Aalto University
School of Electrical Engineering
Degree Programme in Automation and Control Engineering

Juho Salmi

Learning Simulation Game Development

Case: Simulation Game for EPCM Project Management Training

Master’s Thesis

Espoo, November 22, 2015

Supervisor:	Kai Zenger, Docent, D.Sc.
Instructor:	Sampsa Ruutu, M.Sc.

Aalto University
School of Electrical Engineering
Degree Programme in Automation and Control Engineering

ABSTRACT OF THE MASTER’S THESIS

Author: Juho Salmi

Title: Learning Simulation Game Development

Number of pages: 81
Date: 2015-11-22
Language: English

Professorship: Control Engineering
Code: AS-74

Supervisor: Kai Zenger, Docent, D.Sc.

Instructor: Sampsa Ruutu, M.Sc.

Abstract:

This thesis studies how learning simulation games should be developed and used. The empirical case study of the thesis is a development project of simulation game for EPCM (engineering, procurement and construction management) project management training. The theoretical framework of this thesis makes a synthesis of four different topics: 1) complex systems and system dynamics, 2) learning, 3) games, and 4) project management.

The outcome of this thesis is the following suggestion on how simulation games should be developed and used for training purposes: The game development should begin with defining the modelling problem, and then choosing the modelling paradigm and the tools for the model and the game development. The learning goals and the target group should be derived from the problem definition. Thereafter, the model and the game should be developed iteratively, preferably while being tested by the target group. The game sessions should include debriefings to strengthen learning.

Keywords: system dynamics, project management, simulation games, learning games, serious games

Aalto-yliopisto Sähkötekniikan korkeakoulu Automaatio- ja systeemitekniikan koulutusohjelma		DIPLOMITYÖN TIIVISTELMÄ	
Tekijä:	Juho Salmi		
Työn nimi:	Simulaatiopelien kehittäminen oppimistarkoituksiin		
Sivumäärä: 81	Päiväys: 2015-11-22	Julkaisukieli: Englanti	
Professuuri:	Systeemitekniikka	Professuurikoodi: AS-74	
Työn valvoja:	Kai Zenger, Dosentti, TkT		
Työn ohjaaja:	Sampsa Ruutu, DI		

Tiivistelmä:

Tämä diplomityö tutkii, kuinka kehittää ja hyödyntää simulaatiopelejä oppimistarkoituksiin. Diplomityön empiirisenä tutkimuksena kehitettiin simulaatiopeli EPCM-projektinhallinnan koulutukseen. Työn teoriaosio tekee synteesin neljästä aiheesta: 1) komplekseista järjestelmistä, 2) projektinhallinnasta, 3) peleistä ja 4) projektinhallinnasta.

Diplomityö esittää seuraavan suosituksen, kuinka kehittää ja käyttää simulaatiopelejä oppimistarkoituksiin: Pelikehityksen tulee alkaa mallinnusongelman määrittelyllä, jonka jälkeen tulee valita soveltuva mallinnusparadigma ja työkalut pelin kehitykseen. Oppimistavoitteet ja kohderyhmä tulee johtaa mallinnusongelmasta. Tämän jälkeen mallia ja peliä tulee kehittää iteratiivisesti, mieluiten testauttamalla sitä kohderyhmällä. Peli-istuntoihin tulee sisällyttää jälkipuinti oppimisen vahvistamiseksi.

Asiasanat: systeemidynamiikka, projektinhallinta, simulaatiopelit, oppimispelit, hyötypelit

Preface

Human is a complex adaptive system. Human is great at adapting to different environments and even better at changing the environment to meet the human needs. What makes human so adaptive, is its great learning abilities. Humankind has invented education to further improve learning and human adaptive properties. The better we humans adapt to the environment and change the environment to serve us, the better we thrive in this complex world. No wonder that education is thought to be the most powerful weapon.

My education has given me a lot but I have never been completely satisfied with my education. This is perhaps one reason I have grown passionate about education. I want to improve it. As an engineer, I am interested in using technology – such as games and simulations – to improve it. Due to the scalable nature of technology, applying more technology to education might provide us with weapons of mass education.

As this thesis illustrates, education is a difficult subject. That is why I can appreciate the effort my teachers have put in developing the education that I've received.

I thank Kai Zenger not only for being my thesis supervisor but also for being a great teacher in control engineering and being kind of a back bone for the degree programme in automation and control engineering. Furthermore, I thank all my teachers that have given me the mathematical, engineering and systems thinking tools to thrive as an engineer in this complex world.

I thank Sampsa Ruutu for being a great, proactive instructor for this thesis. It is difficult to describe how much you have helped me to crystallise my thoughts for my thesis and about system dynamics in general. Furthermore, I thank VTT for funding this thesis and everybody who were involved in the EPCM Game project or otherwise helped me with the project and my thesis: VTT system dynamicists, VTT Systems Modelling and Simulation team, MODRIO project manager Tommi Karhela, Semantum software engineers, people at the partner company, and games researcher Otso Hannula.

Even though my education has given me a lot, student activities had probably a bigger impact on what kind of a person I have become. Student activities gave me an opportunity to learn how social systems, such as organisations, behave and how to help them thrive. More important, student activities gave me an opportunity to meet amazing people – many of whom I nowadays call my friends. Especially people from AS, AYY and VT have become important to me, not to forget the national teekkari activists, and the people from AaltoES and Aalto COMPLEX. I'm thankful for being able to grow as a person with you.

Thank you, dynamic trio members Tommi & Matti, combined with Martti & Panu; teekkari mentor Mutru; Satu, Rosa & Tommi with his amazing wife Sonja; Cellari dwellers, especially Saana, Laura, Alex, Ville, Erno, Miska; ASH'11 Iiris, Angra, Janna, Elmis, Tuomas, Taffis, sooda, Ian, Joonas & Peter; AYYH'12 Voitto, Noora, Pilvi, Jani, Lauri, Eetu, Anna, Kati & Roope; specialists Lotta, Janne & Juha; engineering education mastermind JP; every single vapaateekkari, especially Hype, Atte, Joni, Käkä, Niko, Hami & Max; and then of course Lotta.

Thank you.

Espoo, November 22, 2015

Juho Salmi

Libertas. Techologitas. Naturalitas.

Table of Contents

1. Introduction

“For the things we have to learn before we can do, we learn by doing.” – Aristotle

There are different ways to learn: studying theory, following an example, reasoning and so on. All means of learning are important and useful. Learning by doing is perhaps the most powerful way of learning. It occurs when trying out different decisions and learning from their outcomes: decisions with good outcomes should be favoured over bad ones. The downside is that the bad outcomes i.e. mistakes cannot be avoided in the learning process.

Making mistakes can, of course, be costly: a mistake by a corporate leader may cause the company to go bankrupt. People may not be willing to take risks and an individual might be satisfied once they discover a single good-enough way of doing things. They might then stick to that and new approaches won’t even be tried. Even if one has the courage to take the action and try out new approaches, it might take years to see the outcome. After years it might be difficult to say which ones were the decisions that led to this outcome.

Virtual worlds – simulations and games – provide a safe environment for testing out different approaches. By using virtual worlds one can see the outcome of different decisions instantaneously. Mistakes in the virtual worlds do not cost anything so there is no fear of failure. Using simulations costs merely the effort to build the simulation model and trying out different approaches with it. Virtual worlds can boost learning and, therefore, help making better decisions. Of course, creating good virtual worlds is neither easy nor free. This thesis studies how to develop one category of virtual worlds: simulation games.

This thesis examines how learning simulation games should be developed and used. Specifically: 1) what is needed from the development process to produce a pedagogically effective game, 2) what aspects should be considered, and 3) what is the pedagogically most effective way of using such games for training purposes?

As an empirical case study for this thesis a simulation game for EPCM (engineering, procurement and construction management) project management training, called EPCM Game, was developed and it was tested with project manager trainees and students. The theoretical framework of this thesis makes a synthesis of four different topics: 1. complex systems and system dynamics, 2. learning, 3. games, and 4. EPCM project management. Some of the novelty value in this thesis lies in applying an extensive set of learning and gaming theories.

The background of this thesis and EPCM Game is that I worked as a research trainee at VTT Technical Research Centre of Finland (Teknologian tutkimuskeskus VTT Oy). The game development contributed to three work packages in MODRIO (model-driven physical systems operation) research and development project (ITEA, 2014). The goal for the work package 6.5: Cloud Computing Service for Simulation-Based Project Management was to develop 1) Simantics System Dynamics and 2) Simupedia. These tools were used to develop EPCM game. My role was to report bugs and request features needed in the game development. This also included me working as the product owner for Simantics System Dynamics.

Simantics System Dynamics is a system dynamics modelling and simulation software built on Eclipse-based Simantics user interface platform (THTH, 2014). Simantics System Dynamics uses Modelica systems modelling language (Modelica Association, 2012). Both, Simantics and Simantics System Dynamics, are being developed under VTT.

“Simupedia is a web service for collaborative planning and communication of model based assessments.” (Semantum, 2014). Simupedia is being developed by Semantum Oy. Simupedia was used to develop the web-based user interface for the game and the user interface was programmed by Semantum Oy software engineers.

The game itself was developed under work package 8.9: Project Management, and Observations and Measurements. The game was developed in cooperation with a Finnish EPCM company with offices in multiple countries to meet their needs for project management training. The project management system dynamics model was also a contribution to work package 7.6: System Dynamics Libraries for Project Management.

2. Complex Systems and System Dynamics

This thesis discusses topics such as learning and organisations. These topics can be viewed as complex systems. There are many definitions of a system (Backlund, 2000) and this thesis uses the definition by Sterman (2000): “a system is a set of interacting components that forms an integrated whole”. As for complexity, there are two kinds: combinatorial and dynamic.

Combinatorial complexity (also known as detail complexity) arises from the number of the interacting components in the system. For example scheduling airline’s flights and their crews is a combinatorial complex problem because there is such a big number of different combinations to choose from. (Sterman, 2000)

This thesis discusses mainly dynamic complexity which exists in dynamic systems. Dynamic system is a system where the states are dependent on the states of the previous time instance. Dynamic complexity is not something that can be perceived just by examining the current state of the system because the dynamic complexity arises from the interactions of the system components over time. Dynamic complexity can be viewed as a system property related to the amount of emergent behaviour in the system. (Sterman, 2000)

Emergence is a behaviour which arises through the interactions among the system components that do not exhibit such properties themselves. Therefore, emergence is something that cannot be explained by studying system components separately, and therefore, a complex system must be studied as a whole. (Bonabeau & Dessalles, 1997; Standish, 2008) The study of complex systems requires both analysis of breaking the complex system into interacting components and evaluation of making a synthesis of the interactions as a whole, holistically. Feedbacks, nonlinearities and history-dependency are some of the main system properties responsible for inducing emergent behaviour (Sterman, 2000).

A model is a representation of a real world system. Thus, modelling is a process of finding such representation. This thesis discusses a lot about modelling complex systems. However, modelling itself is not an interesting goal because typically systems cannot be modelled as such. This is because 1) one can endlessly add environmental factors to the model and 2) the model resolution can be increased endlessly by breaking factors into their factors.

Statistician George E. P. Box once said that all models are wrong but some are useful. Since it is not practically possible to model a system as such we have to make the model or the process of modelling useful. Sterman (2000) suggests that one should model a problem instead of modelling a system. A problem gives a focus for the modelling process and helps keeping the model size reasonable. If a modelling process reveals new aspects of a system or solves a problem then the more or less wrong model becomes useful.

The purpose of the model determines the modelling approach (Sterman, 2000). Agent-based modelling paradigm can be useful when the purpose is in studying interactions of autonomous agents (Janssen, 2005) and discrete event simulation can be useful for analysing discrete processes (Robinson, 2014). We might be interested in the quantitative behaviour of certain variables – whether the growth is 1, 2 or 3 %, for example – or we might be interested in the qualitative behaviour – whether the growth is linear, exponential or any growth at all. The modelling problems in this thesis are related to qualitative behaviour and system dynamics modelling paradigm is used to deal with the dynamic complexity in the socio-technical systems of project management.

This thesis does not conduct any further introduction to system dynamics but the concepts are explained so that anybody should be able to get the big picture around them. However, the basics of system dynamics is required to understand the simulation model in detail.

3. Human as an Adaptive System

A system is adaptive when its properties change over time to satisfy its goals in the changing environment (Sterman, 2000). Learning is one form of adaptiveness of an organism. Schacter et al. (2009) define learning as an organism's act of acquiring new or modifying and reinforcing knowledge, skills, behaviour, values or preferences.

This section discusses human as an adaptive system – how do humans learn and make decisions? To answer these questions this section introduces central learning and decision-making theories relevant for developing learning simulation games.

3.1. Bloom's Taxonomy

Bloom et al. (1956) introduced a classification i.e. taxonomy as a measurement tool for learning objectives. Bloom's taxonomy introduces three domains of learning: cognitive, affective and psychomotor. This thesis concentrates on the cognitive domain and uses the revised version of Bloom's taxonomy developed by Anderson & Krathwohl (2001). The revised Bloom's taxonomy introduces two dimensions i.e. a matrix of learning objectives in the cognitive domain: cognitive process and knowledge.

The cognitive process dimension describes cognitive complexity of learning. There are six cumulative levels of cognitive complexity. Cumulative means that one has to have learned the lower level objectives in order to learn the higher level objectives. The six levels, starting from the lowest, are:

1. **Remembering**: Retrieving, recognizing, and recalling relevant knowledge from long-term memory.
2. **Understanding**: Constructing meaning from oral, written, and graphic messages through interpreting, exemplifying, classifying, summarizing, inferring, comparing, and explaining.
3. **Applying**: Carrying out or using a procedure through executing, or implementing.
4. **Analysing**: Breaking material into constituent parts, determining how the parts relate to one another and to overall structure or purpose through differentiating, organizing, and attributing.
5. **Evaluating**: Making judgements based on criteria and standards through checking and critiquing.

6. **Creating**: Putting elements together to form a coherent or functional whole; reorganizing elements into a new pattern or structure through generating, planning, or producing.

(Anderson & Krathwohl, 2001)

The knowledge dimension describes the kind of the knowledge of a learning objective. The taxonomy defines four kinds of knowledge:

A. **Factual**: Knowledge of facts, terminology, details and elements.
B. **Conceptual**: Knowledge of interrelationships among the basic elements within a lager structure that enable them to function together.
C. **Procedural**: Knowledge of how to do something: methods of inquiry, and criteria for using skills, algorithms, techniques, and methods.
D. **Metacognitive**: Knowledge of cognition and oneself.

(Anderson & Krathwohl, 2001)

An educator can use Bloom's taxonomy as a tool for designing education. First the educator can assess what knowledge the students already have and on which level of cognitive complexity. Then the educator can choose the learning goals. The educator has to design the education to fill the gap between the current level and the goal level of knowledge and cognitive complexity. The empirical framework of this thesis will discuss using Bloom's taxonomy for setting the learning goals for the simulation game for project management training.

Roberts (1978) suggests that the higher levels of cognitive complexity – analysing, evaluating and creating – are congruent with system dynamics. Modelling is a creative process: One cannot build a system dynamics model without first breaking the system into its components and then evaluating them and then forming a functioning whole by putting the components together to form a system dynamics model. Roberts also shows that using system dynamics in education can produce better learning outcomes on all levels of cognitive complexity but especially on the highest levels. Roberts (1978) and Forrester (1992) criticise typical education for focusing on facts and claim that without studying the underlying structures around these facts will be quickly forgotten. Thus, learning on higher levels of cognitive complexity helps retaining what has been learned on the lower levels.

Once a modeller has modelled a system, the modeller should know the system, its components, their interactions, and the overall dynamics well – providing that the model is validated and verified. But how can this knowledge be conveyed to others? Simulations and games can be used for this purpose, as this thesis will demonstrate.

3.2. Zone of Proximal Development and Instructional Scaffolding

Zone of proximal development (ZPD) is a concept introduced by L. S. Vygotsky (1978). Vygotsky defines ZPD as “the distance between the actual development level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or collaboration of more capable peers” (Vygotsky, 1978, p. 86). The idea around ZPD is that there are three types of problems for learners: 1) ones that the learner can currently solve without external help, 2) ones that the learner can currently solve with guidance, and 3) ones the learner cannot currently solve. The zone of proximal development is the intermediate type of problems, ones the learner can solve with guidance. Vygotsky believed that the role of an educator is to guide the learner with the problems in the zone of proximal development until they have learned to solve the problem without guidance. The educator then can move to guide more difficult assignments that the learner could not previously do even with guidance. Vygotsky also raises an idea that two peers are usually able to solve more difficult problems than the individuals by themselves. (Chaiklin, 2003)

This thesis takes the Murray’s & Arryo’s (2002) approach to ZPD which is less strict than Vygotsky’s original approach. Instead of discussing whether it is possible for the learner to solve a problem at all they discuss whether it is efficient and effective for learning. Typically, all problems are solvable given enough time. However, solving a too difficult problem takes too long and is inefficient for learning. It would be more efficient if the learner first solved easier problems to gain enough understanding to solve the previously too difficult problem. On the other hand, solving too easy problems is not only boring but also inefficient for learning. ZPD is the sweet spot, the optimal difficulty of problems for the learner on the current skill level.

Figure 1 is Murray's & Arryo's (2002) illustration of the idea around zone of proximal development in student skill level and content difficulty coordinates. The grey area illustrates the zone of proximal development. Above the grey area are problems so difficult that solving them is inefficient for learning. Under the grey area are problems so easy that solving them is inefficient for learning. The curve is an example educational setting. Idea is that the time interval between the dots on the curve is always the same and the student skill level distance between the dots tells about the efficiency of learning. Efficiency of learning is of course highest in the zone of proximal development. In the example curve the learner first grows bored of the lack of problem difficulty, 1) then the learner is confused of the too high problem difficulty 2) and finally the learner is in the zone of proximal development and the learning is most efficient 3). There, of course, is no discrete edge separating the ZPD when crossed the learning discontinuously drop. Rather, the efficiency of learning is a continuous function of student skill level and content difficulty. ZPD just illustrates the area when learning efficiency crosses certain threshold.

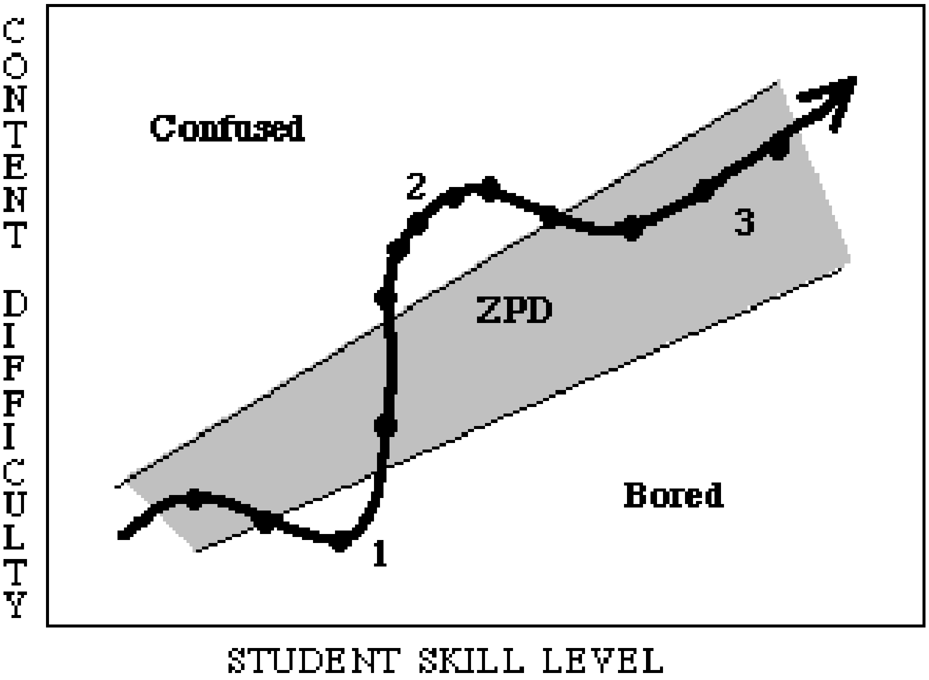


Figure 1. Illustration of zone of proximal development. (Murray & Arroyo, 2002)

Instructional scaffolding is closely related to ZPD. Scaffolding is the support given during the learning process which is the same as the guidance needed in the Vygotsky's zone of proximal development. Scaffolding helps the learner solve problems that would otherwise take too long to solve. The idea is that building scaffolds for education boosts learning. Help of peers can also be considered as scaffolding. (Murray & Arroyo, 2002)

Saye & Brush (2002) introduce a concept that there are two types of scaffolding: soft and hard. Hard scaffolding is instructions planned in advance, "hard-coded" instructions. For example hints given for learning assignments or instructional pop-ups in a computer game tutorial are hard scaffolding. Soft scaffolding, in turn, is responsive. In soft scaffolding the learner's current skill level is first determined and scaffolding is then built along that information. For example, a teacher can adapt teaching methods along the knowledge of the skill level of the students. The teacher can base this knowledge on what has been taught to the students and build the knowledge by testing the knowledge of the students.

I rephrase this idea in the system dynamics point of view. What really separates hard and soft scaffolding from each other is feedback. In soft scaffolding the instructor – human or other machine – adapts instructions it gives based on the feedback it gets from the learner.

ZPD and scaffolding should be considered when designing games and education. When a player starts a game for the first time some scaffolding is most likely needed before the player gets a grasp on the idea how to play the game. The player could give inputs to the game to indicate whether they are rookie or experienced player and the game could adapt the instructions accordingly. Experienced players could skip tutorials whereas rookie players might want to take it slow and learn one trick at the time with the instructions in the game. There could also be options for setting the difficulty so that any player can be in their zone of proximal development.

3.3. Mental and Feedback Concepts in Learning and Decision-Making

From simulation model point of view, decision-making is equivalent to choosing a value for a parameter of a system. This section introduces different approaches to decision-making and mental concepts around them.

Sterman (2000) states that there are two kinds of worldviews around decision-making: event-oriented and feedback-oriented. Event-oriented view approaches decision-making as a single event where only direct impacts need to be considered. Figure 2 illustrates event-oriented decision-making where we have a goal and the current system state. The further the goal is from the current system state, the bigger the problem. Based on the size of the problem a decision is made that lead to certain results. For example, if a project needs 20 % more work input then order people to work 20 % longer days.

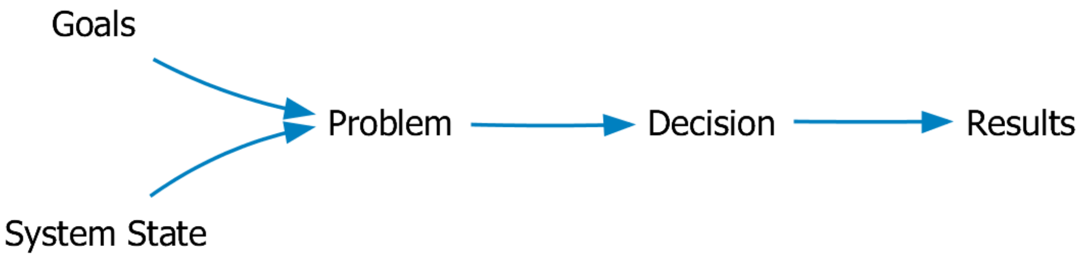


Figure 2. Event-oriented worldview (Sterman, 2000)

This sort of approach does not take into account possible side effects and long-term effects of the decision. In short term the work input, indeed, increases by 20 % but working long days exhausts people in long-term which lowers their productivity and increases errors. The project might end up being completed even later than if nothing had been changed.

Figure 3 illustrates feedback-oriented worldview. Feedback-oriented view takes into account that as decisions alter the system state the decisions have to alter too. Decision-making itself is thus a negative feedback process that tries to get the system state converging to the goal level. Furthermore, systems themselves have typically feedbacks. Feedback-oriented view takes side effects and long-term effects into account and emphasises the understanding of feedback loops in systems.

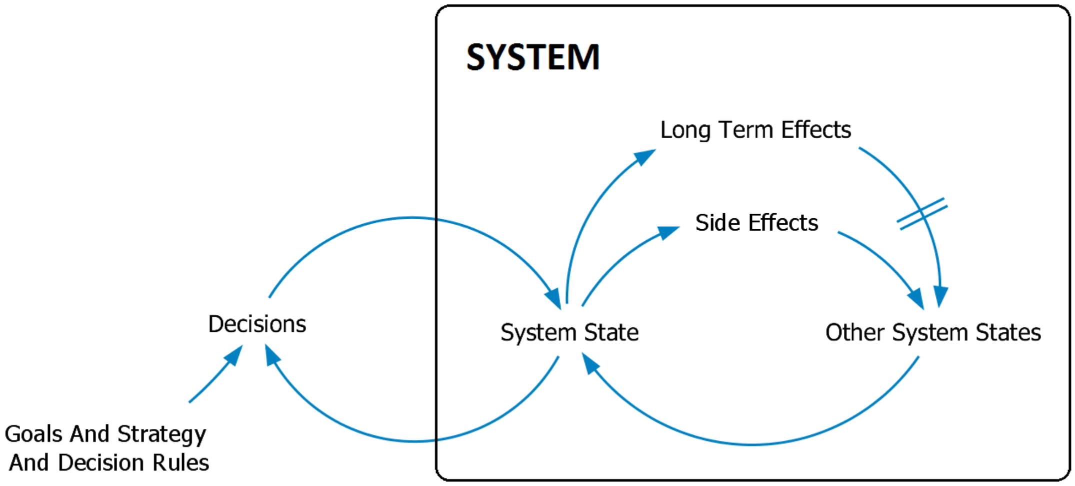


Figure 3. Feedback-oriented worldview[1]

Systems thinking is a concept very discussed and popularised in the system dynamics community. Sterman (2000) defines systems thinking to be an ability to see the world as a complex system, in which we understand that you just can't do one thing and that everything is connected to everything else. As mentioned before, feedback loops are one of the major sources of dynamic complexity in systems. That's why feedback-oriented worldview is essential for effective systems thinking.

Feedback-oriented view is congruent with the concept of learning as a feedback process. Figure 4 illustrates the concept of single-loop learning. Decisions have an impact to the system. These impacts can be perceived which gives information feedback from the system. New aspects are learned of the system. New decisions are then made based on the new knowledge of the system and the goals, strategy and decision rules. (Sterman, 1994)

---

[1] The two lines crossing the causal arrow from Long Term Effects denotes delay (or inertia) in the causality.

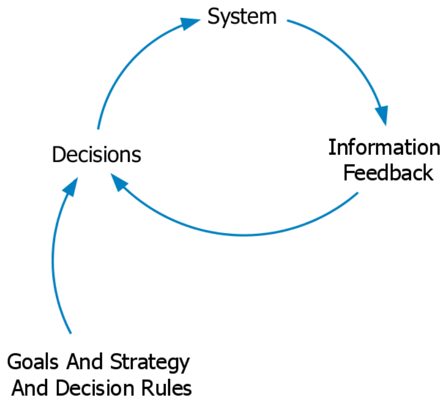


Figure 4. Single-loop learning

However, single-loop learning is very constrained view of learning. Goals, strategy and decision rules remain as an exogenous variable. It is obvious that they, too, change over time. Double-loop learning includes this idea. Figure 5 illustrates the concept of double-loop learning. In double-loop learning the decision-maker's mental models of the system affect how information feedback is perceived from the system. The perceived information feedback updates mental models which then update goals, strategy and decision rules. (Sterman, 2000) To rephrase this: the second loop affects how we perceive the world around us and is responsible for changing the decision-making in the higher level.

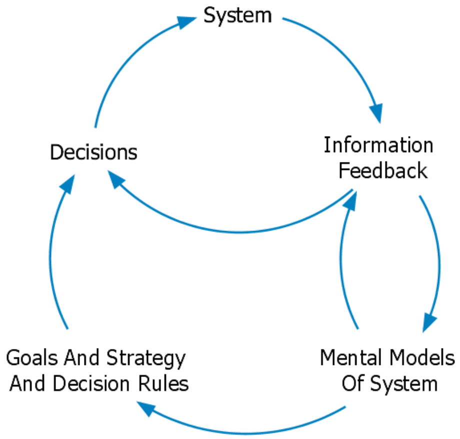


Figure 5. Double-loop learning

Considering double-loop learning, having good mental models is the key to good decision-making. Based on thorough literature analysis Doyle & Ford (1998, p. 19) define that "a mental model of a dynamic system is a relatively enduring and accessible, but limited, internal conceptual representation of an external system whose structure maintains the perceived structure of that system.” This thesis uses this definition. What is relevant for this thesis is that a mental model is how a person perceives the system. As mentioned before, all models are wrong but some are useful. This goes also for mental models. If a person’s mental model resembles event-oriented view or otherwise is not congruent with the real system the person will most likely misinterpret the system state and make bad decisions. The person can, of course, better the mental model by interacting with or otherwise studying the system.

The concept of mental models and double-loop learning can be reflected with Bloom’s taxonomy, ZPD and scaffolding. If a person has too little knowledge of a system then making good decisions is difficult and out of zone of proximal development. With good instructions, however, a person can be scaffolded to the zone of proximal development. Then the person can start building the mental model by learning first to memorizing the system components then understanding how they work and so on climb up the cognitive complexity in Bloom’s taxonomy.

In the higher levels of cognitive complexity the person might start making misinterpretations of the system if the person lacks systems thinking skills. Sterman (2000) describes systems thinking as “the ability to see the world as a complex system, in which we understand that ‘you can’t just do one thing’ and that ‘everything is connected to everything else’”. Systems thinking could be viewed as an ability to form feedback-oriented mental models i.e. models that holistically take feedbacks, delays and nonlinearities into account.

According to Sterman (1994) there are three problems in learning from interacting with the system. The first two problems arise from the time delays of the system. Firstly, if time delay is long then getting the information feedback is slow. Therefore, learning is slow. Secondly, with long time delays it might even be unclear what the role of the made decision eventually was. The third problem is the cost of bad decisions. Finding out better approaches may require trying out different decisions. However, new approaches may result in bad, expensive outcomes. For example a bad decision of a corporate leader may lead to a bankruptcy of the company.

Sterman (1994) suggests that these problems can be overcome by using virtual worlds i.e. simulation and games. In virtual worlds one can try out what kind of results different decisions give. The experiment does not cost anything and one can see the results immediately. Figure 6 illustrates double-loop learning with a virtual world: the information feedback from the real system is delayed but the information feedback from the virtual system is instant. However, building a good virtual world is neither easy nor free. The virtual world must resemble the real world well enough. Otherwise one might learn incorrect dynamic insights which might make coping in the real world even more difficult.

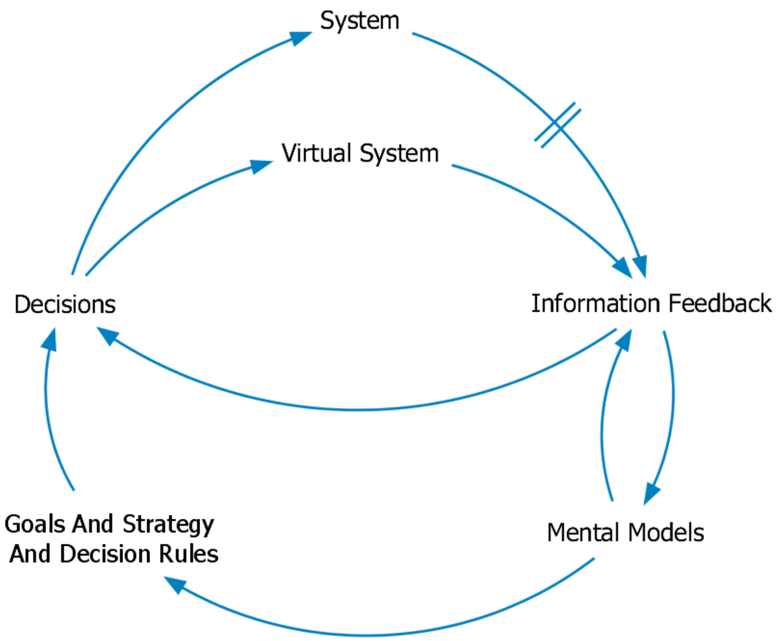


Figure 6. Double-loop learning with virtual worlds

4. Learning Simulation Games

4.1. Games

There are different definitions for a game. For example Abt (1987, p. 6) defines a game as an activity among two or more independent decision-makers seeking to achieve their objectives in some limiting context. However, it could be interpreted that this definition excludes Solitaire and other single-player games without a computer player, for instance. As for Caillois (1961) defines a play as an enjoyable, voluntary and unproductive activity separate from the real world involving uncertainty and rules. Van Daalen et al. (2014, pp. 2–3) and Garris et al. (2002, pp. 442–443) discuss whether there can be an exact definition of a game that includes everything we consider being a game and excludes everything else.

Since it is difficult to give an exact definition for a game, this thesis uses the following definition. Garris et al. (2002) make a literature-based synthesis of different definitions and introduce six dimensions i.e. characteristics that define a game. This is a fuzzy definition: the more of each characteristic there is the more it is a game. The following list are the characteristics with description of how each characteristic is implemented in EPCM Game.

1. Fantasy

Fantasy is an environment that evokes mental images of real life situations but do not exist. In fantasy the player adopts a role of fictional person, be it based on a real person or not (Garris et al., 2002). There are two types of fantasy: exogenous and endogenous. Exogenous fantasy is overlaid on learning context. For example a dragon slayer adventure can be overlaid on a mathematics game. Endogenous fantasy is related to the learning content. For example project management could be learned by playing the role of a project manager in a project management game. (Garris et al., 2002; Rieber, 1996)

EPCM Game has endogenous fantasy as the player takes the role of a project manager in an EPCM project.

2. Rules

When playing a game the constraints of the real world are set aside and the game set of rules is introduced. These rules are the structure of the game. (Caillois, 1961; Garris et al., 2002)

The simulation model lay the rules for EPCM Game and the player is constrained to being able to affect only some of the simulation model variables.

3. Stimuli

The game reality can offer sensations and perceptions that do not exist in the real world. (Caillois, 1961; Garris et al., 2002)

EPCM Game offers new perspectives to project management by letting the player to try out different approaches and seeing their results in an instance.

4. Challenge

Game should introduce challenge and meaningful goals to the player. Individuals desire an optimal level of challenge. (Garris et al., 2002)

The goal of EPCM Game is to get as profitable project as possible while having a happy client and without burning out the workforce. The player can choose to play either the easy or the difficult version of the game.

5. Mystery

The game should seek for optimal level of informational complexity. There should be something to be found, novel sensations or knowledge that makes the player curious. (Garris et al., 2002) Note that especially challenge and mystery are strongly related to zone of proximal development: a good game should provide challenges and information complexity inside the zone of proximal development and the mystery.

Not all the dynamics and the variables are shown to the player in EPCM Game, nor can the player know for sure what kind of effects their decisions cause.

6. Control

The player has to be able to make decisions and thus control some part of the game. (Garris et al., 2002)

The main decision variable in EPCM Game is resourcing i.e. workforce allocation. In the difficult version the player can also make decisions on how to work with the client.

4.2. Simulations and Simulation Games

Banks (2000) defines simulation as an imitation of a real-world system over time. Simulation can be anything from a role play of a negotiation or a mechanical horse ride to a system dynamics simulation of a project. Garris et al. (2002, p. 443) discuss that games and simulations are very similar because games typically have a set of rules that can be interpreted as a simulation model. The main thing that separates a simulation from a game is that the main purpose of a game is not to imitate a real-world system, unless it is a simulation game, and in that case it is both: a simulation and a game.

van Daalen et al. (2014) define that a simulation game is an interactive simulation with game characteristics. They also define that an interactive simulation is a simulation where during the simulation the user gets feedback from the simulation and gets to alter parameters of the simulation model i.e. make decisions. Since decision-making is a game characteristic all interactive simulations are somewhat games. Some authors (e.g. Kopainsky & Sawicka 2011; Maier & Größler 2000) even consider all interactive simulations being simulation games. Adding more game characteristics to an interactive simulation makes it more of a game.

While working on the simulation game project and this thesis it was noticed that some consider roleplaying learning games being simulation games. Some had played business roleplaying games that had been referred as simulation games and roleplaying game was the first association they had of a project management simulation game. Such roleplaying games are built on a pre-determined storyline with no other rules but to empathise with the given role. Such games can, indeed, imitate a real-world system over time but this is up to the mental models of the players and the game master. Additionally, roleplaying characteristics can be implemented in simulation games. However, games without a proper simulation model should not be considered being simulation games. As Frasca (2003) puts it “the potential of games is not to tell a story but to simulate: to create an environment for experimentation”.

4.3. Game-Based Learning

This section discusses learning games, a subcategory of serious games. Many use serious games as a synonym for learning games (van Daalen et al., 2014). This thesis, however, defines a serious game to be any game which has a useful primary purpose other than entertainment.

Typically serious games are learning games but for instance games that are used for human-based computation can be considered serious games (e.g. von Ahn & Dabbish 2004; Chrons & Sundell 2011; Savage 2012). This thesis defines learning games to be games whose primary purpose is learning. Therefore, learning games are a subcategory of serious games.

It is noteworthy to mention that regular entertainment games can also be used for learning purposes. For instance Civilization game series has been used to teach history and power politics (Squire & Barab, 2004; Squire, Giovanetto, Devane, & Durga, 2005), and Minecraft has been used to teach different topics from ecology to computer science (Ekaputra, Lim, & Eng, 2013).

There seems to be a lot of potential in game-based learning. Garris et al (2002) suggest that interactive technologies such as games create learning environments that involve learner in problem solving which enables a shift from learning by listening to learning by doing, from teacher-centred didactic to learner-centred model and from recalling information to being able to find and use information. Simons (1990) suggests that if video games can be transformed so that their users learn, a great many people may come to understand and control dynamic systems.

Most of the definitions of a game included the idea that one of the essential parts of a game is decision-making. Harteveld & Sutherland (2014) even argue that decision-making is the core of what games are. Decision-making is especially important in the learning games context when considering the feedback learning concept. The report of Federation of American Scientists (2006) supports this idea as well. The report suggests that games could be especially effective when teaching higher-order skills and decision-making since games provide an environment where decisions are made all the time. The same is not possible in traditional learning environments.

Garris et al. (2002) introduce Input-Process-Outcome Game Model (Figure 7). The model emphasizes the idea that 1) people learn from active engagement with the environment and 2) this experience coupled with instructional support of debriefing i.e. scaffolding can provide an effective learning environment. The input of the model presents the game itself – the instructional content and the game characteristics – which the player will be exposed to. The process presents the game session. The game cycle in the process of the model is analogous to single-loop learning: 1) The player gets information feedback from the system which the player interprets and judges (System Feedback & User Judgement). 2) The player makes decisions (User Behavior) based on the judgements. 3) These decisions affect the system behaviour. After a game session a good debriefing is needed for the best learning outcome. This can be interpreted analogous to the mental model update in double-loop learning.

Single-loop learning in the game cycle means that the player just learns to play the game. The fear is that the player merely learns how to play without learning to understand the underlying system. Debriefing is the instructional scaffolding making sure that the player not only learns to play the game but also learns the dynamic insights of the simulation model. Without debriefing it would be a lot less likely that the player would learn anything that they would be able to apply in the real life.

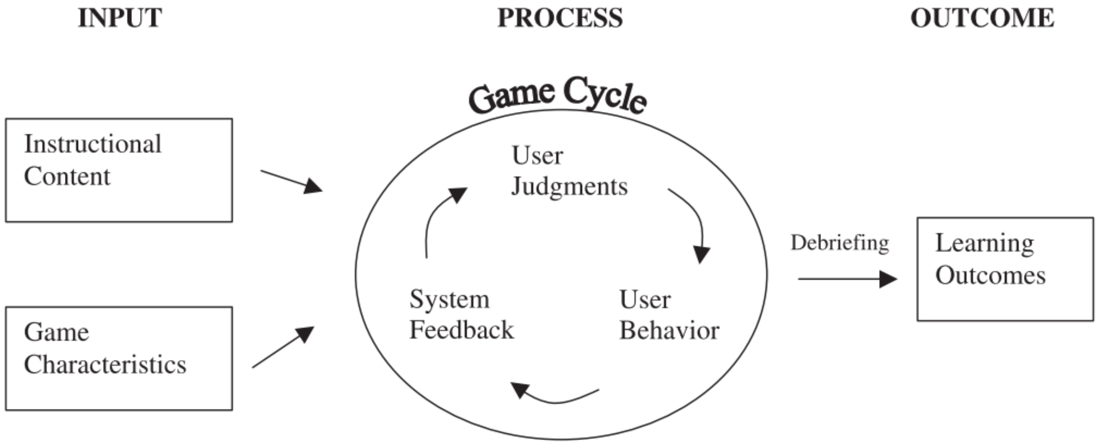


Figure 7. Input-Process-Outcome Game Model (Garris et al., 2002)

4.4. System Dynamics –Based Learning Games

System dynamics –based learning games have existed almost as long as the field itself. The system dynamics community typically calls such games as management flight simulators (van Daalen et al., 2014). Andersen et al. (1990) and van Daalen et al. (2014) discuss that the objective of such games is to teach dynamic insights. Andersen et al. (1990) define dynamic insights being a nuggets of systems thinking. The idea is that systems thinking is a whole approach of thinking whereas dynamic insight means understanding the complex dynamics of a certain system. Having dynamic insights does not imply that one is a systems thinker. However, systems thinking approach can create dynamic insights that can be taught by using simulation games for instance. Learning dynamic insights changes one’s mental models, and goals, strategies and decision rules according to the insight.

When modellers model systems they gain profound dynamic insights over the systems. One of the most profound questions in this thesis is how to convey these insights to others. One way is to build a simulation game around the built simulation model and use that for training. This is not simple. Developing a successful learning simulation game requires expertise over modelling, game development, learning and the domain being taught. Furthermore, Andersen et al. (1990) argue that even if one gained dynamic insights over the modelling process it does not automatically mean that the model would be suitable for conveying these dynamic insights to others.

Sterman (1994) argues that the most effective learning of dynamic insights occurs when people model system problems themselves. When the thoughts in this section are analysed against Bloom’s taxonomy, it is obvious that in order to reach the higher levels of cognitive complexity one has to go as far as analysis and evaluation of the real system which are included in a modelling process. By using a simulation game one can at best learn to apply their knowledge since simulation game training sessions won’t typically provide an environment for analysis and evaluation of the real system.

5. EPCM Game

This section introduces EPCM Game: its development process, simulation model, learning goals and game design. Furthermore, this section describes how the game was used for training, what was the learning outcome and what kind of feedback the game received. This section also compares EPCM game to other project management flight simulators.

As mentioned earlier: one should not model a system as such, one should model a problem. Since a game for training purposes was being developed, the primary purpose of the game was to teach. And since teaching is the primary purpose of the game, the modelling problems should be coupled with the learning goals since there is no point modelling something that we have no intension to teach.

However, setting the learning goals was tricky. The initial requests for the game were really vague: “we want a game we can use to train our project managers”. This sort of request is not a proper modelling problem, and blindly following this request might be more or less modelling system as such. That’s why it was necessary to get to know the environment and extract problems that could be fixed by training project managers. Therefore, at the beginning of the project, five EPCM project experts with project management experience were interviewed at the partner company organisation. The interviews were scheduled over three months.

Even though there was no proper modelling problem, the model development started from the day one by using the established system dynamics project management literature and previous studies conducted in same environment. The model evolved and got new directions as more information was received from the partner company. The next section surveys the established approach to system dynamics modelling in project management, introduces the discoveries that were made at the partner company and describes how these was applied in the EPCM Game development.

Note that the model introduced in this section is a simplified description of the EPCM Game simulation model with simplified equations and less variables. The complete model has over 100 variables with dimensions up to four. Therefore, in the simplified version, all variables and equations are simplified to one dimension, and variables unnecessary for understanding the dynamics are omitted. Appendices B–D are the complete lists of variables, equations, parameters and functions used in the simulation model.

5.1. Project Dynamics

Project Management Institute (2000) defines that organisations perform work which can either be done in projects or in operations. Both are 1) performed by people, 2) constrained by limited human and financial resources, and 3) planned, executed and controlled. Operations are ongoing and repetitive whereas projects are unique and have defined start and end dates. Project Management Institute (2000) defines project management to be "the application of knowledge, skills, tools, and techniques to project activities to meet project requirements". This work typically involves 1) competing demands for: scope, time, cost, risk and quality, 2) stakeholders with differing needs and expectations, and 3) identified requirements. These aspects are included in EPCS Game as well.

Most project management models in system dynamics are built around the idea of rework cycle (Lyneis, Cooper, & Els, 2001; Lyneis & Ford, 2007), an idea originally introduced by Cooper (1980). Figure 8 illustrates the base structure of the rework cycle. As mentioned earlier, projects have identified requirements. In the rework cycle, they are modelled as a stock of *Original Work To Do*. The *Work* flow empties the *Original Work To Do* –stock to the *Work Done* –stock (Eq. 1–2).

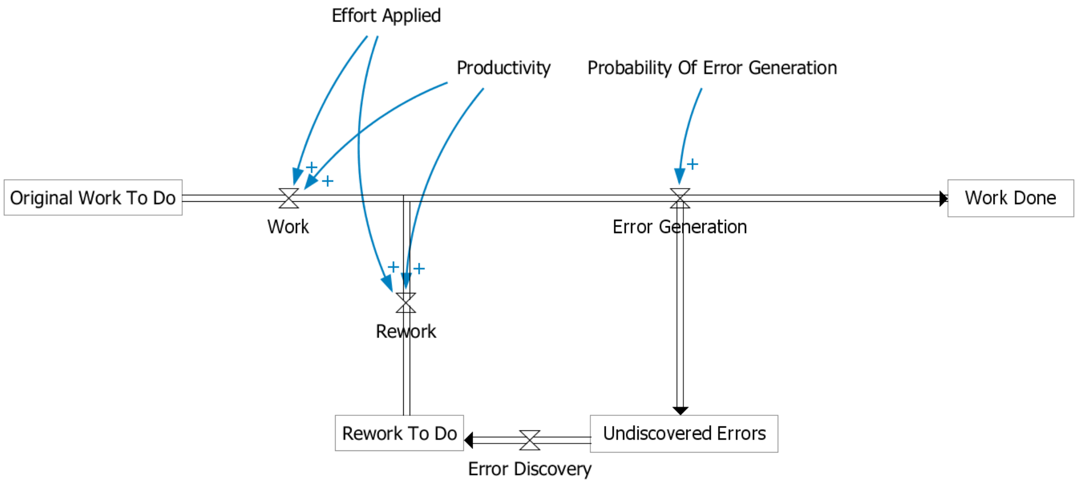


Figure 8. Rework cycle 1/3: basics

This is where the cycle begins. There is a certain *Probability Of Error Generation* which determines how big a share of the total work output flows to the *Undiscovered Errors* –stock instead of the *Work Done* –stock (Eq. 2–4). In most cases *Probability Of Error Generation* is synonymous to work quality. Once errors are discovered (Flow: *Error Discovery*) they flow to the *Rework To Do* –stock from which they can be reworked (Flow: *Rework*) (Eq. 5).

$$\boldsymbol{Original\ Work\ To\ Do} = \int (-Work)\, dt \tag{1}$$

$$\boldsymbol{Work\ Done} = \int (Work + Rework - Error\ Generation)\, dt \tag{2}$$

$$\boldsymbol{Error\ Generation} = (Work + Rework) * Probability\ Of\ Error\ Generation \tag{3}$$

$$\boldsymbol{Undiscovered\ Errors} = \int (Error\ Generation - Error\ Discovery)\, dt \tag{4}$$

$$\boldsymbol{Rework\ To\ Do} = \int (Error\ Discovery - Rework)\, dt \tag{5}$$

$$\boldsymbol{Work} + \boldsymbol{Rework} = Effort\ Applied * Productivity \tag{6}$$

The total working output i.e. the sum of *Work* and *Rework* depends on the product of the *Productivity* and the *Effort Applied* i.e. effective hours put into the project (Eq. 6). Therefore, the project can be finished faster if more effort is applied or productivity is increased. Figure 9 illustrates the balancing loops that affect these variables.

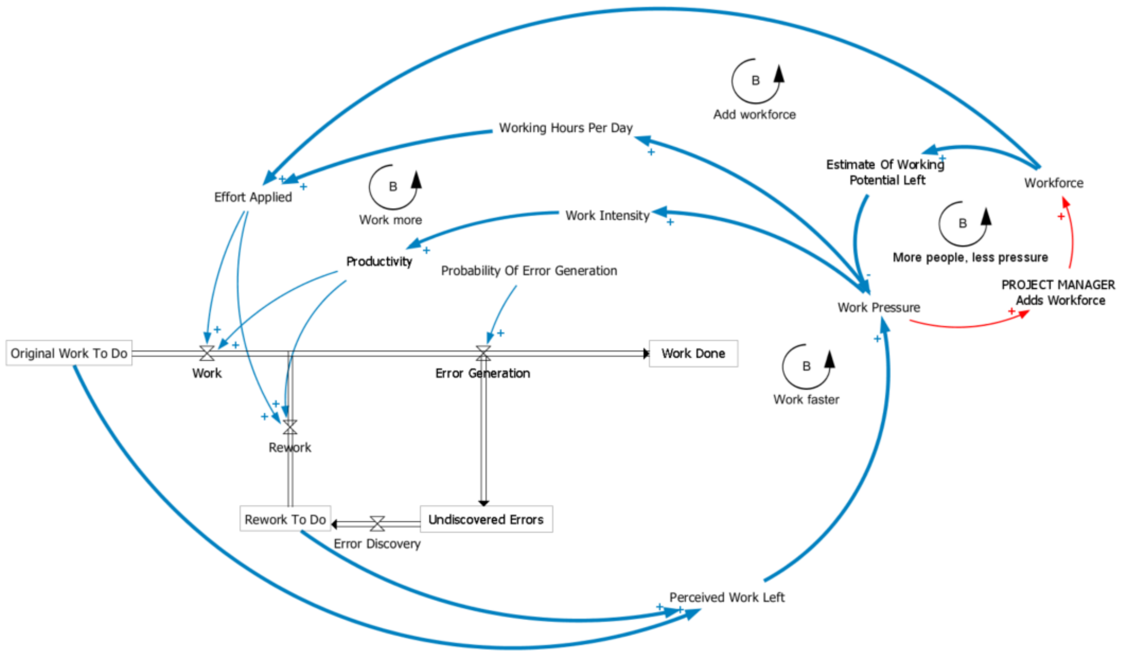


Figure 9. Rework cycle 2/3: balancing loops

When the workers in a project perceive that there is more work left (Eq. 8) than what can be completed with the current working output, *Work Pressure* rises (Eq. 10). This causes the workers to 1) *Work faster* and to 2) *Work more*, and the project manager to 3) *Add workforce*. Working faster means that higher *Work Pressure* causes higher *Work Intensity* (Eq. 11) which increases *Productivity*. Working more means that higher *Work Pressure* increases *Working Hours Per Day* (Eq. 12) which increases *Effort Applied* (Eq. 7).

$$\boldsymbol{Effort\ Applied} = Workforce * Working\ Hours\ Per\ Day \tag{7}$$

$$\boldsymbol{Perceived\ Work\ Left} = Original\ Work\ To\ Do + Rework\ To\ Do \tag{8}$$

$$\begin{aligned}\boldsymbol{Work\ Left} = {} & Original\ Work\ To\ Do + Rework\ To\ Do \\ & + Undiscovered\ Errors\end{aligned} \tag{9}$$

$$\boldsymbol{Work\ Pressure} = \frac{Perceived\ Work\ Left}{Estimate\ Of\ Working\ Potential\ Left} \tag{10}$$

$$\begin{aligned}&\boldsymbol{Work\ Intensity} \\ &\quad = f_{\{Effect\ Of\ Work\ Pressure\ On\ Work\ Intensity\}}(Work\ Pressure)\end{aligned} \tag{11}$$

$$\begin{aligned}&\boldsymbol{Working\ Hours\ Per\ Day} \\ &= f_{\{Effect\ Of\ Work\ Pressure\ On\ Working\ Hours\ Per\ Day\}}(Work\ Pressure)\end{aligned} \tag{12}$$

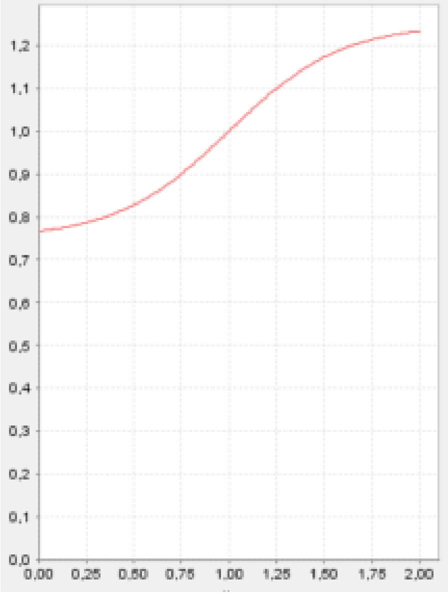


Figure 10. Logistic curve (x = input, y = output)

Standard *Work Pressure* equals 1. Functions in equations 11 and 12 are standard *Work Pressure* – centred logistic functions illustrated in Figure 10. Near the centre at (1.0, 1.0) the changes in hours and intensity are greater than further from the standard. The shape of the curve is adapted from the model by Oliva & Sterman (2010).

The auxiliary *PROJECT MANAGER Adds Workforce* in Figure 9 is not part of the EPCM Game simulation model but it represents the player itself. The red causal arrows denote the player information feedback and decisions. Workforce allocation is the main decision variable in the game, and supposedly the player makes balancing decisions on the workforce allocation which causes the balancing loop *Add workforce* outside the simulation model itself. Once the project manager adds more workforce, *Estimate Of Working Potential Left* increases which relieves *Work Pressure*. This is illustrated as the fourth loop: *More people,* *less pressure*. This loop does not contribute to finishing the project faster but balances the *Work Pressure* when more workforce is added.

Note that lower *Work Pressure* has the opposite effects: working less and slower, and removing workforce from the project. Note also that the workers and the project manager are making their decisions based on perceived information: *Perceived Work Left* and *Estimate Of Working Potential Left*. If the workers and the project manager had perfect information, they would know that the actual work left equals the sum of *Perceived Work Left* and *Undiscovered Errors* (Eq. 8). They could also calculate how much the workforce really is able to finish over time. *Estimate Of Working Potential Left* in EPCM Game is calculated directly from workforce standard working output with standard productivity and work day length without error generation, which causes the estimate to be typically over-optimistic.

If the perceived system state is far from the actual, the project might end up in trouble. For instance, if there are a lot of *Undiscovered Errors*, *Perceived Work Left* might be low which would lower *Work Pressure* which would cause people to start working shorter days with a lower intensity, even though the real situation might be that the project is behind schedule. Once the errors are finally discovered, *Work Pressure* might skyrocket.

There is also a fourth way of catching up with the project deadline in system dynamics literature: postponing the deadline. However, in EPCM projects moving the deadline has to be negotiated with the client, and is therefore not included in the EPCM Game simulation model itself.

Working faster and more, and adding more workforce have their downsides. Figure 11 introduces the reinforcing counterparts of the balancing loops. 1) *Haste makes waste*. In addition to higher *Productivity*, higher *Work Intensity* also increases the *Probability Of Error Generation*. 2) As for working more, in long term, it causes *Fatigue* (Eq. 13) which decreases *Productivity* and increases *Probability Of Error Generation*. This is also known as *Burnout*. 3) The project can become *Too big to manage*. The more there are people in a project, the more there are *Congestion And Communication Difficulties* (Eq. 14). Such difficulties decrease *Productivity* and increase *Probability Of Error Generation* (Eq. 15–16). These are malicious loops. Plummeting *Productivity* and skyrocketing *Error Generation* create more pressure to work even longer days, even faster, with more people.

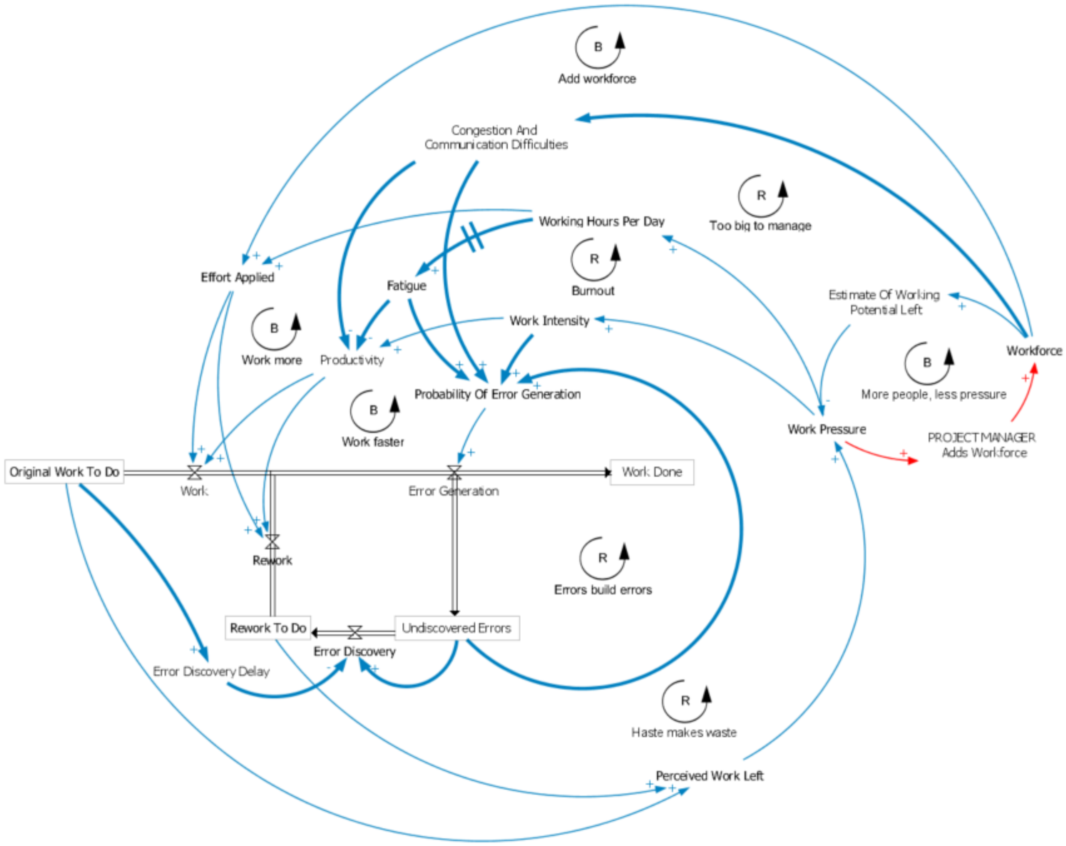


Figure 11. Rework cycle 3/3: reinforcing loops

When there are *Undiscovered Errors* in the project, it might occur that new work is based on the erroneous work. In such case it is very likely that the new work will be erroneous as well. The more there are *Undiscovered Errors*, the higher is the *Probability Of Error Generation* (Eq. 16). This is the fourth reinforcing loop: *Errors build errors*. The functions in the equations 13–16 are adapted from the model by Oliva & Sterman (2010).

$$\boldsymbol{Fatigue} = Delay(Order = 3, Delay\ Time = 20\ working\ days, Input = Work\ Pressure)^{2} \tag{13}$$

$$\boldsymbol{Congestion\ And\ Communication\ Difficulties} = f_{\{Effect\ Of\ Workforce\ Size\}}(Workforce) \tag{14}$$

$$\boldsymbol{Productivity} = f_{\{Productivity\}}(Work\ Intensity, Fatigue, Congestion\ And\ Communication \tag{15}$$

---

[2] 3rd order delay function of *Work Pressure* with the average delay of 20 working days.

$$\boldsymbol{Probability\ Of\ Error\ Generation} = f_{\{Probability\ Of\ Error\ Generation\}}(\\ Work\ Intensity, Fatigue, Congestion\ And\ Communication\ Difficulties, \\ Undiscovered\ Errors) \tag{16}$$

It was also discussed with the partner company experts and applied in the EPCM Game simulation model that the further the project is, the shorter is the delay to discover errors (Eq. 17). *Error Discovery* rate depends also on *Undiscovered Errors* since it is more likely to notice errors when there are plenty of them (Eq. 18).

$$\boldsymbol{Error\ Discovery\ Delay} \\ = f_{\{Effect\ Of\ Original\ Work\ To\ Do\}}(Original\ Work\ To\ Do) \tag{17}$$[3]

$$\boldsymbol{Error\ Discovery} = \frac{Undiscovered\ Errors}{Error\ Discovery\ Delay} \tag{18}$$

It was discussed with the partner company experts that by using quality control policies the project managers could affect *Probability Of Error Generation* and *Error Discovery* rate. This idea was not eventually introduced in the EPCM game since there was already enough complexity for the player to worry about.

Some models include the idea of workforce morale which is decreased by *Fatigue* and low work quality (Lyneis & Ford, 2007). Good morale decreases workforce turnover and *Probability Of Error Generation*. Morale issues, especially workforce turnover, were excluded from the EPCM Game simulation model because they were not considered being relevant enough for the learning goals of the game. Ideas such as “Haste creates out-of-sequence work” and “Errors create more work” were also excluded with similar motivation.

---

[3] The function in equation 17 is a logistic function with a center at (0.5, 0.75), starting at (0.0, 0.99) and ending at (1.0, 0.51).

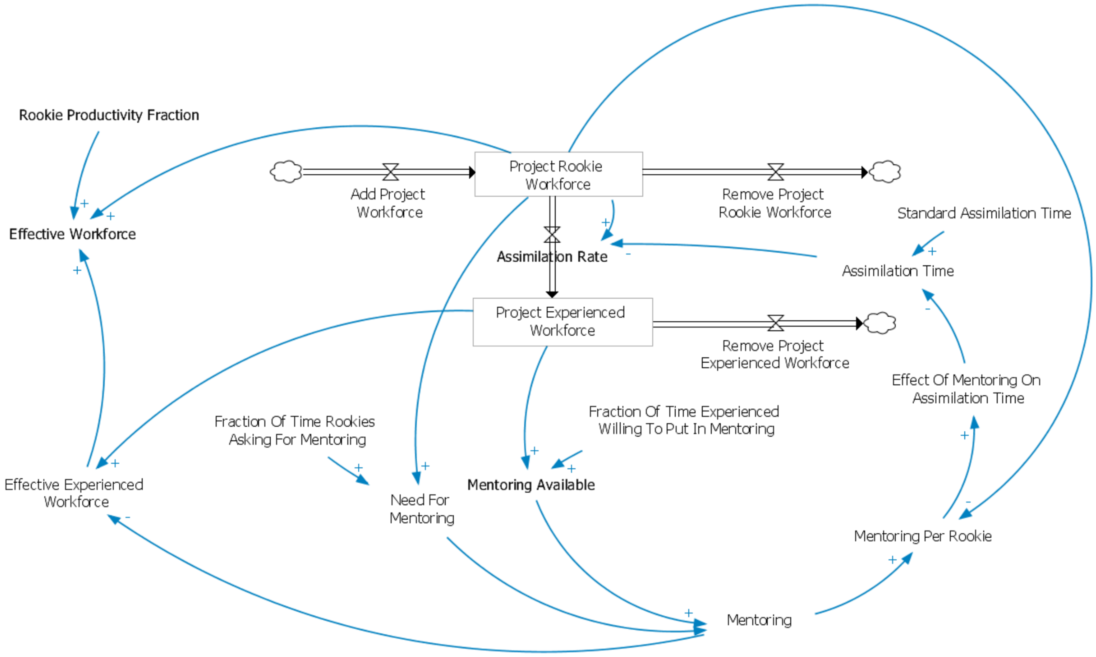

Rookie Productivity Fraction
Effective Workforce
Add Project Workforce
Project Rookie Workforce
Remove Project Rookie Workforce
Standard Assimilation Time
Assimilation Rate
Assimilation Time
Project Experienced Workforce
Remove Project Experienced Workforce
Effect Of Mentoring On Assimilation Time
Fraction Of Time Rookies Asking For Mentoring
Fraction Of Time Experienced Willing To Put In Mentoring
Effective Experienced Workforce
Need For Mentoring
Mentoring Available
Mentoring Per Rookie
Mentoring

Figure 12. Project experience chain

Figure 12 illustrates the project experience chain, an idea originally introduced by Jarmain (1963), which models the learning curve of individual workers added to the project. Once new workers are added to the project (Flow: *Add Project Workforce*) they start as rookie workers (Stock: *Project Rookie Workforce*). As they start working on the project they start figuring out what the project is all about and otherwise assimilate things needed to finish the project (Flow: *Assimilation Rate*). Eventually, they become fully experienced with the project (Stock: *Project Experienced Workforce*). (Eq. 19 and 20)

$$\boldsymbol{Project\ Rookie\ Workforce} = \int (Add\ Project\ Workforce - Remove\ Project\ Rookie\ Workforce - Assimilation\ Rate)\, dt \tag{19}$$

$$\boldsymbol{Project\ Experienced\ Workforce} = \int (Assimilation\ Rate - Remove\ Project\ Experienced\ Workforce)\, dt \tag{20}$$

Since rookie workers are less experienced they are also less effective (Constant: *Rookie Productivity Fraction*) and need mentoring from the more experienced workers (Auxiliary: *Mentoring*) (Oliva & Sterman, 2010). Mentoring consumes time from the experienced workforce decreasing their effectiveness (Auxiliary: *Effective Experienced Workforce*) and resulting in lower workforce output (Auxiliary: *Effective Workforce*). Since mentoring takes time from the more efficient workers, adding workforce to the project can momentarily decrease *Effective Workforce*. (Eq. 21 and 22)

$$\begin{aligned}&\boldsymbol{Effective\ Experienced\ Workforce}\\&\qquad = Experienced\ Workforce - Mentoring\end{aligned} \tag{21}$$

$$\begin{aligned}&\boldsymbol{Effective\ Workforce}\\&\qquad = Effective\ Experienced\ Workforce\\&\qquad + Project\ Rookie\ Workforce\\&\qquad * Rookie\ Productivity\ Fraction\end{aligned} \tag{22}$$

Note that even though it was not shown in the introduction to rework cycle, experience chain takes place when adding workforce. Adding workforce not only causes *Congestion And Communication Difficulties* but new workers can also momentarily lower the working output. Furthermore, it will take a while until new workers are truly useful. An impatient project manager might not realise this and adds one worker after another if the project is behind schedule to suddenly realise that the project was eventually over-manned.

It seems as if the current system dynamics literature does not take it into account that mentoring has an effect on assimilation rate. I included this idea in the simulation model since the idea rose in the discussions with the partner company: in the beginning of a project it takes a longer time to gather all the necessary knowledge to start effectively work on a project since there is nobody mentoring the first workers.

Equations 23–29 describe how mentoring works in the model. The idea is that mentoring has a decreasing marginal utility (Eq. 27): increasing amount of mentoring from 0 % to 10 % decreases *Assimilation Time* more than increasing mentoring from 10 % to 20 %. The more there is available *Mentoring Per Rookie* the faster the *Assimilation Time*. *Mentoring Per Rookie* depends on how much the rookies need mentoring and on how much experienced workforce are willing to put in mentoring.

$$\begin{aligned}&\boldsymbol{Need\ For\ Mentoring}\\&\qquad = Project\ Rookie\ Workforce\\&\qquad * Fraction\ Of\ Time\ Rookies\ Asking\ For\ Mentoring\end{aligned} \tag{23}$$

$$\textbf{Mentoring Available} = Project\ Experienced\ Workforce * Fraction\ Of\ Time\ Experienced\ Willing\ To\ Put\ In\ Mentoring \tag{24}$$

$$\textbf{Mentoring} = MIN(Need\ For\ Mentoring, Mentoring\ Available) \tag{25}$$

$$\textbf{Mentoring Per Rookie} = \frac{Mentoring}{Project\ Rookie\ Workforce} \tag{26}$$

$$\textbf{Effect Of Mentoring On Assimilation Time} = f_{\{Effect\ Of\ Mentoring\}}(Mentoring\ Per\ Rookie) \tag{27}$$

$$\textbf{Assimilation Time} = Standard\ Assimilation\ Time * Effect\ Of\ Mentoring\ On\ Assimilation\ Time \tag{28}$$

$$\textbf{Assimilation Rate} = \frac{Project\ Rookie\ Workforce}{Assimilation\ Time} \tag{29}$$

Some models consider that *Project Rookie Workforce* has a higher *Probability Of Error Generation* (Lyneis & Ford, 2007). This was left out since the experts at the partner company claimed that in their environment, project experience does not really affect *Error Generation* but the overall expertise does.

5.2. Global EPCM Projects

In EPCM project it is topical that work is distributed between different offices in different countries and they require expertise from different disciplines. Ruutu et al. (2011) and Pesonen et al. (2008) include these ideas in their simulation models. In both models the workforce has been divided into different disciplines. This means that workers in each discipline can only work on tasks of their discipline. For example, mechanical engineers can only work on mechanical engineering tasks.

The different disciplines have also interdependencies. Some disciplines might require other disciplines to have reached certain milestones in order have prerequisites fulfilled for their work. For example, mechanical engineers might require process engineers to finish piping and instrumentation diagrams in order to begin their work. However, if prerequisites are not fulfilled, disciplines have to resort to heuristics or they just do nothing. Using heuristics means that a discipline is using their best knowledge and experience to make good guesses on what to do. Skilled workers have a better heuristic accuracy i.e. a fraction of work getting actually done when the rest of the effort becomes *Undiscovered Errors* (Eq. 30). This is why one should not ramp up mechanical engineers before process engineers have finished the piping and instrumentation diagrams.

$$\boldsymbol{Probability\ Of\ Error\ Generation} = f_{\{Probability\ Of\ Error\ Generation\}}(Work\ Intensity, Fatigue, Congestion\ And\ Communication\ Difficulties, Undiscovered\ Errors) * Heuristic\ Accuracy\ When\ Prerequisites\ Not\ Met \tag{30}$$

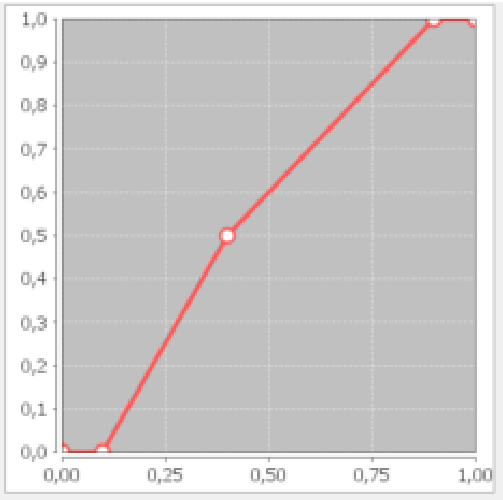


Figure 13. Example prerequisite function (x = prerequisite discipline progress, y = target discipline progress)

EPCM Game simulation model uses the same approach for heuristics as Ruutu et al. (2011). For each discipline there are three values from each other discipline: 1) what the other discipline's progress must be to have prerequisites fulfilled to begin work at this discipline, 2) what the other discipline's progress must be to have prerequisites fulfilled to reach 50 % progress at the discipline and 3) what the other discipline's progress must be to have prerequisites fulfilled to finish the work at the discipline. The prerequisite function is the interpolation over these discipline-prerequisite coordinates.

This is illustrated in the Figure 13 example: Assume that the prerequisite is process engineering (x-axis) and the target discipline is mechanical engineering (y-axis). Process engineering has to be 10 % complete before mechanical engineering has prerequisites to begin their work, process has to reach 25 % for mechanical to reach 25 %, process has to reach 40 % for mechanical to reach 50 % and process has to reach 90 % for mechanical to finish their work. Each discipline has similar function for each other discipline. All prerequisites must be fulfilled in order to have the discipline to work without heuristics.

Having multiple offices, in different countries, is useful for pushing down costs with labour from cheaper countries and for providing client interface workers from the same time zone with a similar cultural background. However, the differences in locations, time zones and cultural backgrounds cause problems inside project organisations. Communication becomes more difficult when people are not working in the same room, at the same time and don't speak the same native language. Even if everybody were fluent in English, the cultural differences cause communication difficulties. In the EPCM Game simulation model these problems are modelled as a part of *Congestion and Communication Difficulties*, and therefore, decrease *Productivity* and increase *Probability Of Error Generation* (Eq. 31).

$$\begin{aligned}\boldsymbol{Congestion\ And\ Communication\ Difficulties} \\ = f_{\{Effect\ Of\ Workforce\ Size\}}(Workforce) \\ + f_{\{Effect\ Of\ Number\ Of\ Offices\}}(Number\ Of\ Offices)\end{aligned} \tag{31}$$

The partner company experts mentioned that having workforce from two offices does not cause too many problems. Typically in such cases it is clear which office is in charge of which part of the project. Furthermore, there is only one way information can flow between offices. However, in the case of three or more offices the information becomes less coordinated since it can flow to two other offices. This also increases the probability of miscommunication due to second-hand information. The responsibilities become less clear as well. Therefore, the effect of number of offices is zero with one office, and the effect of three offices is three times as big as the effect of two offices.

5.3. Learning Goals

This section first introduces the major project management problems in the partner company environment and then the learning goals derived from them. It was eventually decided that the target group for EPCM Game would be anybody at the partner company from project manager trainees to experienced project managers. Considering zone of proximal development: project manager trainees need different challenges and learning goals than experienced project managers. Therefore, there was a need to be able to change the game content. Eventually EPCM Game got two difficulty levels with different learning goals for different audiences.

Easy version. The easy version includes the first three learning goals. The easy version of the game focuses on basic EPCM project dynamics and problems. The classifications of the learning goals according to revised Bloom's taxonomy are written inside the brackets.

1. Robust project manning and active management (3. apply, B. conceptual)

Repenning (2001), Lyneis & Ford (2007) and the partner company project managers agree that projects have often a delayed ramp-up and are often undermanned. This is often due to project managers underestimating required manpower or other projects finishing behind schedule and preventing access to key resources by binding them to the project. Project planning takes place in the early project and if there is not enough time to plan the project carefully, the execution might suffer, especially if the project was originally optimistically manned. Such projects often finish behind schedule, yet again binding the key resources to the project and preventing new projects accessing them. Repenning (2001) calls this "firefighting": instead of planning forward to next projects the organization is doomed to "putting out fires" in the existing projects. In product development projects it might even be beneficial to kill projects that are doomed to bad results to create better conditions for other projects to succeed. However, contracts do not allow killing EPCM projects.

Lyneis & Ford (2007) state that delayed ramp-up and optimistic planning require corrective actions which might overshoot: the project may get overmanned in the middle of the project. Once overmanning is recognised it might yet again be overcorrected if possible undiscovered errors are not taken into account. Once undiscovered errors start unfolding, a new batch of workers might be needed to rework them in time. This may result in double-humped manning outcome (see Figure 18). In addition to labour costs, adding people to the project costs the mentoring, rookie ineffectiveness, and congestions and communication problems.

There is always uncertainty in a project, and when the unexpected happens, more workers are typically needed. However, it might be difficult to find suitable workers in a short notice. That's why the manning plan should be robust. Robust manning plan is such that performs in uncertain conditions. Project manager can seek robustness by adding some buffer to the manning plan or by using more skilled workers that perform with higher certainty since they are less likely to generate insidious errors. The trade-off is that buffers and skilled workforce are expensive.

If a need for more workers emerges, decisions to add workforce should be made as soon as possible because every moment wasted makes it more difficult to find additional workforce. This is what the partner company experts considered “active management”.

Figures 14–19 are examples of different manning plans in EPCM Game. The different colours represent different disciplines. Figure 14 is the shape of a typical manning plan that was seen in the EPCM Game training sessions. Figure 15 is the shape of its progress estimate. If the original plan is robust, only a few changes to the plan is needed during the project: Figure 16 is a manning outcome of a fairly robust manning plan and Figure 17 is its progress outcome with active management. Such project reaches all milestones mostly in schedule and the costs do not bloat.

Even with active management, it is difficult to succeed with a non-robust manning plan. Figure 18 is a typical non-robust manning outcome and Figure 19 is its progress outcome. The plan has two clear humps due to corrective moves and the progress has a clear drop due to errors being discovered. Almost every milestone is reached behind schedule which results in claims. Even though the original plan may have been really profitable, the outcome is far less profitable than with a robust manning plan. Note that since EPCM Game does not introduce a situation where the client has not delivered prerequisite information, the process engineering discipline is relatively easy to man since the project manager does not have to worry about heuristics.

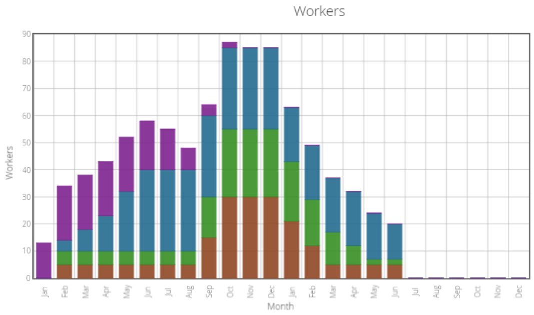


Figure 14. Typical manning plan

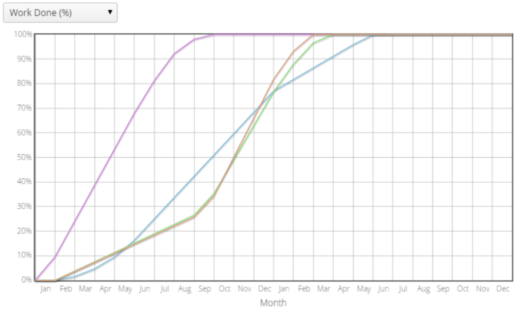


Figure 15. Typical progress estimate

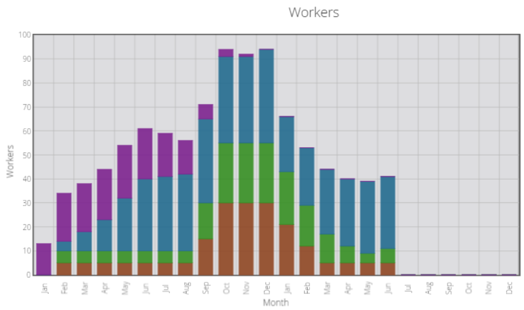


Figure 16. Typical fairly robust manning outcome

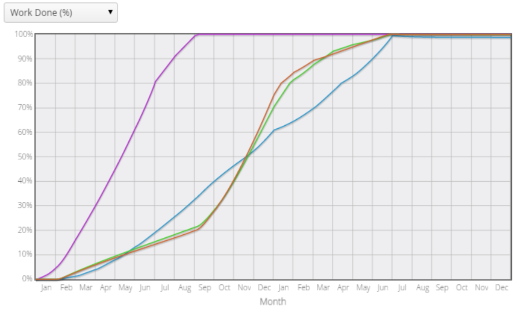


Figure 17. Typical progress outcome of fairly robust manning

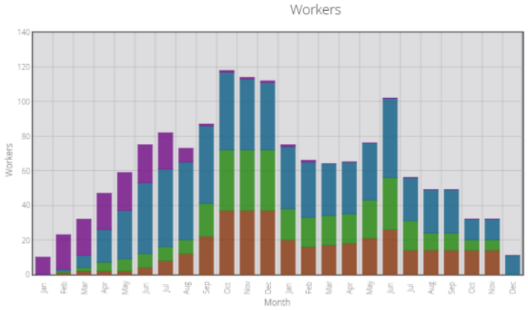


Figure 18. Typical non-robust manning outcome

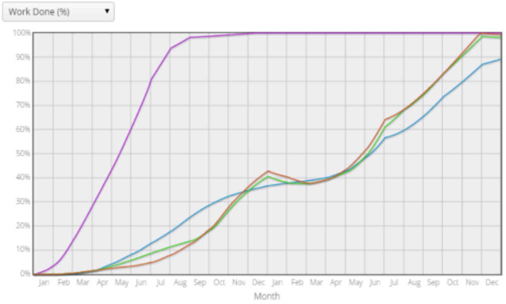


Figure 19. Typical non-robust progress outcome

2. Using the right workers at the right time (3. apply, B. conceptual)

The partner company experts mentioned that in addition to ramping up too late, ramping up disciplines too soon was a typical problem as well. This was typically due to the line organisation having spare workers. Since the line organisation has an incentive to keep up activity rate – i.e. the fraction of time workers are assigned to projects – they push the spare workers to their projects against the will of the project manager. Project managers might not be able to change this but they should understand what problems such behaviour causes.

In the early project, the project should be manned with more senior workers since in early project it is important to get things done and impossible to avoid using heuristics. Experts learn fast even without mentoring and have a good heuristic accuracy. Unfortunately, experts are a scarce and expensive resource, and the whole project cannot be manned with them, so later in the project the ramp up should be done with more junior workforce.

3. Dynamics of rework cycle, experience chain and multiple offices (2. understand, B. conceptual)

The player should also learn to understand the concepts of the dynamics of rework cycle, experience chain and multiple offices: to understand the short and the long term consequences of high work pressure, to understand that there is a learning curve before workers reach their full potential, to learn to lower the workforce costs by using workforce from cheaper offices, and to understand that using multiple offices causes congestion and communication problems, especially with more than two offices.

Difficult version. The difficult version includes the previous learning goals and dynamics but adds events to the game. The events either cause something unexpected or require client interaction. Therefore, the difficult version is more uncertain and requires more robust manning and more active management. The difficult version adds following learning goals to the game:

4. Managing change requests (2. understand, B. conceptual)

The client can also cause problems if not handled carefully. Especially in the early project the client has to provide information such as requirements and specifications so that the work can begin. Otherwise the workers must resort to heuristics. If the contract with the client is well-prepared it should have sanctions for the client for providing information behind schedule which motivates the client to be in schedule. The project manager should in any case make sure that the information arrives in time by working in the client interface.

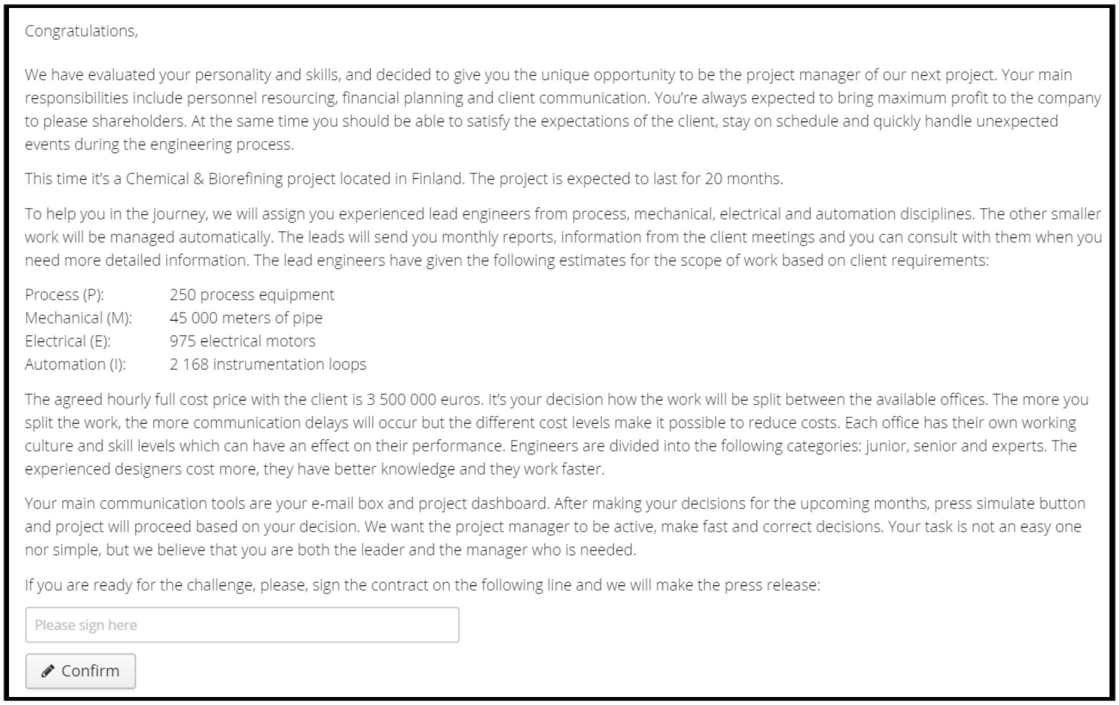
During the project the client might also come up with things that should be changed in the project. Making changes in the project can cause already made work to become obsolete i.e. some of the *Work Done* becomes *Undiscovered Errors* or *Rework To Do*. Since there is typically some lump sum the client is paying for the project, the project manager should make sure to get compensated for the extra work. Often rescheduling the project is needed as well. However, it’s too often that project managers just make a change order without getting project deadlines postponed and without compensation from the client because they don’t realise the impact of the changes and they want to please the client. Especially when the change is small, it does not seem important to negotiate compensations and schedules. But when there are multiple of small changes without compensations, it becomes a bigger problem.

The project manager should try pleasing the client since client satisfactory is important and a happy client is an easy client. However, it is more important to have a satisfied client at the end of the project when project has finished in schedule than a satisfied client during the project who becomes unhappy at the end of the project due to project finishing behind schedule. Project finishing behind schedule not only causes an unhappy client but also claims since contracts typically include sanctions for the EPCM contractor delivering the project behind schedule.

When the client wants to change something the player should learn to negotiate more time and money to compensate the extra work. The player should understand that even though a change seems to be minor from the process engineering point of view, it might require big rework in other disciplines too. The player should learn to find out how great impacts the changes cause in order to make good estimates on required extra work and in order to motivate the time and money demands to the client.

5.4. Game Design

The game is run in a browser. The game begins with an introduction to the project (Figure 20) with the project description. Once the player signs the contract the gaming can begin. The workforce is divided into four different disciplines: process engineers design process equipment (P), mechanical engineers design piping (M), electrical engineers design electrical motors (E) and automation engineers design instrumentation loops (I).



Congratulations,

We have evaluated your personality and skills, and decided to give you the unique opportunity to be the project manager of our next project. Your main responsibilities include personnel resourcing, financial planning and client communication. You're always expected to bring maximum profit to the company to please shareholders. At the same time you should be able to satisfy the expectations of the client, stay on schedule and quickly handle unexpected events during the engineering process.

This time it's a Chemical & Biorefining project located in Finland. The project is expected to last for 20 months.

To help you in the journey, we will assign you experienced lead engineers from process, mechanical, electrical and automation disciplines. The other smaller work will be managed automatically. The leads will send you monthly reports, information from the client meetings and you can consult with them when you need more detailed information. The lead engineers have given the following estimates for the scope of work based on client requirements:

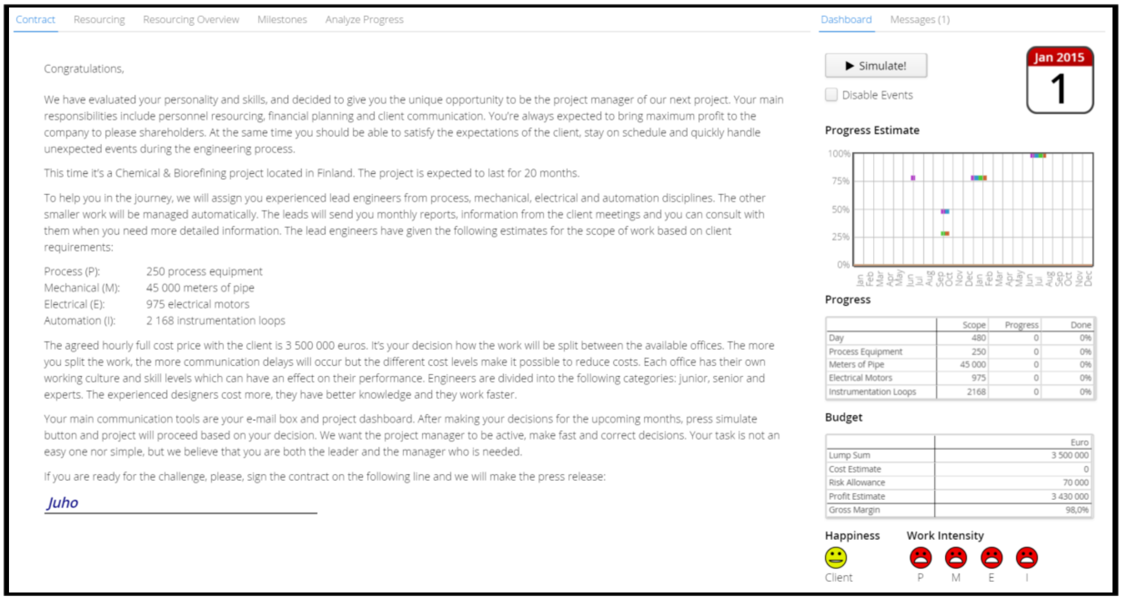
Process (P):	250 process equipment
Mechanical (M):	45 000 meters of pipe
Electrical (E):	975 electrical motors
Automation (I):	2 168 instrumentation loops

The agreed hourly full cost price with the client is 3 500 000 euros. It's your decision how the work will be split between the available offices. The more you split the work, the more communication delays will occur but the different cost levels make it possible to reduce costs. Each office has their own working culture and skill levels which can have an effect on their performance. Engineers are divided into the following categories: junior, senior and experts. The experienced designers cost more, they have better knowledge and they work faster.

Your main communication tools are your e-mail box and project dashboard. After making your decisions for the upcoming months, press simulate button and project will proceed based on your decision. We want the project manager to be active, make fast and correct decisions. Your task is not an easy one nor simple, but we believe that you are both the leader and the manager who is needed.

If you are ready for the challenge, please, sign the contract on the following line and we will make the press release:

Figure 20. The beginning of the game: an introduction to the project



Congratulations,

We have evaluated your personality and skills, and decided to give you the unique opportunity to be the project manager of our next project. Your main responsibilities include personnel resourcing, financial planning and client communication. You're always expected to bring maximum profit to the company to please shareholders. At the same time you should be able to satisfy the expectations of the client, stay on schedule and quickly handle unexpected events during the engineering process.

This time it's a Chemical & Biorefining project located in Finland. The project is expected to last for 20 months.

To help you in the journey, we will assign you experienced lead engineers from process, mechanical, electrical and automation disciplines. The other smaller work will be managed automatically. The leads will send you monthly reports, information from the client meetings and you can consult with them when you need more detailed information. The lead engineers have given the following estimates for the scope of work based on client requirements:

Process (P): 250 process equipment
Mechanical (M): 45 000 meters of pipe
Electrical (E): 975 electrical motors
Automation (I): 2 168 instrumentation loops

The agreed hourly full cost price with the client is 3 500 000 euros. It's your decision how the work will be split between the available offices. The more you split the work, the more communication delays will occur but the different cost levels make it possible to reduce costs. Each office has their own working culture and skill levels which can have an effect on their performance. Engineers are divided into the following categories: junior, senior and experts. The experienced designers cost more, they have better knowledge and they work faster.

Your main communication tools are your e-mail box and project dashboard. After making your decisions for the upcoming months, press simulate button and project will proceed based on your decision. We want the project manager to be active, make fast and correct decisions. Your task is not an easy one nor simple, but we believe that you are both the leader and the manager who is needed.

If you are ready for the challenge, please, sign the contract on the following line and we will make the press release:

Juho

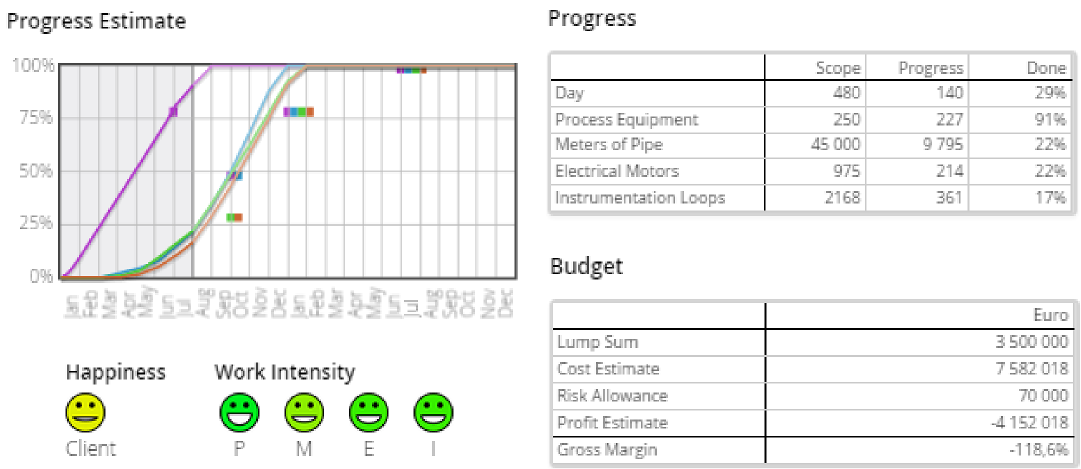
	Scope	Progress	Done
Day	480	0	0%
Process Equipment	250	0	0%
Meters of Pipe	45 000	0	0%
Electrical Motors	975	0	0%
Instrumentation Loops	2168	0	0%

	Euro
Lump Sum	3 500 000
Cost Estimate	0
Risk Allowance	70 000
Profit Estimate	3 430 000
Gross Margin	98,0%

Figure 21. Introducing the user interface

Once the contract has been signed the game view opens up (Figure 21). The left side of the user interface is the main view and the right side is the side view. The upper left tabs open different main views: 1) Contract, 2) Resourcing, 3) Resourcing Overview, 4) Milestones and 5) Analyse Progress. The upper right tabs open different side views: 1) Dashboard and 2) Messages.

The Dashboard shows the current date, progress, budget, client satisfaction and work intensity in each discipline. *Simulate* button is equivalent to *End Turn* buttons in turn-based games. Pressing *Simulate* the simulation is run until the beginning of next month or until an event occurs. By checking *Disable Events* check box, the player gets no unexpected events during the game. It is the game difficulty selector: without events the game difficulty is easy, else difficult.



	Scope	Progress	Done
Day	480	140	29%
Process Equipment	250	227	91%
Meters of Pipe	45 000	9 795	22%
Electrical Motors	975	214	22%
Instrumentation Loops	2168	361	17%

	Euro
Lump Sum	3 500 000
Cost Estimate	7 582 018
Risk Allowance	70 000
Profit Estimate	-4 152 018
Gross Margin	-118,6%

Figure 22. Progress, client satisfaction, work intensity and budget[4] indicators

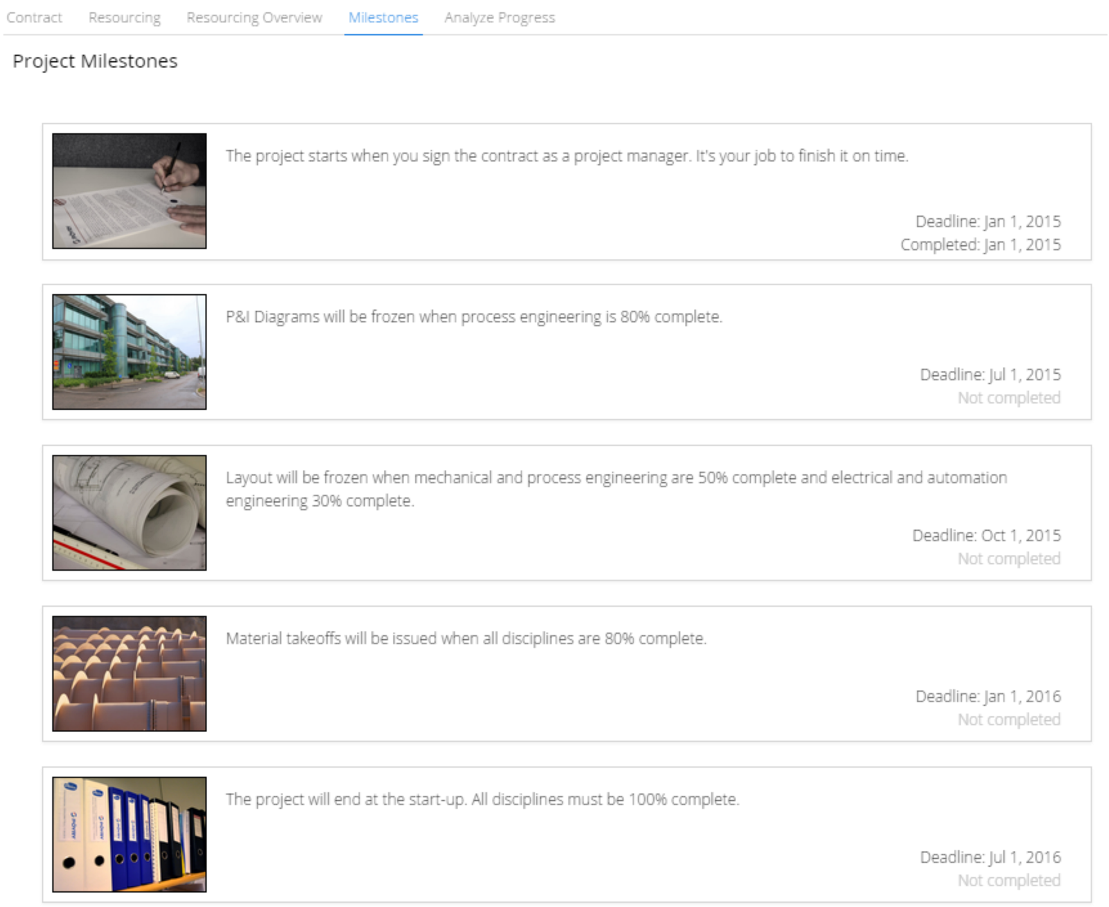
Once the player allocates resources to the project, progress, budget and work intensity indicators in Dashboard get updated (Figure 22). According to the workforce allocation the game makes a best-case estimation on project progress for each discipline over the project time frame. Work intensity is updated as well: the better the estimate looks, the lower is the work intensity. The estimate is plotted on the *Progress Estimate* graph. As the project goes on, the actual progress and the new estimate are updated. The grey area indicates past time and the small squares on the graph indicate the milestone deadlines for each discipline.

The player can take a closer look to the milestones under the *Milestones* tab (Figure 23). The first milestone is signing the contract. The second milestone is that piping and instrumentation diagrams have to be frozen i.e. process engineering has to be 80 % complete by the beginning of July. This is the first small purple square on the graph. The third milestone is freezing the layout: electrical and automation engineering have to be 30 % complete and

---

[4] Risk allowance is 2 % of the lump sum and in real projects it is reserved for unexpected expenses.

mechanical 50 %. The fourth milestone is the material take-off: every discipline has reached at least 80 %. From this point on the construction site starts receiving the materials as planned. The fifth milestone is a completed project: the production start-up.



Project Milestones

The project starts when you sign the contract as a project manager. It's your job to finish it on time.

Deadline: Jan 1, 2015
Completed: Jan 1, 2015

P&I Diagrams will be frozen when process engineering is 80% complete.

Deadline: Jul 1, 2015
Not completed

Layout will be frozen when mechanical and process engineering are 50% complete and electrical and automation engineering 30% complete.

Deadline: Oct 1, 2015
Not completed

Material takeoffs will be issued when all disciplines are 80% complete.

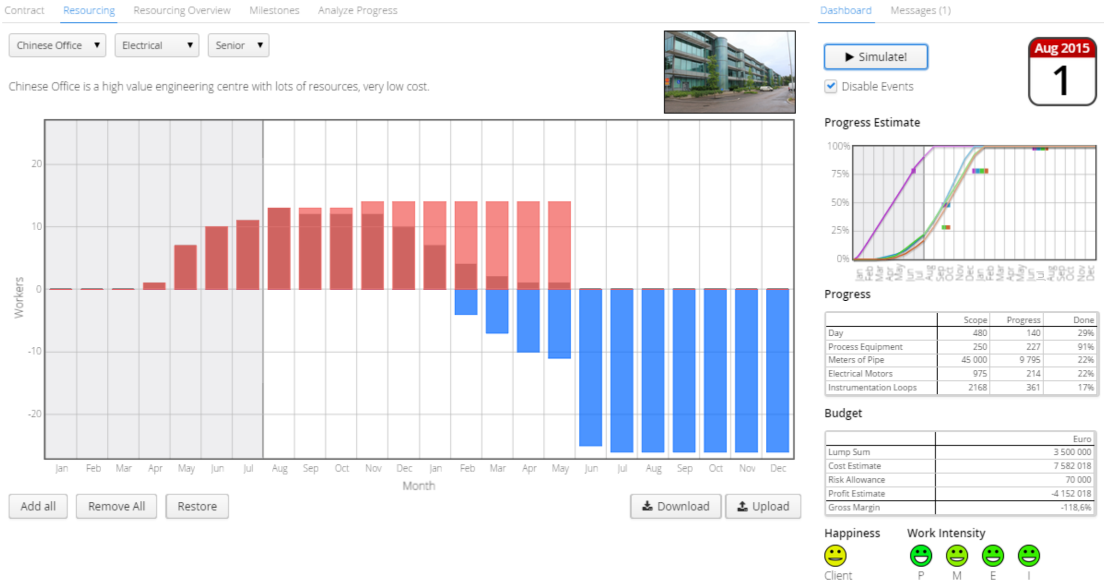
Deadline: Jan 1, 2016
Not completed

The project will end at the start-up. All disciplines must be 100% complete.

Deadline: Jul 1, 2016
Not completed

Figure 23. Milestones

Missing milestones deadlines causes client dissatisfaction and expensive claims from the client. The player will get these claims also in the case when the project seems complete but when there are so many undiscovered errors that the milestone is not actually reached. The claims are proportional to the integral of missing progress over time. Missing the material take-off and the project completion deadlines become the most expensive. Finishing these deadlines late cause the construction process to stand idle. Additionally, if there are undiscovered errors wrong things might be constructed and wrong materials delivered. These cause money to be wasted on extra work and materials.



Chinese Office is a high value engineering centre with lots of resources, very low cost.

Progress

	Scope	Progress	Done
Day	480	140	29%
Process Equipment	250	227	91%
Meters of Pipe	45 000	9 795	22%
Electrical Motors	975	214	22%
Instrumentation Loops	2168	361	17%

Budget

	Euro
Lump Sum	3 500 000
Cost Estimate	7 582 018
Risk Allowance	70 000
Profit Estimate	-4 152 018
Gross Margin	-118,6%

Figure 24. Resourcing

In the *Resourcing* view (Figure 24) the player can choose from the drop-down menus which office, discipline and seniority level to use for workforce allocation. For each four disciplines there are three seniority levels: juniors, seniors and expert, and three offices: Finland, Poland and China.

Seniority levels are not to be confused with project experience levels. Seniority level illustrates the worker's general skill level whereas project experience level illustrates the worker's understanding of the specific project. Workers' seniority levels do not change during the game. Seniority level could be modelled as an experience chain of three stocks (junior, senior, expert) but the time delays of reaching a higher seniority are longer than the length of a project. Therefore, no seniority chain was implemented.

Seniority levels and offices have different features: *Hourly Resource Prices, Heuristic Accuracy, Standard Probability Of Error Free, Standard Assimilation Time* and *Effort Per Outcome.* Basically, juniors are cheaper, generate more errors – especially when resorting to heuristics, assimilate slower and get work done slower. Each of these constants are modelled as 3 by 3 matrices (offices by seniorities). The disciplines have identical seniority and office features. Each office has different numbers of limited resources.

One aspect which was discussed with the partner company experts was that some tasks can only be completed by the more senior workers. The work could have been divided into seniority equivalent difficulty levels that could only be completed effectively by workers with at least the equivalent seniority. It was also discussed that more senior workers might want to leave the company if they have to do too much of low-skilled work. However, this idea was considered to be less important and it would create unnecessary complexity for the game.

The first turn of the game is the project planning phase: the player makes an initial manning plan by allocating different resources from different offices to the project for each month over the project time frame. At the project planning phase the player can access all the resources in the offices.

After the project planning phase some of the resources from the offices are allocated to other projects. Figure 24 is a screenshot of the *Resourcing* view of a situation at the beginning of month 8. The bar chart is the workforce allocation user interface. The red bars above the middle of the chart are the allocated resources and the blue bars under it are the available resources. As seen in the figure, there are no additional senior electrical engineers from Chinese office available for the next few months since they have been allocated to other projects.

One can’t get rid of extra resources easily either. Once the player has received resources from the offices, and therefore, agreed to offer work for the workers, it is difficult to get rid of them if there are too many. The light red part of the red charts illustrates the resources the player can get rid of. Since it is more difficult to find new projects for the workers for the very next months, it is more difficult to get rid of resources for the next month than for months from now.

The resource allocation is done by using the pointer to “paint” the wanted allocation or to “drag” the bars to the wanted position. With the buttons at the lower right corner, the current workforce allocation can be downloaded as a spreadsheet file to be later uploaded and used as a template for another try of the game.

The player can use *Resourcing Overview* (Figure 25) to see the big picture of the workforce allocation. The drop-down menus offer different perspectives. The player can then use *Analyze Progress* view (Figure 26) to inspect cumulative hours and costs, and to see better details over progress estimates.

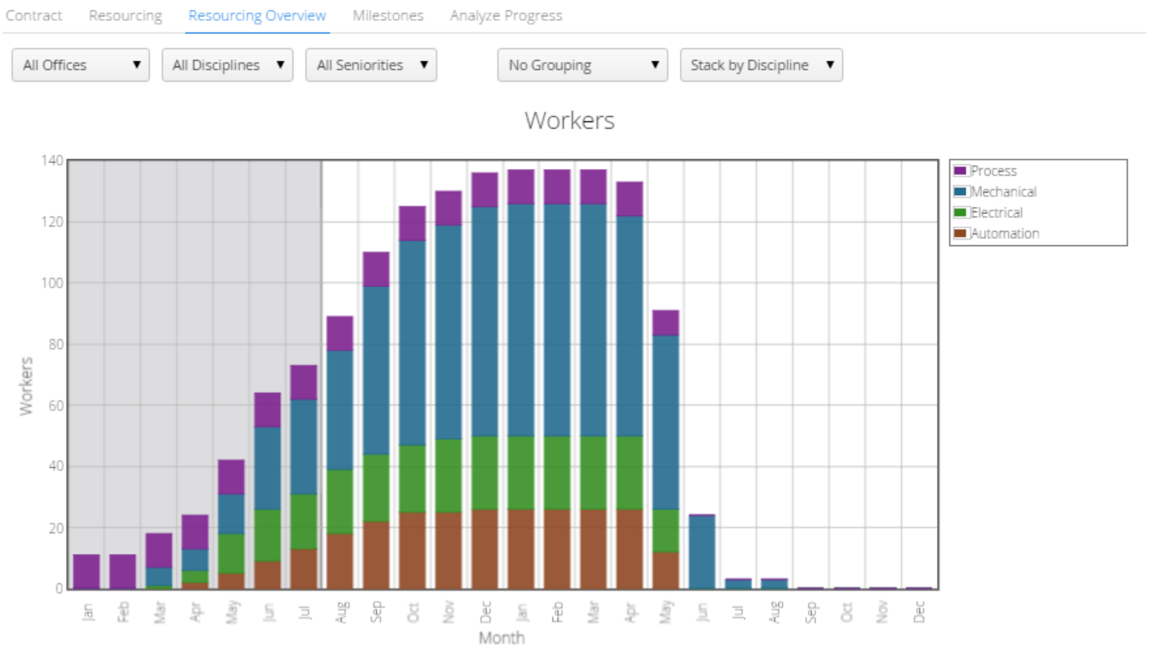


Figure 25. Resourcing overview

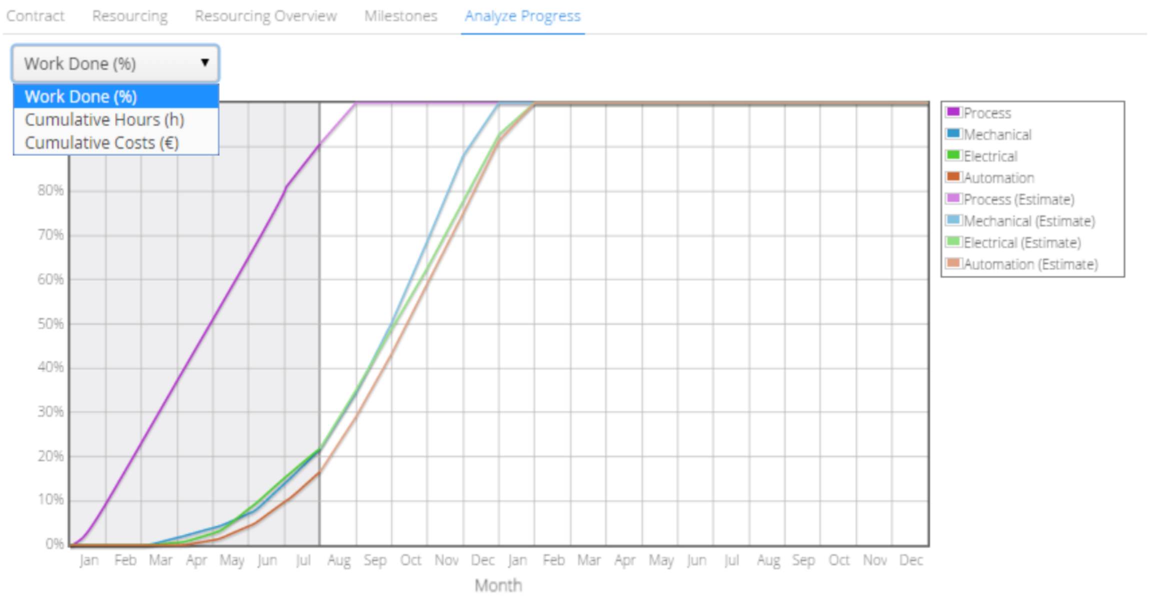
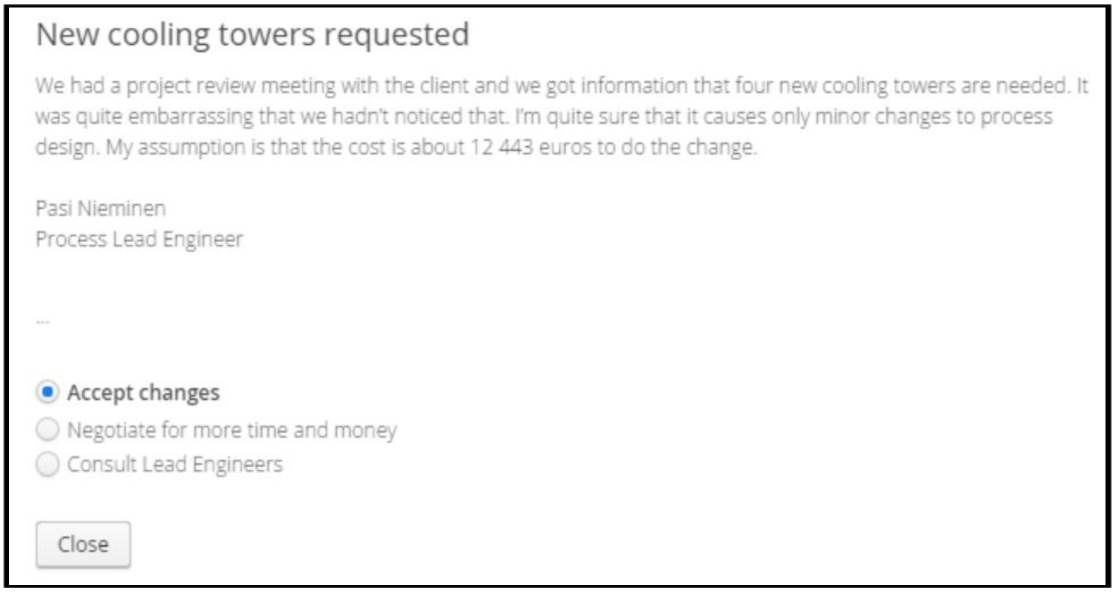


Figure 26. Analyze progress


New cooling towers requested

We had a project review meeting with the client and we got information that four new cooling towers are needed. It was quite embarrassing that we hadn't noticed that. I'm quite sure that it causes only minor changes to process design. My assumption is that the cost is about 12 443 euros to do the change.

Pasi Nieminen
Process Lead Engineer

...

● Accept changes
○ Negotiate for more time and money
○ Consult Lead Engineers

Figure 27. Event: new cooling towers requested

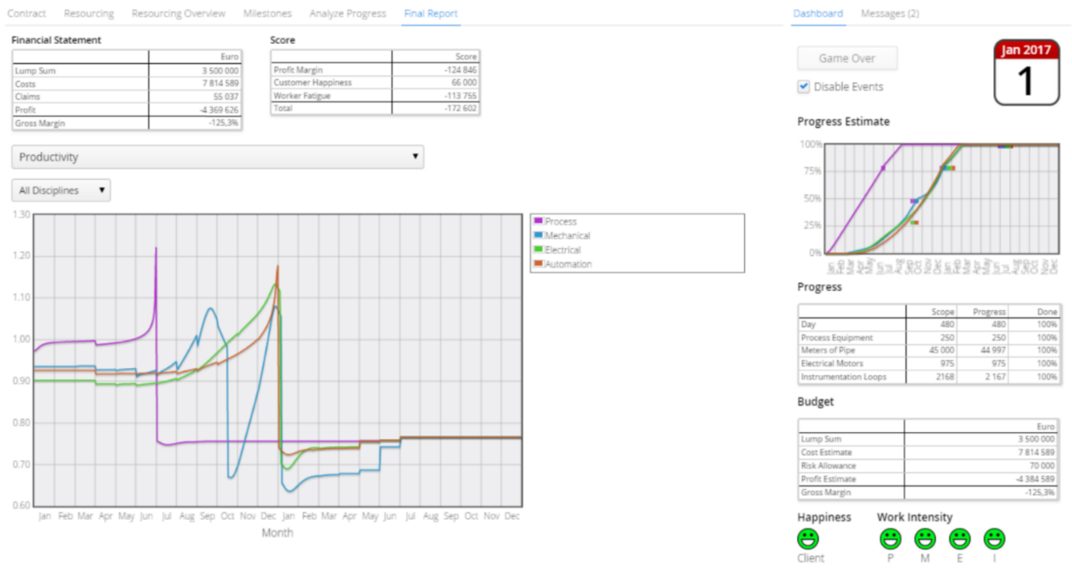
In the difficult version the player gets event messages (Figure 27). There are three kinds of event messages: 1) notifications for reaching milestones, 2) workforce-related problems and 3) change requests from the client. The player should react to workforce-related problems and respond to change requests from the client.

There are three ways to react to a change request: 1) accept changes, 2) negotiating more time and money and 3) consulting lead engineers of each discipline. By accepting the changes, some fraction of *Work Done* becomes *Undiscovered Rework* and *Rework To Do*. The caused rework is costs time and money. That's why the player should understand to negotiate more time and money.

However, unless the player consults the lead engineers of each discipline there will not be knowledge on what will be the impact of the changes on each discipline. For instance, the process lead engineer might tell that the change is minor but even the minor change might cause major rework in mechanical engineering.

The trade-off is that it always takes some time to consult the lead engineers, and to negotiate more time and money. Therefore, the workers will work on the project with the old plans longer. Therefore, a bigger absolute amount of work has to be reworked. Not all client requests are even reasonable. For instance, it might not make much difference to move a door location but it might require major changes to piping.

Typically, the right choice is to first consult the lead engineers, then negotiate for more time and money and only then make the change order. In the case of Figure 27 the player should just accept the changes since as the process lead engineer lets know, it was player’s organisation’s fault the cooling towers were not originally planned. If the player decides not to make changes, the client will later force the player with lawyers to make the changes. Furthermore, due to the delay of changing the plans a lot more has to be reworked.



Financial Statement

	Euro
Lump Sum	3 500 000
Costs	7 814 589
Claims	55 037
Profit	-4 369 626
Gross Margin	-125,3%

Score

	Score
Profit Margin	-124 846
Customer Happiness	66 000
Worker Fatigue	-113 755
Total	-172 602

Progress

	Scope	Progress	Done
Day	480	480	100%
Process Equipment	250	250	100%
Meters of Pipe	45 000	44 997	100%
Electrical Motors	975	975	100%
Instrumentation Loops	2168	2 167	100%

Budget

	Euro
Lump Sum	3 500 000
Cost Estimate	7 814 589
Risk Allowance	70 000
Profit Estimate	-4 384 589
Gross Margin	-125,3%

Figure 28. Final report

At the end of the game the user interface offers an additional view: *Final Report* (Figure 28). It can be used to check how the values of different variables changed throughout the project. This is useful when for pointing out the problems of the project in the game session debriefing.

5.5. Training Sessions, Feedback and Evaluation

Two 2.5 h training sessions were held with the game: one for three engineering students and one for two project manager trainees and their instructor at the partner company. Input-process-outcome approach by Garris et al. (2002) was used as a base for using the game for training.

Both training sessions began with a briefing where the basic concepts were introduced. The project dynamics were not introduced but the participants were told to expect similar behavior to real projects. Since the student group had no experience in EPCM project management, the briefing for them had to be more thorough – i.e. more instructional scaffolding had to be applied to bring the students closer to the zone of proximal development. Only the student group was introduced the interdependencies between disciplines because understanding this was expected from the partner company group. In both sessions, the game was played through twice: once without and once with events. The student group was struggling with the game more than the partner company group. The groups repeated all the typical problems in EPCM projects. The problems were pointed out and the dynamics around them were explained in the debriefing.

At the briefing the participants had to answer a short questionnaire (see appendix A) to test their knowledge on EPCM project management dynamics. The same questions were asked after the debriefing to see whether the attendants had at least briefly learned something. The answers had indeed become better: they were able to 1) describe why and how adding more people to a project affects productivity in short and long term, 2) describe in which order the disciplines should be manned and 3) they could explain better the effects of work pressure. Unfortunately, the questionnaire had to be short in order to fit the whole session in the time frame. There was not much room for other kinds of learning assessment either. Therefore, based on the sessions, it is difficult to claim anything airtight about the learning outcomes.

Both the students and the partner company group showed great interest towards the game. The partner company attendants told that playing the game felt familiar and it had grasped a lot of the important concepts of EPCM project management. They also told that if nothing else, the game would work as a powerful tool for discussing EPCM project management. All the attendants reported that they felt that they had learned something useful related to project management.

The groups also gave thoughts on problems with the game and how to further develop it:

1. Resourcing at the beginning of the game is slow and frustrating
2. The project progress estimate is misleading by being too optimistic
3. The project progress estimate should show also the estimates exceeding 100 % to better get the big picture over resourcing.
4. Process engineering is a too easy discipline since there are no situations when it need heuristics.

One of the worries about the game was that it might be too complex: the resourcing has time, discipline, office, and seniority dimensions. Even though the partner company group assured that the game complexity was not too high, it might still be too high for teaching the basic dynamic insights in project management effectively. However, the game seemed to work well for demonstrating the EPCM specific problems and dynamics. The power of the game seemed to be specifically in being a tool for educational discussion.

The game development suffered from the lack of agile iterative development and resourcing. Game development is a learning process, thus double-loop learning concept applies there. Since it was not originally known what kind of game exactly is wanted, the first version of the game should have been very simple and generic, a minimum viable product. Such a game would have given the partner company better feedback on what they might actually want. Then the game could have iteratively been developed and tested a small set of features at a time. Such approach would have shortened the information feedback delay of the learning loop. However, being agile would have required more commitment and resourcing from the partner company side so that the problems in the organisation would have been discovered sooner and so that the game could have been tested multiple times.

The complexity combined with lack of the agility and resources was a problem for model validation as well. Due to the model complexity there was already a hurry just to get something working which resulted in not enough time for extensive model validation during the model development. However, the simulation model is mostly based on previous research and only complemented with the knowledge extracted from the project management expert interviews. Furthermore, the model also behaves as expected so the model should be fairly valid. For further game development, more extensive model validation should be conducted.

Another problem was also that user interface development resources should have been introduced sooner to have a working user interface prototype earlier in the project. Another minor problem was that the partner company did not seem to have a unified terminology for EPCM projects. This caused confusion already between the partner company experts. This gave the game a new unexpected learning outcome: unified EPCM terminology.

Since the project was partly a technology demonstration, the used technologies set some constraints to the game development. Due to the demonstrated technologies the starting point was that system dynamics will be the modelling paradigm and the game will have a web-based user interface. This is not the optimal setting for learning game development since the tools were chosen before the modelling problem and the learning goals were defined which is against the general modelling and educational principles mentioned in sections 2 and 3. Therefore, instead of first defining the learning goals and then choosing the right tools to reach those goals the learning goals had to be chosen so that they can be taught with the chosen tools.

For instance, one problem that rose from the project manager interviews was that different offices have such incentives that create a game theoretical decision setting which resembles prisoner's dilemma. An example: Office A has a new project starting and needs resources for it. Some of office A's best resources are allocated to an office B's project. To optimise their project office A pulls their resources from office B's project to satisfy the needs of their own project. This is optimal for office A (at least in short-term) but devastating for office B and suboptimal for the company as a whole. This sort of dynamics could have easily been taught by using a board game with the same incentives as rules. The game could be replayed with different incentives that would lead to different outcomes. The same would be more difficult to teach by using system dynamics and web-based user interfaces and teaching such dynamics was, therefore, not chosen as a learning goal. Nevertheless, the results of such incentives could be introduced as events in EPCM Game.

Despite all the difficulties, I argue that the game was successful for being the first version of the game. The game was welcomed positively and there were indications that learning occurred.

5.6. Comparison to Other Project Management Flight Simulators

This thesis is far from being the first time that a system dynamics –based project management flight simulator is being developed. Nevertheless, Lyneis & Ford (2007) recommend to study and develop such simulators even further in their thorough survey on system dynamics applied on project management. This section introduces four system dynamics –based project management flight simulators.

SOFTSIM. Barlas & Bayraktutar (1992) introduce a system dynamics –based game for software project management training called SOFTSIM. Their simulation model is a simplified version of the model developed by Abdel-Hamid & Madnick (1989) which includes many of the dynamics of rework cycle and workforce experience. The objective of the game is to complete the software project within the given time and budget constraints. The game starts by the player choosing a scenario with different settings that affect the game difficulty. The player can choose 1) the project size, 2) whether the project size is estimated correctly, and 3) the length of the hiring delay. The game then runs one turn at a time. Each turn simulates 10 or 20 days depending on the size of the project. In addition to the scheduled completion date, the main information feedback indicators the player gets are 1) cumulative man-days, 2) % development tasks completed, and 3) % testing tasks completed. The player can also view ten less important indicators. Each turn the player makes three decisions: 1) % man power to be allocated for quality assurance, 2) % man-power allocated for rework, and 3) staff addition of removal.

The game was tested with university students and faculty members. The subjects typically had difficulties with the budget but completed the project early. The subjects considered the game realistic, plausible and consistent. The main criticism concerned the game session length which was typically 1.5 hours. The authors suggested the game length to be halved for practical use. The paper did not disclose anything about learning outcomes or whether their game sessions included debriefing.

The Incredible Manager. Like Barlas & Bayraktutar, Dantas et al. (2004) also developed a system dynamics –based game for software project management training. Their game is called The Incredible Manager and the approach is slightly different. Their paper does not disclose much about the simulation model nor its validation. At the beginning of the game the player is given a document describing the project tasks and its function points, quality, schedule, budget demands and constraints. Based on the document the player plans the project: 1) hires developers, 2) determines which developers are assigned to which tasks in the task network, 3) determines number of days necessary to complete each task, and 4) sets policy on quality assurance activities. If the plan does not match the given project description document it will not be accepted by the project stakeholders.

Project execution runs in continuous turns, consuming project resources. The paper is not very specific on the player information feedback but the player is at least shown feedback on time, funds, developer exhaustion and task completion. Based on these, player can modify the original plan on the fly.

The game was tested with software project management students. The test subjects played the game twice and the learning was measured by comparing the individual results between the sessions. Debriefings were held at the end of the sessions. Most subjects got better results in the second session, although, most of the subjects failed to execute a successful project. Despite of failure the game was considered motivating, practical and enjoyable. Continuous-time turns and compelling visual effect were considered adding challenge and entertainment factor to the game. The game was criticised for simplifications of software project management. The game lacked a tool to trace and explain the actions, consequences, lessons learned, and alternative routes for decision-making during the execution of the game and the paper argues that having one would help users evaluate their own performance after executing the game.

BP New Product Development Management Flight Simulator. MacInnis (2004) developed a system dynamics –based game for project management training in new product development. The simulation model is based on single-phase product development process model by Ford & Sterman (1998) which is very similar to basic rework cycle. The model also includes workforce experience dynamics with automated hiring.

The game begins by the player setting up a scenario by choosing values for ten different parameters. Then the game runs one turn at a time. One turn represents one week. The player is shown very detailed information feedback on different variables of the simulation model. There are three decision variables in the game the player can change each turn: 1) hiring rate, 2) overtime authorization, and 3) schedule slip.

MacInnis lists following learning goals for the game: 1) ensure a realistic and attainable schedule, 2) intuitive management actions can exacerbate undesirable project dynamics, 3) it is better to slip intermediate milestones to make project perform better in the end, 4) worse before better –management actions yield better overall results, and 5) aggressively hire personnel early. The game was tested with PhD students and faculty members. MacInnis did not disclose whether game sessions had debriefings. Learning outcome was not measured but the game received positive feedback.

Nokia Game. Pesonen et al. (2008) developed a system dynamics –based game for Nokia product development project and portfolio management training. The goal of the game is “to provoke facilitated discussion in order to gain shared understanding”. Like MacInnis’s game, Nokia Game’s simulation model is also based on product development process model by Ford & Sterman (1998) and includes workforce experience dynamics. Nokia Game is a multi-player game with multiple teams: One of the teams is called the steering group which manages the project portfolio. The other teams each manage their own project inside the portfolio.

The game session starts with motivation and virtual business presentations. The actual playing starts with platform release selection: First the steering group determines project starting dates and the project targets which are either quality, time or cost. Then the projects select the suitable software, engine and electro mechanics releases. Newer releases are better in quality and cost but are more likely to be late. After release selection the project teams request resources for various project tasks from the common resource pool managed by the steering group. The tasks are interdependent which causes bottlenecks in the projects. The resource pool is insufficient for all the needs and the steering group has to prioritise. During the simulation the teams can make changes to their plans: The steering group can change targets, resourcing, terminate projects etc. Game sessions take 3-4 hours and can have from 5 to over 20 participants. Projects can be run by autopilot if there are not enough participants to run every project. The game facilitator can tune the simulation during the session if needed. The game has an automated reporting system which reports after each turn and at the end of the session when the debriefing is held.

Pesonen et al. conducted 30 game sessions with about 500 participants. They discovered that players with different experience levels learn different things: the less experienced learn the basics of project management whereas the more experienced learn how to run the portfolio. Pesonen et al. argue that a tailored simulation game would give better results than a generic business game.

Synthesis. As these studies demonstrate, simulation games can be developed out of different kinds of projects and organisations with different project models and game designs. The basic system dynamics theory seems to be applicable to most kinds of projects but different kinds of projects have different constraints and other properties which require different overall approaches to modelling. See Table 1 for side-by-side comparison of system dynamics –based project management flight simulators.

Even though the main purpose of management flight simulators is learning, these studies did not pay close attention to learning aspects. This is where this thesis differentiates from these studies, in addition to modelling a different kind of project.

Table 1. Comparison of system dynamics –based project management flight simulators

	SOFTSIM	**The Incredible Manager**	**PB NPDMFS**	**Nokia Game**	**EPCM Game**
Goal	complete the software project within the given time and budget constraints	finish project with given quality, schedule, budget demands and constraints	product development scenario given to the player	steering group: successful portfolio, project management team: successful product development project Learning goals: not stated explicitly	manage a profitable EPCM project with a satisfied client and without burning out the workforce
Learning goals	not stated explicitly	not stated explicitly	1) ensure a realistic and attainable schedule, 2) intuitive management actions can exacerbate undesirable project dynamics, 3) it is better to slip intermediate milestones to make project perform better in the end, 4) worse before better – management actions yield better overall results, and 5) aggressively hire personnel early	not stated explicitly	1) robust project manning and active management, 2) using the right workers at the right time, 3) dynamics around rework cycle, experience chain and multiple offices and 4) managing change requests
Information feedback	1) cumulative man-days, 2) % development tasks completed, and 3) % testing tasks completed	1) time, 2) funds, 3) developer exhaustion, 4) task completion	very detailed information feedback on different variables of the simulation model	status reports	1) perceived progress, 2) allocated workforce, 3) progress estimate, 4) costs, 5) client satisfaction, 6) work pressure
Decision variables	1) % man power to be allocated for quality assurance, 2) % man-power allocated for rework, and 3) staff addition of removal	1) hire developers, 2) determine which developers are assigned to which tasks in the task network, 3) determine number of days necessary to complete each task 4) set policy on quality assurance activities	1) hiring rate, 2) overtime authorization, 3) schedule slip	steering group: setting targets for projects, allocating resources to the projects, and terminating projects, project management team: suitable software, engine and electro mechanics releases, and requesting resources for the project	1) workforce allocation over time, 2) client interaction
Criticism	1.5 h length too much, needs to be halved	the game lacked a tool to trace and explain the actions, consequences, lessons learned, and alternative routes for decision-making during the execution of the game	not stated	nothing specific	1) resourcing at the beginning of the game is slow and frustrating 2) the project progress estimate is misleading by being too optimistic 3) the project progress estimate should show also the estimates exceeding 100 % to better get the big picture over resourcing 4) perhaps too complex 5) too long sessions 6) process engineering too easy

6. Conclusions

This thesis introduced the basic concepts around complex systems and then moved to introducing learning and gaming theories that would be used for the game development. The empirical study of this thesis was a simulation game for EPCM project management training which included an introduction to system dynamics in project management along the introduction to the game simulation model. The thesis described the development process, the learning goals and the design of the game, and how the game was used for training purposes, what kind of feedback the game received, and evaluated why there were problems and how things could be improved.

Based on this study, I recommend to get familiar with the learning and gaming theories presented in this thesis when developing simulation games for learning purposes. The theories helped structuring the EPCM Game development. Firstly, double-loop learning gave theoretical – yet pragmatic – justification to using simulation games for training purposes. Secondly, Bloom’s taxonomy combined with zone of proximal development provided a framework for setting the learning goals and finding the optimal difficulty for the learning content. Thirdly, instructional scaffolding and input-process-outcome concept helped designing the game, including the briefing and the debriefing.

The EPCM Game development suffered mostly from the lack of agile iterative development. The game was not tested with the subject group until the version introduced in this thesis. Since the learning feedback loop delay for developing the game was long, the game did not perhaps satisfy the needs as well as it could have.

Since there was only one brief opportunity to test the game with the target group and since there was not opportunity to conduct a proper survey on the learning outcomes, it is difficult to say whether the learning goals were met. The player feedback gave the impression that the game was at least an exciting conversation piece and helped understanding concepts in EPCM project management.

There are good reasons to use simulations and games for training purposes. Simulation games are powerful tools for teaching dynamic insights and decision-making. They make learning very effective as the players do not have to worry about making mistakes and can see the hands-on results of their decisions instantly. This sets them free to explore different approaches. Games can also work as a great tool for learning discussion around the topic.

The research question for this thesis was how learning simulation games should be developed and used. My idea based on the theoretical framework and the empirical study is that there are three tracks in learning simulation game development: 1) the simulation model, 2) the learning aspects and 3) the game design.

In the simulation model track the problem definition is the first thing to do. Once the problems have been identified, one should choose the suitable modelling paradigm, and the tools for simulation model and game development. As mentioned, system dynamics is not suitable for all modelling problems. For some problems, agent-based modelling or discrete event simulation might be better. The game does not have to be a computer game either. For instance, a popular simulation game in system dynamics community, Beer Distribution Game (Goodwin & Franklin, 1994), works probably better as a board game than a computer game. It should also be questioned whether a game is needed at all. Once the problems have been discovered and the modelling paradigm chosen, modelling can continue according to the paradigm. MIT standard method (Hines, 2004), for instance, gives a good iterative framework for mathematical modelling.

The learning track should go parallel with the model development. There are two major decisions to make in the learning track 1) choosing the target groups and 2) choosing the learning goals. The problem definition and the target group are coupled: The discovered problems indicate which groups are in a need for training. On the other hand, the chosen target groups direct the model development. The learning goals should be derived from the problem definition to serve the target group’s learning.

As the model and the learning aspects evolve, the game should be built around them. It would be a good idea to have a working prototype of the game as soon as possible. As the game development advances, the game prototypes should be tested with the target group iteratively to get feedback to develop the game to the right direction.

I agree with Pesonen et al. (2008) that better learning outcomes can probably be achieved by using a tailored simulation game than a generic one. A generic project game would not probably work well as a tool for educational discussion. Furthermore, using a tailored game reduces the need for instructional scaffolding when there is no need to explain, for instance, what is mechanical engineering and in which order it has to be ramped-up. However, a generic game can be good for teaching generic project management skills.

Typically, even a good simulation game does not do the training by itself. The worry with games is that the player might just learn to play the game and not to apply the learning experiences in real life. To tackle this, instructional scaffolding should be used in the form of debriefing. In debriefings, the problems that occurred during the game should be pointed out and discussed, the dynamics around them explained, and better approaches should be introduced. A good debriefing is one that makes the player to reflect the game on real life, to build the bridge between the virtual and the real world.

7. References

Abdel-Hamid, T. K., & Madnick, S. E. (1989). Lessons Learned from Modeling the Dynamics of Software Development. *Communications of the ACM*, *32*(12), 1426–1438. doi:0.1145/76380.76383

Abt, C. C. (1987). *Serious Games*. University Press of America, 1987.

Andersen, D. F., Chung, I. J., Richardson, G. P., & Stewart, T. R. (1990). Issues In Designing Interactive Games Based On System Dynamics Models. In *Proceedings of the 1990 International System Dynamics Conference* (pp. 31–45). International System Dynamics Society. Retrieved from http://www.systemdynamics.org/conferences/1990/proceed/pdfs/ander031.pdf

Anderson, L. W., & Krathwohl, D. R. (2001). *A Taxonomy for Learning, Teaching, and Assessing*. *Longman New York* (Vol. 2003).

Backlund, A. (2000). The definition of system. *Kybernetes*, *29*(4), 444–451. doi:10.1108/03684920010322055

Banks, J. (2000). Introduction To Simulation. In *2000 Winter Simulation Conference* (pp. 9–16).

Barlas, Y., & Bayraktutar, I. (1992). An Interactive Simulation Game For Software Project Management (SOFTSIM). In *Proceedings of the 1992 System Dynamics Conference* (pp. 59–68). The System Dynamics Society.

Bloom, B. S., Englehard, M. D., Furst, E. J., Hill, W. H., Krathwohl, D. R., & Committee of College and University Examiners. (1956). *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain*. *New York* (Vol. 16). doi:10.1300/J104v03n01_03

Bonabeau, E., & Dessalles, J. (1997). Detection and emergence. *Intellectica*, *25*(2), 85–94.

Caillois, R. (1961). *Man, Play, and Games*. University of Illinois Press.

Chaiklin, S. (2003). The Zone of Proximal Development in Vygotsky ' s Analysis of Learning and Instruction. In *Vygotsky's educational theory in cultural context 1* (pp. 39–64). Cambridge, United Kingdom: The Press Syndicate Of The University Of Cambridge.

Chrons, O., & Sundell, S. (2011). Digitalkoot: Making Old Archives Accessible Using Crowdsourcing. *Human Computation*, 20–25. Retrieved from http://www.aaai.org/ocs/index.php/WS/AAAIW11/paper/viewFile/3813/4246

Cooper, K. G. (1980). Naval Ship Production: A Claim Settled and a Framework Built. *Interfaces*, *10*(6), 20–37. doi:10.1287/inte.10.6.20

Dantas, A., Barros, M., & Werner, C. (2004). A Simulation-Based Game for Project Management Experiential Learning. *International Journal of Software Engineering and*

Knowledge Engineering, *19*, 6.

Doyle, J. K., & Ford, D. N. (1998). Mental models concepts for system dynamics research. *System Dynamics Review*, *14*(1), 3–29. doi:10.1002/sdr.476

Ekaputra, G., Lim, C., & Eng, K. I. (2013). Minecraft: A Game as an Education and Scientific Learning Tool. In *Information Systems International Conference* (pp. 239–242).

Federation of American Scientists. (2006). *Harnessing the power of video games for learning*.

Ford, D. N., & Sterman, J. D. (1998). Dynamic modeling of product development processes. *System Dynamics Review*, *14*(1), 31–68. Retrieved from http://scripts.mit.edu/~jsterman/docs/Ford-1997-DynamicModelingOfProductDevelopment.pdf

Forrester, J. W. (1992). *System Dynamics and Learner-Centered-Learning in Kindergarten through 12th Grade Education* (No. D-4337). Cambridge, Massachusetts. Retrieved from http://web.mit.edu/sysdyn/sd-intro/D-4337.pdf

Frasca, G. (2003). Simulation versus Narrative – Introduction to Ludology. In *The Video Game Theory Reader* (pp. 221–235). New York: Routledge.

Garris, R., Ahlers, R., & Driskell, J. E. (2002). Games, Motivation, and Learning: A Research and Practice Model. *Simulation & Gaming*, *33*(4), 441–467. doi:10.1177/1046878102238607

Goodwin, J. S., & Franklin, S. G. (1994). The Beer Distribution Game: Using Simulation to Teach Systems Thinking. *Journal of Management Development*, *13*(8), 7–15. doi:10.1108/02621719410071937

Harteveld, C., & Sutherland, S. (2014). Finding the Game in Decision-Making: A Preliminary Investigation. In W. C. Kriz, T. Eiselen, & W. Manahl (Eds.), *45th Conference of the International Simulation and Gaming Association* (pp. 199–209). Dornbirn: International Simulation and Gaming Association.

Hines, J. (2004). Standard Method. Retrieved from http://ocw.mit.edu/courses/sloan-school-of-management/15-875-applications-of-system-dynamics-spring-2004/projects/stnd_method.pdf

ITEA. (2014). MODRIO Project Outline (v. 2.2).

Janssen, M. A. (2005). Agent-based modelling. In *Modelling in ecological economics* (pp. 155–172). Retrieved from http://pdf.aminer.org/000/566/322/agents_adopting_agriculture_modeling_the_agricultural_transition.pdf

Jarmain, W. E. (1963). *Problems in Industrial Dynamics*. Cambridge, Massachusetts, U.S.A.: MIT Press.

Kopainsky, B., & Sawicka, A. (2011). Simulator-supported descriptions of complex dynamic problems: Experimental results on task performance and system understanding. *System Dynamics Review*, *27*, 142–172. doi:10.1002/sdr.445

Lyneis, J. M., Cooper, K. G., & Els, S. a. (2001). Strategic management of complex projects: a case study using system dynamics. *System Dynamics Review*, *17*(3), 237–260. doi:10.1002/sdr.213

Lyneis, J. M., & Ford, D. N. (2007). System dynamics applied to project management: a survey, assessment, and directions for future research. *System Dynamics Review*, *23*(2), 157–189. doi:10.1002/sdr

MacInnis, D. V. (2004). *Development of a System Dynamics Based Management Flight Simulator for New Product Development by*. Massachusetts Institute of Technology.

Maier, F. H., & Größler, A. (2000). What are we talking about?—A taxonomy of computer simulations to support learning. *System Dynamics Review*, *16*, 135–148. doi:10.1002/1099

Modelica Association. (2012). Modelica ® - A Unified Object-Oriented Language for Systems Modeling Language Specification. Retrieved from https://www.modelica.org/documents/ModelicaSpec33.pdf

Murray, T., & Arroyo, I. (2002). Toward Measuring and Maintaining the Zone of Proximal Development in Adaptive Instructional Systems. In *International Conference on Intelligent Tutoring Systems Toward* (pp. 1–8).

Oliva, R., & Sterman, J. D. (2010). Death Spirals and Virtuous Cycles: Human Resource Dynamics in Knowledge-Based. In P. P. Maglio, C. A. Kieliszewski, & J. C. Spohrer (Eds.), *Handbook of Service Science* (pp. 321–358). New York: Springer. doi:10.1007/978-1-4419-1628-0

Pesonen, L. T. T., Salminen, S. J., Ylén, J.-P., & Riihimäki, P. (2008). Dynamic simulation of product process. *Simulation Modelling Practice and Theory*, *16*(8), 1091–1102. doi:10.1016/j.simpat.2008.04.002

Project Management Institute. (2000). *Project Management Body of Knowledge A Guide to the A Guide to the A Guide to the Project Body of Project Management Knowledge Management Body of Knowledge Knowledge* (2000 Editi.). Newton Square, Pennsylvania USA: Project Management Institute, Inc. Retrieved from http://www.cs.bilkent.edu.tr/~cagatay/cs413/PMBOK.pdf

Repenning, N. P. (2001). Understanding fire fighting in new product development. *Journal of Product Innovation Management*, *18*(5), 285–300. doi:10.1111/1540-5885.1850285

Rieber, L. P. (1996). Seriously Considering Play. *Educational Technology Research and Development*, *44*(2), 43–58.

Roberts, N. (1978). Teaching Dynamic Feedback Systems Thinking: An Elementary View.

Management Science, *24*(8), 836–843.

Robinson, S. (2014). *Simulation: the practice of model development and use*. Palgrave Macmillan.

Ruutu, S., Ylén, P., & Laine, M. (2011). Simulation of a Distributed Design Project. In *29th International Conference of the System Dynamics Society* (p. 16). Retrieved from http://www.systemdynamics.org/conferences/2011/proceed/papers/P1107.pdf

Savage, N. (2012). Gaining wisdom from crowds. *Communications of the ACM*, *55*(3), 13. doi:10.1145/2093548.2093553

Saye, J. W., & Brush, T. (2002). Scaffolding critical reasoning about history and social issues in multimedia-supported learning environments. *Educational Technology Research and Development*, *50*(3), 77–96. doi:10.1007/BF02505026

Schacter, D. L., Gilbert, D. T., & Wegner, D. M. (2009). *Psychology*. New York, NY: Catherine Woods. Retrieved from http://psihologos.files.wordpress.com/2012/11/2009-psychology-schacter-gilbert-wegner1.pdf

Semantum. (2014). Simupedia Web Site. Retrieved from http://www.simupedia.com/

Simons, K. L. (1990). New Technologies in Simulation Games. *System Dynamics Review*, *9*(3), 1047–1059.

Squire, K., & Barab, S. (2004). Replaying History: Engaging Urban Underserved Students in Learning World History Through Computer Simulation Games. In *Proceedings of the 6th international conference on Learning sciences* (p. 8). International Society of the Learning Science. Retrieved from http://ligafutbolnuevaesperanza.com.mx/wp-content/uploads/2010/06/School-Civilitation-3.doc

Squire, K., Giovanetto, L., Devane, B., & Durga, S. (2005). From Users to Designers: Building a Self-Organizing Game-Based Learning Environment. *TechTrends*, *49*(5), 34–42, 74.

Standish, R. K. (2008). On Complexity and Emergence, *1*, 1–6. Retrieved from http://arxiv.org/abs/nlin/0101006v1

Sterman, J. D. (1994). Learning in and about complex systems. *System Dynamics Review*, *10*(2-3), 291–330. doi:10.1002/sdr.4260100214

Sterman, J. D. (2000). *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Boston: McGraw-Hill.

THTH. (2014). Simantics System Dynamics Web Page. Retrieved November 7, 2014, from http://sysdyn.simantics.org/

van Daalen, C. E., Schaffernicht, M., & Mayer, I. (2014). System Dynamics and Serious Games. In *32nd International Conference of the System Dynamics Society* (pp. 1–26). Delft. Retrieved from http://www.systemdynamics.org/web.portal?P1141+0 von Ahn, L., & Dabbish, L. (2004). Labeling images with a computer game. *Proceedings of the 2004 Conference on Human Factors in Computing Systems*, 319–326. doi:10.1145/985692.985733

Vygotsky, L. S. (1978). *Mind in Society*. (M. Cole, V. John-Steiner, S. Scribner, & E. Souberman, Eds.). Cambridge, Massachusetts: Harvard University Press. doi:10.1007/978-3-540-92784-6

Appendix A: Questionnaire

Name: ____

Your experience

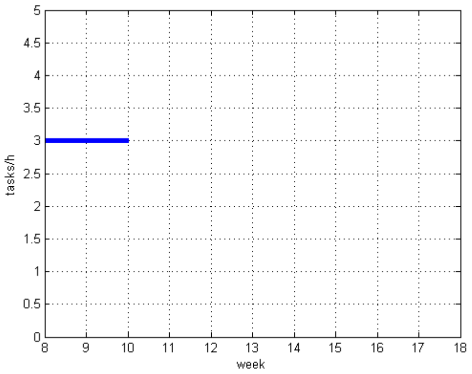
Briefly describe your experience as an EPCM project manager.

____

____

Adding people

There are three people working in the project until week 10. Their total tasks completed per hour until that moment is shown in the figure. What happens to tasks completed per hour when a fourth person is added to the project at week 10? Continue the curve in the figure. Explain.



____

____

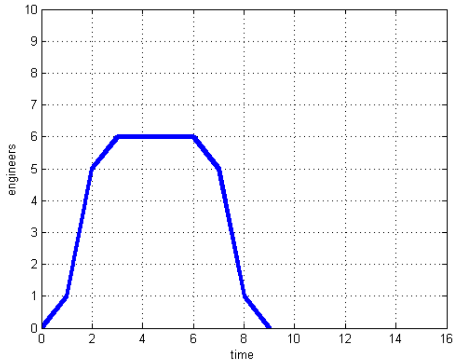
____

____

____

Disciplines

The figure is a manning plan for process engineering. What would then be the optimal manning plan for mechanical engineering? Draw the manning plan for mechanical engineering in the same figure. Explain.



____

____

____

____

____

____

Work pressure

Project seems to be behind schedule. What do project workers do to finish in schedule? What kind of impact does this have on the project? What do you do as a project manager to make the project finish in schedule? What kind of impact does this have on the project?

____

____

____

____

____

____

Appendix B: List of Model Variables

Variable	**Equation**	**Dimensions**	**Unit**
Add Project Workforce	MAX(Change Workforce, zeros(Office.size,Discipline.size,Seniority.size));	Office, Discipline, Seniority	person/day
Assimilation Rate	Project Rookie Workforce./Assimilation Time;	Office, Discipline, Seniority	person/day
Assimilation Time	DisciplineEffects(Standard Assimilation Time, Effect Of Mentoring To Assimilation Time);	Office, Discipline, Seniority	day
Available Office Resources	Office Resources-Resourcing Plan Of Other Projects;	Month, Office, Discipline, Seniority	month
Change Workforce	PlannedResourceImpulse(time, Step Size, 20.0, Resourcing Plan);	Office, Discipline, Seniority	person/day
Claim Generation	sum(Milestone Schedule And Quality Issues.*Claim Costs)		currency/day
Communication Difficulties Between Offices	interpolate(Number of Offices, {{0.0,0.0},{1.0,0.0},{2.0,0.1},{3.0,0.5}});		1
Congestion And Communication Difficulties Due To Discipline Workforce Size	interpolate(DisciplineSum(Total Workforce), {{0.0,0.0},{1.0,0.0},{7.0,0.01},{14.0,0.05},{21.0,0.15},{28.0,0.3},{35.0,0.5},{42.0,0.7},{49.0,0.85},{56.0,0.94},{63.0,0.99}});	Discipline	1
Costs	sum(DisciplineSum(Hourly Resource Prices.*Total Workforce).*Working Hours Per Day);		currency/day
Current Monthly Cumulative Costs Estimate	{Planned Monthly Cumulative Costs Estimate[month] - Planned Current Cumulative Costs Estimate + Cumulative Costs for month in 1:Month.size};	Month	currency
Current Monthly Cumulative Hours Estimate	{Planned Monthly Cumulative Hours Estimate[month,:] - Planned Current Cumulative Hours Estimate + Cumulative Hours for month in 1:Month.size};	Month, Discipline	h
Current Monthly Progress Fraction Estimate	{Monthly Progress Estimate[month,:]./Original Work - Proportional Estimate Of Working Potential Used + Perceived Progress Fraction for month in 1:Month.size};	Month, Discipline	1
der(Claims)	Claim Generation		currency
der(Client Satisfaction)	Satisfy Client;		satisfaction
der(Cumulative Costs)	Costs;		currency
der(Cumulative Errors)	Error Generation;	Discipline	task
der(Cumulative Fatigue)	(Fatigue-ones(Discipline.size)).*DisciplineSum(Total Workforce)	Discipline	1
der(Cumulative Hours)	DisciplineSum(Total Workforce).*Working Hours Per Day;	Discipline	h

der(Departure Fatigue)	DisciplineSum(MAX(-Change Workforce, zeros(3,4,3))).*Fatigue	Discipline	1
der(Original Work To Do)	-DisciplineSum(Work);		task
der(Project Experienced Workforce)	Assimilation Rate - Remove Project Experienced Workforce;	Office, Discipline, Seniority	person
der(Project Rookie Workforce)	Add Project Workforce - Assimilation Rate - Remove Project Rookie Workforce;	Office, Discipline, Seniority	person
der(Rework To Do)	Error Discovery-DisciplineSum(Rework);		task
der(Undiscovered Errors)	Error Generation - Error Discovery;		task
der(Work Done)	DisciplineSum(Rework+Work) - Error Generation;		task
Effect Of Congestion And Communication Difficulties On Error Free	((ones(Discipline.size) - 0.05*ones(Discipline.size).*Communication Difficulties Between Offices).*(ones(Discipline.size) - 0.05*Congestion And Communication Difficulties Due To Discipline Workforce Size))	Discipline	1
Effect Of Congestion And Communication Difficulties On Productivity	((ones(Discipline.size) - 0.1*ones(Discipline.size).*Communication Difficulties Between Offices).*(ones(Discipline.size) - 0.1*Congestion And Communication Difficulties Due To Discipline Workforce Size))	Discipline	1
Effect Of Fatigue On Probability Of Error Free	interpolate(Fatigue, {{0.0,1.0},{1.0,1.0},{1.2,0.98},{1.4,0.96},{1.6,0.93},{1.8,0.9},{2.0,0.86},{2.2,0.82},{2.4,0.75},{2.6,0.65},{2.8,0.43},{3.0,0.0}});	Discipline	1
Effect Of Fatigue On Productivity	LogisticFunctions.LogisticFunctionFit(1.5,0.55,0.55,1,1,Fatigue);	Discipline	1
Effect Of Heuristics On Probability Of Error Free	Heuristics(Progress Fraction, Prerequisites, Heuristic Accuracy);	Office, Discipline, Seniority	1
Effect Of Mentoring On Assimilation Time	LogisticFunctions.LogisticFunctionFit(0, 1, 0.6, 0.4, 0.5, Mentoring Per Rookie)	Discipline	1
Effect Of Unknown Errors Prior To Work On Error Free	interpolate(Proportional Undiscovered Errors, {{0.00,1.0},{0.1,0.98},{0.2,0.96},{0.3,0.93},{0.4,0.9},{0.5,0.86},{0.6,0.82},{0.7,0.75},{0.8,0.65},{0.9,0.43},{1.0,0.0}});	Discipline	1
Effect Of Work Intensity On Probability Of Error Free	interpolate(Work Intensity, {{0.0,1.0},{1.0,0.99},{1.2,0.98},{1.4,0.96},{1.6,0.93},{1.8,0.9},{2.0,0.86},{2.2,0.82},{2.4,0.75},{2.6,0.65},{2.8,0.43},{3.0,0.0}});	Discipline	1
Effect Of Work Pressure On Work Intensity	LogisticFunctions.LogisticFunctionFit(1,1,0.25,0.75,0.9,Work Pressure);	Discipline	1
Effect Of Work Pressure On Working Hours Per Day	MAX(1.0,LogisticFunctions.LogisticFunctionFit(1,1,0.25,0.75,0.9,Work Pressure));	Discipline	1

Effective Experienced Workforce	EffectiveExperienced(Project Experienced Workforce, Mentoring);	Office, Discipline, Seniority	person
Effective Workforce	Effective Experienced Workforce+Project Rookie Workforce*Rookie Productivity Fraction;	Office, Discipline, Seniority	person
Effort Applied	DisciplineEffects(Effective Workforce,Working Hours Per Day);	Office, Discipline, Seniority	h
Error Discovery	Undiscovered Errors./Error Discovery Delay;	Office, Discipline, Seniority	task/day
Error Discovery Delay	{40.0,40.0,40.0,40.0}.*LogisticFunctions.LogisticFunctionFit(0.5, 0.75, 0.25, 0.1, 0.99, Progress Fraction)	Discipline	day
Error Generation	DisciplineSum((Rework+Work) .* Probability Of Error Generation);	Discipline	task/day
Fatigue	Recent Month/Standard Working Hours Per Day;	Discipline	1
Final Profit	Lump Sum-Claims-Current Monthly Cumulative Costs Estimate[MonthFinal]		currency
Final Profit Margin	Final Profit/Lump Sum		1
Heuristics Needed	HeuristicsNeeded(Progress Fraction, Prerequisites);	Discipline	1
Mentoring	MIN(Need For Mentoring, Mentoring Available);	Discipline	person
Mentoring Available	Fraction Of Time Experienced Willing To Put In Mentoring*DisciplineSum(Project Experienced Workforce);	Discipline	person
Mentoring Per Rookie Workforce	{if sum(Project Rookie Workforce[:,i,:]) < 0.001 then 0.0 else Mentoring[i]/sum(Project Rookie Workforce[:,i,:]) for i in 1:Discipline.size};	Discipline	1
Milestone Perceived Ready	geq(Milestone Ready Tolerance, Perceived Fraction Of Work Left);	Milestone, Discipline	1
Milestone Perceived In Schedule	MilestoneInSchedule(Milestone Perceived Ready, Milestone Deadlines, time)	Milestone	1
Milestone Schedule And Quality Issues	MilestoneIssues(Progress Fraction, Milestones, Milestone Deadlines, Milestone Ready Tolerance, time)	Milestone, Discipline	1
Need For Mentoring	Fraction Of Time Rookies Asking For Mentoring.*DisciplineSum(Project Rookie Workforce);	Discipline	person
Number of Offices	CountOffices(Total Workforce);		1
Perceived Fraction Of Work Left	{Milestones[milestone,:] - Perceived Progress Fraction for milestone in 1:Milestone.size};	Milestone, Discipline	1
Perceived Progress Fraction	Perceived Work Done./Original Work;	Discipline	1
Perceived Work Done	Undiscovered Errors+Work Done;	Discipline	task
Planned Current Cumulative Costs Estimate	interpolate(time/Working Days In Month, { { floor(time/Working Days In Month), if time/Working Days In Month < 1.0 then 0.0 else Planned Monthly Cumulative Costs Estimate[floor(time/Working Days In Month)] }, { floor(time/Working Days In Month+1.0), Planned Monthly Cumulative Costs Estimate[min(Month.size,floor(time/Working Days In Month+1.0))]	Month	currency

```
}
 }
);
```

Planned Current Cumulative Hours Estimate		Month, Discipline	h

```
{interpolate(time/Working Days In Month,
 {
 {
 floor(time/Working Days In Month),
 if time/Working Days In Month < 1.0 then 0.0 else Planned Monthly
Cumulative Hours Estimate[floor(time/Working Days In Month),discipline]
 },
 {
 floor(time/Working Days In Month+1.0),
 Planned Monthly Cumulative Hours
Estimate[min(Month.size,floor(time/Working Days In Month+1.0)),discipline]
 }
 }
) for discipline in 1:Discipline.size};
```

Probability Of Error Generation	`ones(Office.size,Discipline.size,Seniority.size) - DisciplineEffects(Standard Probability Of Error Free.*Effect Of Heuristics On Probability Of Error Free, Effect Of Fatigue On Probability Of Error Free .* Effect Of Unknown Errors Prior To Work On Error Free .* Effect Of Work Intensity On Probability Of Error Free .* Effect Of Congestion And Communication Difficulties On Error Free) ;`	Office, Discipline, Seniority	1
Productivity	`Work Intensity.*Effect Of Fatigue On Productivity .* Effect Of Congestion And Communication Difficulties On Productivity;`	Discipline	1
Profit Estimate	`Lump Sum-Current Monthly Cumulative Costs Estimate[25];`		currency
Progress Fraction	`Work Done./Original Work;`	Discipline	1
Proportional Cumulative Errors	`Cumulative Errors./Original Work;`	Discipline	1
Proportional Estimate Of Working Potential Left	`MilestoneEffort(Monthly Progress Estimate, Proportional Estimate Of Working Potential Used, Milestone Deadlines, Working Days In Month, Original Work);`	Milestone, Discipline	1
Proportional Estimate Of Working Potential Used		Discipline	1

```
{interpolate(time/Working Days In Month,
 {
{
 floor(time/Working Days In Month),
 if time/Working Days In Month < 1.0 then 0.0 else Monthly Progress
Estimate[floor(time/Working Days In Month),discipline]
 },
{
 floor(time/Working Days In Month+1.0),
 Monthly Progress Estimate[min(Month.size,floor(time/Working Days In
Month+1.0)),discipline]
}
 }
) for discipline in 1:Discipline.size}./Original Work;
```

Proportional Rework To Do	`Rework To Do./Original Work;`	Discipline	1
Proportional Undiscovered Errors	`Undiscovered Errors./Original Work;`	Discipline	1
Remove Project Experienced Workforce	`MAX(-Change Workforce, zeros(3,4,3))-Remove Project Rookie Workforce;`	Office, Discipline, Seniority	person/day
Remove Project	`MIN(MAX(-Change Workforce, zeros(3,4,3)), Project Rookie Workforce);`	Office, Discipline, Seniority	person/day

Rookie Workforce			
Reserved Profit Estimate	Lump Sum-Reservation-Current Monthly Cumulative Costs Estimate[25];		currency

Resourcing Plan Of Other Projects	{
	if t <= ceil(time/Working Days In Month+1.0) then
	Office Resources[t,:,:,:]-Resourcing Plan[t,:,:,:]
	else
	MIN(round(LogisticFunctions.LogisticFunctionFit(6, 0.5, 0.5, 0.0, 0.99, t-time/Working Days In Month)*Office Resources[t,:,:,:]), Office Resources[t,:,:,:] - Resourcing Plan[t,:,:,:])
	for t in 1:Month.size
	}
		Month,
		Office,
		Discipline,
		Seniority
			month

Rework	WorkFunction(Effort Applied./Effort Per Outcome, Productivity, Rework To Do, Step Size);	Office, Discipline, Seniority	task/day
Score	Final Profit+sum(Departure Fatigue)+sum(Cumulative Fatigue)+Client Satisfaction		1
Satisfy Client	sum(Milestone In Schedule);		satisfaction/day
Time	time;		day
Total Workforce	Project Experienced Workforce+Project Rookie Workforce;	Office, Discipline, Seniority	person

Upper Minimum Resources	{
	if t <= ceil(time/Working Days In Month+1.0) then
	Resourcing Plan[t,:,:,:]
	else
	round(LogisticFunctions.LogisticFunctionFit(6, 0.5, 0.5, 0.0, 0.99, t-time/Working Days In Month)*Resourcing Plan[t,:,:,:])
	for t in 1:Month.size
	}
	if t <= ceil(time/Working Days In Month+1.0) then
	Resourcing Plan[t,:,:,:]
	else
	round(LogisticFunctions.LogisticFunctionFit(6, 0.5, 0.5, 0.0, 0.99, t-time/Working Days In Month)*Resourcing Plan[t,:,:,:])
	for t in 1:Month.size
	};
		Month,
		Office,
		Discipline,
		Seniority
			month

Work	WorkFunction((Effort Applied - Rework)./Effort Per Outcome, Productivity, Original Work To Do, Step Size);	Office, Discipline, Seniority	task/day
Work Intensity	Effect Of Work Pressure On Work Intensity;	Discipline	1
Work Pressure	WorkPressure(Proportional Estimate Of Working Potential Left, Perceived Fraction Of Work Left, Milestone Deadlines, Milestone Ready Tolerance, time);	Discipline	1
Working Hours Per Day	Standard Working Hours Per Day.*Effect Of Work Pressure On Working Hours Per Day;	Discipline	h/day

Appendix C: List of Model Parameters

Parameter	**Dimensions**	**Unit**
Available Office Resources	Month, Office, Discipline, Seniority	month
Claim Costs	Milestone, Discipline	currency/day
Effort Per Outcome	Office, Discipline, Seniority	h/task
Fraction Of Time Experienced Willing To Put In Mentoring		1
Fraction Of Time Rookies Asking For Mentoring		1
Heuristic Accuracy	Office, Seniority	1
Hourly Resource Prices	Office, Discipline, Seniority	currency/h
Hours Target	Discipline	h/task
Lump Sum		currency
Milestone Deadlines	Milestone	day
Milestone Ready Tolerance	Milestone, Discipline	1
Milestones	Milestone, Discipline	1
Monthly Progress Estimate	Month, Discipline	task
Office Resources	Month, Office, Discipline, Seniority	month
Original Work	Discipline	task
Planned Monthly Cumulative Costs Estimate	Month	currency
Planned Monthly Cumulative Hours Estimate	Month, Discipline	h
Prerequisites	PrerequisiteProgress, Discipline, PrerequisiteDiscipline	1
Reservation		currency
Resourcing Plan	Month, Office, Discipline, Seniority	month
Rookie Productivity Fraction		1
Standard Assimilation Time	Office, Discipline, Seniority	day
Standard Probability Of Error Free	Office, Discipline, Seniority	1
Standard Working Hours Per Day		h/day
Step Size		day
Task Prices For Calculating Rework And Extra Work Cost Estimates	Discipline	currency/task
Upper Minimum Resources Copy	Month, Office, Discipline, Seniority	month
Working Days In Month		day/month

Appendix D: List of Model Functions

Function name | **Function code** | **Documentation**

LogisticFunctionFit

```
input Real x_mean(unit = "'p") "x coordinate of
the mean of the logistic curve";
input Real y_mean(unit = "1") "y coordinate of the
mean of the logistic curve";
input Real d(unit = "1") "distance from the
logistic curve mean to the asymptote (d > 0)";
input Real x_1(unit = "'p") "x coordinate of a
point on the logistic curve (x_1 != x_mean)";
input Real y_1(unit = "1") "y coordinate of a
point on the logistic curve  (y_mean - d < y_1 <
y_mean + d, y_1 != y_mean)";
input Real x(unit = "'p") "input of the logistic
function";
output Real y(unit = "1") "output of the logistic
function";
protected Real scale(unit = "1") "scale parameter
of the logistic function";
algorithm
scale := (x_mean-x_1)/log((d+y_mean-y_1)/(d-
y_mean+y_1));
y := ((1/(1+exp((x_mean-x)/scale))) -
0.5)*2*d+y_mean;
```

LogisticFunctionFit(x_mean, y_mean, d, x_1, y_1, x)

Creates and fits a logistic function with given parameters and returns its value with the given input x.

CountOffices

```
input Real[:,:,:] workforce (unit = "person")
"Array of workforce in different offices,
disciplines and seniorities";
protected Integer[size(workforce,1)] officeUsed
(unit = "1") "Keeps track on whether there are
resources from certain offices";
public output Integer usedOffices (unit = "1")
"Number of offices involved in the project";
algorithm
usedOffices := 0;
officeUsed := zeros(size(workforce,1));
for office in 1:size(workforce,1) loop
  for discipline in 1:size(workforce,2) loop
    for seniority in 1:size(workforce,3) loop
      if workforce[office,discipline,seniority] >
0.5 then
        officeUsed[office] := 1;
        break;
      end if;
    end for;
  end for;
end for;
usedOffices := sum(officeUsed);
```

Counts the number of offices in the project

CumulativeCosts

```
input Real[:,:,:,:] plan (unit = "month")
"Resourcing plan";
input Real[:,:,:] prices (unit = "currency/h")
"Resource prices";
input Real workingDaysInMonth (unit = "day/month")
"Number of working days in month";
input Real hoursPerDay (unit = "h/day") "Number of
working hours per day";
output Real[size(plan,1)] cumulativeCosts (unit =
"currency") "Monthly array of cumulative costs";
algorithm
cumulativeCosts := zeros(size(plan,1));
cumulativeCosts[1] := sum(plan[1,:,:,:] .*
prices);
for month in 2:size(plan,1) loop
  cumulativeCosts[month] := cumulativeCosts[month-
1] + sum(plan[month,:,:,:] .* prices);
end for;
cumulativeCosts := cumulativeCosts *
workingDaysInMonth * hoursPerDay;
```

Returns monthly array of cumulative costs.

CumulativeHours

```
input Real[:,:,:,:] plan (unit = "month")
"Resourcing plan";
input Real workingDaysInMonth (unit = "day/month")
"Number of working days in month";
input Real hoursPerDay (unit = "h/day") "Number of
working hours per day";
output Real[size(plan,1),size(plan,3)]
cumulativeHours (unit = "hour") "Monthly array of
cumulative hours in each discipline";
algorithm
cumulativeHours :=
zeros(size(plan,1),size(plan,3));
for discipline in 1:size(plan,3) loop
  cumulativeHours[1,discipline] :=
sum(plan[1,:,discipline,:]);
  for month in 2:size(plan,1) loop
    cumulativeHours[month,discipline] :=
cumulativeHours[month-1,discipline] +
sum(plan[month,:,discipline,:]);
  end for;
end for;
cumulativeHours := cumulativeHours *
workingDaysInMonth * hoursPerDay;
```

Returns monthly array of cumulative hours in each discipline.

DisciplineEffects

```
input Real[:,:,:] availableEffort (unit = "'p")
"Array of available effort from different offices,
disciplines and seniorities";
input Real[:] fraction (unit = "'p") "Array of
fractions in each discipline";
public output
Real[size(availableEffort,1),size(availableEffort,
2),size(availableEffort,3)] effort (unit =
"person*'p") "Product";
algorithm
effort :=
zeros(size(availableEffort,1),size(availableEffort
,2),size(availableEffort,3));
for i in 1:size(availableEffort,2) loop
  effort[:,i,:] :=
availableEffort[:,i,:]*fraction[i];
end for;
```

For each discipline multiplies the effort matrix with corresponding scalar in fraction array.

DisciplineSum

```
input Real[:,:,:] availableEffort (unit = "'p")
"Array of available effort from different offices,
disciplines and seniorities";
public output Real[size(availableEffort,2)] effort
(unit = "'p") "Sum of effort for each discipline";
algorithm
effort := zeros(size(availableEffort,2));
for i in 1:size(availableEffort,2) loop
  effort[i] := sum(availableEffort[:,i,:]);
end for;
```

Returns the sum of the effort matrix for each discipline.

EffectiveExperienced

```
input Real[:,:,:] workforce (unit = "person")
"Array of available effort from different offices,
disciplines and seniorities";
input Real[:] mentoring (unit = "person") "Array
of mentoring in each discipline";
public output
Real[size(workforce,1),size(workforce,2),size(work
force,3)] effectiveWorkforce (unit = "person")
"Array of available effort after mentoring";
algorithm
effectiveWorkforce :=
zeros(size(workforce,1),size(workforce,2),size(wor
kforce,3));
for i in 1:size(workforce,2) loop
  if sum(workforce[:,i,:]) > 0.5 then
    effectiveWorkforce[:,i,:] := workforce[:,i,:]-
(workforce[:,i,:]/sum(workforce[:,i,:])).*mentorin
g[i];
  end if;
end for;
```

Some of the workforce effort goes in mentoring project rookies. Function returns the effective workforce after mentoring.

geq

```
input Real a (unit = "'p");
input Real b (unit = "'p");
public output Integer c (unit = "'1");
algorithm
if a >= b then
c := 1;
else
c := 0;
end if;
```

Greater or equal than function (for elementwise comparison of matrices).

gt

```
input Real a (unit = "'p");
input Real b (unit = "'p");
public output Integer c (unit = "'1");
algorithm
if a > b then
c := 1;
else
c := 0;
end if;
```

Greater than function (for elementwise comparison of matrices).

Heuristics

```
input Real[:] progressFraction (unit = "1")
"Progress fraction of each discipline";
input Real[:,:,:] prerequisites (unit = "1")
"Prerequisites of progress fractions of other
disciplines for each discipline";
input Real[:,:] accuracy (unit = "1") "Heuristic
accuracy of workers in each office and seniority";
output
Real[size(accuracy,1),size(progressFraction,1),siz
e(accuracy,2)] heuristics (unit = "1") "Array of
heuristics accuracy when heuristics needed and 1.0
when heuristics not needed";
algorithm
heuristics :=
ones(size(accuracy,1),size(progressFraction,1),siz
e(accuracy,2));
for discipline in 1:size(progressFraction,1) loop
  for prerequisite in 1:size(progressFraction,1)
loop
    if
      prerequisite <> discipline
      and
interpolate(progressFraction[discipline],
      {
        {0.0,
prerequisites[1,discipline,prerequisite]},
        {0.5,
prerequisites[2,discipline,prerequisite]},
        {1.0,
prerequisites[3,discipline,prerequisite]}
      })
      > progressFraction[prerequisite]
      then
      heuristics[:,discipline,:] := accuracy;
      break;
    end if;
  end for;
end for;
```

Checks whether heuristics are needed i.e. whether prerequisites are fulfilled. If heuristics are not needed the output in the discipline is 1.0 but if heuristics are needed the output for the discipline will be according to the accuracy array.

HeuristicsNeeded

```
input Real[:] progressFraction (unit = "1")
"Progress fraction of each discipline";
input Real[:,:,:] prerequisites (unit = "1")
"Prerequisites of progress fractions of other
disciplines for each discipline";
output Integer[size(progressFraction,1)]
areHeuristicsNeeded (unit = "1") "Boolean
(integer) array that tells if heuristics are
needed in certain discipline";
algorithm
areHeuristicsNeeded :=
fill(0,size(progressFraction,1));
for discipline in 1:size(progressFraction,1) loop
  for prerequisite in 1:size(progressFraction,1)
loop
    if
```

Returns 0 if heuristics are not needed i.e. when prerequisites are fulfilled and 1 when heuristics are needed.

```
      prerequisite <> discipline
      and
interpolate(progressFraction[discipline],
      {
        {0.0,
prerequisites[1,discipline,prerequisite]},
        {0.5,
prerequisites[2,discipline,prerequisite]},
        {1.0,
prerequisites[3,discipline,prerequisite]}
      })
      > progressFraction[prerequisite]
      then
      areHeuristicsNeeded[discipline] := 1;
      break;
    end if;
  end for;
end for;
```

MilestoneEffort

```
input Real[:,:] progressEstimateArray (unit =
"task") "Estimation array of tasks completed at
the end of each month in each discipline";
input Real[:] effortDone (unit = "task") "How much
of the estimated estimated effort is behind";
input Real[:] dl (unit = "day") "Deadline dates";
input Real workingDaysInMonth (unit = "day/month")
"Number of working days in month";
input Real[:] originalWork (unit = "task")
"Original number of tasks in each discipline";
output Real[size(dl,1), size(effortDone,1)]
effortLeft (unit = "1") "Estimate fraction of
number of tasks that could be completed before
each deadline in each discipline";
algorithm
effortLeft := zeros(size(dl,1),
size(effortDone,1));
for milestone in 1:size(dl,1) loop
  for discipline in 1:size(effortDone,1) loop
    if floor(dl[milestone]/workingDaysInMonth) <
1.0 then
      effortLeft[milestone,discipline] :=
interpolate(dl[milestone]/workingDaysInMonth,
      {
        {floor(dl[milestone]/workingDaysInMonth),
0.0},
        {floor(dl[milestone]/workingDaysInMonth +
1.0), progressEstimateArray[floor(dl[milestone] /
workingDaysInMonth + 1.0), discipline]}
      }) ./ originalWork[discipline] -
effortDone[discipline];
    else
      effortLeft[milestone,discipline] :=
interpolate(dl[milestone]/workingDaysInMonth,
      {
        {floor(dl[milestone]/workingDaysInMonth),
progressEstimateArray[floor(dl[milestone] /
workingDaysInMonth), discipline]},
        {floor(dl[milestone]/workingDaysInMonth +
1.0), progressEstimateArray[floor(dl[milestone] /
workingDaysInMonth + 1.0), discipline]}
      }) ./ originalWork[discipline] -
effortDone[discipline];
    end if;
  end for;
end for;
```

Returns a fraction of the estimated number of tasks that could be completed before each deadline in each discipline.

MilestoneInSchedule

```
input Real[:,:] ready (unit = "1") "Boolean
(integer) array of milestone readiness in each
discipline";
input Real[:] dl (unit = "day") "Deadline dates";
input Real Time (unit = "day") "Time variable";
output Real[size(ready,1)] inSchedule (unit = "1")
"Boolean (integer) array of milestone being in
schedule";
```

Returns 1.0 if milestone is completed ahead schedule, -1.0 if milestone has not been completed before deadline, else 0.0.

```
algorithm
inSchedule := ones(size(ready,1));
for milestone in 1:size(ready,1) loop
  for discipline in 1:size(ready,2) loop
    if inSchedule[milestone] > -1 then
      if (Time > dl[milestone] and
ready[milestone,discipline] == 1)  or (Time <=
dl[milestone] and ready[milestone,discipline] ==
0) then
        inSchedule[milestone] := 0;
      end if;
      if Time > dl[milestone] and
ready[milestone,discipline] == 0 then
        inSchedule[milestone] := -1;
      end if;
    end if;
  end for;
end for;
```

MilestoneIssues

```
input Real[:] progressFractionArray (unit =
"task") "Array of tasks completed at the end of
each month in each discipline";
input Real[:,:] milestones (unit = "1") "Array of
milestones for each discipline";
input Real[:] dl (unit = "day") "Deadline dates";
input Real[:,:] tolerance (unit = "1") "Milestone
ready tolerances";
input Real Time (unit = "day") "Time variable";
output Real[size(milestone,1),size(milestone,2)]
issues (unit = "1") "Boolean (integer) array of
milestone being in schedule";
algorithm
issues :=
zeros(size(milestones,1),size(milestones,2));
for milestone in 1:size(milestones,1) loop
  for discipline in 1:size(milestones,2) loop
    if Time > dl[milestone] then
      issues[milestone,discipline] := max(0.0,
milestones[milestone,discipline] -
progressFractionArray[discipline] -
tolerance[milestone,discipline]);
    else
      issues[milestone,discipline] := 0.0;
    end if;
  end for;
end for;
```

PlannedResourceImpulse

```
input Real time (unit = "day") "Time variable";
input Real stepSize (unit = "day") "Step size
variable";
input Real resourcingInterval (unit = "day/month")
"Time interval between resource impulses";
input Real[:,:,:,:] plan (unit = "person")
"Resourcing plan";
protected
Real[size(plan,2),size(plan,3),size(plan,4)] prev
(unit = "month") "Resourcing plan for previous
month";
protected
Real[size(plan,2),size(plan,3),size(plan,4)] now
(unit = "month") "Resourcing plan for current
month";
protected
Real[size(plan,2),size(plan,3),size(plan,4)]
derivative (unit = "month") "Difference in
resourcing plans between current and previous
month";
public output
Real[size(plan,2),size(plan,3),size(plan,4)]
impulse (unit = "person/day") "Resource impulse
released at the beginning of each resourcing
interval i.e. month";
algorithm
```

Function for adding and removing people in the project according to the resourcing plan. Releases impulse at the beginning of each resourcing interval i.e. month.

```
impulse := zeros(size(plan,2),size(plan,3),size(plan,4));
if time/resourcingInterval - integer(time/resourcingInterval) < 0.9*stepSize/resourcingInterval then
  if integer(time/resourcingInterval) == 0 then
    prev := zeros(size(plan,2),size(plan,3),size(plan,4));
  else
    prev := plan[integer(time/resourcingInterval),:,:,:];
  end if;
  now := plan[integer(time/resourcingInterval)+1,:,:,:];
  derivative := now - prev;
  impulse := derivative / stepSize;
end if;
```

ProgressEstimate

```
input Real[:,:,:] EPO (unit = "h/task") "Effort per outcome";
input Real[:,:,:,:] plan (unit = "month") "Resourcing plan";
input Real workingDaysInMonth (unit = "day/month") "Number of working days in month";
input Real hoursPerDay (unit = "h/day") "Number of working hours per day";
public output Real[size(plan,1),size(plan,3)] forecast (unit = "task") "Resourcing plan";
algorithm
forecast := zeros(size(plan,1),size(plan,3));
for month in 1:size(plan,1) loop
  for discipline in 1:size(plan,3) loop
    if month > 1 then
      forecast[month,discipline] := forecast[month-1,discipline] + sum(plan[month,:,discipline,:] ./ EPO[:,discipline,:]);
    else
      forecast[month,discipline] := sum(plan[month,:,discipline,:] ./ EPO[:,discipline,:]);
    end if;
  end for;
end for;
forecast := forecast * workingDaysInMonth * hoursPerDay;
```

Returns a monthly estimation array of tasks completed at the end of each month in each discipline.

round

```
input Real n (unit = "'p");
output Integer r (unit = "'p");
algorithm
r := integer(n+0.5);
```

Function for rounding a number.

WorkFunction

```
input Real[:,:,:] taskCompletionPotential (unit = "task") "Array of potential to complete tasks";
input Real[:] productivity (unit = "1") "Array of productivity in each discipline";
input Real[:] toDo (unit = "task") "Array of work to do in each discipline";
input Real stepSize (unit = "day") "Step size variable";
public output Real[size(taskCompletionPotential,1),size(taskCompletionPotential,2),size(taskCompletionPotential,3)] effort;
algorithm
effort := zeros(size(taskCompletionPotential,1),size(taskCompletionPotential,2),size(taskCompletionPotential,3));
for i in 1:size(taskCompletionPotential,2) loop
  if sum(taskCompletionPotential[:,i,:]*productivity[i]) < toDo[i] / stepSize then
    effort[:,i,:] := taskCompletionPotential[:,i,:]*productivity[i];
```

Function makes sure that all the work that can be completed is completed, no more, no less. Prevents the To Do stocks from going below zero.

```
  else
    if toDo[i] > 0.0 and
sum(taskCompletionPotential[:,i,:]) > 0.0 then
      effort[:,i,:] := toDo[i] / stepSize *
taskCompletionPotential[:,i,:] /
sum(taskCompletionPotential[:,i,:]);
    end if;
  end if;
end for;
```

WorkPressure

```
input Real[:,:] effortLeft (unit = "1")
"Fractional available potential for completing
each milestone in each discipline";
input Real[:,:] workLeft (unit = "1") "Fraction of
work left in each milestone in each discipline";
input Real[:] dl (unit = "day") "Deadline dates";
input Real[:,:] tolerance (unit = "1") "Tolerance
for each milestone in each discipline for
milestone completion";
input Real Time (unit = "day") "Time variable";
protected Real maxPressure := 4.0 "Maximum allowed
pressure";
output Real[size(workLeft,2)] pressure (unit =
"1") "Work pressure in each discipline";
algorithm
pressure := zeros(size(workLeft,2));
for discipline in 1:size(workLeft,2) loop
  for milestone in 1:size(workLeft,1) loop
    if workLeft[milestone,discipline] >
tolerance[milestone,discipline] then
      if Time > dl[milestone] then
        pressure[discipline] := maxPressure;
      else
        pressure[discipline] := max(
          pressure[discipline],
          min(
            maxPressure,
            max(0.0, workLeft[milestone,
discipline]) / effortLeft[milestone, discipline]
          )
        );
      end if;
    end if;
  end for;
end for;
```

Calculates work pressure for each discipline. Closer the deadline, less the effective resources available before deadline and more the work left the bigger the pressure.