Software Engineering coursework specification

Notes:

1) Introduction

Watson Games are a company that produces a range of traditional board games. You have been appointed as a software contractor to develop a computer game version of their popular title "World Conquest", a game similar to the classic "Risk". 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 60 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.

Working as a team, you will need to:

- Develop an appropriate software development process model using elements of Agile. This will involve several phases of design, development and testing. This may involve, as you see fit, determining sets of functional requirements, non-functional requirements and domain requirements to determine more precisely what the software system should do. This may involve going back to the client to clarify exactly what they want, or presenting the client with a range of options to help them develop their own ideas.
- Develop a plan for the design, implementation, testing and delivery of the software system.
- Develop an appropriate software design, expressed using an appropriate combination of high and low level design techniques including, although not limited to, class diagrams, entity relationship diagrams, use case diagrams, sequence diagrams, state transition diagrams, activity diagrams, other UML or UML style/derived diagrams.
- Write the software, using whatever tools, development environments and languages you deem appropriate. Any reasonable set of choices will be accepted, including, although not limited to, Java, JavaScript, C, C++, Scala, PHP, Python and Unity. You may opt for a system using a database backend for persistent data management if you wish. Whatever you think is needed and that you are prepared to justify as part of the design process. The software will include a Graphical User Interface (GUI) component to help you monitor progress of the software simulation. You should bear in mind that the choice of tools and languages is something you should justify, and it must be a choice that is inclusive for all members of your team that will be working on the development of your codebase.
- Develop a test plan for the software, both at a unit testing level and at a system level to demonstrate that your software meets all the user requirements set for it.
- Write a report to document the work that you have done.
- Give a demonstration of your completed software system.
- Demonstrate that you can work as a coherent team, and show that you understand the significance of software engineering principles in the development of a non-trivial piece of software.

Assessment criteria for this team coursework project will be posted on Canvas. No two teams will produce systems that work exactly the same, so there is no gold standard for the operation of the software that you will be assessed against. The assessment criteria will focus on the quality of your

software engineering skills, and your ability to function as a team, to plan and execute work in an appropriate manner and deliver a working and hopefully reliable software system.

You will work in a team of 5 people. You will have an initial window period to form your own teams. If you have not formed a team by the end of that window period, your will be assigned to a team. In that event, there will be no negotiation on who is assigned to your team. Get to know the other students in your team. The assessors reserve the right to amend team membership (but this would be in exceptional cases only), and to merge teams if their membership becomes sub-critical, for example if students drop out from the course for personal reasons.

As this specification represents a set of user requirements, it has been expressed in user-centric terms. It may be incomplete or contradictory in places. Technical terms may be used inconsistently. If there is something you do not understand, you can either ask the customer's representative (course tutors/teaching assistants) or make reasonable assumptions about what is meant. Note that in the former case, the customer may or may not have a view on the query, and in the latter case you should be prepared to justify the choices and decisions you have made.

We are well aware that there are online versions of Risk available, and some open source project versions as well. Do not be tempted to copy the work of others and pass it off as your own. It would not help you using others code as you would not have any of the documentation needed for the project as a whole. Plagiarism is a very serious matter of academic misconduct. Any concerns of plagiarism will be investigated and teams may be required to explain the code that they accumulated.

2) The client: Watson Games

Formed in 1963, Watson Games Ltd are the UK's leading producer of quality board games. In 1986, they expanded upon the acquisition of Wadleys after they ran into financial difficulties arising from the widely reported corporate espionage scandal "Kerplunkgate". Watson Games design, produce and market high end board games designed to appeal to a wide range of ages.

The shareholders for Watson Games feel that the company needs to expand its operations to embrace the digital era and so have decided to develop a computer version of their classic Risk game. Risk was originally designed for up to 2-6 players and has a reputation as a real social experience. However, Watson also realise that it is often not possible in an ever complex world to find players to play the classic game. The head of game development, Quentin Raffles, has hit upon the idea that the computer version (the "simulation") can provide one of the players offering an opportunity for a richer gaming experience for a smaller number of players, or just one player. Your program will therefore require at least one autonomous computer player agent.

3) The rules of Risk

Quentin Raffles has provided the rules as they should apply to the new electronic version of World Conquest. These have been helpfully written up into a document "World Conquest Rules" . A copy of this document can be found on Canvas.

Versions of the game

The game can be played in two main versions:

- **The full game:** In the full version, the game is played until one player remains.
- **The abridged game ("See Capital Risk"):** In the abridged version, the objective is to capture all opposing Headquarters whilst still controlling your own.

- territory.

There are other variations available – see the document on Canvas for more details.

4) Other user requirements

As Watson Games have no experience with software developments, all they can do is to provide some general statements in respect of what they hope to achieve:

- The electronic version should be for desktop machines, and ideally should be playable on both Mac and PCs. If this is difficult, then PC development should be preferred.
- There are no plans for a mobile version at this stage.
- The game should be fun to play and have a colourful and intuitive interface that reflects the spirit and character of the original board game.

5) Core elements of the software system

You will need to design and implement a number of key elements to make this simulation work. As well as all the necessary classes, code, interface GUI and other infrastructure elements, you will need to create:

- **A game player agent:** An agent that can take the role of 1 (or more) of the players. This would allow for a limited number of human players to enjoy a richer gaming experience. The game player agent should play a respectable game of Risk.

You will not be assessed against how well the autonomous game agent performs, but there is the potential for a competitive element in terms of the software design team that comes up with the best performing game player agent. In the first instance the game player agent could just be based on random decision making.

You will also need to ensure that your simulation has:

- **A means of uploading initial data:** The game has a lot of card and other physical assets that need to be managed, such as cards and playing pieces. This kind of resource data will be loaded on start-up from external files, this means that the game is easily customised and Watson Games see this as a valuable selling point of the new electronic version.
- **A means of monitoring the performance of the simulation:** including the current status of each of the players and game assets that they own. This should be available for all to see as it is the current board game version.
- **A means of being tested:** to ensure that the market is operating properly in accordance with the rules of the game, and to demonstrate that the software is working correctly

It falls to your team to determine the form, structure and means of implementation of any user interface that your system will need to meet the requirements. It should be noted, however, that a flashy GUI does not compensate for a poor simulation.

6) The game player agent

The game player agent should be able to play the game to the same extent that a human player would. The key point here is that the game player agent needs to be able to play the game, but it does not necessarily need to be any good at it, at least in the first instance.

A game player could incorporate some simple rule for making in game decisions, or could feature relatively sophisticated Artificial Intelligence. But a game player could just operate on purely random

decision making. That would still allow the game to be played. It is suggested that your make the game player agent perform random decision making initially, and then if time permits, look to increase the sophistication of the agent. The marking scheme places emphasis on the delivery of a working agent, and only a small amount of its relative game playing sophistication. If enough teams deliver an agent with some more developed decision making, then a competition will be organised to pit player agents against each. But this will depend on there being enough teams with sufficiently sophisticated game player agents.

7) Integrity of the game.

Player may not borrow or lend game assets of any kind to one another.

As stated in the rules, certain assets owned by players are a matter of public record and that information must be available to all players at all times.

Any dice used in the game must be fair with each dice have an equal probability of landing on one of its six sides.

8) Deliverables

You are responsible for the delivery of this project as a group. Submission will be via the Esubmissions system on Canvas. You will need to produce:

- **A project plan:** showing the tasks that you identified for your team, how much effort you planned for each task, when it was scheduled to start and complete, and which team member(s) were responsible for it. This will likely take the form of a PERT style analysis or a Gantt chart. Bear in mind that the execution of the plan may well up end quite different from the initial plan. This is quite normal as sometimes things turn out differently to what you expect. If the variations are small, you don't need to keep the plan up to date,. If the execution is widely different from the plan, you should consider updating the plan and presenting both the initial and revised plans in your submission, together with commentary to explain why the plan fell apart.
- **A process document**: This project will be developed over a series of incremental developments ("sprints" in the Agile terminology). Your process document should record what you did for each incremental phase. For each phase this will include, but is not limited to, the target objectives, user requirements or user stories you set for yourself, any functional, non-functional or domain requirements that you identified that needed to be addressed, any other reasonable assumptions that you made as part of any requirements analysis process, any key design and implementation decisions that you took, and a review of how successful you were. You can blend elements of the Waterfall and Agile development process models as you see fit.
- **A design document:** documenting both high level and low level designs of your software systems. Such a document should include, although is not limited to, class diagrams, sequence diagrams, activity diagrams, state transition diagrams, use case analyses and other UML style diagrams. The design does not need to be exhaustive. You only need to do as much design as your team deems necessary to deliver each of the incremental development phases.
- **The code base:** including all non-code files or other data necessary to allow your submission to be compiled and executed by the assessors. The code should be properly documented (e.g. using Javadocs if you are working in Java), and be commented to an appropriate

standard. As well as the completed codebase, you should retain a snapshot of your codebase at the end of each incremental development phase as set out in your process document.

- **A testing schedule**: showing how you tested the software at both unit test and system test levels. It is acceptable for bugs and other issues to remain in the software as long as they are clearly identified, documented and that you have provided appropriate commentary to explain why they exist and what you would do about them.
- **A group report**: documenting how the project went for you, with a critical analysis of the performance of your team, and describing what worked well for your team and what did not. The report must also contain records of any group meetings that you had, together with a summary of actions agreed and a weekly log recording your progress.
- **A peer assessment/review:** A document where the team agrees the distribution of a pool of marks to reflect the relative contributions of each team member to the group project as a whole. Your individual mark for this Software Engineering module will be a composite of your team score and an individual score determined by your peer assessment. If any individual does not make a contribution to the group effort, the assessors reserve the right to award an individual score of 0 for that team member for the module as a whole.
- **A video:** Providing evidence of the final working state of your program. The purpose of this video is to assist the markers in evaluating the working state of your final delivered program, and to enable you to showcase any particular features that you are particularly pleased with.

The process documents should contain documentation for each of your Agile sprints. The process document is not intended to be a heavyweight account of the sprint, but it must provide evidence of the work that you have done. There is a template document on Canvas. The sprit cycle process document for each individual sprint should record:

- **The sprint objectives:** the target objectives, user requirements or user stories you set for yourself, any functional, non-functional or domain requirements that you identified that needed to be addressed, any other reasonable assumptions that you made as part of any requirements analysis process.
- **Design:** documenting both high level and low-level designs of your software systems. Such a document should include, although is not limited to, class diagrams, sequence diagrams, activity diagrams, state transition diagrams, use case analyses and other UML style diagrams. The design does not need to be exhaustive. You only need to do as much design as your team deems necessary to deliver each of the incremental development phases.
- **Testing**: showing how you tested the software at both unit test and system test levels. It is acceptable for bugs and other issues to remain in the software as long as they are clearly identified, documented and that you have provided appropriate commentary to explain why they exist and what you would do about them.
- **Post sprint review:** A review of how successful the sprint was in achieving the objectives you set out for it.

Your team will also be required to attend progress review meeting(s) with one of the assessors/seminar teaching assistants to ensure that you are keeping on track. You can request additional meetings if you need them and the assessors will endeavour to provide support as needed. But your first port of call for any problems should be the scheduled seminar sessions.

When you upload your deliverables, please use Word or PDF formats for text documents. Do not use Mac Pages files as these are not widely supported. For code deliverables, please remember to

include any project (e.g. Eclipse, BlueJ), initialisation or other external files your code needs to be run. If assessors cannot run your code, they will likely ask for demonstrations of the operation of key parts of your code. Do not upload files that are in obscure formats. Assemble all of your documents together and create a single ZIP file with everything inside. Upload the zip file onto the Esubmissions system. Check that your ZIP file unpacks correctly before uploading it.

You will be also be required to give a short demonstration of your project at the end of the semester. This will be an informal demonstration to your lab seminar leaders or the module convenor.

9) Peer assessment

Each team has 20 points to distribute per member of the group. Each member gets a peer mark, which is a non-negative integer (including 0) such that the sum of all members' peer marks is 20 x the number in the group. The idea is that the peer mark reflects the contributions of each member to the project. It is perfectly acceptable to give each member the same mark.

Here is an example. Assume the group has members: Alice, Bob, Charles, Diana, Eric, Fiona, George and Herbert.

- Alice: 25
- Bob: 0
- Charles: 30
- Diana: 20
- Eric: 15
- Fiona: 15
- George: 35
- Herbert: 5

In this case, George contributed most to the group, followed by Charles and Alice, and so on. Bob is a "ghost" - who never contributed to the group work at all. I take that as a signal from the rest of the group to that effect, and he will not be awarded the grade for the project the others receive. I think that's fair. Note that a ghost's marks count as some measure of compensation for the others for the increased workload they are under.

When submissions are graded, each member of the team receive a base mark (except for ghosts) which reflects the quality of the design process, how well teamwork has gone on, and so on (actual code, while it inevitably has an influence, is the least important element from the point of view of this module). The team score will then be combined with the peer mark. A standard peer mark of 20 will count for around 10 percentage points, or about one grade.

Only one submission per team of your agreed marks is required, but it will need to be clear that it is agreed between you (excluding ghosts). If there is really no agreement, I will make the final decision. To do so, the team will be called to a follow up meeting to explain the reason(s) for their disagreement.

10) Final observations

There is no single right way of designing and delivering this Software Engineering assignment. You will be assessed according to your ability to come up with a coherent set of requirements, project plan, software design, implementation, testing and production of a final report. A set of marking criteria will be published on Canvas.

If you do not understand something, use the Software Engineering seminars to ask questions and raise discussions on the assignment. You can also discuss ideas with other teams, but do not copy the work of other teams. If you require clarification on any points arising from this document, any such clarification will be copies to all students and teams via your Sussex email address. You should also keep an eye on your Sussex emails in case the Watson Games project manager decides to issue an updates, amendments or clarifications on their user requirements. The customer is always right, even when they are changing their mind. As a consequence, you should try to anticipate any requirements shift that might arise and try to design your software so that changes can be easily accommodated. This is not always an easy thing to achieve in practice.

You will need to ensure that you team meets often, documents decisions that it makes and that the workload is spread fairly between team members. The most common cause of problem with team based assignments such as this is a breakdown in team dynamics. A part of the assessment process will require teams to submit a peer assessment to evaluate the relative contributions of each team member to the overall delivery. If a team member does not contribute to the team effort as a whole, I reserve the right to award that team member a score of 0. Do not leave problems unresolved. If you are having team issues, you should try to resolve them yourselves first, and take the issue up with one of the teaching assistants in the seminars, or through a private meeting between one of the teaching assistants or me as you see appropriate.

So, work together, and have some fun working on this team assignment.

And keep regular backups of your work ……. “I lost the USB stick” just doesn’t cut any ice.

Dr Kingsley Sage
School of Informatics and Engineering
Khs20@sussex.ac.uk
January 2024

END.