Change Report

Group 17

Team Loading

Joel Warren
Charlie Wilson
Jake Keast
Hari Bhandari
Ishrit Thakur
Joshua Hills

2. A) Introduction to Change Report

To plan, make, keep track of, and review changes to the following Assessment 1 inherited work, we first researched change management and software maintenance, to help best prepare us. Hence, we then followed the subsequent process of ownership change:

- Familiarise with the project, deliverables and code. “Knowledge transfer”.
- Understanding stakeholders -> Customers / customer requirements
- Understanding the team -> Organisation
- Knowing what is required -> Customer requirements / product brief.
- Planning / Making / Keeping track of and Reviewing changes, as described below.

<u>Change management for assessment 1 deliverables</u>

- **Plan** - Analysed existing deliverables, identifying any missing assessment 1 requirements and then comparing with the assessment 2 brief to plan what needs to be added.
- **Make** - Make a copy of the original deliverables in our shared drive and make changes/additions there.
- **Keep Track Of** - Using an ‘updates log’ to list out any major changes, as well as Google Docs’ inbuilt ‘Version History’.
- **Review** - Compare changes to original documents to ensure changes were sensible and assessment 2 was properly covered with any additions.
- **Conventions** - Updated indexing and titles to differentiate between assessment 1 and assessment 2.
- **Tools** - Google docs version history

<u>Change management for inherited code (+ website)</u>

- **Plan** - Initial code review to determine what needs changing and how additional code will be implemented.
- **Make** - Updates to code made using Git commits, with commit messages and code comments with each change to ensure keeping track of and reviewing changes is made easier.
- **Keep Track Of** - Commit messages and comments to keep track of what has been changed with each commit.
- **Review** - Review commit messages, code comments and requirements to ensure all changes are sensible and code additions adequately cover assessment 2.
- **Conventions** -
    - Comments to highlight new/changes to code
    - README file
- **Tools** -
    - Change request tracking system - GitHub Issue
    - Git commit history

Change report 2. B)

Requirements

URL of original: https://eng1-loading.github.io/assessment_2-website/requirements.html
URL of updated: https://eng1-loading.github.io/assessment_2-website/pdf/ReqUpdated.pdf

Updates to the essay

The essay on requirements elicitation was updated to reflect the process our team used to inherit and successfully update the 3 requirements tables to take into account assessment 2. The original requirements elicitation process was kept the same because it describes the previous team's (team 21) methodology. Here we explicitly explained which parts of their methodology we used and which we didn't, with corresponding explanations. These updates went underneath 'Inheriting requirements'.

User Requirements added

- UR_ENDLESS_MODE - Added to reflect additional requirements in assessment 2 product brief.
- UR_MOVEMENT_WASD - After testing the game, we felt like having the option to use WASD controls would also be suitable, instead of the current restricted point and click controls.
- UR_PURCHASE_COOK - Added to reflect additional requirements in assessment 2 product brief.
- UR_POWERUP - Added to reflect unexpected requirements given by our customer.
- UR_DIFFICULTY_SELECT - Added to reflect unexpected requirements given by our customer.
- UR_SAVE_GAME - Added to reflect unexpected requirements given by our customer.
- UR_RECIPES - Added to reflect additional requirements in assessment 2 product brief.

Functional Requirements added

- FR_BURGER_BURN - Added to reflect additional requirements in assessment 2 product brief.
- FR_CUSTOMER_GROUPS - Added to reflect additional requirements in assessment 2 product brief.
- FR_CUSTOMER_LEAVE - Added to reflect additional requirements in assessment 2 product brief.
- FR_REPUTATION - Added to reflect additional requirements in assessment 2 product brief.
- FR_MULTIPLE_CUSTOMERS - Added to reflect additional requirements in assessment 2 product brief.
- FR_DIFFICULTY - Added to reflect unexpected requirements given by our customer.

- FR_SAVE_STATES - Added to reflect unexpected requirements given by our customer.
- FR_POWERUP - Added to reflect unexpected requirements given by our customer.

Updates to Non-functional Requirements

Non-Functional Requirements added:

- NFR_NO_VIOLENCE - Previously missing requirement from requirements table, where no violence is necessary because the game must be family friendly.
- NFR_JOYFUL - Likewise, joyful and colourful themes are important for family friendliness.

Non-Functional Requirements removed:

- NFR_SYSTEM_COSTS - Requirement not requested thus not needed.

Lastly, we updated the ‘indexing’ row for each table, because we felt it was inconsistent and confusing. The index format was different between user and functional requirements. Hence, indexing was updated to distinguish between assessment 1 and assessment 2 requirements. With a prefix 1 or 2 for assessments 1 and 2 respectively (any additional requirements given by our customer went under assessment 2). We did not add indexing to the non-functional requirements table because it did not feel necessary as all non functional requirements were applicable to both assessments.

Some pre-existing requirements were related to assessment 2, hence their indexing was updated:

- UR_REPUTATION
- UR_EARNINGS

Architecture

URL of original: https://eng1-loading.github.io/assessment_2-website/architecture.html
URL of updated: https://eng1-loading.github.io/assessment_2-website/pdf/ArchUpdated.pdf

Updates to state diagrams

- Fig 1 - The “waiting for next user input” was removed because this could be made implicit that cooks are always waiting for the next user input. The action for clicking station without relevant ingredients was removed because this was also made implicit and not needed architecturally. “Cook x selected” introduced to show change in state for cook.
- Fig 2 - “waiting for user input” removed, “cook x selected” introduced both as described for Fig 1.
- Fig 3 - “waiting for user input” and “waiting for action to be completed” removed because they are implicit, and offer little architecturally.
- Fig 4 - “cook x selected” introduced as described for Fig 1.
- Fig 5 - “settings screen” introduced for UR_DIFFICULTY_SELECT. “Mode selection screen” introduced to represent UR_ENDLESS_MODE and UR_DIFFICULTY_SELECT architecturally.
- Fig 6 - “waiting for user input” removed as described in Fig 1. Assumes the cook is already selected.

New state diagrams

- Fig 7 - Architecture introduced to represent UR_POWERUP, FR_POWERUP
- Fig 8 - Architecture introduced to represent UR_PURCHASE_COOK

Removed state diagrams

- (old) Fig 6 - This diagram was not needed because it could be avoided by making this implicit when a cook is selected or not. Nothing architecturally was being offered, hence the architecture was removed.

Updates to class diagrams (fig 10)

For most architectures, the class diagrams have been simplified to not directly include actual java classes but more high-level architectural concepts. Practically this has meant that not every attribute/method of the implementation class has been included but only the most important ones architecturally.

- Screens - Added SettingsScreen to reflect new settings screen
- Food - Nothing to add since the previous team had already added pizza + jacket potato.
- Ingredient - Added “burned” boolean to represent if ingredients are burned, implementing FR_BURGER_BURN.

- Customer - Simplified by by removing move/generateOrder/Orderoptions to remove unnecessary java jargon
- Cook - Simplified by making concepts more high level instead of java concepts
- Order - Made comprehensible by removing both get methods as they were unnecessary
- PiazzaPanic -Removed high level java methods because they are not needed architecturally
- Screen - No changes because it was already simplified
- MenuScreen - made it simpler by removing methods as they were not needed architecturally
- CreditsScreen - Removed high level java method because they are not needed architecturally
- EndGameScreen - Removed high level java methods because they are not needed architecturally
- GameScreen - Made more intelligible by getting rid of the many unnecessary java methods
- SettingsScreen - Added settings screen to represent the requirements of changing difficulty etc
- Recipe - Made few changes to make it more simpler architecturally
- Pizza - Added recipe and cash value, representing UR_EARNINGS
- Burger - Added recipe and cash value, representing UR_EARNINGS
- JacketPotato - Added recipe and cash value, representing UR_EARNINGS
- Salad - Added recipe and cash value, representing UR_EARNINGS

Updates to the essay

Since assuming responsibility of the project, architecture had to be updated to reflect improvements as well as new requirements. Hence, the design process was expanded to include how we extended architecture. The design evolution was also extended to reflect how the design has further evolved since our team assumed responsibility.

We also removed a lot of unnecessary detail in the justification. This is because based on the feedback for the previous teams architecture, they included a lot of unnecessary java classes and based the design evolution on how implementation changed over time. Instead, this was changed to reflect how the design of the system and its concepts evolved over time, such as from new requirements.

Updates to requirements traceability

The following updates to requirements traceability was to reflect the additional requirements introduced from assessment 2 being reflected by the following architecture:

Updates to class Gamescreen
- UR_COOK_STACK
- UR_MOVEMENT_WASD
- UR_PURCHASE_COOK
- UR_POWERUP
- UR_RECIPES

- FR_REPUTATION
- FR_BURGER_BURN
- FR_CUSTOMER_GROUPS
- FR_CUSTOMER_LEAVE

Updates to class Cook
- UR_MOVEMENT_WASD

Updates to class Ingredient
- UR_RECIPES

Updates to class Customer
- UR_CUSTOMER_VIEW
- FR_CUSTOMER_GROUPS
- FR_CUSTOMER_LEAVE

Updates to class EndGameScreen
- UR_MAX_SERVE

Further added traceability for the newly added settings screen.

Method selection and planning

URL of original: https://eng1-loading.github.io/assessment_2-website/planning.html
URL of updated: https://eng1-loading.github.io/assessment_2-website/pdf/PlanUpdated.pdf

For each of the following sections of method selection and planning, the previous team's approach was kept the same, with our team's (team 17) approach being added underneath each section respectively.

This was mainly because different sections to the previous team's approach were still relevant to our team's approach, such as their descriptions and justifications of different software engineering methods which were also used by our team. Keeping the previous teams essay also helped with clear comparisons between the two if needed.

Software engineering methods + development/collaboration tools

The essay was updated to reflect the process our team went through in inheriting the previous teams software engineering methods and different development/collaboration tools, as well as to present our own methods/tools etc.

The essay also included details of how we had to learn new tools, such as Lucidchart, needed because we had never used this tool before.

Team organisation

The previous project breakdown was for assessment 1, and thus not largely relevant. Hence, a new "Assessment 2 Project" breakdown was created to reflect different deliverables etc.

Deliverables ownership was updated to reflect different team ownership. Difference in team organisation was also added in the essay to reflect difference in team dynamics arising from different team members but also difference in assessments.

Systematic plan

A new gantt chart was created to reflect the difference in assessments and team members, which required a new systematic plan. While the plan is different, we created it similarly to the previous team using plantUML using a similar format too, laying out key tasks, start/finish dates, task priorities and dependencies.

Similar to assessment 1, weekly snapshots of the (updated) gantt chart were added to the website [https://eng1-loading.github.io/assessment_2-website/snapshots2.html]. The final snapshot of the gantt chart was also added to the essay.

The essay then updated to include a discussion on how the plan evolved throughout the duration of the project. This was necessary because the old discussion was no longer relevant, given the systematic plan has since changed and evolved differently. The updates to the essay reflected these changes.

Risk assessment and mitigation

URL of original: https://eng1-loading.github.io/assessment_2-website/risk.html
URL of updated: https://eng1-loading.github.io/assessment_2-website/pdf/RiskUpdated.pdf

Updates to the essay

The essay on the risk management process was updated to reflect the process our team used to inherit and successfully adapt the risk register for assessment 2, as well as to explain the risk management process that our team used.

The original risk management process was kept the same because it describes the previous team's methodology. Since we used a very similar process, we simply referred to it under the updated essay.

We updated the essay under 'Inheriting the risk register' to describe the process we took of systematically updating the risk register.

Updates to risk register

We updated the indexing of the risk register to distinguish between old risks (still applicable to us), i.e. R1.__, and new risks denoted by R2.__. This was to aid with the change report.

Risks removed: (old indexing - refer to original risk assessment)

- R7 - Removed because it was in the same scope as R1.09
- R8 - Removed because it was too similar to R1.01

Removing these unnecessary risks helped to keep the risk register focused and easier to monitor.

Risks added:

New risks were added to the risk register to reflect risks associated with taking on another team's project, as well as any risks our team felt were missing from the previous team's risk register.

- R2.01 - Potential risk because the customer may unexpectedly give new requirements (which did happen).
- R2.02 - Necessary risk because our team was taking on new code
- R2.03 - Potential risk because of new assessment requirements
- R2.04 - Necessary because our team was taking on unfamiliar deliverables
- R2.05 - Necessary risk because refactoring is vital to implementation
- R2.06 - Missing risk from assessment 1 about gameplay, necessary because game functionality not working is always a possibility with potentially devastating consequences.

- R2.07 - Missing risk from assessment 1 about gameplay, necessary because an enjoyable experience is a ‘must’ customer requirement.

Inherited risks updated:

Risks were updated, such as changing the likelihood and severity to reflect brownfield development vs greenfield, as well as a change in team with different strengths and weaknesses.

- R1.04 - Description generalised to be more appropriate. Likelihood increased from L to M due to our team taking longer than expected to complete certain tasks like architecture.
- R1.18 - Likelihood decreased from M to L due to our team already being familiar with strengths so being able to work on tasks that match this.
- R1.19 - Severity increased from L to M to reflect significant delays from low work output
- R1.02 - Severity increased from M to H to reflect seriousness from missing customer requirements

The rest of the R1 risks were unchanged because we felt that their likelihood, severity and mitigation strategy were still suitable for our team and assessment 2 of the project.

Ownership for each of the risks was also changed to reflect the change in team ownership and thus different team members.