Software Engineering Group Project
End of Project Report

Author:	Ben Dudley, David Fairbrother, Jonathan Englund, Josh Doyle, Liam Fitzgerald, Maurice Corriette, Oliver Earl, Tim Anderson
Config Ref:	SE_05_DEL_06
Date:	2016-13-02
Version:	1.0
Status:	Release

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

CONTENTS

1. INTRODUCTION

1.1 Purpose of this Document

This document contains the teams learning outcomes, critical feedback and performance reviews. It additionally specifies improvements for future projects [1].

1.2 Scope

This report describes the final state of the project and outcomes following the completion of the project.

1.3 Objectives

The reader of this document should understand new mitigation techniques for the difficulties encountered during the completion of the project. It should additionally provide new processes for software developments in future projects.

2. MANAGEMENT SUMMARY

We set out to create a task assignment system with three components, TaskerCLI, TaskerMAN and TaskerSRV. These components communicate to each other via TaskerSRV and allow for task assignment and synchronisation. The client provided a description of each component and their functional requirements [2] to ensure each component meets their expectations.
All members of the team were open and friendly from the beginning. This meant everyone quickly learnt individual strengths and weaknesses which allowed us to split into two separate teams; web and client development. Every member of the team provided strong contributions to the project and was happy to receive constructive criticism and incorporate changes leading to a strong feeling of comradery.

The biggest difficulty continuously encountered throughout this project was lack of experience or exposure to elements used within this project. This included building GUIs, using JDBC in Java, displaying modal windows and handling a variable number of elements in the management component. Prototyping provided some initial experience and lead to subsequent design changes. During the projects implementation bugs arose due to the team not fully understanding the technologies; this led to subsequent refactoring of earlier code as more principles were understood.

Within TaskerCLI race conditions and threading lead to unexpected and difficult to reproduce bugs. This ranged from the database disconnecting randomly to all windows of the program becoming invisible. The client development team developed a stronger understanding of how to use the debugger to reproduce these race conditions and prevent them.
The team developing TaskerMAN was struggling to post data to modal windows. Initially they created several prototypes using AJAX however they later realised that through the use of PHP GET and POST methods they could easily push data to the page. This had the additional benefit of sanitising data when combined with prepared statements.

At the conclusion of this project out of the twelve functional requirements we completed and demonstrated all but two functional requirements. The two functional requirements not completed are FR7, filtering and sorting of tasks in TaskerMAN and FR9, task synchronisation in TaskerCLI.
FR7 was not implemented as the requirement was misread as being optional. FR9 is implemented however two bugs prevent the correct operation of saving elements and updating a task on reconnection.
All documentation is currently in a good state as they contain the latest information on the project and meet QA standards.

3. HISTORICAL ACCOUNT

Once the initial functional requirements had been provided for the Tasker system, the group set about deciding upon the platform that the Tasker system would be built on and tested against. They then produced a high level design which included user interface mock ups and UML diagrams to "map out" the overall functionality. This high level design was shown to the client to ensure that the group effort was being applied in the correct way. From that meeting with the client the group learnt that our understanding of Task Elements was incorrect; this delayed development until the group figured out the best way to implement the changes in the user interface design, and the design of TaskerCLI, TaskerMAN, and TaskerSRV.

Once initial designs had been agreed upon, the group was separated into two sub-groups to separately work on development of TaskerCLI and TaskerMAN. This was done to allow the group member's skills to be applied more effectively in their strongest area.

3.1 TaskerCLI

The first task for the team handling TaskerCLI was to begin work on the Design Specification whilst the Test Specification was being developed.

The Design Specification contains some contents from the high level design document, which had already been completed. However, due to changes stemming from the production of the high level design documentation, these diagrams could not be copied over without modification. We decided that the team working on TaskerCLI would not contribute heavily to the Test Specification and only work on the Design Specification for TaskerCLI. This was done to ensure that all changes could be made before the Design Specification deadline arrived.

This change to how Task Elements were handled, as pairs of descriptions and comments as opposed to a single element, required changes to the user interface design as well as a lot of the logic of loading and storing data. Interfaces and Class Diagrams also had be developed in accordance with this new way of handling Task Elements.

After the Design Specification had been submitted, work began on building the prototype of TaskerCLI for demonstration to the client. The major difficulty in developing this prototype was that TaskerSRV was not functional during development, meaning the desired functionality (downloading Task data) could not be implemented in time for the prototype demonstration. Fortunately, however, TaskerCLI had the ability to work in an "offline mode" so the basic functionality could be displayed.

Development of the final software was a generally straightforward task, however there were a series of bugs found during testing that extended the development time. The most difficult of the errors to find and fix were race conditions that arose when working with TaskerSRV. These errors were arduous and took many hours to fix, meaning that some team members were required to work extra hours in order to complete the development of the software.

3.2 TaskerMAN

The first task for the PHP team was the design specification in joint progress with the TaskerCLI team whom were also working on the design specification. The creation of the design specification initially when on without much of an issue, however there were timetabling conflicts which resulted in TaskerMAN team meetings without all members present.

After the submission of the test specification, the feedback introduced a curveball which affected the design stage, as we were unaware of the complexity behind task elements. We believed that Task elements was nothing more than a long description of the task.

During the prototyping stage of the project the entire team ran into issues due to TaskerSRV not being functional which held back all other development. Thankfully TaskerMAN was somewhat functional as we had modal windows and login working correctly.

When it came to actually implementing the PHP the hardest issue to deal with was finding a creative solution for inserting a user defined number of task elements which was eventually tackled but not without its struggles.
Aside from this difficulty the implementation was relatively calm without any major hiccups or setbacks. Bugs were identified and patched as coding week progressed until a fully operational website was up. All functional requirements were met aside from the filtering of tasks. This was a functional requirement that was unfortunately completely forgotten during implementation

3.3 TaskerSRV

TaskerSRV was planned with the functional requirements in mind to contain three tables: Users, Tasks and Elements. Whilst the users and tasks table design did not change throughout the project a misunderstanding about foreign keys led to the elements table being redesigned.
Previously the elements table contained two foreign keys and a string to allow elements to map to a user and task. After discussing with the client the team realised we needed to store a comment and the current design would not work.
This lead to us redesigning the elements table to hold a primary index, comment and element description and a foreign key to the relevant task.
The next step was creating this schema from the design, for this task we used MYSQL workbench which allowed us to graphically prototype the database. After some minor changes such as the length of a column being too short we verified TaskerSRV's operation.
Using the queries generated from MYSQL we created a BASH script to generate a fresh database. This script can be run interactively or from a single command and will detect an existing installation, or partial installation and generate clean tables with the user's permission. The final step was taking test data produced and using the bash script to automatically insert this data when instructed to create test data by the user.

4. FINAL STATE OF THE PROJECT

The project is viewed as meeting all but two functional requirements at its completion.
TaskerSRV correctly implements all functional requirements; this includes storing the following information within its schema:

- Full name and Email Address for Users
- Task Title, Start Date, End Date, Status and Allocation
- Task Element with free text comments

TaskerMAN meets all but one functional requirements; these are:

- Adding, Updating and Deleting team members
- Adding, Updating, Viewing and Deleting task data
- Reallocation of tasks
- Identification of managers

This includes input validation and additional checking such as enforcing a member cannot have any tasks before being deleted. Filtering and sorting by column specified in FR7 has not been implemented so FR7 is not completely met.

TaskerCLI implements all but two functional requirements:

- Connecting to database and storage of connection details
- Allowing users to login and local storage of this list
- Local copies of tasks periodically updated from TaskerSRV (every minute currently)
- Intuitive GUI (IR1)

Due to a bug in saving task elements an updated comment will not synchronise back to the database. This same code path also is responsible for uploading changes made offline as soon as a connection is available thus offline changes are discarded. Despite this changing a Task's status still synchronises correctly whilst online.
Secondly whilst offline the database will automatically attempt to reconnect on a timer, this leads to multiple dialog boxes being displayed every time a connection is attempted if the application was previously online. This can make the program difficult to use due to dialog boxes appearing on every change or periodically.

5. LOCATION OF GROUP REPOSITORY

The group repository is located at:

https://github.com/antibones/CS221_Group_05

The final branch for delivery is the "master" branch.

6. PERFORMANCE OF EACH MEMBER

6.1 Ben Dudley

Ben worked mainly with the TaskerCLI and TaskerSRV team throughout the group project.
After an unsure start to the project, Ben was able to show his strengths within the group and proved to be an important member of the team. Initially he had limited experience with many of the aspects that were involved. However, he managed to learn many tasks with assistance from the group including Junit Testing, Git commands, UML mark-up using Visual Paradigm and integration testing which was completely new to him.
Ben was keen to gain experience in new areas; an example of this is when he prototyped a script to automatically create a TaskerSRV instance. The team had no previous experience in in this area, his experience gained whilst implementing the prototype helped create the final scripts for setting up the database.
Ben showed hard work and dedication throughout and was always keen to help other members of the group wherever possible.

6.2 David Fairbrother

David took the role of project leader and a java programmer in this project.
To effectively lead he organised the project into assignable tasks and created deadlines to ensure delivery times were met. If problems or ambiguity occurred he communicated with the client or team members to resolve those issues. He additionally created two development teams, this way team leaders can get a stronger idea of their member's strengths and weaknesses and decide the best member to assign a task to.
David also participated as a Java programmer. During this project he researched and implemented a Database class. This utilised JDBC to connect to TaskerSRV to perform task synchronisation. He then additionally researched debugging threaded applications and taught other Java programmers more advanced debugging.
David has been a strong team member both managing the project whilst providing support and valuable contributions to other individuals within the team.

6.3 Jonathan Englund

At the start of the project Jonathan was assigned to be a part of the Java team. Despite him having more experience in web development he wanted to gain experience in Java by completing spike work as well as debugging code that the others had created. He additionally created several diagrams and mock ups used within multiple design deliverables. During development Jonathan used his new experience to create several methods which were essential for example email validation on the login screen.
After the first day of development he helped the Web team as they required more people to complete additional work for the final deadline. Whilst working with the web team he developed the taskAdd function, to integrate with TaskerSRV he developed a function for converting input from the user into SQL queries. Although Jonathan had less experience than some others in the team he did provided strong valuable contributions and was willing to accept feedback.
Jonathan has learnt many things during this project and looks forwards to his next project where he will be able to contribute more as a result of these new skills.

6.4 Joshua Doyle

Before the development of the software started, Josh worked very closely on the development of the design of TaskerCLI, as he was one of the most experienced members in using Java. Josh designed the user interface for TaskerCLI and worked with David Fairbrother to design the Class structure of the TaskerCLI program.

When it came to development Josh was the head of the team working on TaskerCLI. In taking on this role, Josh was in charge of distributing tasks to everyone working on TaskerCLI. Josh also worked on writing code directly due to his long term experience of programming. Josh used Window Builder in Eclipse to create the GUI windows for the entirety of TaskerCLI, and wrote the code that saved and loaded Task and Member data. He performed extensive debugging and resolved many complex bugs discovered during the testing phase.

Josh's code was well-written however the code required to load Task Elements was quite complicated and was not commented sufficiently. This lead to him subsequently refactoring the code resolving several bugs discovered during testing.

6.5 Liam Fitzgerald

Throughout the project Liam attended all meetings and provided valuable contributions within them despite being initially shy. One of his first tasks he completed was UI spike work for TaskerMAN, the initial ideas from this spike work still form the foundations for our final website design.
Liam also worked with the TaskerMAN team and produced several documents for the design deliverable such as sequence diagrams for TaskerMAN. He later produced the basic HTML and laid the CSS foundations.
Unfortunately due to personal problems Liam was not able to work from implementation week onwards. After contacting the project leader to inform him of the situation early in the week the team reorganised the tasks allowing the project to still be completed. This was only possible due to him informing as early he did.

6.6 Maurice Corriette

From the beginning of the group project Maurice was determined to work hard in order to support the group and achieve the best grades possible. From the outset he has been committed, attending all group meetings and formal meetings whilst providing strong input in all.

From the start of the project Maurice identified his experience in testing and thus was selected as the group testing lead. As the testing lead Maurice took on the sole responsibility of the testing specification and testing report, with input from Oliver and David in QA sessions. All versions of the testing specification have been detailed and extensive enabling delivery of a high quality product which conforms to documentation. Furthermore due to Maurice's meticulousness in creating the test specification based on the requirements specification he noticed the design deviated from the requirements. This allowed him to alert the group of this deviation and his input was valuable in checking future design documentation conformed to the client's requirements.
During Testing and coding week Maurice worked with the developers. He tested the system extensively, opening issues on GitHub, walking the team through reproduction of those bugs, and ensuring the system adhered to the functional requirements.

6.7 Oliver Earl

Oliver undertook the roles of lead QA as well as the lead web developer in this group project. As the most experienced PHP programmer in the group, the development and bulk of the implementation of the TaskerMAN component fell under his jurisdiction. This includes the underlying logic that generates database queries and the presentational layer that the user navigates. He also worked closely with members in the group responsible for testing to ensure that the HTML5 and JavaScript validation was fit for purpose.

Oliver also ensured other members of the Web Team were fully involved wherever possible with tasks such such as presentational work to ensure everyone had work to complete.

Oliver carried out his QA duties in meetings, working alongside the Deputy QA and Team Leader to ensure documentation and all deliverables were in accordance with the functional requirements. Their thorough, systematic approach resulted in the achievement of consistent, positive feedback.

Punctual, hard-working and sometimes a bit of a perfectionist, Oliver ensured that he attended as many necessary meetings and group work sessions as possible to ensure work was done at a consistently high quality.

6.8 Tim Anderson

Tim took the role of deputy leader during this project. Tim fulfilled the duties in this role at the start of the project when the team leader wasn't present. He picked up the partially completed tasks and communicated the projects status to the client during the meeting ensuring work could continue.
Tim proved to be hardworking and a good member of the team who was able to communicate well with all. As a part of the web team he showed a good understanding of CSS, being responsible for the complete overhaul of the old design and crafting the new one individually. He helped with debugging and developing new HTML for TaskerMAN. This ensured the layout and presentation of the finished product matched the design requirements.
Furthermore he completed additional research work independently and used that knowledge to aid the advancement of the project. For example he researched PHPUnit however we did not manage to implement them.

7. CRITICAL EVALUATION

The team performed very well during this project even during setbacks faced throughout the project. All members clearly knew their areas of expertise and weakness allowing the team to best utilise individual skills. An area for improvement is during quieter weeks additional training could have been performed ensuring skill sets are more equally spread across team members. During this project all members were present throughout however if somebody with a unique essential skillset went ill it would have caused delays.

The project could have been improved in two major ways. Firstly the final product did not have any unit tests built for it. Initially we were planning on using test driven development; however our lack of experience with the technology meant whilst tests would pass the program would not function correctly.
Following this the team agreed additional prototyping would have paid dividends. Our initial prototypes proved concepts such as using window builders to create GUIs but did not complete additional steps such as refreshing the displayed data. This lead to large deviations from the original plan which assumed certain functionalities that could have been avoided with more rigorous prototyping.
From this extra prototyping new unit tests should be built, a common issue during implementation was regressions which went largely undetected from the lack of unit tests. Whilst unit tests would have taken additional time to produce, the time saved would have paid dividends.

The team developed an understanding of the need for a well-developed project plan. By completing research and identifying risks in new technology additional testing and training can take place. This prevents large design changes, unnecessary refactoring and bug introduction. In larger projects with multiple teams these changes would significantly impact on other teams who may depend on the old incorrect design.
Finally within more complex methods it is worth breaking them into sequence diagrams. Whilst the implementation may seem trivial and everyone will still approach that algorithm differently. This leads to people assuming the algorithms behaves in a certain manner as it is not documented and leads to subsequent bugs stemming from those assumptions.

REFERENCES

[1] N. W. Hardy, C. J. Price and B. P. Tiddeman, *SE.QA.11 1.9 - Producing a Final Report,* Aberystwyth University: Software Engineering Group Project, 2016.

[2] N. W. Hardy, *Tasker Team Tasking System - Requirement Specification 1.2,* Aberystwyth University: Software Engineering Group Project, 2015.

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	13/02/2016	Original Version	DAF5