the quality factors were picked up from the organization's project, through interviewing the project staff and through documents. To add a broader perspective, results and ideas were taken from software literature. Influence diagrams were built including the different quality factors' relation to each other, but primary their effects on lead-time and product quality.

The third phase, operating the simulation model, started with translating a small part of the theoretical model into the simulation tool. A short test showed that the simulation tool worked properly. More features were added from the theoretical model into the simulation tool and more test runs were performed. The verification and validation of the model was made stepwise through the input of the whole model into the simulation tool, and the yardstick to compare with was given by documents and discussions with the assigner.

3. Developing the simulation model

In the simulation domain there are two main strategies: continuous and discrete modelling. The continuous simulation technique is based on system dynamics [1], and is mostly used to model the project environment. This is useful when controlling systems containing dynamic variables that change over time.

The continuous model represents the interactions between key project factors as a set of differential equations, where time is increased step by step. In the standard system dynamics tools, these interconnected differential equations are built up graphically. A system of interconnected tanks filled with fluid is used as a metaphor. Between these tanks or levels there are pipes or flows through which the variables under study are transported. The flows are limited by valves that can be controlled by virtually any other variable in the model. Both this mechanism and the level-and-flow mechanism can be used to create feedback loops. This layout makes it possible to study continuous changes in process variables such as productivity and quality over the course of one or several projects. It is however more problematic to model discrete events such as deadlines and milestones within a project [9, 10].

In the discrete model, time advances when a discrete event occurs. Discrete event modelling is for example preferred when modelling queuing networks. In its simplest form, one queue receives time-stamped events. The event with the lowest time-stamp is selected for execution, and that time-stamp indicates the current system time. When an event occurs an associated action will take place, which most often will involve placing a new event in the queue. Since time always is advanced to the next event, it is difficult to integrate continually changing variables. This might result in instability in any continuous feedback loops [9, 10].

To suit the purpose of this study, which is to visualize process mechanisms, continuous modelling was used. The continuous model was chosen in order to include systems thinking [13] and because it is better than the discrete event model at showing qualitative relationships.

3.1. Problem definition

The study is based on a process that is similar to the waterfall model [14]. The whole process is shown in Figure 1, but the simulation model was focused on the

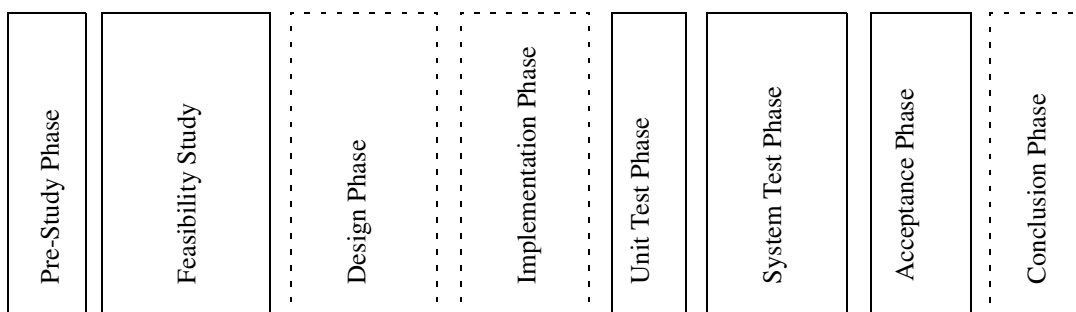


Figure 1. Process description

requirements phase and the test phase. The other phases, with broken lines in Figure 1, were excluded to get a less complex model. The requirements phase includes the pre-study phase and the feasibility study phase. The test phase involves the unit, system and acceptance tests. All these types of tests are included, since the data available did not separate between test types and they overlapped in terms of time.

3.2. Simulation planning

This step included identifying factors that affect the quality of the developed software and the lead-time of the project. This was made through interviews with project staff and based on information in literature [5, 7].

The relevant project staff consisted in three persons with whom discussions were held continually during the entire study. Among the factors discovered during interviews, only those considered relevant to software development

Table 1. Factors that affect quality and lead-time

Number of personnel in the project
Level of personnel education
Level of personnel experience
Level of personnel salary
Level of personnel turnover
Communication complexity
Geographical separation of the project
Software and hardware resources
Environment, e.g. temperature, light, ergonomics
Amount of overtime and workload
Level of schedule pressure
Level of budget pressure
Amount of new market requirements
Amount of requirements changes
Level of inadequate requirements
Amount of review
Amount of rework
Level of structure in the project organization
Standards that will be adhered to e.g. ISO and IEEE
Amount of software functionality
Testing and correcting environment and tools
Productivity
Amount of program documentation
Level of reusable artefacts, e.g. code and documentation

processes were selected. The identified factors are listed in Table 1. Discussions with concerned personnel pointed out the most important factors in respect to both quality and lead-time. The factors considered to affect quality and lead-time the most were chosen to be included in the influence diagrams, see Figure 2.

Influence diagrams [12] for the requirements and the test phase were built to show how the chosen factors affect the lead-time and the software quality. Each factor's importance for each phase was considered together with the relationships between the factors. The influence diagram for the requirements phase is shown in Figure 2. The factors in the influence diagram are further explained below.

- *Amount of functionality* is the estimated software functionality to be developed.
- *Amount of new market requirements* is a measure of the change in market expectations.
- *Amount of requirements changes* is a measure of the changes made in the requirements specifications.
- *Amount of review* involves reviewing requirements specifications.
- *Amount of rework* is the effort spent on reworking both new and inadequate requirements.
- *Communication complexity* is an effect in large project groups where an increasing number of participants increases the number of communication paths.
- *Level of inadequate requirements* is a measure of the requirements specification quality.
- *Level of personnel experience* is a measure of knowledge of the current domain.
- *Level of schedule pressure* is the effect of the project falling behind the time schedule.
- *Number of personnel* is the number of persons working with requirements specifications in the project.
- *Productivity* is a measure of produced specifications per hour and person.
- *Time in requirements phase* is the lead-time required to produce the requirements specifications in this project.

It is beyond the scope for this paper to present all details of the simulation model. In this paper the simulation model and related models, such as influence diagrams, are presented in some detail for the requirements phase. The requirements phase is by its nature more intuitive and easy to understand than the test phase. For a presentation of details of the complete simulation model with all related models refer to [2]. For example, the influence diagram for the test phase is presented in [2] and not here.

At the same time as the influence diagrams were constructed, causal-loop diagrams were built to get a basic

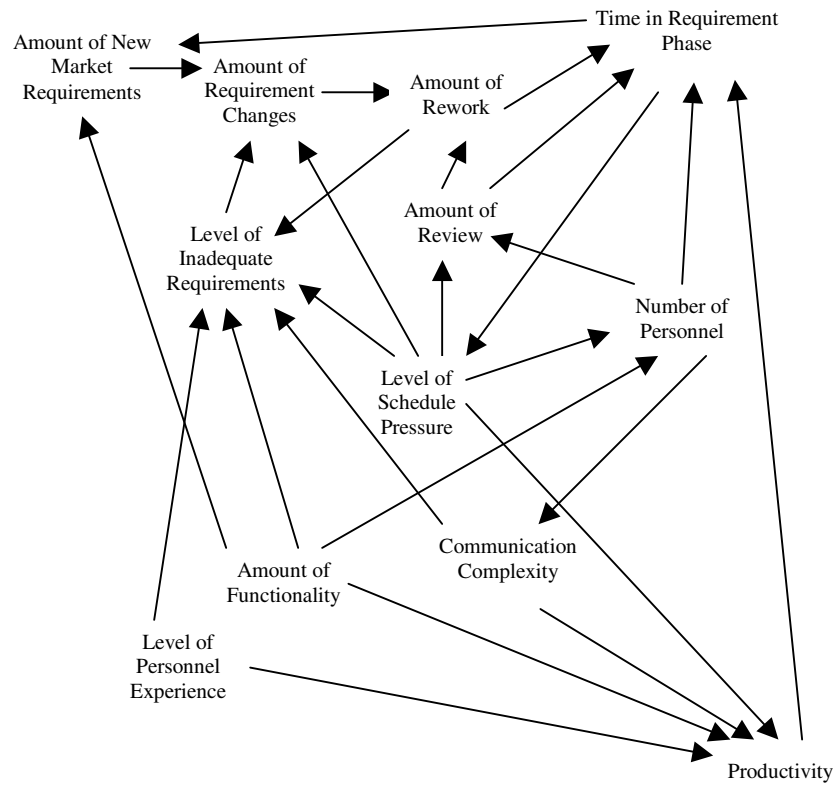


Figure 2. Influence diagram for the requirements phase

understanding of the feedback concepts. Causal-loop diagrams are often used in system dynamics to illustrate cause and effect relationships [1]. When examining these relationships isolated, they are usually very easy to understand. However, when they are combined into long chains of cause and effect, they can become complex. The causal-loop diagrams increase the understanding of these complex relations. Figure 3 illustrates how the schedule pressure affects the time spent in the requirements phase. An increased schedule pressure increases the error generation, due to a higher stress level. A high error density increases

the amount of necessary rework and thereby increases the time in the requirements phase, which in turn increases the schedule pressure. At the same time, high schedule pressure increases the productivity because of its motivational role. Increased productivity decreases the time spent in the requirements phase, which in turn decreases the schedule pressure.

Information about the relationships between the factors in the causal-loop diagram is shown by adding an “O” or an “S” to the arrows. An “O” implies a change in the opposite direction, while an “S” implies a change in the same direction.

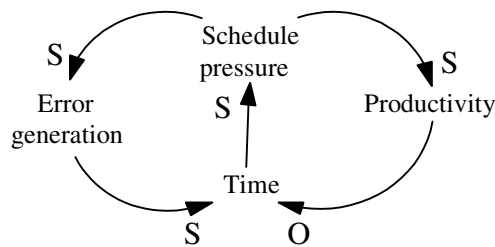


Figure 3. Causal-loop diagram

3.3. Simulation operation

The simulation model was built based on the knowledge gained from creating influence diagrams and causal-loop diagrams. The idea behind the model of the requirements phase is based on a flow of tasks, from customer requirements to finished specifications. In the requirements phase there is a transformation from uncompleted to completed tasks by the production of specifications. A fraction of the specifications are not acceptable and needs to be taken care

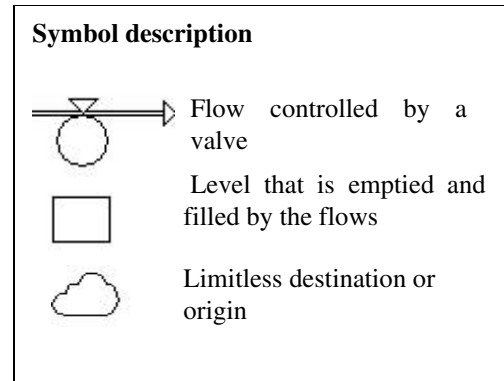
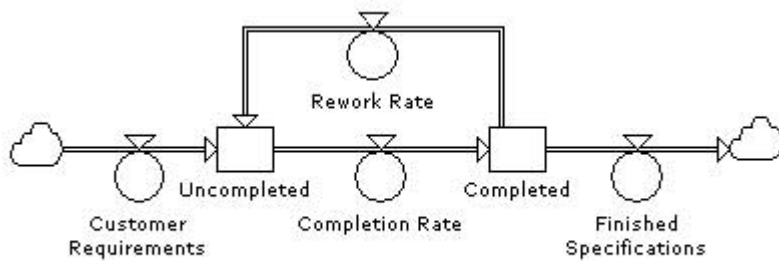


Figure 4. Basic model of the requirements phase

of in the rework loop, see Figure 4.

The test phase in the model is based on the same idea as the requirements phase and is built in a similar way. A flow of test cases is performed, a certain percentage of the functionality has to be corrected and retested, and the rest is supposed to be acceptable.

This basic model was built in the Powersim simulation tool [11] and further developed with help from the factors in the influence diagrams. Factors from the influence diagrams were added to the model in order to affect the levels and flows. The causal-loop diagrams were also considered during the development, to ensure that the model was adapted to systems thinking.

To avoid getting a too complex model, all of the factors in the influence diagrams were not included in the simulation model. Some factors were included indirectly in the parameters in the model. These can be extracted from the parameters and are thereby possible to affect from the user interface, for example the communication complexity which is included in the productivity. The construction was made step by step, by adding a few factors at a time and then running the simulation. The values of the parameters were taken from project documentation except one that was taken from [7], *Amount of new market requirements*. This parameter was not available in project documentation but the value from [7] is an average from several software projects and was considered to be valid also for this project. Some values were estimated by iteration and verified by discussions with concerned personnel at the organization. The verification of the simulation model was made through checking that the amount of code that is used as an input to the model is the same as the output amount of code. The verification also included comparing the time in the simulation to the time according to the project documentation to ensure that the estimations were correct.

The final model for the requirements phase is seen in Appendix A. The flows in Figure 4 is the base of the final model, which is then further developed. To get a measure of the quality of the specifications, another flow was

included, which counts the inadequate specifications. This measure affects the amount of defect code that is produced in the design and implementation phases which in turn affects the test phase. The design and implementation phases are in the simulation model modelled as a delay. A second flow is added to the basic model to terminate this phase and start the following phases.

The rest of the additions to the basic model can be described in four groups, where each group originates from the influence diagram.

- The first group, *Lines of code* and *Functionality*, describes the functionality of the code to be developed. This group controls the inflow to the phase.
- The second group is *Percentage*, *Effort* and *Duration*. The *Percentage* allocates a percentage of the planned total effort to the requirements phase and is controlled from the user interface. This makes it possible to study how the amount of resources in the requirements phase affects the lead-time and quality.
- The third group, *Productivity* and *Duration*, controls the completion rate of the specifications. The *Duration* also affects the amount of inadequate specifications because of the schedule pressure that might increase during the project's duration.
- The fourth group, *Amount of rework* and *Functionality*, decides how much of the specifications that needs to be reworked after the reviews.

Note that some factors are part of more than one group. This is because some factors affect more than one other factor.

4. Results from the simulation

The final model was simulated to show how a relocation of resources to the different process phases affects the quality of the software products and the lead-time of the project. This model included both the requirements phase and the test phase. The model was run several times with different values of the percentage of the planned project effort, spent on the requirements phase. The results are

given in precise figures but since there are a number of uncertainties they should be broadly interpreted. For example, the results are uncertain because of the difficulty in measuring the values of the included factors. It is the tendencies in the results that are important and not the exact figures.

The simulation runs indicate that the effort spent on the requirements phase has a noticeable effect on the lead-time of the project. The decrease in days, when increasing the effort in the requirements phase, arises from the increased specification accuracy. A more accurate specification facilitates the implementation and decreases the error generation and will result in a higher product quality from the start. This decreases the amount of necessary correction work and thereby shortens the time spent in the test phase. At a certain point the total lead-time will start to increase again because the time in the test phase stops decreasing while the time in the requirements phase continues to increase. The time in the test phase stops decreasing because there is always a certain amount of functionality that needs to be tested at a predetermined productivity. The number of days in Figure 5 is the total lead-time for the whole project.

In the same manner, the quality increases when increasing the effort in the requirements phase to a certain extent. The simulation runs indicate that the quality optimum appears in the same area as the lead-time optimum. The increase in quality originates from a higher specification accuracy, which is explained above. However, if too much effort is spent in the requirements phase, the quality will start to decrease again because there is less effort left for design, implementation and test tasks.

As a step in the verification of the results, they were compared to results in the software literature [7, 15]. This literature points at the same magnitude of effort in the requirements phase for a successful project as the simulation results.

To summarize, the simulations indicate that there is an optimum for both the quality and the lead-time. If the effort in the requirements phase is lower than the optimal value,

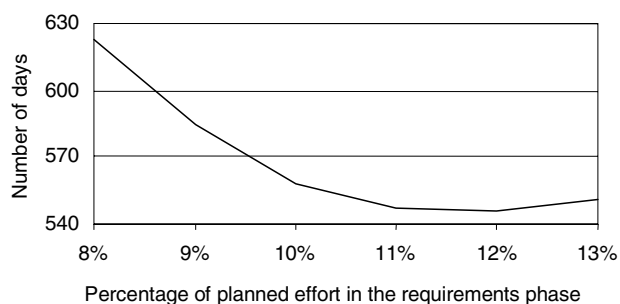


Figure 5. Simulation results for the total lead-time

increasing it towards the optimum will result in increased quality of the developed software and decreased lead-time.

5. Discussion

One result of this study is a simulation model that visualizes different relations in a software development process. A simulation of this kind can contribute to enhancing the systems thinking in an organization. Thereby it is easier for the members of the organization to understand the relationships between the quality factors in the process.

The results from this kind of simulation shall not be interpreted precisely since there, of course, are a number of uncertainties. It is the tendencies and the behaviour in the results that are important and by changing the parameters in the model it is possible to get a picture of how the process mechanisms interact. This is a simplified model of the reality and therefore there are a number of sources of uncertainty. The included factors might not be the ones that affect the model the most, the assumed relations between the factors might not be correct and the values of the factors can be incorrectly estimated. However, the results, that there is an optimum for the effort that is spent in the requirements phase, can be intuitively expected for many projects in software organizations.

A simulation of this kind can also be used to increase the motivation of the organization to work with quality issues and to increase the product quality early in the project.

One part of the knowledge gained from simulations is received in the model building process. The procedure to build the model forces the participants to communicate their mental models and to create a common image of the organization's direction.

To summarize, it seems to be feasible to build and use this kind of model for this kind of process. There are, however, a number of uncertainties which are important to take into account when the results are interpreted. This is a first model, based on one project, that needs to be further elaborated in order to obtain a model that can be applied on other projects. Thus, the model has not been empirically validated in real projects after it was developed. As far as the authors know, the model is not currently in use at Ericsson.

The impression after developing and getting feedback on the model is that it is uncertain whether most knowledge is gained by developing the model or using it. This is one of a number of issues that need to be further investigated in the area of software process simulation.

The model could either be used, for example by a project manager, by only changing the parameters, or it could be used by changing also the structure of the model, for example by adding or deleting factors and adding or deleting relationships between factors. It may be that users

of the models need to understand the internal structure of the model and not only the interface to it. This would limit the choice of modelling techniques, and it would for example mean that models with an internal design, that is not easy to understand for the users of the models, would not be suitable in all cases.

Acknowledgement

The authors would like to thank Wladyslaw Bolanowski and Susanne S. Nilsson at Ericsson Mobile Communication AB for all their help with this study. This work is partly funded by the Swedish Agency for Innovation Systems (VINNOVA) under grant for Centre for Applied Software Research at Lund University (LUCAS).

References

- 1 Abdel-Hamid, T., Madnick, S.E., *Software Project Dynamics: An Integrated Approach*, Prentice Hall, 1991
- 2 Andersson, C., Karlsson, L., "A System Dynamics Simulation Study of a Software Development Process", CODEN:LUT-EDX(TETS-5419)/1-83/(2001)&local 3, Department of Communication Systems, Lund Institute of Technology, 2001
- 3 Banks, J., Carson, J.S., Nelson B.L., *Discrete-Event System Simulation*, Prentice Hall, 1996
- 4 Burke, S., "Radical Improvements Require Radical Actions: Simulating a High-Maturity Software Organization", Technical Report CMU/SEI-96-TR-024, Software Engineering Institute, Carnegie Mellon University, Pittsburg, USA, 1996.
- 5 Fenton, N.E., Pfleeger, S., *Software Metrics: A Rigorous & Practical Approach*, International Thomson Computer Press, 1996
- 6 Höst, M., Regnell, B., Natt och Dag, J., Nedstam, J, Nyberg, C., "Exploring Bottlenecks in Market-Driven Requirements Management Processes with Discrete Event Simulation", Accepted for publication in *Journal of Systems and Software*, 2001.
- 7 Jones, T.C., *Estimating Software Cost*, McGraw-Hill, 1998
- 8 Kellner, M.I., Madachy, R.J., Raffo, D.M., "Software Process Simulation Modelling, Why? What? How?", *Journal of Systems and Software*, Vol. 46, No. 2-3, pp. 91-105, 1999.
- 9 Martin, R., Raffo, D., "A Model of the Software Development Process Using both Continuous and Discrete Models", *International Journal of Software Process Improvement and Practice*, 5:2/3, June/September, pp. 147-157, 2000.
- 10 Martin, R., Raffo, D., "Application of a Hybrid Process Simulation Model to a Software Development Project", proceedings of *PROSIM 2000*, July 12-14, London, UK
- 11 Powersim corporation, www.powersim.com, 010903
- 12 Rus, I., Collofello, J.S., "Assessing the Impact of Defect Reduction Practices on Quality, Cost and Schedule", proceedings of *PROSIM 2000*, July 12-14, London, UK
- 13 Senge, P.M., *The fifth discipline*, Random House Business Books, 1990
- 14 Sommerville, I., *Software Engineering*, Addison-Wesley, 1996
- 15 Stewart, R.D., Wyskida, R.M., Johannes, J.D., *Cost Estimator's Reference Manual*, John Wiley & Sons Inc, 1995
- 16 Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publisher, 2000

Appendix A

The simulation model of the requirements phase

