UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science
Information Technology Curriculum

Taavo-Taur Tammur

Web-Based Single-Player Project Simulation Game

Bachelor's Thesis (6 ECTS)

Supervisor: Dietmar Pfahl, PhD
Seminar Supervisor: Margus Niitsoo, PhD

TARTU 2014

Web-Based Single-Player Project Simulation Game

Abstract:

The goal of this thesis is creating a simulation model for software development and implementing it as a part of a web based single-player simulation game.

Keywords:

Software Project, Simulation, Game

Veebipõhine Üksikmängija Tarkvaraprojekti Simulatsioonimäng

Lühikokkuvõte:

Selles lõpputöös tehakse tarkvaraarenduse simulatsioonimudel ja selle rakendamine veebipõhise üksikmängija simulatsioonimängu osana.

Võtmesõnad:

Tarkvaraprojekt, Simulatsioon, Mäng

Table Of Contents

Introduction

Software development is a complex process that is constantly influenced by decisions and events that happen during it. As methods mature, higher quality and faster development is demanded from the resulting projects. However as found by McKinsey and the BT Centre for Major Programme Management at the University of Oxford [1], 66% of software projects end up costing more than budgeted, and 33% of software projects overrun their schedule.

In order to try and improve the situation you could give the project managers experience on how (not) to fail at running their projects. One possible way to do that is creating a simulator for project managers. Simulators have already shown value in training for other fields, such as transportation in training plane pilots [2] and train drivers [3].

The goal of this thesis is creating a simulation model for software development and implementing it as a part of a web based single-player simulation game using modern tools.

Related Work

Previous work on this topic includes “Design and Implementation of a Scenario for Simulation-based Learning in the Domain of Software-Engineering” (2000) by Marco Klemm [4]. However it's shortfall is the dependency on a custom browser plugin for running the simulation, especially as recently people have become hesitant to running browser plugins due to it being the main vector for malicious attacks [5]. Hence one of the goals of this thesis is also creating it without using any external plugins.

In addition there have been numerous video games made in the simulation genre that are related to software development, namely “Game Dev Story” by Kairosoft [6], “Game Dev Tycoon” by Greenheart Games [7] and the GameBiz series by Velocigames [8]. These however focus on the business side of development, where running a successful business is the focus of the games.

Thesis Overview

In Chapter 1 of this thesis, a model is created for representing software development. In Chapter 2, the game side of the project is explained. Lastly, the last Chapter recaps the completed work and any possible future extensions to the project.

1. Model

The game is built around a model that represents software development. It is based on agile software development principles, where the development is composed of sprints/iterations (1~2 week periods where a subset of user stories are developed into the product). The model has been simplified in order to ease implementing it in JavaScript for the game.

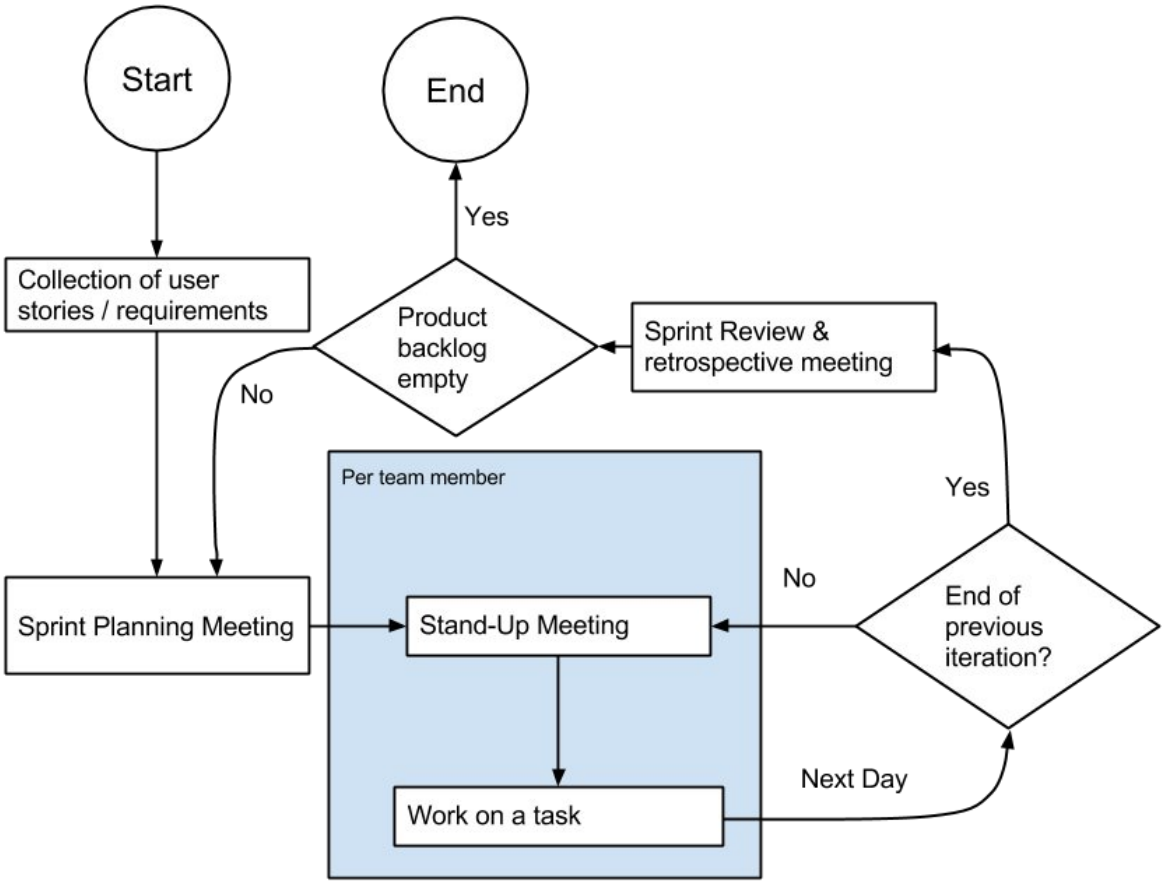

Figure 1: Software Development Model

In the model the following is done:

1. **Collection of user stories / requirements** – User stories and tasks that are needed to be worked on are generated
2. **Sprint Planning Meeting** – User stories that are being worked on in the sprint are selected by the manager. The tasks of the user story go from product backlog into the sprint backlog.

3. **Stand-Up Meeting** – At the beginning of every day all workers take part of a stand-up meeting where team members report the task they are working on.
4. **Working on a Task** – Workers work on their tasks either until the task is done or when the day ends. Once a task is complete it is removed from tasks being on worked on and the worker chooses another task that will be worked on.
5. **Sprint Review and retrospective meeting** – At the end of each iteration there is a review of the completed sprint and it's results (completed user stories). If the project isn't complete (all user stories are completed), the model returns to the sprint planning meeting step (2.)

1.1. Model Entities

The model consists of the following entities:

1.1.1. User Story

User stories are requirements that the project must complete. Each user story is composed of tasks that must be completed

1.1.2. Task

Tasks are what the workers must complete to fulfil the requirements of the user story. Each task has a set amount of effort (in man-hours) that must be done by the worker in order to complete the task.

1.1.3. Worker

Workers complete tasks by spending effort on the task selected by them. The tasks that the workers work on are automatically chosen by the workers from the sprint backlog. Each worker also has a set amount of effort they produce per hour (which can be different from 1 man-hour)

1.1.4. Manager

Managers decide on which of the user stories get worked on and what decision should be done as random event happen. In-game this role is represented by the player.

1.2. Model Variables

The following variables are used to control the simulation model: (text in parenthesis is the variable name inside the model)

1. **Seed** (*seed*) – The number used as the base seed in the random number generator;
2. **Hours In Day** (*hoursInDay*) – The amount of hours in a workday;
3. **Days Per Sprint** (*daysPerSprint*) – Sprint length in days;
4. **Stand-up Meeting length** (*standUpMeetingLength*) – The length of the daily stand-up meeting (in minutes);
5. **User Stories** (*stories*) – The amount of user stories that are generated in the model;
6. **Tasks Per User Story** (*tasksPerStory*, *tasksPerStorySigma*) – The amount of tasks that are generated per user story. Has a mean value and variance (sigma) that is used as arguments for the random numbers;
7. **Effort Per Task** (*effortPerTask*, *effortSigma*) – The amount of effort (in man-hours) the tasks will take. Has a mean value and variance (sigma) that is used as arguments for the random numbers;
8. **Workers** (*workers*) – The amount of workers that the player's project is given;
9. **Worker's output** (*workEffort, workEffortSigma*) – The amount of effort per hour that a worker is able to output per task.

1.3. Random Events

During the game random events may happen that affect the simulation. These are saved with the scenario definitions. Depending on the event, the player may be given a choice between 2 or more options for what to do. The default scenario includes the following events:

1. Worker falls ill for N days, meaning the worker will not be productive for N days (N is randomly generated). Player has a choice of finding a replacement for half the output or doing nothing about it.

2. New user stories getting added due to new requirements.

3. User stories getting removed due to being no longer relevant. For completed user stories this can mean being replaced by a task that represents removing the feature's code. For user stories not yet done, it can be simply removing it.

2. Design Of Application

The game has been built as a single-page application [8], providing the user experience comparable to a desktop application while also being cross-platform.

2.1. Gameplay (User Story)

Also see appendix 2 for the use case representation of the user story.

The application is started by visiting the homepage of the game using a web browser. Once the application is loaded, the player has to choose a scenario from a list of pre-made scenarios. These define the default values for the variables used in the simulation model.

After the player has chosen a scenario they are shown the game view where the player is given information on the state of the game, including lists of workers and the user stories that must be completed during the project.

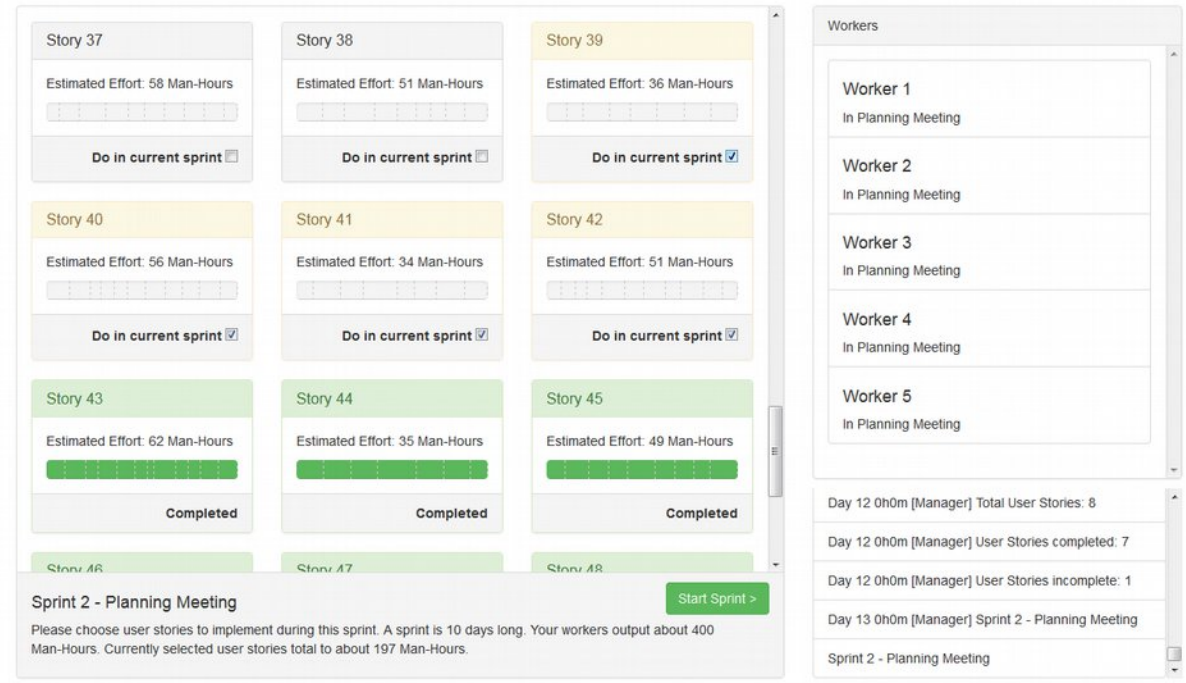

Figure 2: Screen shot of the game view

During each sprint, the player chooses user stories that the workers will work on. To help the player choose, information on the estimated effort the user story takes and a rough estimation on the amount of work the workers can do are given.

As the sprint goes on, random events may happen in the simulation to randomise the output of the model and to make it less predictable. Depending on the random event, the player may be given a choice between choices.

At the end of the sprint the player is given an overview of the sprint results. If there are incomplete user stories, the player can start next sprint.

2.2.Technology

The game has been implemented in JavaScript, allowing running it natively on most modern web browsers. To help speed up development the following tools and libraries are used:

2.2.1 SIM.JS

SIM.JS is a Discrete Event Simulation library which helps modelling discrete time event systems [10]. It also includes a seeded random number generator library which allows random numbers have same values across different simulation runs as long as the situation (model variables, player choices) are the same. As the library is in beta and not updated since 2012,

2.2.2. Lo-Dash

Lo-Dash [11] is a utility library which provides a large amount of functional programming helper functions, while ensuring cross-browser compatibility. The choice of using this is mostly a personal style decision by the author.

2.2.3. JQuery

jQuery is a Javascript library for HTML document manipulation, event handling and animation [12]. It is used on the front-end side of the application for manipulating the user interface. While jQuery also includes some functional programming helpers, these aren't used on the model side to ensure better quality for alternative ways of running the model (for example any future changes that would run the model on the server side.)

2.2.4. Mustache.js

Mustache.js [13] is a logic-less templating engine based on the mustache templating system, which is supported by many languages [14]. It is used to generate views on the game side,

avoiding the complex process of creating DOM trees in javascript side.

2.2.5. Bootstrap

Bootstrap is a front-end framework that provides a base CSS stylesheet for web procjets [15]. It also provides pre-made components that improve the look, that are used to style the project. Mainly because the author isn't confident at his design skills.

2.2.6. RequireJS

RequireJS is a JavaScript file and AMD module loader [16]. It is used to help structure the program in a modular matter, allowing for better organisation of the code in separate files. Additionally the text plugin is used to load files as strings (for example Mustache templates).

Using RequireJS also allows usage of it's optimisation tool, which compiles the code into one file that is automatically compressed [17].

2.2.7. Grunt

Grunt is a javascript task runner [18]. It is mainly used as the build tool for compiling files to be served in the production environment.

In the project Grunt has been configured with following tasks:

- *build* (default task) – Compiles a production ready version of the project into the build directory, compiling the source JavaScript & template files into one compressed and optimised file.
- *serve* – Used to launch a static webserver that can be used for development. Includes livereload option which automatically refreshes the page if source files have been modified by saving it.

Conclusions And Future Possible Work

The goal of this thesis was to create a single-player project simulation game based on an agile development model. While the objective was completed, the author considers it to be in the proof of concept phase and while it's ready for real use, there isn't much value in playing the game. This is mainly due to the project using the latest concepts in website development (Single-Page Application, modular architecture) and lack of materials on creating simulation models in Javascript. However it can be a good base for future projects, and the author would like to offer some suggestions for future improvements:

1. **Measuring the Success of the Simulated Projects**: Adding more variables that can be used to score the player on his/her choices. For example adding in values for user stories, different ways to measure the quality of the project (how many bugs are in the code, how many automated tests are there)

2. **Long-term Choices**: Currently the only ways the player influences the simulation is by choosing the user stories each sprint and how to act in case of random events. There would be much value in being able to modify the development cycle itself with longer-term choices that can be chosen by player at any time, for example by introducing extra concepts such as code review procedures.

3. **Better Feedback to the Player**: Adding more visual information on how the sprint went, adding charts about progress (for example burn-down charts)

4. **Recording and Comparing Player's Choices**: There's no one known best way to solve the problems. One way would be to crowd source the data by recording the player's choices and having multiplayer leader-boards for best managed projects. It would also give an additional feedback vector where players could compare how they are doing to other players.

References

[1] McKinsey & Company in collaboration with the University of Oxford, 2012. “Delivering large-scale IT projects on time, on budget, and on value" [Online] http://www.mckinsey.com/insights/business_technology/delivering_large-scale_it_projects_on_time_on_budget_and_on_value Last Visited: 13.05.2014

[2] Flight Simulator [Online] http://en.wikipedia.org/wiki/Flight_simulator Last Visited: 13.05.2014

[3] Discovery Channel Asia “Super Japan: On-time Metro” [Online] http://youtu.be/_rtxm30NULU Last Visited: 13.05.2014

[4] Marco Klemm (2000), “Design and Implementation of a Scenario for Simulation-based Learning in the Domain of Software-Engineering”

[5] Dan Goodin, Ars Technica “Pwn2Own carnage continues as exploits take down Adobe Reader, Flash” [Online] http://arstechnica.com/security/2013/03/pwn2own-carnage-continues-as-exploits-take-down-adobe-reader-flash/ Last Visited: 13.05.2014

[6] Kairosoft, “Game Dev Story” [Online] http://kairopark.jp/ Last Visited: 13.05.2014

[7] Greenheart Games, “Game Dev Tycoon” [Online] http://www.greenheartgames.com/ Last Visited: 13.05.2014

[8] Velocigames, GameBiz series [Online] http://veloci.dk/gamebiz/ Last Visited: 13.05.2014

[9] Single-page Application [Online] http://en.wikipedia.org/wiki/Single-page_application Last Visited: 13.05.2014

[10] SIM.JS [Online] http://simjs.com/ Last Visited: 13.05.2014

[11] Lo-Dash [Online] http://lodash.com/ Last Visited: 13.05.2014

[12] jQuery [Online]
http://jquery.com/ Last Visited: 13.05.2014

[13] Mustache.js [Online]
https://github.com/janl/mustache.js Last Visited: 13.05.2014

[14] Mustache template system [Online]
http://mustache.github.com/ Last Visited: 13.05.2014

[15] Bootstrap [Online]
http://getbootstrap.com/ Last Visisted: 13.05.2014

[16] RequireJS [Online]
http://requirejs.org/ Last Visited: 13.05.2014

[17] RequireJS Optimizer [Online]
http://requirejs.org/docs/optimization.html Last Visited: 13.05.2014

[18] Grunt [Online]
http://gruntjs.com/ Last Visited: 13.05.2014

Appendix

1. Project Code

The source code for the project can be accessed at https://github.com/t2t2/projectsim. The readme file includes instructions for installing and running the development server.

A running version of the project can be found at http://projectsim.t2t2.eu/.

Project Structure

This Section explains the project structure and the code files. It is mainly targeted to readers who are interested in working on the project in the future.

- */gruntfile.js* – Configuration for Grunt, the task runner/build tool.
- */public/index.html* – The HTML for the main game page.
- */public/runner.html* – Older version of the model runner that was used to test the model before the game was done. Isn't used anymore.
- */public/css* – Folder for the stylesheet files
  - */public/css/bootstrap.css* & */public/css/normalize.css* – Files from the bootstrap library.
  - */public/css/game.css* – Game specific stylesheet that covers styles that can't aren't part of bootstrap.
  - /public/css/simulator.css – Simulator runner. See note above about */public/runner.html*
- */public/js* – Folder for the javascript files
  - */public/js/game.js* – The starter file for game runner, includes configuration for requireJS and loads the game related modules.
  - */public/js/InputException.js* – A special exception that is used to indicate from model to the UI that the user should do some sort of an input.
  - */public/js/runner.js* – The starter file for simulation runner, see the notes about */public/runner.html* and */public/js/game.js* before.
  - */public/js/sim.js* – Modifications to the Sim.JS library to provide extra features that aren't present in the current version.

- */public/js/game* – Folder for the javascript files related to the user interface of the game.
  - */public/js/game/ui.js* – Code that handles the user interface for the game.
  - */public/js/game/scenario/default.js* – Contains the scenario definition for the default scenario. Scenario definitions include the default variable values and random events that may happen during the scenario.
- */public/js/simulation* – Folder for the javascript files related to the simulation model.
  - */public/js/simulation/simulation.js* – The simulation model, main responsibility is initialising the entities used in the model.
  - */public/js/simulation/entities* – Entities used in the model
    - */public/js/simulation/entities/eventer.js* – Responsible for generating random events.
    - */public/js/simulation/entities/manager.js* – Entity that represents the project manager in the simulation model.
    - */public/js/simulation/entities/worker.js* – Entity that represents workers in the model.
  - */public/js/simulation/objects* – Data structures that are used in the model
    - */public/js/simulation/objects/task.js* – Tasks that are to be completed by the workers.
    - */public/js/simulation/objects/userstory.js* – In-game representations of the user stories.
- */public/js/vendor* – folder for so called vendor code, as in libraries.

- */public/js/templates/game* – Templates used by the UI.
  - */public/js/templates/game/interface.html* – Template for the game UI
  - */public/js/templates/game/layout.html* – Template for the game page, mainly used for the scenario selection
  - */public/js/templates/game/random-event.html* – Template used for the modal popup that informs player of random events.
  - */public/js/templates/game/review.html* – Template used to render sprint reviews.
  - */public/js/templates/game/story.html* – Template used to render user stories in game.
  - */public/js/templates/game/story.html* – Template used to render workers in game.

- */public/js/templates/game/status* – Templates used to render sprint status messages

While the simulator runner isn't used any more, it's files are left in the repository as a possible base for implementing one in the future.

2. Use Case

1. The player enters the website. The player sees a page with a list of scenarios that can be played;
2. After the player has chosen a scenario, the player sees the game view. It shows:
    - A list of user stories (top-left),
    - Overview of the current sprint (bottom-left),
    - Log of recent events (bottom-right),
    - Overview of workers (middle-right).

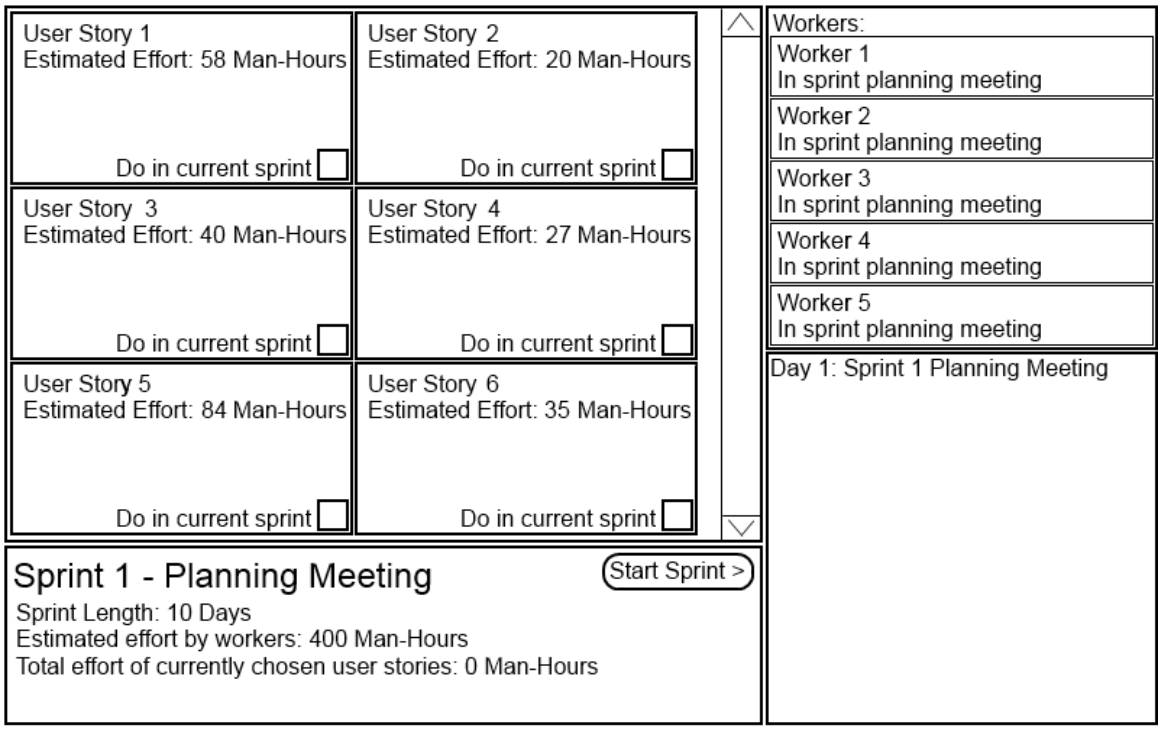

Figure 3: Game View Wireframe

3. In the sprint planning meeting player can choose which user stories will be included in the current sprint. To help player, hints about estimated worker output are given. After choosing the stories to work on, player can hit the "Start Sprint" button.
4. During the sprint, the player's screen is updated with the state during daily stand-up meetings (for example what a worker is/was working on as of the stand-up meeting).
5. Also during the sprints random events may happen that require player to make a decision between 1 or more choices. The decisions affect the simulation model either by manipulating the variables or simulated entities.
6. After the sprint has been simulated, the player is shown an overview of the sprint

results. If there are still incomplete user stories, player can enter next sprint, going back to step 3. Should there be no more user stories to complete, the player can go to the final screen (next step).

7. When the player reaches the end of the project, an overview of the game just played is shown.

3. License

Non-exclusive licence to reproduce thesis and make thesis public

I, Taavo-Taur Tammur (date of birth: 16.09.1992),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis “Web-Based Single-Player Project Simulation Game”, supervised by Dietmar Pfahl,

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 13.05.2014