Property Tycoon Documentation

Jonathan B. Morris, Alex Homer, Joseph K. M. Lee, Thomas Muffett, Joseph A. Corbett

January 2020

1 Introduction

Watson Games are a company that produces a range of traditional board games. We have been appointed as a software contractor to develop a computer game version of their most popular title Property Tycoon, a property trading game similar to the classic Monopoly. Watson Games have never commissioned such a system before and so do not have a formal requirements document to specify how the software should operate. They do however, have over 50 years of experience developing and marketing board games and have put together a set of user requirements to describe in their own terms what they want the system to do. This document contains all their user requirements. It is possible these requirements might evolve as the project progresses, but the core themes of the requirements will not change. You should, however, design with flexibility in mind.

The Group Leader(s): Ben Morris

Development strategy: Combination of waterfall and agile.

Group organisation: Controlled centralized

2 Sprint 1

2.1 Success Criteria

2.1.1 Planning Elements

2.1.1.1 Project Plan

Clear sense of individual tasks - In the PERT diagram we will use box's to describe individual tasks and time frames.
Order of Achievement - We will use lines to designate dependency in the PERT plan as well as horizontal lines to delegate specific tasks before specific large milestones.
Assigning members to tasks - We will use a box underneath the time for the allocation of the Pert Plan.

2.1.1.2 Proof of organisation

We will take recordings of each meeting, stating time and date as well as who is in attendance and where the meeting is.
Additionally, Project leader will keep a separate file documenting time and date, where and who is in attendance.
Meetings will have a review of Risk analysis and requirement analysis.
During the meetings there will be a review of progress made prior to the meeting
During the meetings there will be a goal setting exercise to decide what targets should be meet before next meeting.

2.1.2 Process documentation

For clear evidence of organisation into incremental development phases - we will use Github to document the developmental process.
Setting of targets with relative priorities - During design and later stages we will implement a shared priority queue document to be able to set what the most important tasks are at any given time. To be changed during meetings.
Analysis of risk factors - For this we are developing an analysis using the format provided in lecture 2 for software engineering. We will be approving and discussing improvements and changes in meetings.
Determination of requirements - For this we are developing a requirements analysis based on a lecture in software engineering by breaking down the requirements to determine what we will functionally need for the completion of the project. We will be reviewing this each meeting for changes to do and improvements.

2.1.3 Design documentation

2.1.3.1 High Level

We will write use cases We will use UML (unified modelling language) to create component based diagrams to model our software

2.1.3.2 Low Level

We will create class diagrams to show the variables and methods used. We will create sequence diagrams to show the flow of the program.

2.1.4 Software documentation

We will use Javadocs to document functions as well as other electronic documents for describing Completeness of documents covering key sections - for this we will be creating a document covering the implementation of our design. Describing key aspects of core components - For this we will use javadoc to create our description of the key aspects of the core components.

2.1.5 Code

General good coding practice and that the code produced reflects any design documentation - we will use variable names that make sense. We will use ALL CAPS for constants. We will use camelcase for variables. We will capitalize the first letter of functions and classes. We will use consistent indentation. We will follow the DRY or Don't Repeat Yourself principle. We will try to avoid deep nesting. We will comment code to make sure others, including those that aren't part of our team, can understand the intent of the code. Useful function names to summarise the intent of the code.

2.1.6 Testing Documentation

Anyone who tests the code will be required to keep a document containing their tests. Whether they be informal tests of function and functionality, like clicking a button to see if it works, or more formal JUnit tests. Testing can be done on other people's code as it is uploaded to GitHub.

2.1.7 Report

Clear summary of what works and what does not, and description of relative successes and failure of the project as a whole. - This will be a Latex document describing what of the requirements were met and not met.

2.2 Risk Assessment

2.2.1 Type

Risk Type	Possible risks
Technology	Deletion of Data (1) Software modules aren't as reusable as planned (2)
People	Team member leaving (3) Task may be out of team members field of experience (4)
Organizational	Management change (5) Team structure change (6)
Tools	Software/Network unavailability (7)
Requirements	Changes to requirements that require major design rework are proposed (8) Customers fail to realise the impact of requirements change (9)
Estimation	The size of the software is underestimated (10) Specification delays (11) Development delays (12)

2.2.2 Description

Risk	Affects	Description
Deletion of data	Project	A team member or error may cause data loss
Software modules aren't as reusable as planned	Project and Product	Software modules aren't as reusable as planned because of defects in them
Team member leaving	Project	A team member will leave the project before project completion
Task may be out of team members field of experience	Project and Product	A team member might fall behind task having to learn new information for their task
Management change	Project	There will be a change in who runs the project and the change will have different priorities/ideas
Team structure change	Project	There will be a change in the structure of the group as to how it is run.
Software/Network unavailability	Project	Delays caused by unavailability of software or network access
Changes to requirements that require major design rework are proposed	Project and Product	Larger number of requirements changes than anticipated suggested by designers in later sprints
Customers fail to realise the impact of requirements change	Project and Product	Larger number of requirements changes than anticipated by client
The size of the software is underestimated	Project and Product	The size of the system was underestimated
Specification delays	Project and Product	Specifications of essential interface are not available on time, or the specifications for the project overall are not completed on time
Development delays	Project and Product	Team members fall behind on assigned tasks for design or coding as the task is more complex than originally accounted for

2.2.3 Severity and Likelihood

Risk	Likelihood	Severity
Deletion of data (1)	Low	Moderate
Software modules aren't as reusable as planned (2)	Moderate	Low
Team member leaving (3)	Moderate	Severe
Task may be out of team members field of experience (4)	Moderate	Moderate
Management change (5)	Low	Low
Team structure change (6)	Low	Low
Software/Network unavailability (7)	Low	Moderate
Changes to requirements that require major design rework are proposed (8)	Low	Severe
Customers fail to realise the impact of requirements change (9)	Moderate	Severe
The size of the software is underestimated (10)	Likely	Moderate
Specification delays (11)	Low	Moderate
Development delays (12)	Likely	Moderate

2.2.4 Consequences and Solutions

Risk	Consequences	Solutions
Deletion of data (1)	Project set back	Separate backup of data, use version control software
Software modules aren't as reusable as planned (2)	Extra time required to fix issues	Change the design, create new modules with redundancy or modify design on next sprint to make modules more reusable
Team member leaving (3)	Reduced manpower	Reduce the amount of features for the end project and modify times
Task may be out of team members field of experience (4)	May require extended time to complete task	Assign another team member to help, modify deadlines to allow more time
Management change (5)	Need to rework design	Make design modular enough to incorporate design changes
Team structure change (6)	May need to find new method to manage time	Not Applicable
Software/Network unavailability (7)	Project set back	Modify the time scale for stages interrupted by unavailability
Changes to requirements that require major design rework are proposed (8)	Could set back project significantly	Change the design for the next sprint and lean more heavily into the Extreme Programming style of design
Customers fail to realise the impact of requirements change (9)	Could set project back significantly.	Could be solved by leaning more heavily into Extreme programming, may require restart of project if changes are large enough
The size of the software is underestimated (10)	May not be able to finish project	May have to change design strategy or miss out features or do extra work beyond expected load
Specification delays (11)	Will cause delays to other aspects of the program	May require more people to work on specification or remove features
Development delays (12)	Will cause delays to further sprints and releases of the program	May require more people to work on specification or remove features

2.2.5 Indicators

Risk	Indicators
Deletion of data (1)	Missing data in the github folder, comments in discord of panic
Software modules aren't as reusable as planned (2)	Attempting to reuse a module in a seperate area of the program is unsuccessful
Team member leaving (3)	Team members discussing leaving/quitting the team or university
Task may be out of team members field of experience (4)	Questions about how to perform a task that they have been assigned
Management change (5)	Talk of overthrowing the current leader
Team structure change (6)	Project leader talking of leaving, talk of using a different management style
Software/Network unavailability (7)	Sudden unavailability of team members or in-person complaints about software
Changes to requirements that require major design rework are proposed (8)	The sprint for the project resulting in a sub-optimal design that is unfeasible to continue working on
Customers fail to realise the impact of requirements change (9)	An E-mail, or other method of communication, from the client with massive requirements changes or changes to fundamental aspects of the project
The size of the software is underestimated (10)	Failing to meet deadlines
Specification delays (11)	Failing to meet project planning deadlines
Development delays (12)	Failing to meet design or coding deadlines