university of groningen

faculty of mathematics and natural sciences

FASD: A Functional Architecture for Serious game Development

Bachelor Thesis Computer Science

August 6, 2016

Student: V.R. (Vincent) Velthuizen

Primary supervisor: Prof. Dr. Ir. M. (Marco) Aiello

Secondary supervisor: Dr. N.B. (Nick) Szirbik

Contents

Chapter 1

Introduction

For a long time games have been associated with not only fun, but also learning. A game like chess, for example, teaches and trains strategic military thinking. This has led to the more modern concept of the "serious game". "Serious game" is a term with many definitions and to help narrow our perspective this thesis will look at one of several types of serious games which will be referred to as "interactive simulations". In an "interactive simulation" the goal is to simulate some domain, but, instead of simulating all human actions within this domain, actual humans will make some of those decisions. Having humans making those decisions introduces some uncertainty in the outcome, in exchange for credibility if an appropriate group of humans (players) was selected.

Aim of this thesis

An important down-side of "interactive simulations" are the costs associated with developing and playing these games. If a game can be developed to serve multiple researchers, this will help share the financial and organisational load. To help do this in the best way possible it would be good to learn from past experience, giving a solid starting point for any development effort, and a touchstone for checking if the project is on track. The goal of this preservation of information is not to force a certain implementation. Instead the purpose is to conserve the functional elements for reimplementation in a subsequent project. This might lead to reuse of implementation, but does not mandate it.

In other words, the question that this thesis would like to answer is: can we provide guidelines to help speed up the development of serious games and make the resulting design qualitatively better? Such a set of guidelines would not describe a general design for serious games, but at a more meta level describe how to achieve a good design.

Methodology

Systems engineering, a field concerning itself with the integration of different disciplines into a successful system, provides tools for very high-level description of such a system. One such tool is functional analysis and the resulting artefact is typically a functional architecture. A functional architecture describes the different functions that need to be performed, leaving out who, how and where they should be performed. Seeing the overall structure can inform implementation choices regarding this, but do not enforce them to still be compliant with the architecture. Such an architecture is continuously adapted over time as more experience is gathered within it.

In order to have an architecture that can help solve the problems described before, it should be based on a project that had to deal with many of those problems. GISEGas is a game that was used by multiple researchers each with their own needs. Because of the multitude of researchers, and the possibility of more joining later, the game was designed to be as generic as possible, with regards to initial setup, player groups, run time, etc., while also implementing some advanced domain specific concepts, the domain being: energy from (bio-)gas. Analysing this game should result in a useful architecture. The artefact helps the design of new interactive simulations by preparing the designers for the problems they are likely to encounter and allow them to make sure their design can withstand them. This will prevent late design changes and the high cost associated with them. The architecture can be used and subsequently be improved upon by incorporating additional lessons learned by the designers who have used it.

The functional architecture, together with the guidelines for using it form a framework. The framework will help future serious game development projects that use it to achieve better results faster. The framework is named the: Functional Architecture for Serious game Development Framework or "FASD Framework".

Structure

The structure of this thesis is as follows: In Chapter 2 the concept of "serious games" in general and "interactive simulations" more specifically is explored further. It results in the research question: What development guidelines are useful for creating an interactive simulation? In order to answer that question, it is first explored how to answer that question. Chapter 3 delves deeper into systems engineering, functional analysis, functional architecture and how they can be applied to answer the research question. To apply these techniques however a case is needed that can be studied. The GISEGas project is explained further in Chapter 4.

Chapter 5 shows and explains the functional architecture resulting from the analysis of the GISEGas project. This architecture is the core of FASD, but by itself not yet very useful. Guidelines for using FASD are presented in Chapter 6. Together the architecture and guidelines form the FASD framework that will help develop future serious games. Chapter 7 presents a different serious game and shows an initial validation of the framework. Finally Chapter 8 will conclude this thesis.

Chapter 2

Background

In 2014 the serious games market was 2.46 billion dollars [20]. For an industry of that size, however, there is no clear single definition of what constitutes a serious game. New definitions are still being offered like by Møller in 2016 [17]. What is clear is that there are several types of serious games:

Interactive Simulation Simulate situations and let humans interact with those to multiple ends amongst which: learning how players react, learning how players interact, allowing players to learn from otherwise costly mistakes. Useful for exploring system, process and/or organisational (re)design from a development or investment perspective.

Educational This type of game is used to supplement or replace other teaching materials and used to convey knowledge.

Gamification Projects that use this are not necessarily games, but use gaming elements to help motivate certain behaviour.

In 2016 there are no "holistic" development frameworks to aid the design of serious games, especially the type of game the focus is on here: interactive simulation.

This chapter first explores the definition of serious game, concluding in a definition for a "serious game" or as it is specified an "interactive simulation". After that the need for examining such a system and creating a development framework for it is discussed.

2.1 Defining Serious Games

Common definitions for serious games include the use of a computer [17], and are similar to that of Susi: "(digital) games used for purposes other than mere entertainment" [22]. As a high level definition this works, but, when narrowing down the scope, more detail is needed. The definition by Susi does not state what the (primary) goal is, merely that it is not entertainment. A consequence of playing a game, whether you win or lose, and whether that has real-world effects, or not, is: the player gains experience playing the game. If the game reflects some real-world scenario, the experience gained can help the player learn, and seeing what a player does can help an outside party learn about how people in general may react to certain situations. In short, playing a game can help us gain knowledge. The following is a list of types of learning that can happen from a (serious) game.

Player learning explicit knowledge This type of learning occurs when the knowledge that needs to be transferred is explicitly known. The intended "lesson" is programmed into the game steering the player to a certain outcome [22]. A reason to employ this type of game is to allow the learner to fail and learn from his/her mistakes in a safe environment where making those mistakes has no real world consequences.

Player learning implicit knowledge When the knowledge is not explicitly known this type of learning can occur. Implicit knowledge tends to come from experience, sometimes getting that experience is not achievable in the real world. Games can allow for these experiences by for example: speeding up time, or creating extreme circumstances. In such games a learner, possibly with help from a trainer or expert, can acquire the needed experience.

Observer learning implicit knowledge In games like described for "Player learning implicit knowledge" an observer can attempt to make the knowledge being acquired explicit, for example by using data mining techniques [21]. This also has applications with lay-man players to create completely new insight in a topic.

Observer learning group behaviour Multi player games allow an observer to research how groups of people can try to reach a common goal, or how they can collaborate/compete with each other on individual goals.

Besides a definition, Susi also gives a goal for serious games: "Experience situations that are not possible in the real world due to: safety, cost, time constraints" [22]. Combining the learning discussed before with this goal leads to the definition of serious games that this thesis will use:

> **"A system that allows players to experience and learn from situations that are not feasible in the real world".**

Because such a system, as described above, needs to reflect the real world in some way and such a system would need to accurately reflect the real world in order to lend credence to these experiences [21], it can also be considered a simulation. Having players influence this simulation leads to it being interactive, resulting in the term "interactive simulation" to be used to describe (serious) games adhering to the definition given above.

Many serious games, including interactive simulations, focus only on the first type of learning (Player learning explicit knowledge).

2.2 State of the art

At the start of the GISEGas project a literature research was done and no helpful frameworks were found. During the last stages of writing this thesis the literature research was repeated using Google Scholar to see if the state of the art has changed. Terms used were: "Interactive Simulation Design", "Interactive Simulation Framework", "Serious Game Design", and "Serious Game Framework". A framework for interactive simulation was not found.

As stated there is not much literature on interactive simulations. For a good overview of the state of the art the existing literature is presented below in four categories. The first category covers any relevant literature on interactive simulations. The second covers educational games and the third the use of gamification. Finally some literature regarding "regular" games is presented.

2.2.1 Interactive simulations

Papers describing interactive simulations cover the following three main subjects.

The first subject is the psychology of the learner, exploring how interactive simulations can be more effective than existing tools and where they fall short. An example of this is the work by Chittaro and Sioni [8] showing how interactive and non-interactive simulations of terror attacks can be used to prepare the population.

The second subject deals with using innovative interfaces powered by interactive simulations to aid learning. This is well illustrated by the work-in-progress paper by Oh, So, Park, and Kwan [19]. They present the work they are doing on a museum exhibit describing how light travels in a multi-user interactive simulation using augmented reality.

The third subject is that of subsystems. For example the adaptive support feature by Kardan en Conati [15]. They describe how they provide feedback to a learner in an interactive simulation, this is not trivial due to the lack of a clear narrative in such a learning situation.

Though all of these subjects provide valuable information about interactive simulations, none aid the development of a new one. Additionally, most of this type of work is specific to a certain game, and does not translate easily to other applications of interactive simulation.

2.2.2 Educational games

Over the past years a lot of research has been done on the design of educational games. These games do not try to generate new knowledge, but focus on transferring explicit knowledge or skills to learners in an efficient way. In 2015 Carvalho [6] proposed an "activity theory-based model for serious games" (ATMSG) for analysing and design serious games. It is based largely on the LM-GM model. These frameworks provide most value to a designer working on an interactive simulation. For example a service-oriented architecture was setup around it also by Carvalho [5], which could prove useful when designing UI elements for an interactive simulation.

Other frameworks that are designed for educational games but are not a good fit for interactive simulation are:

- Mechanics, Dynamics, Aesthetics (MDA) Framework [13]
- Hierarchical Activity Based Scenario (HABS) [16]
- The Game Ontology Project (GOP) [26]
- The Four-Dimensional Framework [9]
- The Game Object Model II (GOM II) [2]

2.2.3 Gamification

Gamification is the practice of using tropes from games to engage people in a certain process. There are many ways this can be done. For example, by using positive or negative reinforcement for certain tasks. Having counters run down for unwanted activity and awarding points to stimulate behaviour. The field of gamification deals a lot with expectation, since it is trying to address an unconscious part of the brain to have people work harder, live healthier, etc.

Mora [18] gives a literature review of several gamification frameworks. A table is presented where the different properties of the frameworks are compared. Since gamification tries to manipulate users into certain behaviours, using it in a research context might skew the results. However when used to reinforce gameplay it can help make the game more interesting, only influencing aspects of player behaviour the research is not interested in.

2.2.4 Commercial games

There are books on game design, but due to the commercial nature of blockbuster games the companies making them are interested in keeping the state-of-the-art processes that develop them proprietary, and not much is published about this. Ushaw [24] draws parallels between games from big publishers and a few examples of medical games. These health related games focus mostly on either gamification (to encourage certain exercises) or educational games and therefore are similarly unfit for our purpose as the ones described above.

2.3 The need for high level design in serious games

Budgets for research are always constrained, and making games is expensive. In 2008 Anderson et al, called for general research into reusable components for games. He states: "The available research has mainly focused on game engine subsystems, such as rendering, AI (artificial intelligence) or networking. However, issues regarding the overall architecture of engines, which connects these subsystems, have merely been brushed over." [3].

The tools and implementations available for multi-player support and rendering computer graphics are already available. However, the engine of an interactive simulation is always highly specific to the research needs. Tools for creating such an engine are hard to find. Many research/development teams using interactive simulation are reinventing the proverbial wheel.

The research question for this thesis is: What development guidelines are useful for creating an interactive simulation? ($RQ1$). To answer this question another question has to be answered first: What form should such guidelines have? ($RQ1_a$).

By answering those question this thesis responds to Andersons research call at a very abstract level. In it a framework for design that can support any combination of the types of learning mentioned before is presented. Such a framework will help future serious game projects find design problems earlier, when they are cheaper to fix.

Chapter 3

Research Method

To answer the research question presented in the previous chapter, first the sub-question needs to be answered: what form should the answer be given in? In this chapter the fields of “Design Science”, “Design Science Research” and “Systems Engineering” as well as the technique for creating a functional design are introduced and shown to deliver a high quality design document: a functional architecture, answering $RQ1_a$.

Section 3.1 covers “Design Science” and “Design Science Research” as described by Hevner [1, 11] and Wieringa [25]. Subsequently, Section 3.2 gives an overview of the “Systems Engineering” discipline. The type of document produced by systems engineering is known as a “Functional Design”, or “Functional Architecture” and is explained in Section 3.3. Finally Section 3.4 describes the modelling language used to write the functional design in: IDEF0.

3.1 Design Science Research

Design science uses a scientific approach to solve design problems. It can be used for any engineering problem though within design science there are specialisations for the different types of engineering. Design science research (DSR) takes a scientific look at design science, essentially a meta approach to design science determining how to design design science. The relationship between “real-world problem”, “design science”, and “design science research” is captured by Hevner [1] in the design science research cycles described in section 3.1.2. In section 3.1.3 the application of this to the design problem of interactive simulations is discussed.

3.1.1 DSR Glossary

Within DSR some words are used that have less well defined definitions outside of design science research. Since this text will use those words with there strict definitions, those definitions are given here.

design artefact A number of disciplines have design as the central element of what they do. Examples of such disciplines are: Architecture, Computer Science, Software Engineering, and Systems Engineering. Not every discipline uses the same definition of artefact [11]. Here we consider the working design, be it a system in use, a compiled program or a built building to be the result of applying the design artefact: the model for the system, the source code, or the blueprint.

meta artefact An artefact that when applied results in another meta artefact or a design artefact. An example is a software architecture, which informs designing source code before it can be compiled into an application.

design problem When the solution to a problem needs a design artefact, the designing of that artefact is what is called a design problem. It is a subset of the larger problem, but should involve all aspects of it. For example: if the problem requires a building, the challenges in actually creating that building inform the design process and resulting blue prints.

3.1.2 Design Science Research Cycles

This section describes the elements of the DSR cycles as depicted in Figure 3.1. First all of the domains (the rectangles) and then the cycles. The domains are: Environment, Design Science Research and Knowledge Base.

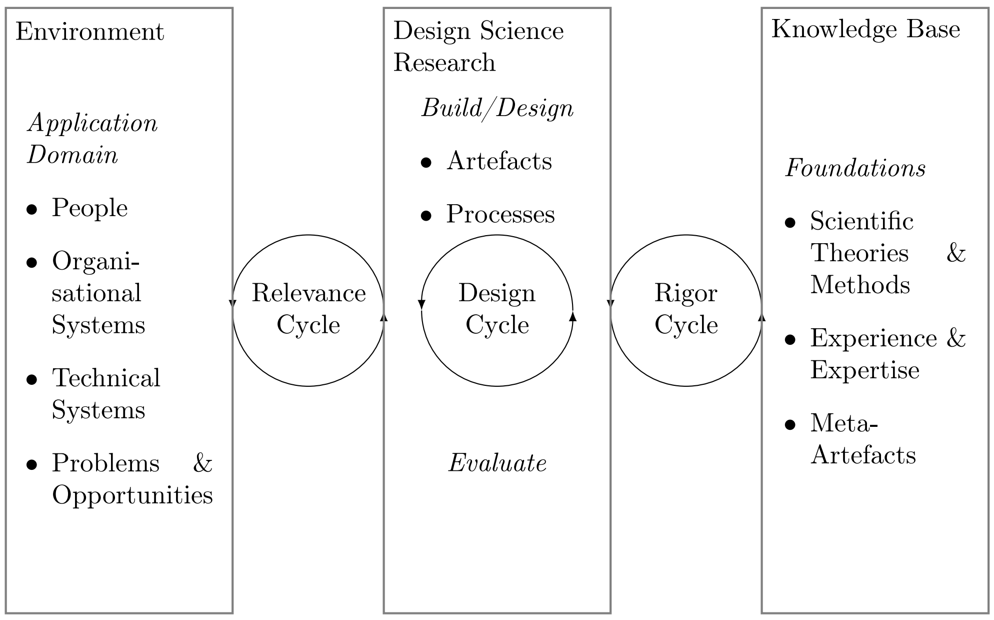


Figure 3.1: Design Science Research cycles [1]

There are 3 cycles in the DSR cycles. The "Relevance cycle" translates between a real-world and a design problem, applying design artefacts to real-world problems in an attempt to solve them. The "Design cycle" concerns itself with creating design artefacts that can solve design problems and evaluates those designs. Finally the "Rigor cycle" attempts to extract generalised data from the design artefact and design process that created it. The rigor cycle creates, amongst other things, meta-artefacts. Meta-artefacts can be used to aid the design process in a subsequent design cycle.

Environment Domain

The environment represents the real world. More specifically, it relates to the real-world components of the problem that is being investigated. It contains all the people and organisations that are involved with the problem as well as any systems that interact with it. As a part of mapping the environment, opportunities will arise: people, organisations and systems that can help solve the problem in some way.

Design Science Research Domain

This domain encompasses the design of artefacts and processes that attempt to solve problems in the environment. An attempt is also made to estimate how well a designed artefact or process will do in a theoretical evaluation.

Knowledge Base Domain

In this domain resides the knowledge that is applied by the DSR domain. It contains theories on how to design quickly and efficiently. These cover techniques and heuristics for good design, but can also cover meta artefacts. The knowledge base aims to be a repository of existing experience and expertise such that new projects can benefit from lessons learned in earlier ones.

Relevance Cycle

The relevance cycle ties together the environment and DSR domains. It defines the design problem as a set of requirements. The requirements serve as the basis for the design and evaluation. Finally the cycle carries back a proposed solution to be used in the environment.

Design Cycle

In the DSR domain might already exist a direct answer to the design problem which can immediately be evaluated as a potential solution, or a new one can be designed. Whenever a design is finished it should be compared to other designs and the requirements to check if it is a valid fit, and if there are multiple solutions, which is the best. The design cycle iterates through finding/creating designs and validating them until a design fits, and thus is good enough.

Rigor Cycle

Whenever design theories are applied, good designs are created or otherwise lessons are learned, the rigor cycle can carry that experience to the knowledge base. In order to add to the knowledge base in a meaningful way however, that experience should be drafted into an artefact. Whenever the DSR domain requires scientific backing, these meta artefacts form the knowledge base that can be consulted.

3.1.3 Applying DSR cycles to Interactive Simulations

DSR provides an appropriate framework to meet the challenges in the interactive simulation domain set forth before. This section shows how those challenges map onto the DSR domains and cycles.

Environment Domain

The list below describes some of the stakeholders and what their relation to the interactive simulation might be.

Researchers Wants to answer their research question, possibly using a designed artefact like an interactive simulation.

Domain experts Needs to be motivated, by the interactive simulation and the design process, to share their expertise and knowledge.

Financers Whether funding comes from business, university or elsewhere, they need to be convinced their investment will yield the results they are looking for.

Amongst the requirements facing an interactive simulation are constraints set by the organisations to which your domain experts belong, or their governing bodies. It cannot be assumed everybody is allowed to share their knowledge and experience for the greater good. Another source of requirements comes from limits on the technical systems that can or should be used. This can be due to monetary limitations or for example licensing or integration needs.

As discussed previously a potential problem with getting an interactive simulation realised is getting it funded, since it can be a costly enterprise. However, if the interactive simulation is setup to support multiple avenues of research simultaneously it is possible to assemble a group of researchers and businesses behind a single development project. Doing this should greatly increase the chances of securing the funding needed.

Design Science Research Domain

As a domain this would contain any previously designed interactive simulations that might be reused or adapted to be applied to the requirements coming from the environment through the relevance cycle.

Knowledge Base Domain

As a result of applying the rigor cycle, the knowledge base will grow specifically with a meta-artefact helping the design process for interactive simulations. This thesis takes the lessons learned from GISEGas, turns them into a meta artefact in an effort to add to this knowledge base.

Relevance Cycle

The purpose of the relevance cycle is to define the design problem by determining a set of requirements from the environment to be solved by the design cycle. In the context of interactive simulations the relevance cycle is applied every time a (series of) game(s) has to be run. A researcher or group of researchers is applying the interactive simulation provided by the design cycle to accumulate data to answer their research question. Convincing domain experts to play the game is achieved when they believe there is something in it for them. One way this can be achieved is by having a well designed, “fun” game. It is also important to make sure that any conditions set when securing the financing are met. Examples of such conditions are: Producing a certain amount of scientific output, or providing a business model for making money with the produced artefact.

Design Cycle

In the context of creating an interactive simulation the design cycle should design the system that will execute the simulation. This will likely be a collection of elements including:

visuals Something for the players to grasp what is happening within the game. This can be a physical or virtual game board for example.

game rules Describing what players are and aren’t allowed to do.

simulation Crunching the numbers of time progressing the game world and making sure that player actions have realistic effects on that game world.

player interaction Some control over the communication the players have with each other isn't unreasonable to expect. Whether it is facilitating and recording these interactions, or preventing certain players from communicating or over-hearing is crucial to some of the research associated with interactive simulations.

These elements might already exist within the DSR domain and thus do not have to be designed from the ground up. Meta-artefacts and experience from the knowledge base should help finding the right elements to combine and help determine which parts, if any, still need to be designed.

Rigor Cycle

The core of this thesis is the application of the rigor cycle to learn from a previous design process creating an interactive simulation, and creating a meta-artefact that will help subsequent designs of interactive simulations qualitatively better and quicker leading to a reduction in cost.

3.2 Systems Engineering

Systems engineering is an engineering discipline that concerns itself with "the bigger picture". Its goal is to combine any skills needed, mostly from other engineering disciplines, into a coherent system. Such a system does not only contain for example: software or machines, but also people and organisations to give a whole picture of what is happening.

The aim for this thesis is to create a meta-artefact for the knowledge base. In the artefact the complex interactions between stakeholders from different organisations that result in a game are described. That game is likely to contain software components. Systems engineering provides the tools and language needed to create a meaningful and understandable artefact.

The artefact created if a functional design also known as a functional architecture and will is described in section 3.3. The design is visualised as a model and the modelling language used for that is described further in section 3.4.

3.3 Functional Design

One of the tools used in systems engineering is the functional design, or functional architecture. It describes a system as a series of functions with inputs, outputs and controls. It shows the processes within a system and how they "talk" to each other. There is also a "physical" architecture which describes the actual components. Note that there is not a one-to-one mapping between these architectures as a component can perform multiple functions and a functions implementation can be spread over multiple components.

3.4 IDEF0

IDEF0 is the first member of the IDEF family of modelling languages. IDEF stands for Integrated DEfinitions for Functional modelling. The focus for IDEF0 is to be used to create a functional or process model of a system [4].

3.4.1 Hierarchy

Functions in an IDEF0 model are numbered. Whenever a function is decomposed into sub-functions those are numbered with the number of the super-function followed by a ".", and a number indicating its rank within the super-function. The rank is determined by what is most important to the design. The system that is being described has number A.0 and is usually shown in its context by showing its super-level: A.-1.

Functions have inputs on the left, describing any thing that has to go into the function. Resulting information/artifacts come out of the right and are called the outputs. At the top of function, parameters for that function can be set, these are called controls.

3.5 Method

Figure 3.2 shows the steps that led to the creation of the meta artefact based on the cycles from design science research. First a literature search is done to establish the existing knowledge base, allowing the design cycle to draw from that. Next an interactive simulation is developed. To close the loop on the rigor cycle the newly created knowledge is distilled and fed back into the knowledge base. In this case the feeding back includes the creation of the new FASD framework (Functional Architecture for Serious game Design Framework).

The functional architecture is how the research question is answered, thus answering the sub research question ($RQ1_a$). The functional architecture combined with guidelines for using it form the actual answer to the research question ($RQ1$).

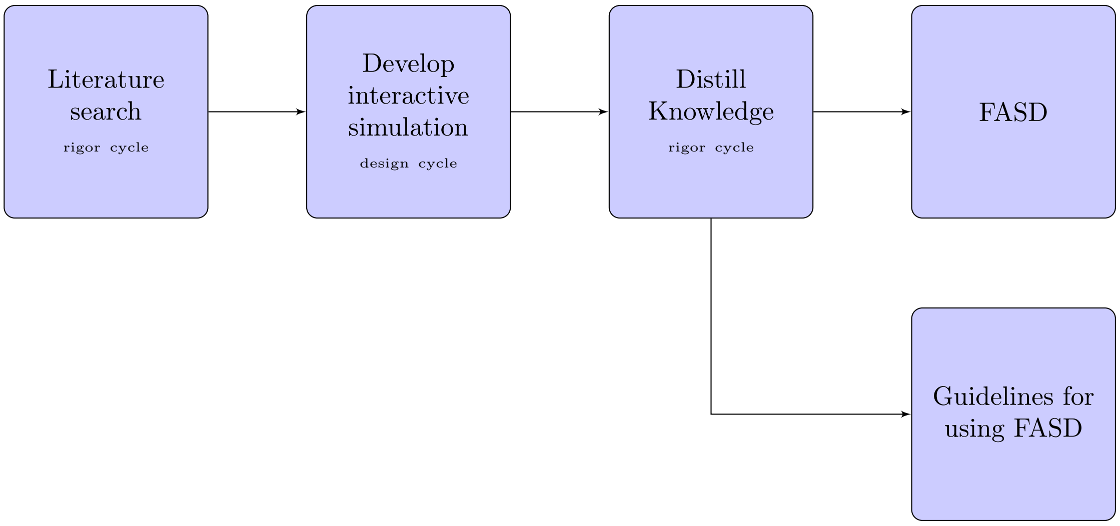


Figure 3.2: Steps taken to create FASD

Chapter 4

Developing an interactive simulation

Part of the method presented in the previous chapter is developing an interactive simulation. This chapter describes the development of the GISEGas game and the development process ending with the lessons that can be distilled from that process.

In Chapter 2 the definition of an interactive simulation is given. GISEGas fits that definition in the broadest sense, forming a good basis to be used for the systems engineering approach of functional design as described in Chapter 3. To support both these points an extensive description of the GISEGas game is given that ties into both previous chapters.

First an overview of the GISEGas project is given in Section **??**. Section 4.2 describes how GISEGas is setup, and section 4.3 how it is played. In sections 4.4 and 4.5 the implementations of the board- and video game, Gasboard and GISEGas respectively, are described. How the game is used to answer research questions is elaborated upon in section 4.6. Finally in section 4.7 the implementation of GISEGas will be linked to some general components that will lead to the functional Architecture presented in Chapter 5.

4.1 GISEGas: an overview

The GISEGas project started with a game called Gasboard. Gasboard is a serious board game, and is designed to serve as a catalyst for conversation about the introduction of biogas into a traditional west-european gas network. Gasboard is included into a larger project called GISEMarket and when it was decided to make it into a computer game renamed GISEGas. The goal for GISEMarket is to investigate the investment process in to gas infrastructure in the Netherlands [12, 14]. Over time more potential research related to the GISEGas/Gasboard game was identified and explored by multiple companies and researchers.

GISEGas was build as a collaboration between multiple companies, at least nine different programmers worked on it over the course of several years. Table 4.1 gives the result of using the cloc software (`https://github.com/AlDanial/cloc`) on the source folders of different elements of the project, showing the size of the final source code to be approximately 32.000 lines of code. The software was used by more than 5 researchers and in business to various ends.

The author has spend at least 300 hours with the project and was responsible for several of the major design choices and was at several points the only or lead developer on the project.

Language	files	blank	comment	code
Java	185	3.802	3.983	14.425
XML	3	0	1	204
C#	135	2.986	2.942	16.390
SQL	1	306	272	1.413
Sum	324	7.094	7.198	32.432

Table 4.1: Number of lines of code of the major parts of the GISEGas project

4.2 GISEGas: the game

This section explains the way the Gasboard and GISEGas games work. First the way the world is setup is given. Second the different roles that are available for a player to choose are explained.

4.2.1 What the game looks like

The initial game of Gasboard was modelled on games like "Settlers of Catan" [7] and "Power Grid" [10]. There is a "world" made up of a hexagonal grid and players can buy game pieces to put on the intersections. These game pieces can be connected to a network by putting pipes on the edges. Next are the available pieces and their effects on the emerging network.

Well This game piece represents a natural resource of gas. It is typically part of the setup of the board and additional ones cannot be bought. It has a limited amount of gas available and not all gas can be pumped out at once in accordance to the laws of nature.

Outside source Some game setups can have an external supplier/buyer of gas. It will fix an unbalanced network but will pay very little for excess gas it gets rid of, and charge very much for extra gas it has to supply.

Digester This piece represents a farm where biogas is produced. Any gas that is produced but not pumped out is converted to electricity. How much money is earned from selling that electricity depends on the scenario. A digester can be upgraded to produce more gas. Upgrading a digester is a smaller investment than building one at an additional location. The gas coming from this game piece is of a lower quality than that from the natural or outside source.

Factory This game piece uses gas, it creates products which can be sold and thus generates and inflow of money into the system. The owner of the factory can indicate how much gas it would like to get from the network, but if this gas is not being supplied to the network that can lead to an imbalance. By default the factory can only accept gas of the quality that the fossil gas producer provides, but it can be upgraded to also allow lower quality gas, like from the digester to be used. Like the digester a factory can be upgraded to have a higher capacity in a single location.

Pipe Is used to transfer gas from one intersection to another. The first player to put a pipe at an intersection owns that intersection and can decide who can connect to it and under what conditions. Pipes have a limited capacity that can be upgraded. Whenever something is connected to the pipe network gas can be pumped in or out, this can lead to imbalance, to be rectified by the outside source or by systems shutting down. It can also lead to the quality of gas in the network dropping causing non-upgraded factories to no longer be able to use that gas and shutting down. A fee is charged automatically for pumping gas into or out of an intersection to the owner of that intersection this is called the tariff.

Substation This game piece disconnects the pipe networks connected to it, it will also convert the quality of gas from one network to the other like a lock in a canal. This game piece is typically very expensive but can be used to connect digesters to an old network without anything having to change in the "old" high quality gas network.

4.2.2 Roles in the game

In Gasboard and GISEGas not all players are the same. Not everybody can own every type of game piece for example. This is to reflect the current situation in the real world where there are also distinct roles. This list gives an overview of the roles typically found in a game, note that not all roles have to be played by a human being and some roles might by present multiple times in the game.

Fossil Gas Producer (fgp) The player with this role is the owner of the natural gas sources and the sole provider of gas to the consumers at the start of the game. Depending on the scenario this role may have contracts selling more gas than is available in their gas source. A fgp is allowed to build pipes and substations.

Putin This player represents a foreign power able to provide or store a large amount of gas and owns the "Outside Source" game piece. The idea is that if the player group makes a mistake and too much gas is taken out of the system or too much is put in, Putin can take care of the deficit or excess, at unfavourable prices to the players. The "Outside Source" game piece is the is the only game piece this player is allowed to have. This role was automated in GISEGas games.

Bio Gas Producer (bgp) A player with this role owns one or more Digesters. Their low-quality bio gas is being turned into electricity but not for much profit. A bgp can build additional digesters, pipes and substations. If they want to earn more money, an attempt to get connected to a network so they can sell the gas should probably be made.

Consumer A player with this role has factories that consume (initially only fossil) gas to produce goods and therefore money. Though this role tends to start with a contract for gas with the fgp, they might look for cheaper and longer lasting sources of energy. A consumer can build more factories, pipes and substations.

Grid Operator Typically only one player has this role. The player with this role owns all of the initial infrastructure on the board and is required to attach a game piece to the network when asked. The grid operator can charge a fee for that, limits for which are set by the government. This connection fee is in addition to the transportation fee, also known as tariffs, that were described at the pipe game piece, of which the grid operator has a lot.

Government Whenever disputes arise it is the government's task to resolve it, to this end it can set limits to the tariffs the grid operator is allowed to charge and it is allowed to fine/subsidise players. The player with this role, there tends to be only one, is not allowed to buy any game pieces. However, when a player goes bankrupt its assets are taken over by the government.

(Investment) Bank There can be multiple roles of "bank". This role keeps track of money and can provide loans. In GISEGas this role was never assumed by a player, instead two simulate players were created. One is responsible for the bank accounts for all of the players and allowed them to go “in the red”. The other could make investments based on strictly formatted proposals from the players. Players with this role are not able to own any game pieces.

Deus Ex Machina (DEM) This role is taken by the “game master”, he or she keeps an eye on the proceedings and can manipulate the world if needed. Though disputes should be solved by the player in the governmental role, the DEM can step in to help out the players with tougher roles. This role would typically by taken by the researcher running the game.

4.3 Typical Game Session

A GISEGas game typically starts with the game organiser (usually also the DEM) explaining about the world the players are about to participate in. It includes, depending on the level of expertise of the participants, some background about gas. Most certainly the game organiser will indicate the state of the game world: Fossil gas is running out, Bio gas is up and coming but the consumers have long lasting contracts with the fossil gas producer. What is explicitly left vague is the number of rounds to be played, and in general nothing is said about who “wins” or even what constitutes a “win”.

For a group of players who have not played the game before an introductory game is started where the goal is to not go bankrupt in the first round. This allows the players to get familiar with the game controls and mechanics without having to worry a mistake ruins the rest of their game. In the second round, which hopefully all players reached, the instruction becomes “go bankrupt” which isn’t that difficult but also gives a lot of insight in what to look out for. After this the game is reset and a “real” game is started.

The game organiser will have selected a scenario corresponding to the goal of the session, goals can be for example a research question or a learning goal for the players. A scenario consists of specifying how many players there are, what roles they have and what there starting assets are (game pieces and money). Also included into the scenario can be to give some strategy or goal to each player, for example to see whether having certain instructions changes behaviour, and can help reach an optimal solution quicker.

After a couple of rounds a post-mortem would be done, discussing what went right, what went wrong and, while the game was still in active development, if the players had any input for that development.

GISEGas has been applied in multiple academic and business contexts. Though the business context can be relevant to secure funding, the focus here is on how GISEGas is academically relevant.

In order to provide academic results, experiments need to be repeatable. It is good to realise that this type of game or simulation depends in large part on the people that are participating in the interactive part of the simulation. A lot can be done in software but selecting the right player group and instructing them correctly is paramount for a successful experiment, and having sound results for the research being conducted.

4.4 Gasboard: the implementation

Initially this game was developed to be played at a table. It consists of a board, depicting the hexagons and wooden game pieces that can be placed on that board. Players take turns based on their role to buy pieces and negotiate contracts. At the end of a round (i.e. every player has

Figure 4.1: Gasboard being played, image courtesy of Marcel Volkerts

had a turn), the state of the board is put into an excel sheet which gives some information about how the state of the world changes by turning the clock, gas flows through the system, contracts are resolved etc. Putting information into Excel and interpreting and communicating the results is the role of the DEM.

4.5 GISEGas: the implementation

The GISEGas game consists of three major parts: the game client, the game server and the calculation module. The game client can be compared to the game board in gasboard, it is what the players see and interact with. Everything the players put into the client is sent to the game server, it holds the "true" game world and determines what is and is not allowed. Whenever a round ends the server calls the calculation module (calcmod) to simulate the network and determine what happens when time moves forward given the current setup. The result flows back, through the server to the client and a new round is ready to be started.

4.5.1 The game client: Unity (C#)

The game client is the face of the game, it is what the players see. Knowing this, effort was spent early to make it look good. This helps keep people interested in the game and convince stakeholders to invest in it. This meant that some elements of the game where developed in the client before they existed in the server. While developing the server this led to some challenges where the client reacted different then expected based on the design.

As the game became more complex and the number of (types of) messages between the client and the server grew, it became more and more apparent that the communication had to be formalised in to an Application Programmer Interface (API). Writing the API documentation

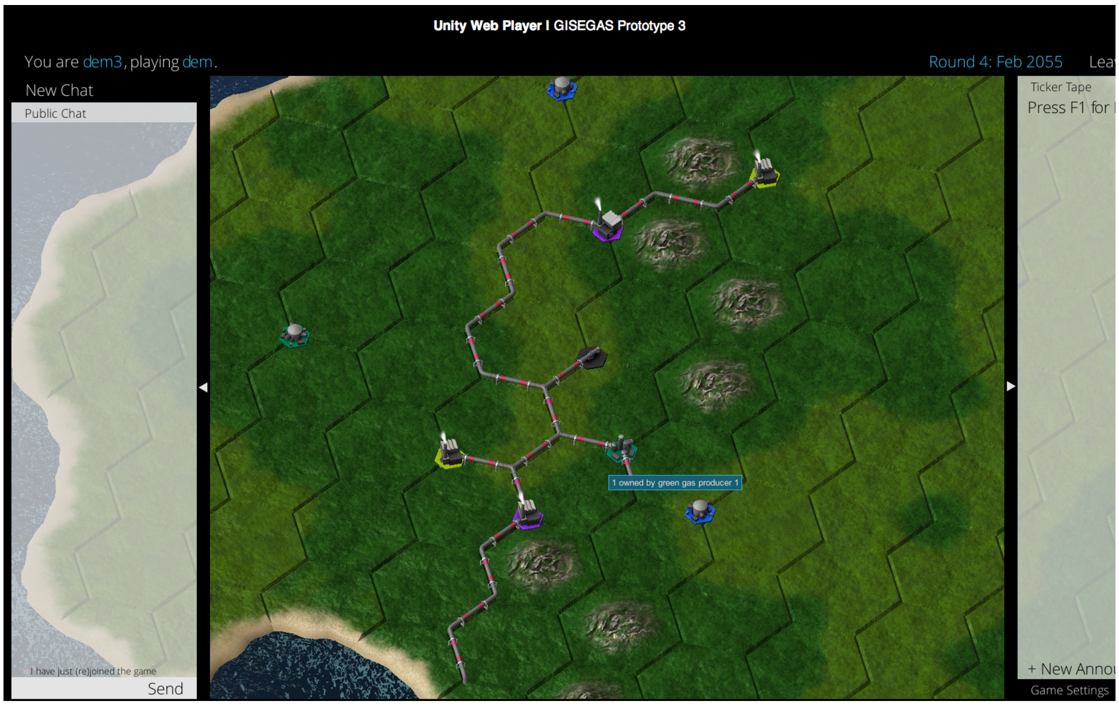

Unity Web Player | GISEGAS Prototype 3
You are dem3, playing dem.
Round 4: Feb 2055
Lea
New Chat
Public Chat
Ticker Tape
Press F1 for
1 owned by green gas producer 1
I have just (re)joined the game
Send
+ New Anno
Game Settings

Figure 4.2: The UI of the game client

first allows for it to be well designed and the client and server to follow that documentation strictly, instead of testing having to be done after the fact to figure out what is going on.

Late in the project, while refactoring the code, it was discovered that there were several implementation of solutions for essentially the same problem. In those final stages of the project much effort was put forth in finding and solving problems occurring from inconsistency between client and server, which was one of the things that was inconsistently implemented.

4.5.2 The game server: SmartFoxServer (SFS) Extension (Java)

The core (in the project also referred to as kernel) of the extension was started by modelling the interactions between the agents[1] in the game. The language used to describe these agents and the interactions between them is called TALL (The AgentLab Language). All roles as described before are linked to an agent, each agent can subsequently be linked to a human or a virtual player.

A representation of the game world was developed to reflect what was being shown in the game client. The goal being to share the world state between players. It was realised the implementation was the wrong way around. The "truth" about the state of the game world and by extension the game board should reside in the server and the client should reflect the servers state. Rules where added to govern which agents could perform what (type of) action and a mechanism for denying actions was implemented. From this point on the control was definitely in the server.

[1]In the context of describing this process the term "actor" should be used here. As the discussion turns to software however, the term "agent" is more appropriate.

At this point development was shifted to increasing the scope of the represented world. Elements were added like a monetary system including a bank, bank accounts, loans and interest. Additional game pieces with more intricate rules were added and UI work was done adding more information for players to see and, when appropriate, manipulate. As more options were given to the players it was discovered that the rules for the world were not always clear "what should happen when a player goes bankrupt?" and "how do you define bankruptcy?".

The ad-hoc decision making by the DEM on how the world should react was interesting, showing how much room for interpretation there is. A human can easily deal with this complexity but the rules needed to be made explicit for the server to implement them. Play testing also led to changes being made in choices that turned out not to work well and more ad-hoc solving of problems that were found with game flow, and ease of use. This way of working reflects the agile workflow of the development, but led to unnecessary extra work.

The final product was reasonably consistent, though the proverbial wheel was reinvented more often than seems necessary on retrospect.

4.5.3 The calculation module: MOPED (AIMMS)

MOPED, also referred to as calcmod, provides the SFS Extension with simulation data about how gas flows in a network. Whenever a time step is taken in SFS it will ask MOPED to calculate how much gas can be pumped in and out of the network, compared to what players setup their game pieces to do. This allows SFS to apply game rules like what a player has to pay in tariffs.

The communication between MOPED and SFS was done via a database both could read from and write in. Flags in a certain table where used to indicate between them what they where doing (flags indicating "please calculate" set by SFS and result flags like "done" or error codes set by MOPED). This turned out to be a design flaw when trying to move MOPED to a different machine than SFS and this effort was eventually abandoned.

Throughout development only minor changes where needed to MOPED, mostly due to bugs found over time. The initial estimation for the scope of this module appear to have been largely correct.

4.6 Applying GISEGas to research

Over the course of the GISEGas project the game was played many times. There were test games played mostly with the development team: programmers, researchers, and business partners; these sessions where merely to test or demonstrate functionality of the game. The game was also applied in a research context. These are several examples of how the game has been used.

GISEGas has been applied to aid students in a Systems Engineering course understand the problem they were facing. Topic of interest in these sessions were: "Does playing a Serious Game help design a system in the domain of that game?", "Does designing a system help play a serious game in the same domain as the system?" and "Does playing a serious game after having designed a system in that same domain and then having the opportunity to redesign the system change that design?". These questions are part of an effort to determine whether serious games can be used to help teach courses that combine multiple disciplines (also known as integrative courses) [23].

Another researcher was interested in "the prisoners dilemma" in the context of negotiations and investment decisions. The dilemma boils down to: it might be better to not take short-term gain in order to prevent mutual destruction. By documenting all decisions being made and trying to find instances of the prisoners dilemma the negotiations leading to that decision could be analysed.

Figure 4.3: The view of a room of students playing GISEGas in an experiment setup with the screen of the DEM shown

GISEGas is used by business as a revenue generating enterprise. The company doing this provides hosted games for real world stakeholders at least some of whom are operating in the real-world domain of gas trade, infrastructure and investment.

Whenever a gaming session was held, the results and feedback were used to refine the design of the game. This resulted in more relevant data for researchers that was better accessible and a smoother game experience for the players.

4.7 Applying the rigor cycle

4.7.1 Using the game

Playing the game or "running the simulation" requires some functions to be executed. At the top level two functions can be identified for the software to perform, namely: Keeping track of some representation of the world and its rules, and secondly determining the actions for Agents that are not controlled by a human player. A third function is for the human players: perform actions for Agents that are controlled by a human player. The software has to provide an interface to do this.

Representing the world

In GISEGas the domain model consists of Java objects that are mapped to a database using Object Relational Modelling (ORM). Whenever a time step needs to be taken, like at the end of a round, the gas network is simulated by AIMMS. Handling the passing of time for things like contracts and loans, is done in another simulation engine, the "bankcalcmod", implemented in the SFS Extension. For a long time there was a struggle as to how to handle when what was allowed to be changed, which led to the need to enforce game rules.

Domain model The domain is modelled in Java objects, this allows it to be directly used by the SmartFoxServer (SFS) extension. Using the Java Persistence API (JPA) allows for easy storage in a MySQL database which formed the backbone of the interaction with AIMMS and persistency through server restarts.

Simulate domain The core simulation for GISEGas, the gas flows, is simulated using AIMMS. It read the current location for all objects and the associated settings (how big are they, should they function at full capacity, etc.) directly from the MySQL database that was kept up to date by SFS.

Anything else that changed over time, like money automatically changing hands based on contracts, was done by the SFS extension.

Apply game rules The development team struggled with how to implement the enforcement of game rules. Eventually a system was setup where, whenever a request comes in, it is first checked for permission (is this player allowed to do this at this time). Then a probe is made into the model, not actually changing anything but checking if the change would lead to any issues (e.g. upgrading a factory already of the maximum size), if no problems are found the model is updated to reflect the change. Whenever a move was not allowed, the player attempting the move is notified.

In retrospect the moves made by simulated players should also flow through this system. This was not done initially due to the assumption that a simulated player would never attempt an illegal move. This assumption might not hold when computer controlled agents try to play smarter and try to find the bounds of what is allowed or contain otherwise game breaking bugs. All moves should be checked for validity.

Agents not linked to a virtual player

In GISEGas any actor that could make decisions was modelled as an agent. Representing who can own game pieces have a bank account etc. is always an “Agent” object. Some agents can be marked as Non-Player Characters (NPCs), these agents do not appear in lists where players choose an agent to control or interact with other players. The NPCs implement some logic, respond to specific questions, like “can I get this loan from you?”.

Examples of NPCs in GISEGas are the bank and investmentBanker. Especially the investment banker could be interesting allowing the virtual player to review the assets of the agent asking for an investment and determining the investment the investment banker is willing to make based on how much collateral the requesting player can put up. Unfortunately this system was not implemented by the time the project ended.

Agents linked to a human player

Most agents in the game would be linked to a player. These players can interact with each other using the built-in chat system. This proves to be useful for analysing games and interactions after the game has finished. The players interact with the game world by using a browser plugin. This shows them the state of the world (gotten from the game server) and any changes, for example when another player makes a move.

Different clients can be created and used based on the needs of the users. To not have to deal with different systems players might use, they would always run the web client. Sometimes the DEM system was outfitted with a special version of the client. A spectator client was also created to have a game board but no other UI, to show on a central screen for people spectating the game, or as centre piece for the players to look at.

Chapter 5

A Functional Architecture for Serious game Design (FASD)

In Chapter 2 the need for a high level architecture of an interactive simulation is underlined. In Chapter 3 the fields of design science and systems engineering are described as useful and appropriate to reduce the problem They enable taking a case and extracting from it the generic features needed. Chapter 4 describes such a case in GISEGas and starts to extract some of those generic features.

In this chapter a generic functional design for an interactive simulation is explained, it is created based on the GISEGas case. The modelling language used to describe the architecture is IDEF0, which allows enough expressiveness to capture the necessary details, and at the same time is simple enough for non-experts to quickly grasp. When designing a game, or validating such a design, the designer can check whether all functional parts as described here are implemented in the design.

The model is built hierarchically from functions, as described in section 3.4, with the function describing the entire process of running a game as A.0. The explanation of the model starts at level A.-1 showing the context for level A.0. Whenever a function is elaborated on, first a hierarchical diagram is shown to give an overview of the sub-functions. Then an interaction diagram shows how these sub-functions interact with each other.

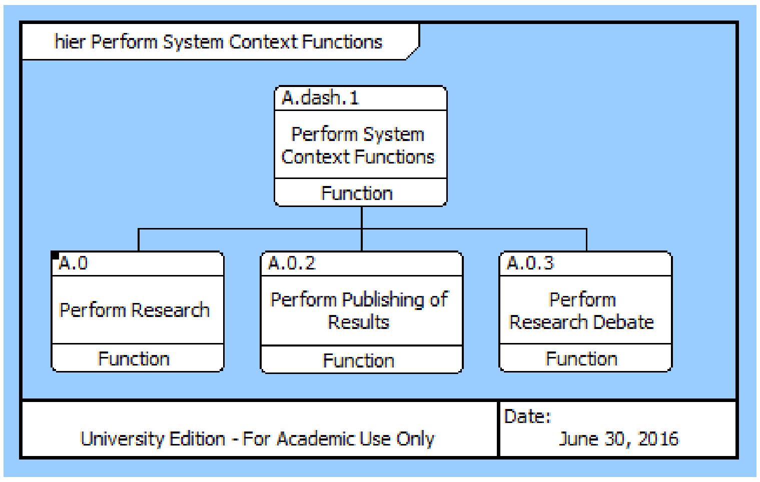
5.1 A.-1 System Context

The A.-1[1] level presents the environment the system exists in; the constraints acting upon it. As the goal of the system is knowledge gathering, the function associated with reaching that goal is "Perform Research". The context for the A.0 research presented here is academic in nature. The context can also be provided by business instead of academia, such a change should not change A.0, but would influence the requirements and motivations of the stakeholders.

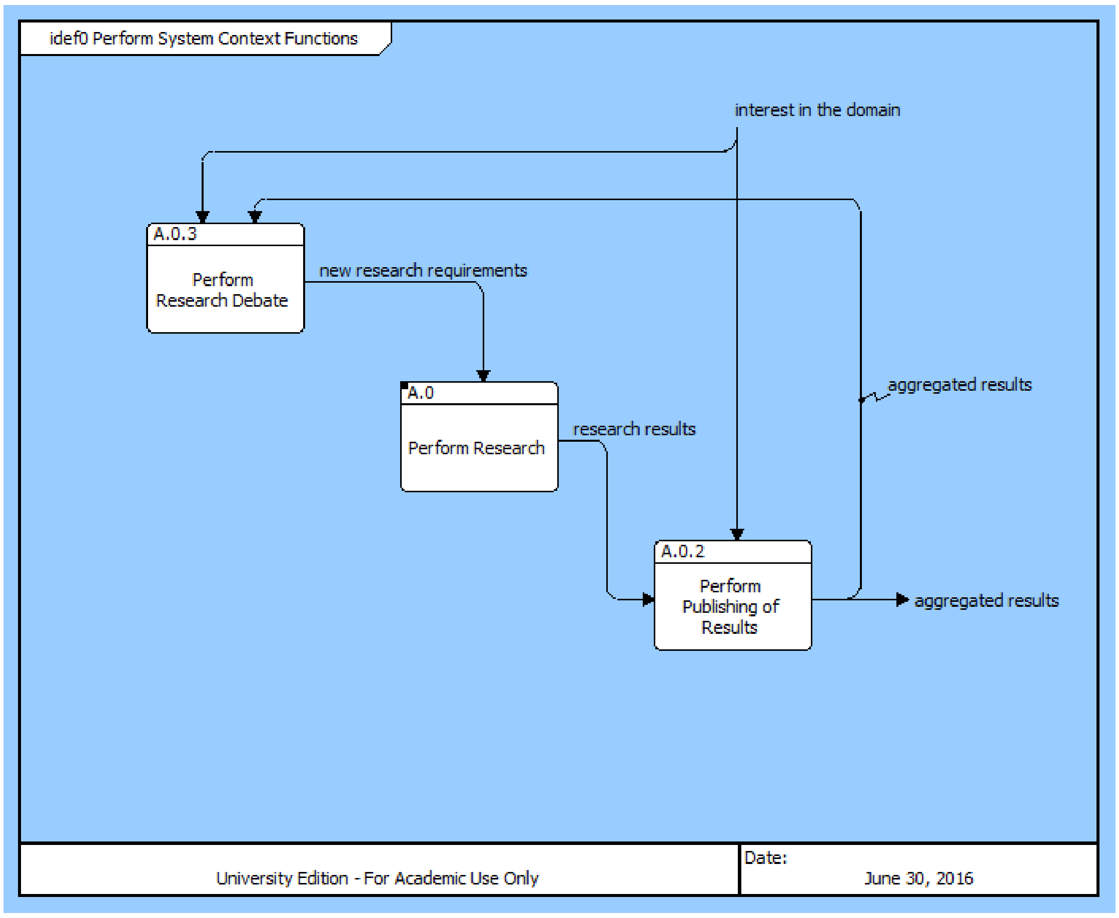
At the top of the interaction diagram we see the outside "interest in the domain". This is the external trigger causing the interest in a topic. There can be any number of actors and motivations there, but this is not a topic further explored in this thesis. It suffices to assume there is interest in gaining knowledge. From this interest stems a desire to debate and publish about a topic. The two of these are linked by our system in the sense that out of the debate come research requirements. The research then takes those requirements and hopefully output

[1]A.dash.1 in the diagram due to limitations in the software used to draw it

Figure 5.1: Level A.-1


hier Perform System Context Functions
A.dash.1
Perform System Context Functions
Function
A.0
Perform Research
Function
A.0.2
Perform Publishing of Results
Function
A.0.3
Perform Research Debate
Function
University Edition - For Academic Use Only
Date:
June 30, 2016

(a) Hierarchical diagram



(b) Interactions between the context and system's function some results which can then be published. Those publications should lead to further discussion, new or refined requirements which can then be used for further research and eventually lead to more publications.

In short the A.-1 function transforms interest in the domain into (published) results, using the scientific method.

5.2 A.0 Perform Research

Level A.-1 is generic and can be applied to any form of research. Here, at level A.0 the assumption is made that the research will be conducted using some form of (interactive) simulation.

At the beginning of the research project the requirements are formulated and lead to the type of research that needs to be performed. How this breaks down into sub-functions and adopts the simulation is discussed further in A.2 (Section 5.3). After that, those parameters are used to perform the simulation discussed further in A.1 (Section 5.4) and finally the results of the simulation will be analysed to form the results of the research and feedback into the research design to help refine the theories, restarting this chain of events.

5.3 A.2 Design Research

The designing of research aimed at using the interactive simulation described in this thesis revolves around the testing of hypotheses. A hypothesis can be formulated using: previous theories, the conducted research and their results, and the research requirements coming from the research field and business. The hypothesis is then tested using the interactive simulation.

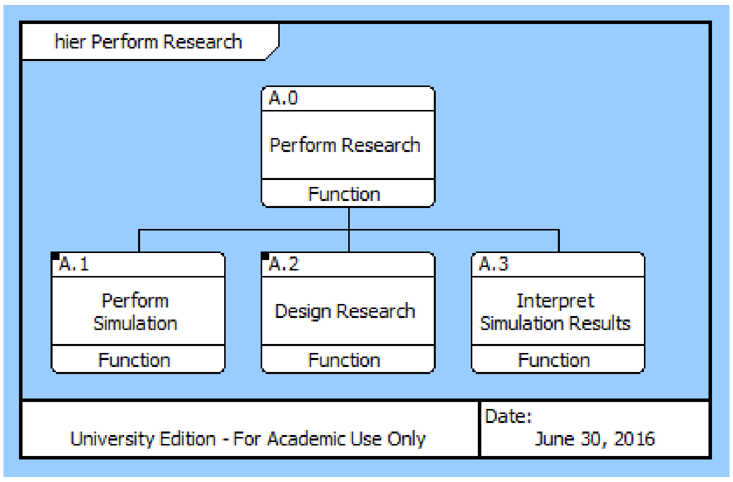
After a hypothesis is formulated the researcher should define scenarios that can verify or reject that hypothesis. Such a scenario gives a starting state of the world (structural, dynamic and data components need to be defined), and rules that will govern the game, like how time will progress. Given one or more scenarios, sessions can be planned. Not only does this mean a time and maybe place, but the selection of the "player" group can be very important. Questions like: "Should the group play multiple games?", "Can a player have prior experience", and "Does the players background influence his decisions in a way that might skew the results?" need to be answered.

When trying to compare different groups of players it is difficult to predict all variables. Opinions regarding for example how liberal or conservative the government should be can play a large factor in how a player approach their role. Careful player selection is part of the solution, another is giving the players instructions like a goal or strategy. This can ensure multiple players in the same role start the game with a similar mindset helping to compare them.

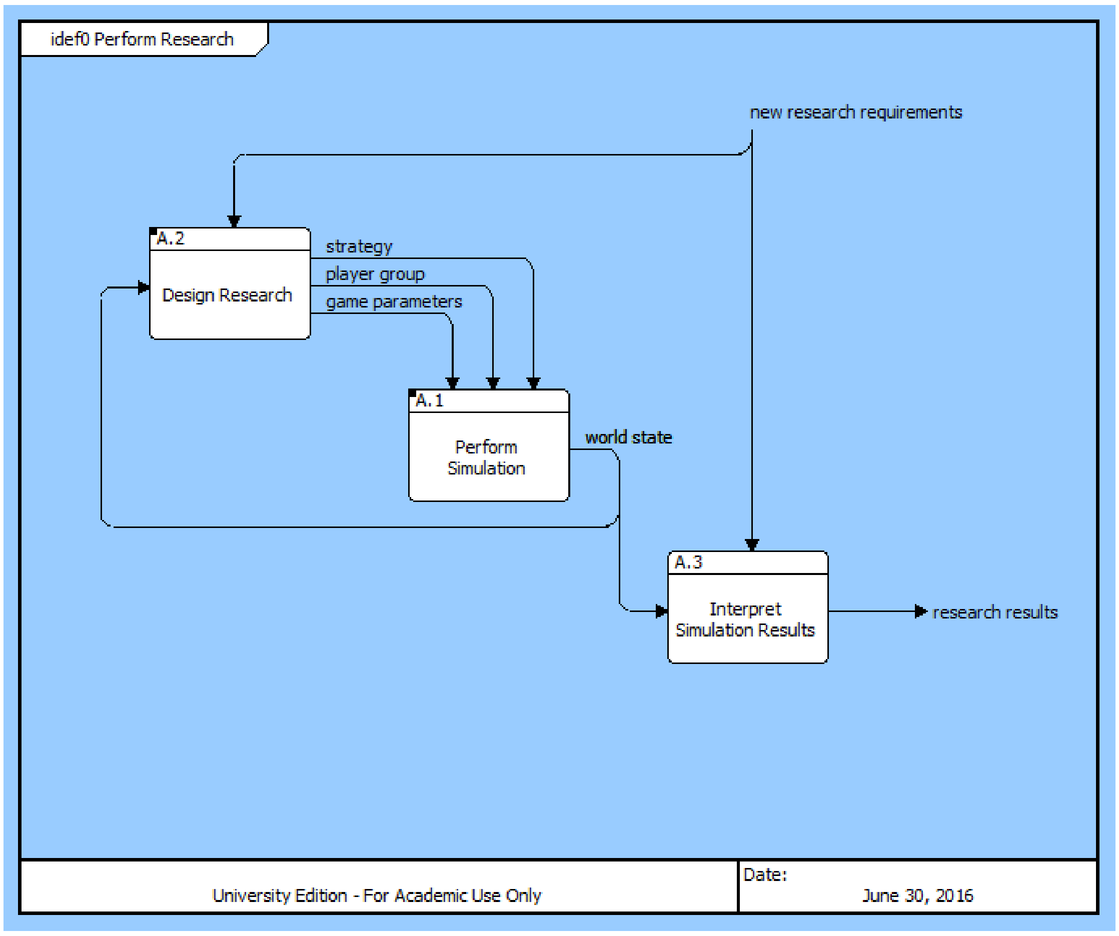
5.4 A.1 Perform Simulation

When performing the simulation there will be some representation of the simulated world. This representation can be very close to "reality" or very abstracted from it. An example of a world very close to reality is that of a racing simulator used by formula 1 drivers to train with. An example of a very abstract representation is that of chess to learn about battle tactics. Either way, the representation serves as the basis for the simulation. Over the course of the game the state of the simulated world changes. The result of the simulation is the final state of the world. Research can involve the state the world reached trying to predict the future given certain input.

Figure 5.2: Level A.0

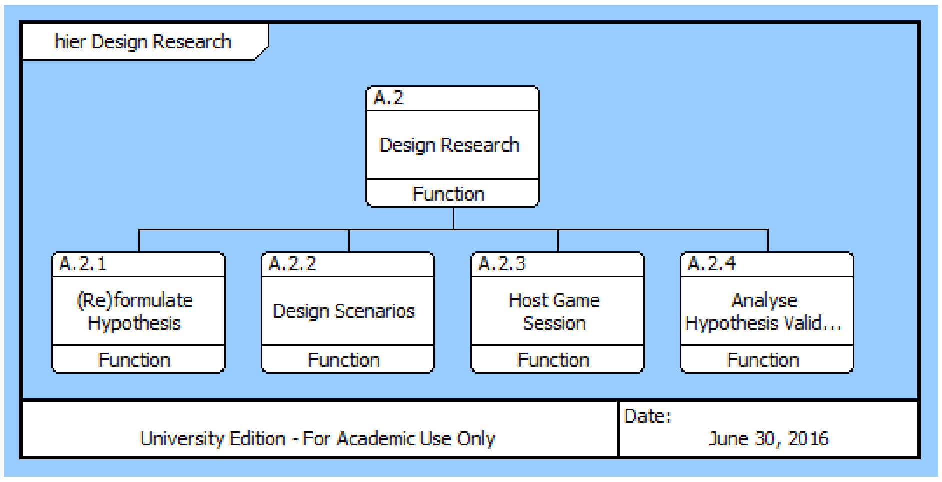


(a) Hierarchical diagram

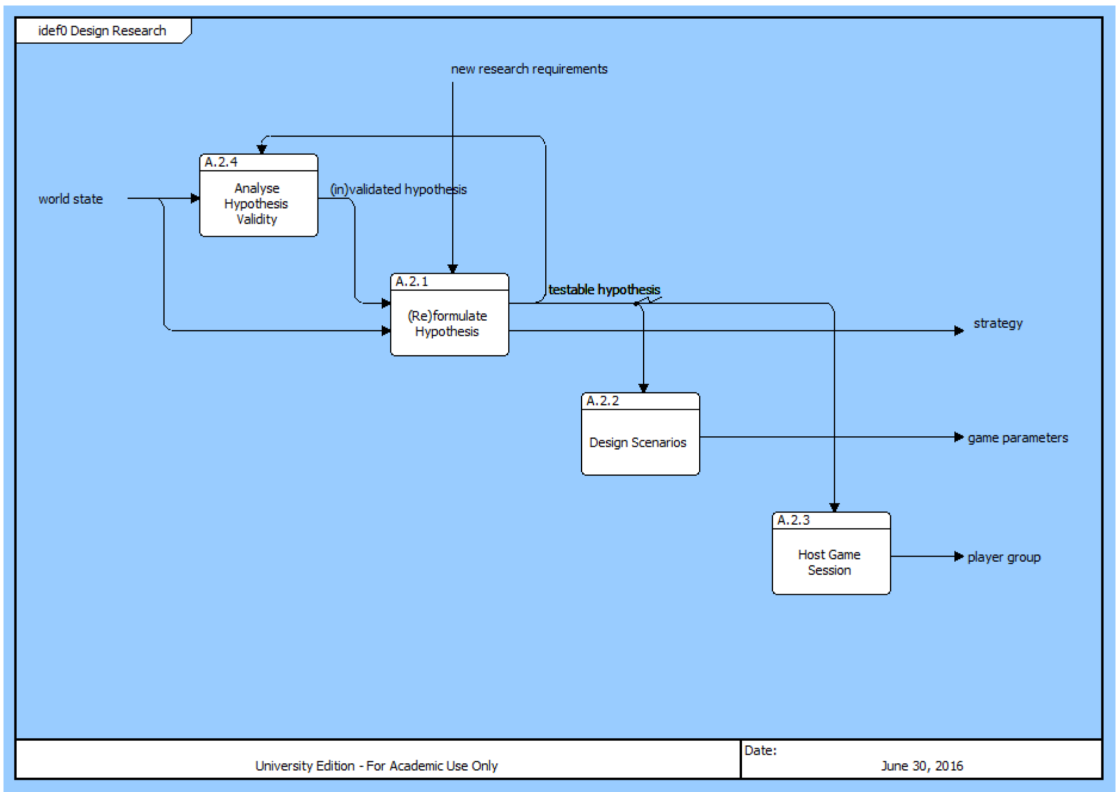


(b) Interaction diagram

Figure 5.3: Level A.2

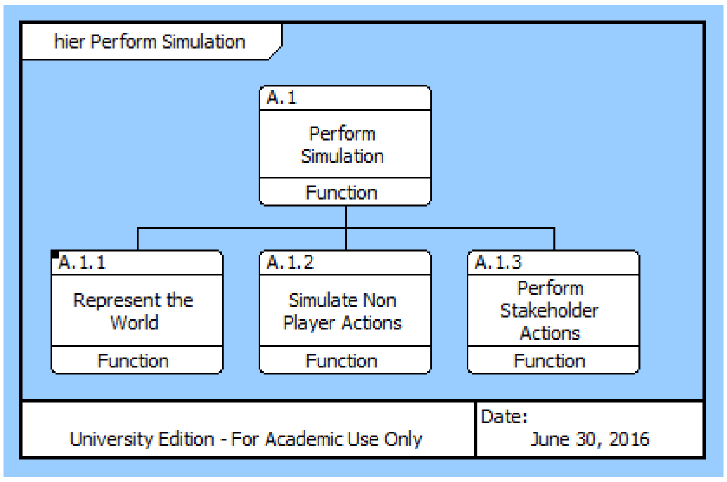


(a) Hierarchical diagram

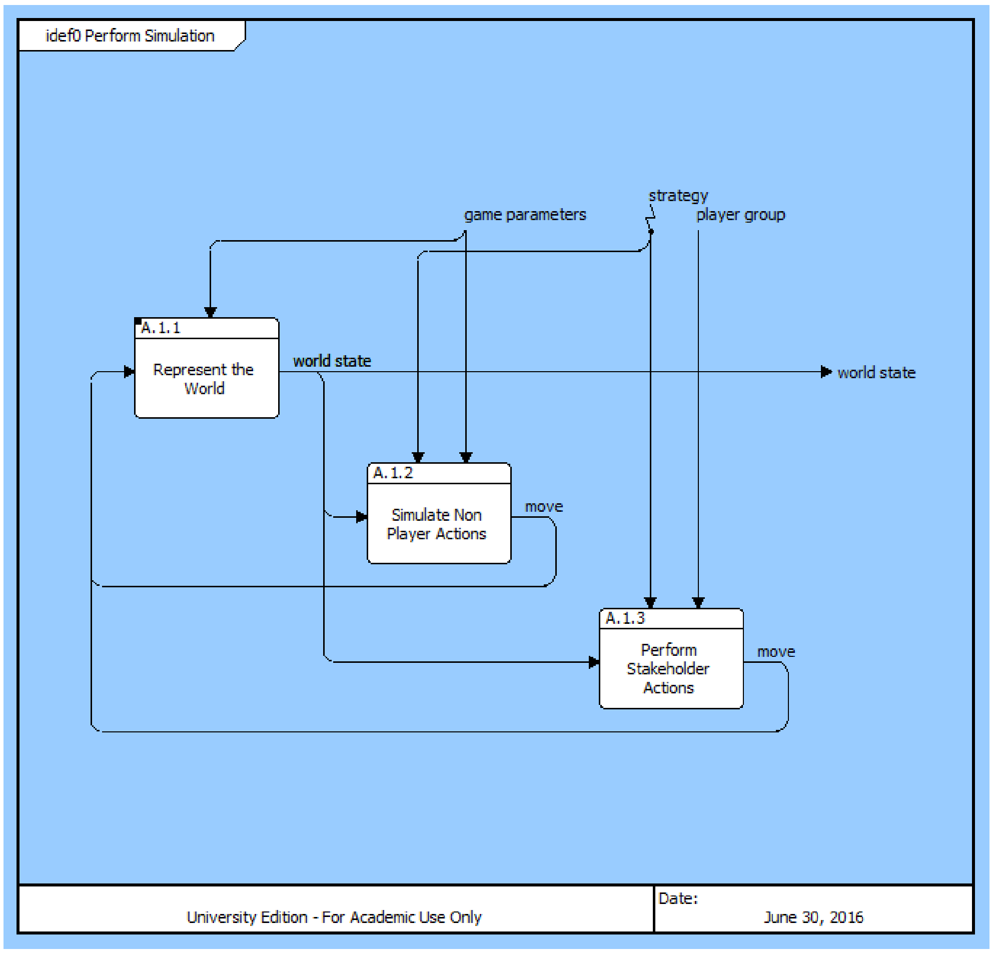


(b) Interaction diagram

Figure 5.4: Level A.1



(a) Hierarchical diagram



(b) Interaction diagram

Alternatively, research can try to find what is needed to reach a particular future. Such research can be used to determine what actors need to do in the near term to reach their long term goals.

Over the course of the simulation several events change the state of the world. Some changes come from within the world, like from the passing of time in the simulated world. Those will be explored when taking a closer look at A.1.1 in section 5.5. Other changes come from the agents in the world. The group of agents can be split in two: agents that are being controlled by the simulation, and those that are not. If the group not being controlled by the simulation does not exist, this would just be a simulation. Adding input to the state of the world from outside makes this an interactive simulation. It should be noted that this architecture could be used to create a simulation by making sure all agents are completely automated.

Agents controlled by the simulation will need some kind of artificial intelligence(AI), at the current level of technology their decision making should be expressible as a set of rules. For example "the bank" in Monopoly has no autonomy, it simply follows exactly the very clear rules and this would be a prime candidate for a simulated agent. When decisions become more complicated it becomes more difficult to write rules that accurately predict what would happen in the real world. To make the behaviour of an agent that should make complex decisions better, a human, preferably a domain expert, can take those decisions. Since that human is now interacting with the simulation, the simulation has become interactive.

5.5 A.1.1 Represent the World

This function deals with everything concerning the simulated world, serving as a gatekeeper for any changes to it, and responsible for keeping a valid and accurate representation of it. Its input are actions from players, simulated or real, and its output is either a changed world, or an error to the user saying why the world hasn't changed. Certain moves may not be allowed based either on the rules of the world, or the rules of the game. A filter is used to check the validity of the moves presented. Some changes to the world happen because of passage of (game) time. Whenever a time step is taken, the progression of the world is simulated, for example gas flowing through a pipe or a factory producing goods. Though it is the Function A.1.1.2 that simulates the domain and thus "consumes" time, the game rules and therefore function A.1.1.3 governs the passage of time in the game. The game rules determine when and by how much in game time should change, activates the simulation to simulate for that amount of time. Triggers for changing in game time can be: real world time passing or players indicating the end of their round.

The main purpose for this function is to maintain the valid state of the world. Whenever someone needs to know what the world looks like, they can get an accurate and up-to-date representation. Note that the visual representation of the game world is not a part of this function, though indicating a new version is available should be.

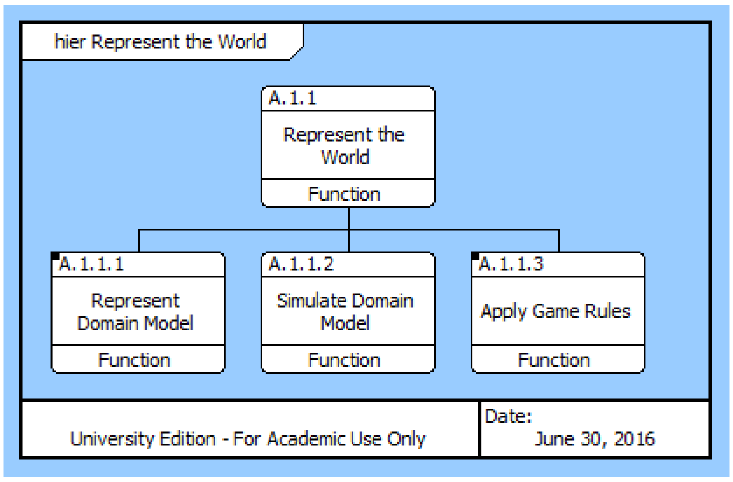
Keeping a record of the moves and domain model events allows a researcher to precisely reconstruct what happened in a game, since those are the only ways to change the state of the world. An obvious candidate to log this events is the apply game rule function, which keeps track not only of all legal moves, but also attempts to do things that are not allowed.

5.6 A.1.1.1 Represent Domain Model

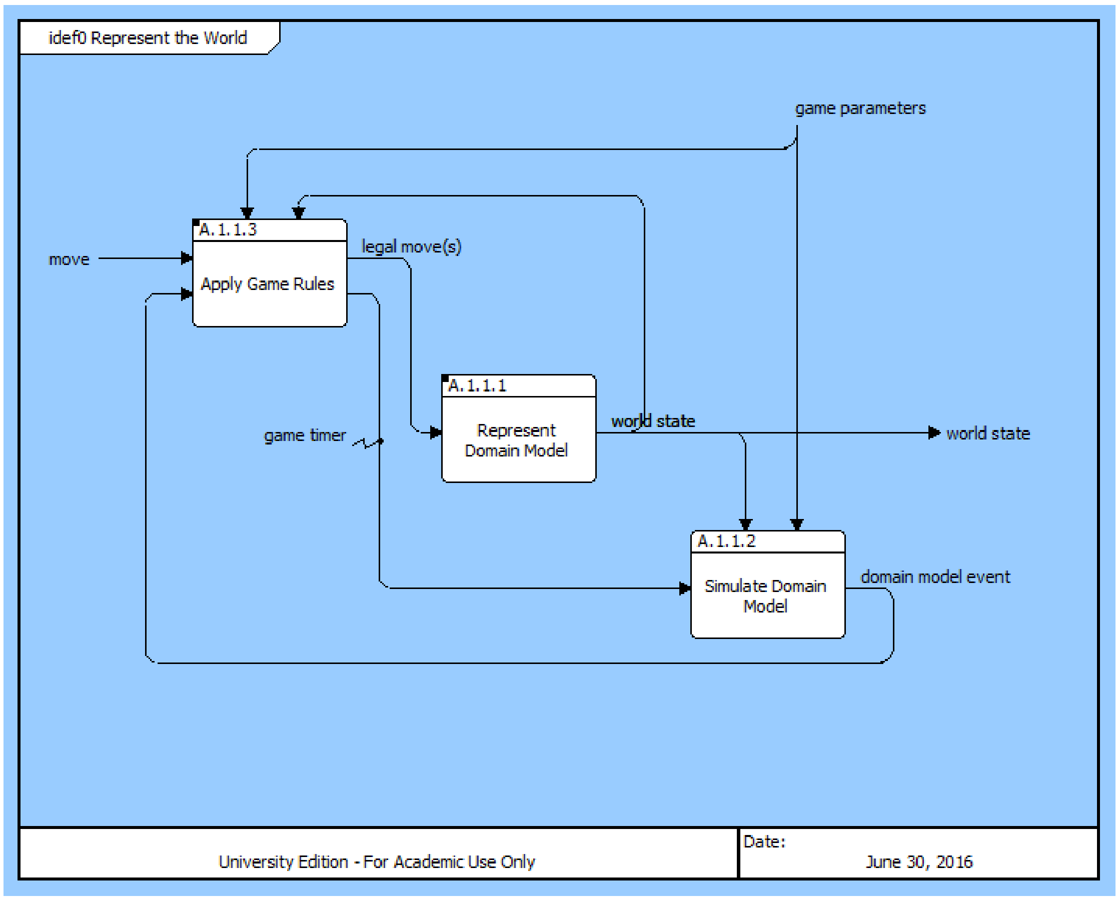
Within the domain model (the state of the world), three distinct types of information are stored:

- The "physical" blocks the world is made up of, in a board game these would be the pieces and the board. These are known as the structural components.

Figure 5.5: Level A.1.1



(a) Hierarchical diagram



(b) Interaction diagram

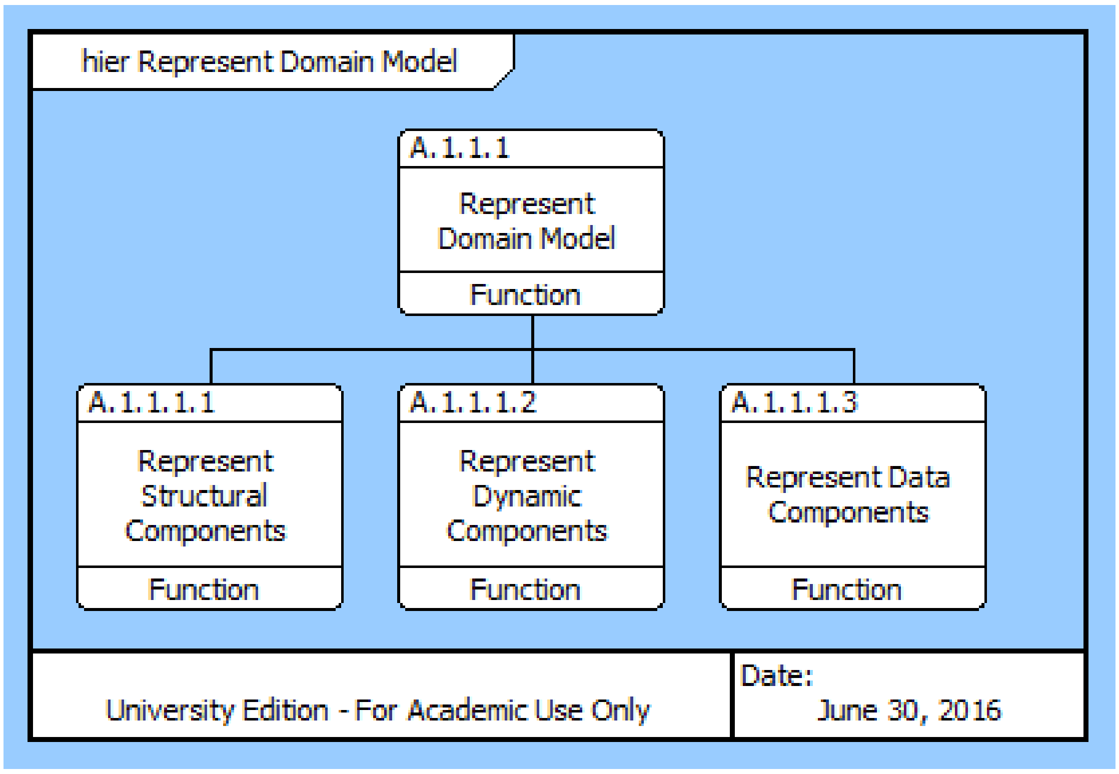

hier Represent Domain Model
A.1.1.1
Represent Domain Model
Function
A.1.1.1.1
Represent Structural Components
Function
A.1.1.1.2
Represent Dynamic Components
Function
A.1.1.1.3
Represent Data Components
Function
University Edition - For Academic Use Only
Date:
June 30, 2016

Figure 5.6: A.1.1.1 Hierarchical diagram, there are no interactions between these functions

- The dynamic components. They represent the things “flowing” through the game.
- “data”, representing anything intangible. Examples of data components are prices for goods that are set by the scenario instead of a (simulated) player, rates for natural disasters and so on.

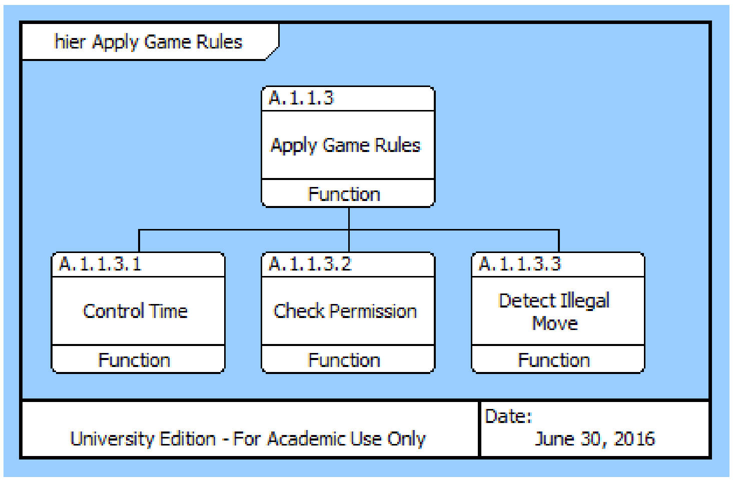
5.7 A.1.1.3 Apply Game Rules

This function is responsible for checking whether a player is allowed to perform a move at the current time. Presumably players will test the limits of the rules and those rules should therefore be strictly enforced. When the move is allowed or if the change comes from simulating the world, a check is made whether this would put the world in an illegal state. It is up to the game rules to resolve the illegal state before changing the representation of the world, thus ensuring the validity of the move. An example of an illegal state: After handling interest payments a player has less money then allowed. A resolution for this can be: bankrupting the player and eliminating him from the game.

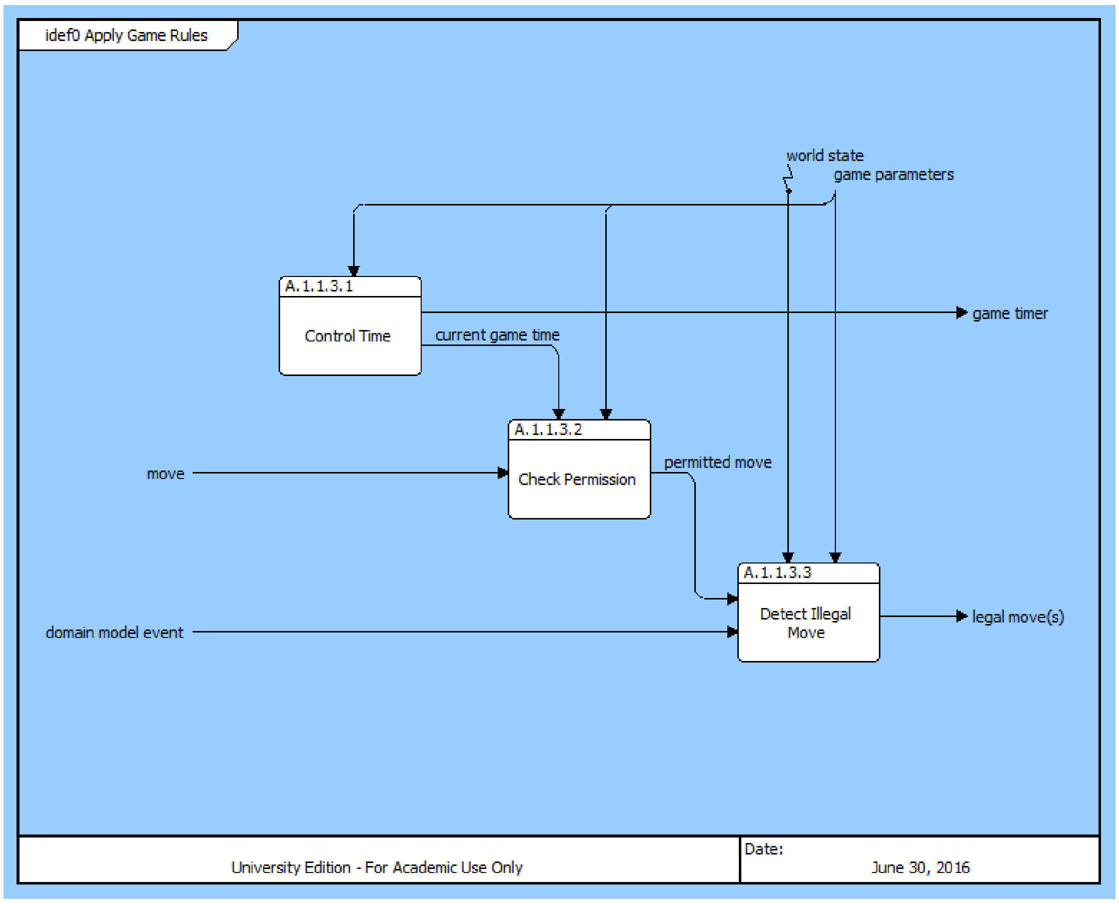
5.8 How to use this architecture

The architecture presented here serves at a starting point for aggregating information on the experiences of designing interactive simulations. When a designer is creating an interactive

Figure 5.7: Level A.1.1.3



(a) Hierarchical



(b) Interaction diagram

simulation, or is looking to validate it, this architecture can serve as the knowledge base for that design. When an interactive simulation has been designed, successful or not, the experience and knowledge gained can be added to this architecture to make it better for future designers. In other words, the rigor cycle should be applied to use and improve this meta-artefact.

Chapter 6 gives general remarks about how to use a functional architecture. It also gives step-by-step guidelines of how an agile development process would use the architecture. The architecture and guidelines together form the FASD framework. In Chapter 7 the process is described of how the architecture can be used to validate the game, that also gives validation of the architecture in the process.

Chapter 6

Guidelines for applying FASD

6.1 General comments about using the architecture

This functional architecture can be applied at every stage of the design process. The way to use it to validate a design, for example in order to compare it to other designs, is presented in Chapter 7. Here using the framework at the start of a project and during development are discussed.

6.1.1 At the beginning

Ideally this framework is used from the outset of the project. The functions of the architecture should serve as a common vernacular for the project so that it helps align the team.

A minimal viable product should be created by getting the cheapest implementation for each of the functions possible. This might include creating a game board with pencil and paper, and calculations being done by a "game master" in a spreadsheet.

Requirements for the research, and thus the game, will inevitably change. When this happens anybody involved, be it: project management, researcher, domain expert or anyone else, should be able to identify in what function(s) the change should happen. Having this clear distinction for the whole team will allow the nature of the change to be determined faster and the impact assessed more accurately. All of that combines to help the project stay on schedule.

Since all functions are implemented in some form from the start of the project the game should always be in a playable state. This will allow "play testing" to happen from the beginning. In this context "play testing" refers to playing the game to find shortcomings in the design, whether it be unwanted restrictions imposed by implementation, or elements that are frustrating for new or experienced players. These tests will help requirements evolve more quickly, hopefully stabilising earlier in development.

Because modularity of the design is imposed from the beginning expanding a single element will be less work than when the entire architecture would have to change to do that. An example is the rule checking module that was added to the GISEGas project very late and thus at much greater cost then it would otherwise have been.

6.1.2 During development

If a project has already started, applying the functional architecture can still be beneficial. The first step should then be a validation of the current design, fitting the design (or more importantly the implemented parts) to the functions of the framework. If any parts are missing they should be added in the same “minimal viable product”-way as described before. This will get the design on track for all of the time saving benefits also described above.

Getting the team to use the “new” terminology might prove difficult. But during the evaluation of the results of the previous step it is wise to check whether having this hierarchy of functions highlights a misunderstanding between the “implementation-people” and the rest of the team. At least a connection should be made between the terminology in use so far and the model, possibly supplementing it with elements that were missing or over looked up to now.

6.2 Using the architecture step-by-step

1. A.0 Perform Research - When wanting to use this framework, that must mean a research topic has been selected. With regards to design cycles, this means the environment is fixed and it can now be mapped, stakeholders approached etc.. Note that it is possible to try to answer questions from multiple topics with a single project, making the project more complex, but the base for funding wider.

2. A.2 Design Research - As the goal for going through this architecture is to apply the design cycle to create an interactive simulation, first the relevance cycle should be done.

   This means creating one or more research questions the stakeholders want to answer with this project. These research questions should be linked to hypotheses (A.2.1). Given these hypotheses scenario’s can be created (A.2.2) which lead to the requirements for the game.

   The requirements for the game should cover anything the game is able to do and anything a player is able to do during the game. The requirements should be prioritised, for example using the MoSCoW scale (Must have, Should have, Could have and Would like to have, a.k.a. Won’t have)).

   It is unlikely for the requirements to stay the same through the project and this step will have to be repeated several times during the course of the project to make sure the requirements and priorities are still in line with the project goals.

3. A.1.1.1 Represent the domain model - Given the domain(s) that are the topic of research for this project, determine what approximate granularity the domain should be recorded at. It will help finding what to represent and how by categorising the components into the sub-functions of A.1.1.1 (Structural, Dynamic and Data). Perhaps realising the similarities and dissimilarities between certain components.

   Now, decide how to present the domain model to the player, and try to find a cheap way to implement it. For example, if you are emulating an existing (board) game, try reusing its game pieces. Also make sure that when changes are made to the model, they are reflect in the representation the players get. Essentially implementing the Model and View parts of the Model-View-Controller pattern.

4. A.1.1.2 Simulate domain model - now that there is a domain model determine what the passing of time should do to the model. How this is implemented determines how often this can be triggered (i.e. having to do a lot by hand makes it infeasible to have it up date continuously). Having a domain expert that can explain how the domain works and/or show the tools of the trade will be a tremendous help here.

5. A.1.1.3 Apply Game Rules - When looking at the Model-View-Controller pattern this would be the controller, any changes that happen to the model should be “approved” and done by

this controller. Initially this might be the game master running the game. The controller is also responsible for making sure time passes in the game, triggering simulations cycles whenever appropriate, notifying players of their turn etc. (A.1.1.3.1).

Some of the game rules are there to make sure the simulated world corresponds to the real-world. It can be very useful to have a domain expert help determine what the rules might be. These will be mostly in the "Detect illegal move" (A.1.1.3.3) category. Others are directed at making sure the flow of game play works as planned, i.e. "Check permission" (A.1.1.3.2). For some rules it might not be clear in what category the fall, be aware that the domain expert might have an opinion on a rule that is there for a good reason that is not theirs to comment on.

6. A.1.2 Simulate non player actions - These should be actions that players are able to do, but not willing to, for example because it is tedious. Initially these tasks can be done by extra players or the game master, depending on how much time they take to perform.

7. A.1.3 Perform Stakeholder Actions and A.2.3 Host Game Session - Now it is time to host a first (test) game session. The player group should be "cheap" so not CEO's from all the big companies in the domain, but people who can provide criticism, preferably stakeholders close to the project. This session will allow evaluation of all of the previous steps and fine-tuning or even flat-out changing the requirements set up at the beginning.

   Playing the game even at this early stage might lead to formulation of new hypotheses or the reformulation of old ones. The scenarios should be checked to still match the hypotheses, the requirements should still match the scenarios and the implementation the requirements.

8. Iterating - Now parts can be build out, keeping in mind the research goals. Each iteration should make it easier to play/perform the research. If the goal of the research is to play the games a lot, making a game require less work from the researcher is most important (thus automating the things being done by the game master). Maybe having a continuously updating model is necessary, which means the cycle of updating the model and seeing those changes should be very short and time should be spent on automating that. After each iteration step 7 should be repeated, making sure the project is still on-track for all stakeholders and allowing them to correct course if necessary. Eventually an iteration will be "good-enough".

9. Research - Once the game is deemed "good-enough" research can start. New issues can crop up and can be fixed by going back steps 7 and 8.

Chapter 7

Validation & Discussion

As outlined in Chapter 3 the knowledge base and design science are linked through the rigor cycle. What this means is we can use the knowledge base, like to FASD framework to validate artefacts create by design science. The application of design science can subsequently feedback new insight into the knowledge base. In Section 7.1 a serious game project, other then GISEGas, is validated using FASD. The succes of that validation inspires confidence in the usefulness of the framework for other projects. In Section 7.2 potential criticism of FASD and the method that created it is presented and waylaid.

7.1 Validating FASD

The FASD framework is a meta-artefact. To unequivocally prove its value, two similar projects should be followed, one using the framework and one not. At the end metrics like cost efficiency and how well designed the end product is can then be compared to show if there is a benefit to using the framework. However, finding two projects similar enough to do that, and letting one intentionally "fail" is not commercially viable.

There is another way to show the frameworks value, using again design science research. The framework can be used in the validation step of the design cycle of an artefact where the framework could have been used. If the designed artefact is valid according to the framework, but fails in application or is invalid but succeeds, that disproves the framework. When the design proves to not be completely valid, the framework can be applied in the next iteration of the design cycle to make sure it becomes valid. This approach makes it difficult to say exactly how much the framework contributes, but is much cheaper and easier to do. Every time a designer feels their project improved from using the framework, that strengthens the claim it is correct. When the the lessons learned in the project lead to changes in the framework, that should make it even better.

Jan Willem Veeningen, who designed the "From Energy to Synergy" game, expressed interest in improving his game using FASD. This made a candid discussion about the state of his game and the possibilities to improve it a mutual interest. Section 7.1.1 describes his game, addresses which parts where valid, how the project could benefit from the parts that where not corresponding to the framework yet, and finally how this discussion validates FASD.

7.1.1 From Energy to Synergy

At the University of Groningen, Jan Willem Veeningen is working on the game: "From Energy to Synergy". In the game producers of renewable energy are challenged to sell their energy to industrial consumers directly, bypassing the main energy suppliers network. Outside of the main network they need to reach agreements on the price for the energy and how to cover the cost of infrastructure. Available game pieces are: windmills, electric lines, factories and batteries. The application of batteries to this problem was only recently made possible through technological advancement and is the main topic of research for this game.

Validating the design

Most of the higher-level functions are easily recognised. There is a representation of the world and some way for users to interact with that representation. Within the representation of the world there are the concepts of structural, dynamic and data components, and the domain model can run through a set of calculations that represent the passing of time (simulate domain model).

Some of the more specfic functions did not have clear answers yet. At the research level for example, the player group composition was not well defined. This can however be attributed to the project not yet having reached an iteration where this was very significant.

Improving the design

The framework is intended for use in a software development scenario. As such it is based on traditional Object Oriented concepts like encapsulation and delegation. It also intends to suggest useful superclasses which can be extended for specific implementation. "From Energy to Synergy" was not yet being developed in a higher level programming language and relied mostly on spreadsheets, like gasboard. This limits the usefulness of the framework for the current code base, but can greatly help in identifying which parts do what currently, and how that should translate to the next fase of the project.

The function: "Apply Game Rules" is not covered by the current implementation. Violation of the rules is dealt with by indicating to the player when an illegal move is about to happen and trusting that the player will correct his or her input. When a player neglects to act on the warning this might invalidate the rest of the experiment. Confusion about what is valid, and who should input what exactly has already lead to issues during game sessions. Having the game give clear feedback about what cannot happen and only applying what can happen can help resolve this problem.

"From Energy to Synergy" Is played by players editing the world state directly in a Google Docs Spreadsheet. The spreadsheet is both the input/output mechanism and representation of the game world. Though Google docs, as a tool for collaboration, does versioning and keeps track of who makes changes, this setup presents challenges for analysing how the world state develops. When actions are clearly specified and approved before being allowed to change the world state finding how a state came to be is much easier and controlling what can and cannot happen can be controlled more strictly. The decision for implementing the game like this can be defended by the phase the game is in, but will limit the game from growing beyond its current application.

Validating the framework

No elements of FASD where invalidated. Whenever a function is not performed by any part of the implementation this is identifiable as a part where the game needs to be developed further. It also seems, subjectively, that having the framework will boost further development giving clear guidelines for what categories need to be covered, allowing for a translation between the current project and future iterations.

7.2 Limitations

The main counter agains the method used to develop FASD is that it relies on only a single example of a serious game. This does limit how good this version of FASD can be, but there are two reasons why this is not a big problem. First, having only one game is mitigated by GISEGas being a very good game to base this initial version on. Second, the framework needs to be iterated upon anyway, this is merely a starting of point.

GISEGas is a good case because it addresses the needs of many diverse stakeholders. Those stakeholders provided input while the game was still under development, leading to a very agile game suitable for many purposes. The framework drawn from GISEGas is as diverse, and therefore generally applicable, as the applications the game was used for.

Applying design science [25], the way to get a better (meta-)artefact is to iterate on it. The functional architecture presented here should be a sound first iteration, but by no means the last. Through the "From Energy to Synergy" project it has already contributed to the development of another game, proving its value and significance. In addition to that, Jan Willem has indicated his intention to contribute his findings back to the meta-artefact adding a function dealing with the creation of scenarios.

Chapter 8

Conclusion

The first question this thesis answers is $RQ1_a$ asking how to convey development guidelines for serious games in general and interactive simulations especially. The main challenge for this type of project is that they are one of a kind, dedicated software projects. The answer was found in the field of Design Science Research combined with Systems Engineering. Guidelines should be captured in an IDEF0 diagram which shows the functions that need to be covered by any interactive simulation implementation, regardless of the domain it covers. Such a meta-artefact, describing not the general design of an artefact, but how to approach designing an artefact is called a functional architecture.

Answering the real research question, $RQ1$, creating the guidelines as a functional architecture requires studying an example of an interactive simulation. Since this case should serve as a template for as many projects as possible the very broadly applied GISEGas is an excellent example. The lessons learned have been converted into a functional architecture describing all of the major elements that need to be covered when designing an interactive simulation. A series of steps describing how to apply the architecture to a project are also provided. Together the steps and architecture form the FASD framework. FASD should help any serious game, but especially interactive simulation, achieve a better design faster.

The FASD framework was compared to an ongoing interactive simulation development project validating its usefulness and correctness to a promising degree.

8.1 Future work

8.1.1 Validate further design through usage

When a game is designed from the ground up using the FASD framework, the success and/or specific failures of such a project will show the validity and shortcomings of this framework. As with “From Energy to Synergy”, even ongoing research projects involving serious games can use this framework without having to start from the beginning. Instead the researcher might find components that are needed earlier, and adjust the planned development efforts accordingly.

Whenever such a project leads to new insight, whether it be validating the existing framework or improving on its shortcomings, they should feedback into the knowledge base. Allowing subsequent projects to benefit from the lessons learned.

8.1.2 Improve design based on more real-world data

A different approach can be taken, checking existing serious game designs against the framework. Checking whether being valid according to the framework and the project being a succes can give insight into the correctness of the framework. Looking at the difference between succesful and failed projects in the context of a functional architecture can also provide insight into functions that can be added or expanded upon in the framework.

8.1.3 Software architecture and components

Though by its very nature a functional architecture does not mandate a specific physical or software architecture, one could be created to fit this framework. Such a framework could provide even more of a springboard for interactive simulation development. An example of a component that could benefit from such treatment is the GISEGas Kernel, which provides a template for player interaction using TALL (The Agent Lab Language), handeling turn permissions and restrictions.

Chapter 9

Bibliography

[1] Jinsoo Park & Sudha Ram Alan R. Hevner, Salvatore T. March. Design science in information systems research. *MIS Quarterly*, 28:75–105, 2004.

[2] Alan Amory. Game object model version ii: a theoretical framework for educational game development. *Educational Technology Research and Development*, 55(1):51–77, 2007. URL: `http://dx.doi.org/10.1007/s11423-006-9001-x`, `doi:10.1007/s11423-006-9001-x`.

[3] Eike Falk Anderson, Steffen Engel, Leigh McLoughlin, and PeterComninos. The case for research in game engine architecture. In *FuturePlay*, pages 3–5, November 2008.

[4] Dennis M. Buede. *The Engineering Design of Systems: Models and Methods.* Wiley, 1999.

[5] M. B. Carvalho, F. Bellotti, J. Hu, J. B. Hauge, R. Berta, A. D. Gloria, and M. Rauterberg. Towards a service-oriented architecture framework for educational serious games. In *2015 IEEE 15th International Conference on Advanced Learning Technologies*, pages 147–151, July 2015. `doi:10.1109/ICALT.2015.145`.

[6] Maira B. Carvalho, Francesco Bellotti, Riccardo Berta, Alessandro De Gloria, Carolina Islas Sedano, Jannicke Baalsrud Hauge, Jun Hu, and Matthias Rauterberg. An activity theory-based model for serious games analysis and conceptual design. *Computers & Education*, 87:166 – 181, 2015. URL: `http://www.sciencedirect.com/science/article/pii/S0360131515001050`, `doi:http://dx.doi.org/10.1016/j.compedu.2015.03.023`.

[7] Catan. Official website for settlers of catan [online]. URL: `http://www.catan.com/`.

[8] Luca Chittaro and Riccardo Sioni. Serious games for emergency preparedness: Evaluation of an interactive vs. a non-interactive simulation of a terror attack. *Computers in Human Behavior*, 50:508 – 519, 2015. URL: `http://www.sciencedirect.com/science/article/pii/S0747563215002757`, `doi:http://dx.doi.org/10.1016/j.chb.2015.03.074`.

[9] S. De Freitas, G. Rebolledo-Mendez, F. Liarokapis, G. Magoulas, and A. Poulovassilis. Learning as immersive experiences: Using the four-dimensional framework for designing and evaluating immersive learning experiences in a virtual world. *British Journal of Educational Technology*, 41:69–85, 2010.

[10] Board Game Geek. Power grid review on board game geek [online]. URL: `https://boardgamegeek.com/boardgame/2651/power-grid`.

[11] Alan Hevner and Samir Chatterjee. *Design Research in Information Systems.* Springer, 2010.

[12] George Huitema, Nick Szirbik, Hans Wortmann, Ashwin Ittoo, and Austin Dzousa. Embedding flexibility in combined gas and electricity infrastructures. In *4th Research Day EDGaR*, 2012.

[13] Robin Hunicke, Marc LeBlanc, and Robert Zubek. Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, volume 4, page 1, 2004.

[14] Ashwinn Ittoo, Nick Szirbik, George Huitema, and Hans Wortmann. Serious gaming augmented with natural language processing for learning on energy infrastructure investment decisions. In *11th Annual Industrial Simulation Conference*. 11th Annual Industrial Simulation Conference, 2013.

[15] Samad Kardan and Cristina Conati. Providing adaptive support in an interactive simulation for learning: An experimental evaluation. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3671–3680, New York, NY, USA, 2015. ACM. URL: `http://doi.acm.org/10.1145/2702123.2702424`, `doi:10.1145/2702123.2702424`.

[16] Tim Marsh and Bonnie Nardi. *Spheres and Lenses: Activity-Based Scenario / Narrative Approach for Design and Evaluation of Entertainment through Engagement*, pages 42–51. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. URL: `http://dx.doi.org/10.1007/978-3-662-45212-7_6`, `doi:10.1007/978-3-662-45212-7_6`.

[17] Louise Møller and Poul Kyvsgaard Hansen. Framing serious games development as a matter of business. *International Journal of Serious Games*, 3(1), 2016.

[18] A. Mora, D. Riera, C. Gonzalez, and J. Arnedo-Moreno. A literature review of gamification design frameworks. In *Games and Virtual Worlds for Serious Applications (VS-Games), 2015 7th International Conference on*, pages 1–8, Sept 2015. `doi:10.1109/VS-GAMES.2015.7295760`.

[19] Seungjae Oh, Kyudong Park, Soonmo Kwon, and Hyo-Jeong So. Designing a multi-user interactive simulation using ar glasses. In *Proceedings of the TEI '16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '16, pages 539–544, New York, NY, USA, 2016. ACM. URL: `http://doi.acm.org/10.1145/2839462.2856521`, `doi:10.1145/2839462.2856521`.

[20] ReportBuyer. Serious game global market outlook - trends, forecast, and opportunity assessment (2014-2022) [online]. October 2015. URL: `http://www.prnewswire.com/news-releases/serious-game-global-market-outlook---trends-forecast-and-opportunity-assessment-2014-2022-300168238.html` [cited 01-04-2016].

[21] Adam M. Ross, Matthew E. Fitzgerald, and Donna H. Rhodes. Game-based learning for systems engineering concepts. In *Conference on Systems Engineering Research (CSER 2014)*, 2014.

[22] Tarja Susi, Mikael Johannesson, and Per Backlund. Serious games - an overview. Technical Report HS- IKI -TR-07-001, School of Humanities and Informatics, 2 2007.

[23] Nicolae Szirbik, Christine Pelletier, and Vincent Velthuizen. *Enhancing an Integrative Course in Industrial Engineering and Management via Realistic Socio-technical Problems and Serious Game Development*, pages 541–548. IFIP Advances in Information and Communication Technology. Springer, 2015.

[24] Gary Ushaw, Richard Davison, Janet Eyre, and Graham Morgan. Adopting best practices from the games industry in development of serious games for health. In *Proceedings of the 5th International Conference on Digital Health 2015*, DH '15, pages 1–8, New York, NY, USA, 2015. ACM. URL: `http://doi.acm.org/10.1145/2750511.2750513`, `doi:10.1145/2750511.2750513`.

[25] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineerin" Excerpt From: Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014.

[26] José P. Zagal and Amy Bruckman. The game ontology project: Supporting learning while contributing authentically to game studies. In *Proceedings of the 8th International Conference on International Conference for the Learning Sciences - Volume 2*, ICLS'08, pages 499–506. International Society of the Learning Sciences, 2008. URL: `http://dl.acm.org/citation.cfm?id=1599871.1599933`.

Appendix A

Enhancing an integrative Course in Industrial Engineering and Management via Realistic Socio-technical Problems and Serious Game Development.

Enhancing an Integrative Course in Industrial Engineering and Management via Realistic Socio-technical Problems and Serious Game Development

Nick Szirbik[1]( ), Christine Pelletier[2], and Vincent Velthuizen[3]

[1] IEM/Operations/University of Groningen, Groningen, The Netherlands
n.b.szirbik@rug.nl
[2] IBS/Hanze University of Applied Sciences, Groningen, The Netherlands
c.m.pelletier@pl.hanze.nl
[3] Trias Informatica, Groningen, The Netherlands
vincent.velthuizen@triasinformatica.nl

Abstract. This is a position paper. It discusses specific educational issues encountered during the Systems Engineering Design course at the Industrial Engineering and Management master program at the University of Groningen. It explains first the concept of an integrative course, an innovation that was applied first in this master program. It explains the causes and effects of two observed educational shortcomings of this course, and it links these to the extant literature. Finally, the paper proposes two ideas to address these shortcomings.

1 Introduction and Background

According to the current standards on engineering education [1], the courses in an engineering study program can be classified either as lecture-centric (LC), problem-based (PB or inquiry- based), project-assisted (PA), or project-centric (PC sometimes, these courses are named capstone courses). In the Industrial Engineering and Management (IEM) study program at the University of Groningen (RUG, in the Netherlands), an innovative learning paradigm has been developed and applied in the last decade: the integrative course. This kind of course is one that is placed at the end of the curriculum schedule, and requires the application of knowledge and skills learnt from the majority of the previous courses. The integrative courses are meant to apply both paradigms of problem-based learning and project-centric learning. This concept may sound similar to the capstone course [10].

Currently, at IEM, there is an accumulated experience of teaching integrative courses over the past 8 years, and there has been confirmation that the outcomes of these courses are the desired ones. The first source of confirmation has been the short-term and long-term feedback given by the graduates and alumni from the last decade. The second source was the industry, and the third source was

S. Umeda et al. (Eds.): APMS 2015, Part I, IFIP AICT 459, pp. 541–548, 2015.
DOI: 10.1007/978-3-319-22756-6_66

the formal accreditation reports of the IEM program, revealing an increase in the quality of the master theses produced in the last years.

Despite the apparent success of the integrative courses revealed by these sources, the teachers who are involved in their development are aware of two major shortcomings that reduce the effectiveness of teaching in the integrative manner. The extant literature [6] and exchange of experience (via personal channels) on the subject show that these shortcomings are not unique at RUG. The first shortcoming is the lack of realism and veracity of the "problem" that is to be solved by the designs developed during the integrative study. The second shortcoming is the lack of cognitive engagement between the teams and individual students who are doing the designs for the coursework, leading to a fragmented project development, and also a lack of communication and alignment in the integrative process. Later in this paper, the authors propose two ways to mitigate these shortcomings.

In the Netherlands, the higher education in engineering is provided by some of the existing public universities. In addition to these, there are a few dozen universities of applied sciences (UAS) and offering bachelor and more recently master level programs in various engineering disciplines.

During the 1990s, two change tendencies emerged in Dutch universities. The first change was driven by the industry, which asked more vocally for engineering students with skills like team work, context and problem complexity understanding, the ability to diagnose complex problems, communicate the findings, and the preparedness to coordinate solution efforts. At its core, this request challenged the old style of "chalk and talk" (i.e. LC) teaching. The universities responded in various ways, one being by applying PB learning, and another by inviting more and more professionals as guest lecturers. It was quickly learned later that PB learning is not the single or even the best solution for engineering.

The second change was to respond to the demand of the industry for a more multi-disciplinary kind of graduate. The response was a new type of engineering study program, namely the IEM. These master degrees are supported in each of these universities by a similar bachelor study program in IEM. Students who earn a bachelor degree in Industrial Engineering from an UAS can follow a special adaptation coursework for half a year and then start one of these master studies in IEM.

These IEM programs appeared exactly when PC and PB learning started to be applied in various engineering programs universities in the world. The disciplines that can be taught in PB style are those where knowledge is rather encyclopedic and less hierarchical (that is, the order in which the concepts are learned is not so important). The emphasis in these courses is to investigate a "situation/problem", and find its causes (diagnostic). This is similar to the initial application of problem-based learning in medicine [11], and students learn and put together the pieces of knowledge along the way in finding the causes of the investigated "problem" (i.e. finding a diagnostic for it).

For those disciplines where knowledge is strictly hierarchical, the style of teaching is either PA or PC. The emphasis in PA courses is on analysis. The "problem/opportunity" that triggers the design is given by the teacher, and its causes and diagnosis are clearly established beforehand. The students have to analyse and specify requirements for a future design.

The disciplines taught in PC style, like for example software project management and control systems design (for robotics), have a strong emphasis on the design itself - the problem, stakeholders, and the requirements are mostly given beforehand by the teacher. Typically, project teams are smaller, of 3–5 students, each having a small portion of the design to tackle. Design of different teams are sometimes integrated into bigger designs, spurning collaboration between teams. Periodic presentations of the teams help that knowledge and acquired skills (and again, mistakes) are shared and discussed.

2 The Integrative Course and Its Shortcomings

At RUG, the master IEM program is ending with courses that apply an integrative approach. The emphasis is on technical integration, but both courses (the course in Systems Engineering Design and the course in Sustainable Engineering) have still a strong managerial/economic-related part.

In the integrative courses, all the learning styles (LC, PB, PA, and PC) are applied. There are plenary lectures, which present mostly examples of problems, stakeholders, requirements and design solutions. The "problem" has to be found/defined/invented by the students - with teacher's assistance. Then, students have to analyze specify the requirements of a complex and multi-disciplinary system, and finally develop a conceptual design that meets the requirements and can stand a formal design assessment - an evaluation phase that is also performed by the students. Finally, the teachers role is that of the "buyers" of the design, who are involved in the design in iterative stages of its development, revealing its shortcomings, gaps, and inconsistencies.
The course Systems Engineering Design is the "oldest" integrative course in the IEM master program at RUG. Students are organized in teams of 3–4 students, and teams are organized in "triads" (three teams) of 10–13 students each. Each triad has a separate student who plays the role of system integrator. The tasks of the project are to define first - separately for each triad - a specific multi-disciplinary problem (which has to be more or less realistic), and explore the "stakeholders' " wishes. Next, the teams have to analyse and specify the requirements of a system that addresses the effects of the problem - each team in the triad specializes in a specific set of requirements. Finally, the system has to be designed in terms of a context-placed operational architecture, comprising a functional architecture described in IDEF0 and a generic physical architecture with alternatives for components and interfaces. The final deliverable is a dossier that is supposed to participate in a Request for Proposals (RfP) for such a system. The project is quite equivalent (albeit much shorter in time and smaller in scope) to the pre-inception phase [2] in a system development process.

The operational architecture has to be built by using a systems engineering CAD system (CORE^{tm}9 from the Vitech Corp.), which incorporates also requirements engineering tools. This CAD system allows for collaborative design and the integration of separate structures/interactions of functions and components that were designed by different teams. A recent example of system designed was a biogas producing and storage infrastructure, which was supposed to replace/upgrade an existing infrastructure for fossil gas in a clearly delimited region. Because this course runs in parallel with the Sustainable Systems course, many problems are related to environmental issues, alternative fuels, and novel energy systems.

The students are encouraged to view this course as a "serious game", where they compete to "sell" their design to the issuers of the RfP (see presentation [12]), and there is a constant competition for the best team and best triad. Moreover, the teachers (as a separate team) are developing each year a system in parallel with the students, sometimes as part of a triad with two other student teams. If a student team delivers a better project than the teachers, who get always a mark 9 (on a scale from 5 to 10), that student team gets the maximum mark. Each year, one or even two student teams manage to "beat" the teachers' team.

This course is popular with other students than the IEM master program, for whom the course is compulsory. Each year, 15–20 % of students are taking this course as elective, and they are coming from master programs like Computing Science, Energy and Environmental Science, Chemical Engineering, Econometrics, and even Law. They bring even more disciplinary knowledge and skills into the triads, and also ways of thinking that are new for the IEM students and even the teachers. The course is highly sought by exchange students also, who are studying at RUG only for a semester. However, there are also shortcomings.

Problem-based learning assumes that the problem is real, existing in a real context. In PB courses in medicine for example, the students are given a real patient case, which they have to investigate and diagnose. In PB courses in engineering, the students go into the field (typically a company) and are given or identifying a real problem. However, due to time and resource limitations, courses that have a design project at their core, hardly can cope with overloading the coursework with a diagnostic phase. Another typical issue is that teachers that are teaching in PC style, have limited experience with the PB approach. The net result is that in PC course that starts with a "problem" that is made up, typically by the teachers, who try to communicate its details as well as they can to the students. However, the problem "exists" only in the mind of the teachers, and many times, students do not grasp or have a really good understanding what the problem really is. The single way to figure out, is to continually "interview" the teachers about the problem, but many times this can lead to even more confusion. The net effect is that students are losing interest in the problem and in the project, going out from the "immersion", "make-believe", and "flow" state that make them feel like in an engaging game.

A simple solution (applied currently in the course Systems Engineering Design) is to let the students define the problem/opportunity themselves (with some teacher feedback). They have to describe it in a scenario-like narrative, and expand the details as they go with the design. However, this approach leads to other undesired effects. The problem becomes part of the design itself, and in order to have an elegant and technologically interesting design, the problem is changed in ways that bend reality. Especially the quantitative aspect of the problems are suffering, and this can lead to problems that do not have a holding in physical reality, because the assumptions made lead to the violation of the laws of physics. The net effect is that students realize the lack realism of their problem at some moment, and again, they lose interest in the project overall. If they remain immersed and that happens many times, this creates an even more dangerous learning effect, because the students may remain convinced that this kind of unrealistic problems and designs exist in reality. If the students are "brought back to reality" during the final presentation or the previous feedback sessions with the teachers, this will create a sense of failure, and again, they lose interest and are not engaged anymore. These effects have been observed by other researchers who attempted to use elements of PB learning in PC settings [3,8].

The second shortcoming is related to the design phase; after the problem, stakeholders, and requirements have been clearly established. Students working in design teams tend to divide the work in chunks that are doable by single students (using the given CAD system) and integrate this separately made work later. Except for the feedback sessions, the work does not need to be integrated, and students tend to work in isolation. This leads to fragmentation lack of focus and understanding of the system as whole - and less cognitive engagement between team members and the teams in the triad. In this phase, the students put most of the effort in mastering the CAD environment and produce the functional and physical designs. Unfortunately, this leads to less communication, low team performance, no collaboration-induced creativity, friction between team members (because they do not understand properly what the others are doing), frustration, and in the end boredom or anxiety. The "flow" mental state sought by using the serious game in the design competition is not achieved any more by some teams. There is no motivation to finish with an exciting result, and there is no intrinsic curiosity left to explore and find alternatives and new ideas for the design. Because the used CAD system is a folder and menu based system that has a graphical interface for IDEF0, the whole work seems to become 99 % mouse clicks and menu navigation, at odds with a creative process, and distracting the students from team communication and reciprocal creative thinking. This kind of effects have been observed for many years in real design projects that use CAD systems [5,7], and it is not surprising to find these in PC courses that use CAD systems and teamwork.

An interesting finding came out last year, when members of a triad of students played a digital multi-player serious game, which was developed by the gas industry with RUG collaboration. This triad was the one that developed the bio-gas system, and it had the opportunity to visualize the system they were designing in the form of a business game, which mimicked the development of the bio-gas infrastructure, with implications to the gas markets and investments. At the end of the game, these students were interviewed to find out if the game playing helped them in the process to design their bio-gas system. The most interesting finding of these interviews was that the game actually improved communication within the triad in the "boring" design phase (the game-playing took place in the second half of the course), and helped them to keep an eye on the whole of the system, and remain engaged in achieving an satisfying result. This was in line with findings in the literature [4]. The identification and validation of these two shortcomings have been done via qualitative research: observation of how students work during tutorial classes, end-of-course survey via open-question forms, and "post-mortem" workshops with students who volunteered to participate, where various aspects of the course were discussed openly, and the findings formalized.

3 Proposed Solutions

For the first shortcoming (problem's lack or realism), the proposed solution is to have a real problem owner, and a real problem. The IEM students who are doing their master thesis in the study year that follows the Systems Engineering Design course are following a three phase curriculum: first a LC course on Research Methodology (with an emphasis on Design Science Research aspects), second a Research Project, and third a Design Project. The last two phases, or at least the last phase, take place in the context of a company, which is the problem owner for the design of the student.

When the Systems Engineering Design course starts (in April), the senior master students who are doing their master thesis project are well advanced in their track, that is, in the middle of the Design Project phase. At that moment, the problem they have to address by their design is very clear, and they started to implement the improvement, or a novel process, or a novel product, or a system that addresses this problem. There are in total 40–50 students whose thesis and design project is in this status. The idea is to recruit a number of students as teaching assistants (TAs) for the Systems Engineering course which is smaller than equal than the number of triads in course. The selection would be based on their previous performance in the course, and the nature of their design (if its scope is a complex, multi-disciplinary system, the better). These students will play the role of problem owners and main stakeholders in the design of a similar system by a triad. There is no need to have an exactly similar scope and nature of the system, but it is expected the problem owner will keep the assumptions made more realistic.

This approach is expected to be advantageous not only for the triad, which will be helped to keep the problem and the design realistic, and drive the design from the problem and not vice-versa. The problem owner student can use the triad as an exploratory design team, and apply their ideas in its own thesis work.

For the second shortcoming (fragmentation of focus during the design phase), the idea is to develop a simple board game that mimics the development of the proposed system. This idea was actually proposed by the students who played the serious game for bio-gas infrastructure. They complained that the digital game was too restrictive and impossible to adapt to their own ideas about the proposed bio-gas system. Because the game had a previous non-digital version, a board-game where game pieces mimicking the components of the system where place on a map, they argued that it would have been more interesting and useful to have a board game that was easy to change according to the changes they made in their own design. The board can be hand drawn each time the game is played, and new pieces (representing new components) can be improvised easily from old board game pieces, or even 3D printed. The rules of the game should reflect the behavior of the system and its development. The main issue with such a game is its final purpose, which is important for its developers [9]. The current view is that such a game is to be developed to allow stakeholders to play to understand better the system design, and also its potential development. The players who will play the roles initially will be students themselves, but they will design the game having in mind that the real players will be stakeholders. If a problem owner student is attached to a triad that designs a game, this student can participate as a real stakeholder, and also bring the game within the company where the real design project takes place, and engage the potential stakeholders to play the game, communicating the proposed design in a user-friendly manner.

However, the main educational purpose of the game is to prevent fragmentation and bring together the triad members in the phase when they tend to work too much in isolation. To enforce that the game is played regularly, the triad coordinator student should be tasked to organize gaming session twice per week, and have sometimes teacher participation if that is possible. For example, the last feedback session of the course should involve the teachers as players in the newly developed games.

4 Discussion and Future Work

When applying these ideas, some problems can be envisaged. For example, if there are 8–9 triads in a course, that will necessitate an early planning and strict coordination. The 8–9 appropriate problem owners within the group of senior master students have to be found before the course starts. Also, these problem owners should have design tasks, or at least contexts, that qualify as “complex system designs”. The intention is that for the first year of idea’s application, only a few triads (2, maximum 3) will be matched with a problem owner. At the time of writing, 2 master students who are finishing their master thesis, and a PhD student are already allocated to the role of problem owner. When the course will be taught this year, 3 triads will face a real problem from a company, and their assumptions will be “guarded” by a student tasked for this role.

For the design of the board game, expertise in game design is needed - and it is currently sought after. Initially, there will be only one triad that will attempt a board game design, and only those students who have experience and interest with games will be asked to volunteer for this triad. The intended system design for this experiment will be an electric energy storage infrastructure, modelled on the previous bio-gas infrastructure and its related board game. The teachers and the study program will invest supplementary effort and resources in this triad, and the experience will be carefully documented.

An important follow up to applying these ideas is to assess the impact of these improvements, short term and long term, and communicate the results and eventually interact with programs that have similar approaches. A related research theme is to investigate if there is an advantage in designing a serious board (+digital) game for the stakeholders in the pre-inception and inception phases of the design of real systems.

5 Conclusions

The use of serious games seems to be promising for improving the effectiveness of PC learning and integrative learning in engineering education, especially in IEM programs. For the moment, there is little experience in applying them, and quick and efficient ways to develop board games that mimic a system's development are yet to be discovered and evaluated.

References

1. ABET: Criteria for Accrediting Engineering Programs (2012). http://www.abet.org/uploadedFiles/Accreditation/Accreditation_Step_by_Step/Accreditation_Documents/Current/2013_-_2014/eac-criteria-2013-2014.pdf
2. Buede, D.M.: The Engineering Design of Systems: Models and Methods. Wiley, New York (2009)
3. Fink, F.K.: Integration of engineering practice into curriculum 25 years of experience with problem based learning. In: ASEE/IEEE Frontiers in Education Conference, Session 11a2, 7–12 (1999). http://fie.engrng.pitt.edu/fie99
4. Honey, M.A., Hilton, M. (eds.): Learning Science Through Computer Games and Simulations. National Academies Press, Washington (2011)
5. Kosmadoudi, Z., Lim, T., Ritchie, J., Louchart, S., Liu, Y., Sung, R.: Engineering design using game-enhanced CAD: the potential to augment the user experience with game elements. J. Comput. Aided Des. **45**, 777–795 (2013). Elsevier
6. Lattuca, L.R., Terenzini, P.T., Volkwein, J.F.: Engineering change: a study of the impact of EC 2000, executive summary. ABET, Baltimore, MD (2006)
7. Luczak, H., Beitz, W., Springer, J., Langner, T.: Frictions and frustrations in creative-informatory work with CAD systems. In: Bullinger, H.-J. (ed.) Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals. Elsevier (1991)
8. Mills, J.E., Treagust, D.F.: Is problem-based or project-based learning the answer. Eng. Educ. (4) (2003). http://www.aaee.com.au/journal/2003/mills_treagust03.pdf
9. Pasin, F., Giroux, H.: The impact of a simulation game on operations management education. Comput. Educ. **57**, 1240–1254 (2011)
10. Paulik, M.J., Krishnan, M.: BA competition-motivated capstone design course: the result of a fifteen-year evolution. IEEE Trans. Educ. **44**(1), 67–75 (2001)
11. Smith, K.A.: Inquiry and cooperative learning in the laboratory. In: Proceedings of the ABET/Sloan Conference on Distance Learn. Practice Oriented Professions (2002)
12. Szirbik, N.: Experiencing the systems engineering process as a serious game. Presentation at the CORE-users Conference, Washington, DC (2013). Available as video at https://www.youtube.com/watch?v=0C-mMIrxqyA