Change Report

Assessment 2

Group 6:

Igor Smollinski <is942@york.ac.uk>
Pranshu Dhungana <pd861@york.ac.uk>
Zack Tyler-Kyle <ztk500@york.ac.uk>
Phoebe Russell <pbr508@york.ac.uk>
Sanjna Srinivasan <ss3264@york.ac.uk>
Sam Savery <sgs527@york.ac.uk>
Ewan Hutcheson <@eh1776york.ac.uk>

Change Report A

In light of phase 2 of the assessment, we have adopted a brownfield development plan for making changes to our project. Brownfield development is a strategy used to build upon already existing software projects, in our case, the inherited piazza panic game from team 3. It involves understanding the existing architecture and discovering potential changes and adaptations to make the product better, developing said changes into the software and testing whether the changes fit the existing software. We used this as a basis to conduct our changes to our project.

After receiving team 3's project we had a meeting as a team to try to understand all the existing code, documentation and deliverables to see what we are going to work with before making any changes or development and marking areas of concern. We also obtained the team's assessment 1 feedback to get an idea on what went wrong with their project, and how we can change certain areas where they went wrong. We then split up into groups based on what sections each member was comfortable with and then worked on stated change in each section.

To keep track of all the changes made, we already have an existing google drive, to put in any rough drafts of the changes we have documented. We are also using github and gitkraken as our version control to monitor developments made to the code. We also ensure that whenever a team member commits code changes to their branch, they include a short description of what exactly they have changed, hence making it easier to document about it in the change report. The code commits are reviewed by our team member Igor.

Before writing about any changes to the deliverables, we ensured to discuss it with the rest of the team. When making changes we tried to incorporate the existing architecture for Phase 2 before deleting or starting sections from scratch to save time and unnecessary changes.Our group also had regular discord meetings to discuss the progress made in the change report, and to document the changes made to the code by the members who are working on implementation.

While working on updating the deliverables with the changes, we took insight and some pointers from the members who were previously working on each of the sections during phase 1 of the assessment. After committing a change we made sure that the change we made still fitted within our current projects objectives and currently working systems and continued to update these changes if we needed to.

Testing is an integral part of this game development to ensure that the code works as intended to. After choosing the team's project, we set up some base structure for tests to ensure that we haven't inherited any dysfunctional code, and then went from there, making changes and building upon the existing code wherever deemed necessary according to the new set of user requirements. We also set up continuous integration testing in the github workflow to run automated tests on the code we inherited.

Change Report B

Requirements Changes

Original: https://neves6.github.io/PiazzaPanic2/team3/Req1.pdf
Updated:https://neves6.github.io/PiazzaPanic2/deliverables/Req2.pdf

Introduction

One change to add upon this section is how we elicited the new set of user requirements for phase 2. We will also be adding references to how we conducted the requirements table and backing up our technique with the relevant literature, as team 3's introduction lacks proper research into literature.

User Requirements:

- UR_ENDLESS_MODE to be added as a part of phase 2 of the assessment as described in the product brief since not all Phase 2 requirements were included by Team 3. Functional and Nonfunctional requirements of the same need to be added.
- UR_POWER_UPS, UR_SAVE_STATE, UR_SUPPORT_DIFFICULITY new user requirements demanded by the client, to be added to phase 2 of the assessment. We need to add functional and nonfunctional requirements for the above three user requirements.
- UR_LOCKED_RESOURCES needs to be added as a part of phase 2 user requirements mentioned in the product brief.

Functional Requirements:

- UR_COOK, UR_NUMBER_OF_CUSTOMERS needs to be updated to FR_COOK, FR_NUMBER_OF_CUSTOMERS as they are functional requirements and not user requirements as they describe how the system works rather than what the user needs..Both descriptions need to be updated to include Phase 2 requirements due to Team 3 not including them in the original tables.
- FR_JACKET_POTATO and FR_PIZZA need to be added, as they are to be implemented as new recipes required for phase 2.
- FR_ACTION: Combine FR_CUTTING and FR_FRYING into one functional requirement, as both of them, even though they are different actions, ultimately serve the same purpose as a user requirement by using the cooking stations.
- FR_PREPARE_STAGE needs to be deleted as it seems irrelevant. Other requirements already explain what FR_PREPARE_STAGE does.
- NFR_PLATFORM_AVALABILITY needs to be changed to a functional requirement FR_PLATFORM_AVALABILITY as it describes which operating systems the game and we only need to provide for two operating systems as stated by UR_PLATFORM

Non-Functional Requirements:

- UR_FRAMERATE should be a non-functional requirement and be renamed to NFR_FRAMERATE as the description of UR_FRAMEATE is a fit criteria for the UR_UX requirement and doesn't stand alone as a description for a requirement.
- NFR_AVALABILITY: Playable state is too vague and needs to be redefined to chances of crashing and the rates of crashes needs to be a lot lower as not crashing 95% of the time is too high.
- NFR_RESPONSIVENES the fit criteria must be researched and changed as the timing of responsiveness for player input is inaccurate.
- NFR_TIMER_PRECISION has to be reworded so the fit criteria make sense. Error of margin for the timer should be less than 1%.
- NFR_OPERABILITY needs to be reworded to say once a player has been through the tutorial they should be able to play the game without any help as that is the whole point of the tutorial to make sure that the player knows how to play the game.

Method and Tool changes:

Original: https://neves6.github.io/PiazzaPanic2/team3/Plan1.pdf
Updated: https://neves6.github.io/PiazzaPanic2/deliverables/Plan2.pdf

One change made is that instead of having just 3 gantt charts which, we are having bi-weekly gantt charts representing each of our sprints, as part of the SCRUM strategy we have decided to follow into Phase 2 of the assessment. This is because only having 3 gantt charts did not accurately show or monitor the progress of the assessment throughout and does not allow us to make changes to deadlines or work flow easily.

Furthermore we are also resorting to the use of Kanban Boards for phase 2 of assessment so that every team member knows what task they are assigned to, and we can easily view the progress of all the tasks which are assigned. And it also shows the priority of the tasks assigned and the size of said task, to allow us to estimate how long a task is going to take.

Addition of the use of GitKraken for version control to monitor and track changes to the github repository. This is because our team used it in our previous project and where we are comfortable with it.

During Phase 2 we are making each of our roles more clearer and everyone has been split into role groups to make sure that one person is not working on a section on their own. We are now holding regular meetings, and also implemented 2 week sprints as a part of scrum. This way, we focus on different stages of design, Architecture, Documentation and implementation, on each of the sprints to check up upon progress, monitor changes, and be clear upon everyone's roles to avoid confusions. We also have a designated person that makes the final decisions about certain tasks and conflicts. This is because none of this was mentioned in detail in the original report which can cause problems with the flow of the project.

Another change we have is that we are going to be using IntelliJ as our only IDE instead of multiple IDEs as team 3 used IntelliJ and Visual Studio Code. This is due to the fact that using multiple IDEs can potentially cause confusions, and also some of the features available in one IDE might not be the same as the other one. It can cause version conflicts while commiting to github.

We will also update the plan and plan evaluation section of what we planned to do for the major section of the project and what actually happened during the major section of the project during the evaluation of the plan. We have dedicated ourselves to do this plan in bi-weekly sprints with a planning , review and retrospective section as a part of the SCRUM strategy.

Risk Changes:

Original: https://neves6.github.io/PiazzaPanic2/team3/Risk1.pdf
Updated: https://neves6.github.io/PiazzaPanic2/deliverables/Risk2.pdf

We have added the following new risks as a part of phase 2 of our assessment due to the new risks that come with inheriting a groups project and continuing the development of it:

- R17 - Not having a well documented code from team 3. This can make the classes and methods difficult to interpret. It can also lead to difficulties in making unit tests due to unclear code comments.
- R18 - Inheriting another team's code can cause confusion regarding the overall structure of the code and make it difficult to adapt the code. It can also prove to be challenging to update methods/classes, and also be difficult to implement new requirements.
- R19 - Inheriting another team's deliverable can cause issues with the planning of the project due to Requirements, Architecture, Risks and Methods not being structured the proper way. This leads to more changes needed to be made to the deliverables making changes to deliverables from new requirements from the client harder to incorporate into the new design.
- R20 - Shifting from the github (GHC) repository of team 3 to gitkraken. We have to make sure it is compatible and all the branches are integrated correctly into our repository.
- R21 - Since we are now using gitkraken, gitkraken server going down can be added as a new risk.
- R11 - Risk 11 is unnecessary and needs to be removed from the table. This is because the discontinuation of a library doesn't actually affect the projects itself. The previous features are still usable as you still have the previous version of the

library and only new features from the library that you might want from the library won't be available which can be solved by finding a new library with said features.
- R22 - We need to add another risk about the usability and user experience of the game as this was missed during the original table. This will include how easy the controls are and how they feel to players and the actual game play and whether its pleasant user experience i.e. whether the game was hard to play or if users experienced bugs etc.
- Team 3 didn't do any risk monitoring during the time of their project which can cause issues with the project if risks are not monitored correctly. We are going to indicate when a risk has happened and take appropriate procedures to try and minimise the effects of the risk. If no risks have happened during the project we should say that we haven't encountered any risks and that we project has run smoothly.
- We will need to assign names for the risks to our group specially so we know who is reasonable for each risk as the previous risks from team 3 carry forward to our project and we need to follow the appropriate procedure to correct issues.
- R5 - Need to make the description of this risk more clear as it's too vague and only mentions a specific case such as each potential merge needs to be peer-reviewed by the team before its commit to prevent bad updates to the code.
- R23 - We need to add a risk for if we encounter memory leaks and how we would try to resolve them during development. This is because we inherited team3 code. There was a major memory leak we needed to fix before developing the code further and this was a major risk to development.
- R10 - This risk needs to be redefined as its description is too vague. It should highlight how and when requirements change during the development of the project and how we handle that in the requirements tables.

Architecture

Original: https://neves6.github.io/PiazzaPanic2/team3/Arch1.pdf
Updated: https://neves6.github.io/PiazzaPanic2/deliverables/Arch2.pdf

- We split the UML diagrams into different parts so they are readable, understandable and easier to visualise including Power, Food, Stations, Screens, GameScreen, Entity , Utility and the Full Diagram.Along with some other interesting ways of visualising the UML relationships by exploring CRC collaborations.
- We started by creating new CRC cards for changes in responsibilities and new classes. We had to generate new CRC cards for the new set of requirements which emerged for phase 2. We updated the CRC cards for the existing classes due to some of the new features depending upon these classes. The updated CRC cards include stationManager, GameScreen and OvenStation, PrepStation and ServingStation. The new class CRC cards include Power, PowerUnit and OrderCard.

- We ended up using https://echeung.me/crcmaker/ for generating CRC, since the previous website Lucid charts ended up requiring additional payment to create a lot more cards. The reasons are fully explained within the architecture.
- We also added state diagrams for some of the new features because they were the best way to showcase the features properly. For other features we just added sequence diagrams. The state diagrams we have added were for the new features for Saving and Loading, different game modes and difficulty levels. We added sequence diagrams showing how the Power ups and Hiring chef and unlocking stations worked.
- These state diagrams were made in Lucid charts before it cost additional money.While the the Sequence diagrams were made in plantuml due to its simplicity and the final uml relationship diagram showing the implementations in the end were done with Intellij and formatting with plantuml.

-