FINAL REPORT

Customer Value and Scope

The chosen scope of the application under development including the priority of features and for whom you are creating value

(A) We have created a rudimentary city simulation game with a focus on sustainability, relating to UN Goal #11, “Sustainable cities and communities”. The game has basic functionality regarding placing and removing buildings, as well as resource management. The player’s score is measured through their impact on the environment.

We prioritized learning how to use Unity’s toolkit to best suit our game design, which created value for us, the developers. We prioritized basic functional gameplay and aesthetics over meaningful gameplay to create value for the customer. This provided us a useful foundation but no realistic depiction of our vision.

(B) Going forward we should re-evaluate our scope more often, refining the ideas and features we decide to keep and possibly discarding ideas that no longer align with our goal. We should also commit ourselves to create clear and unambiguous documentation regarding the changes in scope we decide upon. Since we had no product owner or official target group we had a hard time evaluating customer value since we pretty much regarded ourselves as the customer.

(A->B) To achieve our goal of re-evaluating scope regularly, we can utilize a meeting agenda, to make sure we go through the process of each weekly meeting. During each meeting, we can then document any changes to the scope we have decided on, in dedicated documents. Included in this document should be a clear ranking of our priorities. Additionally, a formal target group should be defined.

The success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort)

(A) We have an aesthetically pleasing game that has basic resource-turnover functionality and could be built on to achieve our initial vision. Our goal was to create a game that puts focus on sustainability and the challenges of air pollution and a decaying environment. We want the player to be faced with choices of short-term gain versus long-term sustainability. Our goal is to make a simple but hopefully challenging game that simulates the difficulty of such choices.

It seems our common goal was to familiarize ourselves with game development and get a glimpse of how it could work on a medium scale.

(B) As for what we wanted to achieve with the project, it was very ambitious from the start. We had an undocumented understanding that none of us were very unfamiliar with this kind of and thus accepted the risk of not reaching our full goal. In the future, the scope should perhaps be questioned whether or not it's reasonable for the allotted time given to finish a project. More time diverted to structuring in the initial stage of the project might have allowed for a smoother workflow.

(A->B) Achieving a balanced and challenging game the fastest way possible would require studying existing well-developed City-sim games with entities that consume and produce resources in a seemingly balanced manner. We wouldn't have to reinvent the wheel.

Playing our game repeatedly and trying to exploit the mechanics and tweaking them accordingly would also be very useful. Research (opinions) about what players prefer would be informative and help where to put the focus of resources available. A well thought out and more extensive tutorial could be added to accommodate new players.

Your user stories in terms of using a standard pattern, acceptance criteria, task breakdown, and effort estimation and how this influenced the way you worked and created value.

(A) Our user stories have evolved during our sprints going from some basic user stories that have no description, no clear title that show the goal of the user stories to clearly defined and easily understood tasks. During the first few sprints tasks were more focused on what we needed to do generally to finish the user story but then they were improved to a more structured standard pattern with well-defined acceptance criteria, task breakdown, and effort estimation. A definition of done was later created to separate things from being "in progress" to "done".

(B) Although our stories have been more structured towards the end, they could still have more clearly defined acceptance criteria. The ideal user story could be adopted, fully understood, and completed by anybody without any questions to the reporter. The problem is how much time should be spent crafting the user story and when it becomes overkill. This is a line we're not really sure when we are crossing due to our lack of experience.

(A->B) A dedicated per-sprint story-master could go over all created stories before the sprint begins and determine whether a story is clearly defined and properly sized. We could make a general acceptance criteria validation protocol specified in a document.

Your acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders

(A) Acceptance tests were generally performed by the person who was assigned to complete the task. This worked fine since most of the time the acceptance criteria had been discussed with the team during meetings.

(B) Ideally both the acceptance criteria and the acceptance test should be written and performed by a person that is not directly involved with the task which would add an extra layer of validation.

(A->B) In the future, we should assign two people to handle each user story. The first one is the one who is going to work on it and the second one is the one who is gonna take responsibility for writing and checking the acceptance test.

The three KPIs you use for monitoring your progress and how you use them to improve your process

(A) For this project we used three different KPIs; a stress level sheet, a burndown chart, and a velocity graph. We found that the usefulness of the KPIs differed a bit. The burndown chart did not really represent the actual progression since it only tracked whole user stories and not individual sub-tasks. The stress level sheet was useful because it gave us an indication of when the team felt overworked so that we could adjust the amount of work we put on a sprint. The velocity chart gave us a good overview but wasn't really helpful as a tool to improve our project planning.

(B) We wanted to have 3 KPIs that would reflect on the current status at each sprint. The KPIs would be used for measuring progress and keeping track of our mental state. The goal was to be able to use them to see if the amount of work we put on each sprint were enough or if the user stories were completed too quickly. They would be used as a tool to discuss and evaluate our progress after each sprint.

(A->B) In future projects, we think we need to pay more attention to our KPI especially if the project is big so we can get a good understanding of the progress and try to solve the problems that we get before it gets even bigger.

Social Contract and Effort

Your social contract (Links to an external site.), i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project (this means, of course, you should create one in the first week and continuously update it when the need arrives)

(A) **Link to social contract**. The social contract helped us with having regularly scheduled meetings. The technical content included "comment your code" and what platforms to use when managing code. The social content included "maintain a good tone when facing conflict". This seemingly worked out since no conflicts broke out.

(B) The social contract doesn't include what should happen when one of these rules are broken. Who should enforce them and how should they be enforced?

(A->B) Everyone followed the social contract and no conflicts occurred. The next time we would need to have a more concrete plan of action to deal with troublemakers and ruffians.

The time you have spent on the course and how it relates to what you delivered (so keep track of your hours so you can describe the current situation)

(A) We tried to estimate the time each task would take to finish when we planned the sprint. After each task is completed we evaluate how much time was spent. Given our situation and experience with game development, we felt that the time that was given was reflected by our deliveries.

(B) If we would have done the project again we would probably have prioritized things differently. More time should have been dedicated to the planning phase because we felt like we did not have enough time to talk everything through. We planned every sprint in 2-2.5 hours but we probably would have needed about 4 hours to get everything organized and sorted out.

(A->B) Optimize usage of time and remove redundant repetitive tasks by automation or a simplified less time-consuming method.

Design decisions and product structure

How your design decisions (e.g., choice of APIs, architecture patterns, behaviour) support customer value

(A) We chose Unity as a game engine that has a lot of prebuilt libraries so we didn't have to reinvent the wheel. This allowed us to make a decent game in a short amount of time. We split up the work with the intention to help each other out if anyone got stuck, and let the individual have a lot of responsibility for the elements they created for the game.

(B) Looking back at the experience we should have created a more solid base for our project whilst also making sure that everyone was on the same page on how to design and further develop the game. The split between themes could have been better balanced as some people had a lot more to do some weeks. This however was our own fault as the project was very ambitious from the get-go. The innocent choice of using dictionaries to store data later made it so we failed to implement a save game feature due to the difficulty of the task (due to dictionaries containing dictionaries, unable to convert to saveable data). Choosing a way to structure the project to facilitate future implementations from the start would be a better way to approach.

(A->B) It would yield much to sit down together and let everyone be a part of how to shape the game, time estimates, how to design a grid with belonging applications to reduce technical debt, and allow for a modular game for future expansion with more implementable features. This would give everyone without previous knowledge equal footing and a good start in understanding what we are to create and also remove a lot of the (sometimes) parallel work that was done on some tasks.

Which technical documentation you use and why (e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents)

(A) For now we only use Jira which helps us with planning and executing our sprint and we use google docs to write our notice, brainstorming, and definition of done. We didn't use any other diagrams in this project because we didn't see the need and we didn't have the time for it.

(B) Having Class diagrams and Domain models could be beneficial, since it would help other people to understand how our application works, and for developers to understand its structure.

(A->B) Now that we know more about how unity works a good plan could be, to begin with, a Domain model to explain how the program will work, and then we can go and plan our code with a Class diagram. After that, we can begin making user stories, epics, and tasks.

How you use and update your documentation throughout the sprints

Our documentation is used as a reference in our planning and in our work. We update this documentation after discussion.

How you ensure code quality and enforce coding standards

We haven’t yet considered enforcing coding standards, but it is present in our social contract. The code quality will be reviewed before it is pushed to master by the person who checks that the user story is completed.

Application of Scrum

The roles you have used within the team and their impact on your work.

(A) The only role we assigned was a SCRUM master since we did not feel that we would benefit from additional roles in our project.

(B) As mentioned, weekly rotating roles would be useful. Code reviewers can ensure good quality and structure of code while also making sure code is well documented. User story reviewers can ensure user stories and their tasks are well described and modeled.

(A->B) Each week, a rotation should be applied and roles will be assigned. Everyone should be assigned as frequently as sprints pass. There should be documents defining the process of validation including code standards and user story validation protocols.

The agile practices you have used and their impact on your work

(A) We have used JIRA as a tool for our application of a SCRUM board. The first sprint we wrote a list of things we wanted to complete. We discovered that this was not ideal since we ended up working on many different features at the same time without structure. We changed to use theme-based sprints to focus on a specific set of features within each sprint. This structure allowed us to work in parallel to complete each sprint.

(B) In future projects, we would use the knowledge we have gained about agile practices to have a better structure for sprints and user stories. It would be good to decide a structure for user stories from the beginning, how to define subtasks, definition when tasks and stories are completed, and make it clearer for everyone in the group.

(A->B) More time should be spent to decide how to structure epics and stories in the beginning, before the first sprint. This would make it easier for everyone to understand our stories and how to perform and complete them, including showing the product owner our progress in a simple way. Since we did not have any experience in agile practices when we started the project, we had some issues understanding the value of well-defined stories. After supervision, we learned to develop them into something that was very clear. Next time we can use this knowledge to make a definition in our social contract from the start.

The sprint review and how it relates to your scope and customer value (Did you have a PO , if yes, who?, if no, how did you carry out the review? Did the review result in a re-prioritisation of user stories? How did the reviews relate to your DoD? Did the feedback change your way of working?)

(A) At the end of each sprint, we sat down and discussed potential left-overs. If there were any, we would discuss it with the assignees. If there is still work to be done, those issues were carried over to the next sprint. Following this, we had a meeting with our supervisor to review our sprint structure. We took the feedback and discussed it to refine our next sprint.

(B) Having a project owner would give us additional feedback on our project that could be cultivated in our discussions.

(A->B) Getting a project owner would be the next move to do if we want to improve from where we are right now and getting one or more people to join the sprint review would be a good idea so that we get more opinions about our sprints.

Best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)

(A) We chose to use JIRA in our first meeting, as a member of the group had used it during the summer and had some experience with it. We went through the basic functionality of it together and then members were left to explore on their own. After each supervised meeting, we obtained a deeper understanding of the tool and added features such as time estimates, logged time, and other functions. The usage of git has been mixed, some have used the built-in GUI version, and some push through the terminal. Problems occurred quite a lot when pushing due to merge conflicts or Unity wanting to do something quirky but we solved it together as a team.

(B) In the future a deeper understanding of the platforms used would be very useful as we didn’t utilize the full functionality of it and misunderstood some parameters and how they

worked. It would also be beneficial to know when others were working on the project and pushed certain builds or changed classes because it would have reduced the number of collisions we had while constantly updating the project. After the supervised meetings, it would've been good if we took detailed notes of what had been said since it is easy to forget.

(A->B) At the start of a project, a meeting to get a feel of everyone's proficiency level with the tools would've been good as we were very vague with our proficiency levels. A thorough walkthrough or instructions to watch videos should be conducted. Reading up on documentation would suffice to make sure everyone gets a basic level of understanding of how everything works. Making sure there is unambiguous documentation that is available to everyone and that it's updated is an easy way to make sure what has been said is not forgotten and makes it easier to reflect and improve. Guidelines of git procedures could also be decided to ensure a smoother workflow.

Relation to literature and guest lectures (how do your reflections relate to what others have to say?)

(A) As course literature goes we really only used the lectures and the weekly supervision to help us guide the structure of our project. Most of our knowledge about Unity we got from video tutorials and documentation.

(B) We felt that we got enough information from our lectures to be able to work with the project in an agile way.

(A->B) We should probably have looked more into what a good quality product backlog item would have looked like by using the INVEST-criteria as a base template when we created user stories and tasks.