CSC3600 Final Report

By
USQ Learning Emporium
Richard Dobson, Jesse Hare, James McKeown, Vincent Roberts, Ryan Sharp

Examiner
Dr. Xiaohui Tao
Senior Lecturer (Computing) – School of Sciences - PhD QUT

Project Supervisor
Assoc. Prof. Stijn Dekeyser
Associate Professor (Computing) - School of Sciences - PhD Antwerp

School of Management and Enterprise

Due: Friday 18th of October 2019

Contents

1. Executive Summary

This report provides an analysis and evaluation of the development stages completed for the harvest metadata.

2. Methodology

2.1 Methodology Statement

The team used the scrum implementation of the agile methodology to complete the project. The scrum process allowed the team to incrementally build the program over the entire project duration. As there was constant input from the the project supervisor and the client, the scrum process permitted the team to make quick changes.

2.2 Justifications

The following sub sections will describe why scrum was chosen, the benefits of scrum and the disadvantages of scrum.

2.2.1 Why Scrum Was Used

As stated in the previous section, scrum was ideal for this project as it allowed the team to make quick changes during the development process. Another reason why scrum was chosen was because there was no team leader. This provided an environment where all team members could be equal and address issues as a whole group.

2.2.2 Benefits of Scrum

Scrum provided a framework that allowed the team to build in sprints, which suited this type of project. After each sprint, a meeting with the project supervisor was organised so that feedback could be given in preparation for the next sprint. In addition to this, scrum forced the team to prioritise requirements which allowed for faster development. Without this, lower-priority requirements may have been implemented at the cost of the higher-priority requirements.

2.2.3 Disadvantages of Scrum

As time constraints were a major factor in the development of the project, it was difficult for the team to have as many daily meetings as scrum required. Scrum requires the implementation of daily meetings, which wasn't feasible with this project. Another disadvantage with scrum in the team's case, was that each member of the team wasn't experienced with the process and the roles involved. This meant that the roles outlined in the scrum process weren't followed 100% correctly.

2.3 Discussions

The following sub sections will discuss how the scrum process was utilised and how it will be used for future projects.

2.3.1 How Scrum Was Followed

All team members simulated the *Product Owner* and *Scrum Master* roles. Due to the inexperience with scrum, it was decided that this was the best option to follow. Daily meetings were reduced to at least 1 meeting a week, with 2 meetings a week the average throughout the duration of the project. Sprints were carried out every 1 to 3 weeks, depending on the size of the individual tasks. Once a sprint was complete a whole team meeting was organised to discuss the next sprint. Meetings were then held with the project supervisor to demonstrate what had been completed and to gain new requirements for the project.

2.3.2 How Scrum Will Be Used In The Future

The obvious improvement to the scrum process in future projects would be to have a dedicated product owner and scrum master. These two roles play a key part in the scrum process, and when included, make the development of the project a lot easier. In addition, having more time to have daily meetings would help tremendously in both planning and developing the project.

Due to this being the first time each member had used the scrum process, it can be presumed that it wasn't as efficient as it could have been. As with all projects of this nature, the more experience you have, the faster and more efficient you get. It is the goal of all members, that with each time they use the scrum process they get more proficient with using it.

3. Project Process

3.1 Team Organisation

There were a total of 5 members in the USQ Learning Emporium team. As this project was initially designed for teams of 3, the team needed to strategise ways in which to divide the work evenly between members. The first

solution was to divide the main project task into 2 sub teams; a back-end team (3 members) and a front-end team (2 members). Within these teams each member had different roles.

From these sub teams, each had their own Facebook Messenger group and their own Slack channel for discussing specific project tasks. This stopped unnecessary information from being communicated between the sub teams, i.e. the back-end team didn't need to know the specifics of the front-end and vice versa. Within these messenger platforms, jobs were distributed and agreed upon between members.

3.2 Team Structure and Roles

As stated above, the group was divided into 2 main groups; a back-end team (3 members) and a front-end team (2 members). The back-end team consisted of: James McKeown (Team Leader), Richard Dobson (Programmer) and Vincent Roberts (Programmer). The front-end team consisted of: Jesse Hare (Team Leader), Ryan Sharp (Programmer).

James McKeown and Jesse Hare both lead their respective teams. This allowed for better and more effective team co-ordination. It helped all members stay on task and stopped individual team members from deviating from the plan and doing their own work. Both of these members (James and Jesse) took on the most technical aspects of the project, especially in regards to the planning the project (in terms of coding). Other members then utilised this research to program their respective tasks within the project.

3.3 Communication and Meetings

The following sub sections will discuss how well the team communicated throughout the project duration.

3.3.1 Effectiveness of Communication

Communication between members was very quick and effective. The team decided to use a private Facebook Messenger chat as the primary point of contact as each member regularly used this application. This made it easier to notify team members of urgent matters. In addition to Facebook, the team decided to use a Slack channel for more technical and project specific subjects. Slack was new to all members and

was therefore not utilised to it's full potential.

The main problem faced with the Facebook Messenger chat, was that the information couldn't be organised. It would have been better to use Slack as the primary messenger application as it was more suited to this type of project; especially because we had 5 team members and 2 sub teams. In future projects organisation of information needs to be a higher priority.

3.3.2 Team Meetings

Team meetings were run through the Zoom video conferencing application. At least once a week the whole team would meet to discuss the progress of the project. Meetings for the back end and front end team were often held more than once a week to discuss new ideas for their respective tasks. These meetings were highly effective as each member could screen share what they were working on and clearly communicate that with the team.

These meetings could be improved by having a set agenda before each meeting. Occasionally the meetings started slow as we all had to organise what we were going to discuss. It would also be ideal to have one member as the designated scribe to take notes for each meeting and relay this back to the team.

3.4 Documentation

The activity log sheets, task summary sheets and meeting minute documents all assisted in tracking how the project was developing. The activity log sheets provided a good point of reference each week in deciding what tasks needed to be worked on next. It also helped ensure that each task was completed. Without the activity log sheets and task summaries it would have been difficult to keep track of the progress of the project. However, it was often the case that some weeks would be forgotten and team members would need to go back and fill in the information. This was obviously not ideal, but did not have a noticeable effect on the productivity of the team.

The meeting minutes documents were essential for the project. Without them, members of the team would forget what was discussed in each meeting. This would result in a massive amount of lost time as information would

need to be repeated. After each meeting with the project supervisor (Stijn Dekeyser), the team would stay back and document what had been discussed in these meetings. This helped tremendously when working on specific parts of the project. This is because we now had a reference to look at, to see if certain features needed to be implemented. These meeting minutes can be found attached to the bottom of this document.

3.5 Process

The following sub sections will discuss the process used to develop the project, the difficulties encountered with this process and beneficial strategies used throughout the project duration.

3.5.1 Overall Process

The team (and it's sub teams) decided on using the agile methodology for the project. Each week a meeting was held between members to discuss what tasks needed to be completed that week. Every 2 - 3 weeks a meeting was held with the project supervisor to discuss improvements that needed to be made. These improvements (as tasks) were added to the list of tasks to be completed.

3.5.2 Difficulties Encountered

Due to time constraints, university commitments and conflicting schedules, some weeks were more challenging than others. As a result some tasks did not get completed on schedule. This resulted in the rescheduling of tasks which hindered productivity. Any difficulty faced was getting team members schedules to align. Occasionally, tasks would get worked on individually and passed on to another team member. Ideally in the real world, each member would have this project as their number one priority with no external distractions (such as university commitments, work etc.).

3.5.3 Beneficial Strategies

Weekly video conference meetings were essential for the project completion. These meetings allowed each member to discuss, and show, exactly what they were working on. Especially as the group implemented the agile methodology, it was important to discuss any im-

provements to the project each week.

Using Trello boards also assisted in the development of the project. It allowed all of the tasks to be laid out for each member to see. Once a task was completed it was moved from the 'to-do' card into the 'completed' card. As there were 5 members of the team, it was extremely beneficial as the tasks could be marked as 'to-do', 'doing' and 'completed'. This helped in reducing and avoiding work that would otherwise be repeated.

4. Project Report

4.1 Project Outcome

4.1.1 The Project

The aim of this project was to implement a software solution to our client's need to search for and examine file metadata from a potentially large dataset. In order to do this, we had to design and develop a solution that consisted of two components. The first of the two, the file metadata harvester, is used to gather the file metadata from files, whereas the second component, the searcher, will make use of a GUI to display that harvest information, and enable the client to interact with it, and perform a variety of functions that will aid in the user's search for particular files.

4.1.2 Project Outcome

The deliverable for our client at the end of this project consists of two software components, and their supporting documentation. We have managed to produce working versions of both components, and now we will go over the outcomes for each of these components in more detail

4.1.3 Metadata Harvester Outcomes

The Harvester program we have developed can be used to search a specified directory, either recursively or non-recursively, and extract the metadata for any files found within the search area. A recursive search will begin in the target directory but will also expand to within and subdirectories within the search directory and continue till there are no more subdirectories to search within. If the user specifies that a recursive search is to be performed, it can be expected to take a little more time than a search performed on a single

directory.

When the Harvester program is invoked, before it's runtime is over it will write all the file metadata to a user-specified file. This file is what is used by the searcher program as input to display the file metadata in a more accessible and interactive GUI format.

The harvester program extracts metadata using the following methods:

- Hachoir external Python Library – this tool is the most valuable in regard to metadata extraction, it is able to extract data from a large range of file types, however it still comes with quirks and it's own set of challenges
- PyPDF2 external Python Library – this library is used to extract metadata from pdf files. We had to use this library because Hachoir deals with most file types except text-based ones, such as pdf, docx, .txt etc.

• Os Python module – this useful module native to Python is invaluable in gathering standardised file metadata such as type, creation, modification and accessed dates. This module can extract the same set of data from any type of file, so we have used this module to ensure all files represented in the searcher GUI have at least this subset of metadata information available to the user. There are some filetypes in which the data extracted will be slightly lacking, and the reason being is simply due to time constraints our team didn't have the time to find more metadata extraction libraries that were focused towards those files that we had difficulty in obtaining data for. If our project does get continued funding, our priority for this component will be the broadening of the types of metadata that can be extracted from a larger range of filetypes. This is not exactly a difficult task, although it is time consuming as all metadata extraction libraries extract their data in different ways, and for our software to work we need it all in the standardised csv format when output to a file. We are confident however that our client will be satisfied with this portion of the project deliverable, as they have indicated to us that their main intentions were to harvest data from their own personal files, and the types of those files they will be harvesting are for the most part supported by our metadata harvester program. It is worth noting, that as could be expected, the harvesting operation can take time when harvesting many files, and although some optimisation was attempted, we feel that we could maybe do more in the way of reducing waiting time for

the users. All in all, we as a team feel that the harvester meets the client's requirements, and although there is room for improvement in regard to performance and the range of files from which unique metadata is gathered, we feel that the harvester will meet our clients requirements.
Metadata Searcher Outcomes The fundamental aim of the Searcher program was to develop a GUI that displayed the harvested file metadata. This metadata could be interacted in a way that would facilitate sorting/searching and filtering on file metadata values to enable a user to located certain files or groups of files that share common attributes easily. This aim was achieved quite early on, however over the course of development, there were additional requirements introduced that improved the searcher programs usability and utility. The GUI design and layout itself changed a few times over development, as well as the functionality and usage of GUI elements. We decided to make use of the external Python library PyQt5 to handle the GUI implementation, but all other libraries used were modules that were native to Python. The most important module we used beside PyQt5 was the SQLite3 library. This library enables the creation of Relational Databases using the Structured Query Language. This proved invaluable in storing the harvested metadata, and interacting with it through SQL queries, which enabled us to manipulate and interact with that data in many ways, opening a lot of possibilities regarding the Searcher program's functionality. In the end we ended up meeting all critical and non-critical requirements gathered from our client over the course of development. The functionalities that the Searcher program provides for it's users include: • Sorting/Searching of Metadata based on one or more fields • Ability to open files when clicking on the filepath in their respective record • Filtering of results based on one or more fields • Responsiveness • Ease of Use Although we are confident that this component has lived up to our client's expectations, we realise that there is always room for improvement. Further development on the searcher program will likely be focused toward performance improvements, and possibly even additional functionality such as deletion and renaming of files, which could prove to be of great utility for its users. Although we focused a lot of resources towards performance/optimisation, the user should still expect that the searcher program may take some time to gather and collect metadata from large csv files. Once the GUI has launched however, all subsequent operations can be expected to perform without delay. The reason for this is that operations on a Relational Database using SQL tend to be quite efficient. As a team we are thoroughly satisfied with what we have developed regarding the searcher program, and we hope our client feels the same way. We are confident it has met all their needs and expectations

and that it remains an important utility for them for some time to come.

Project quality

Usability

Regarding the harvesting of metadata, using the harvester program, is simple. To harvest metadata from a directory, the user simply needs to provide the harvester script, the '-r' option if they wish to harvest from a directory recursively, and the path to the output file. If the user enters the arguments incorrectly a help dialog will print to the screen instructing them on the correct command to execute the harvester program from the CLI. Once executed, the user simply must wait briefly until the harvester program has been executed. With the Searcher program, execution is even simpler from the CLI. All that is required for an argument is the csv file produced by the harvester program. After a brief wait, the GUI appears on screen. Most of the elements are accompanied by labels, and tooltips on mouseover, and those that aren't resemble constructs seen in many other types of applications, following common conventions in relation to GUI design. Examples of the intuitive features are things such as double clicking on a column header to sort, and the results filter that appears above the displayed metadata. The only part of the GUI where a user may get caught out or feel unsure, will provide information on incorrect user input that instructs the user on the correct way of using that widget. Another intuitive feature we included is the ability to double click on file paths to open the file, which is a common practice in many file manager/explorer applications, and it is something that most users may try to do automatically.

Completeness

As far as the harvester program is concerned, all requirements have been met, and it performs as it should. That being said, there is always the opportunity to expand on the range of filetypes that the harvester program can extract metadata from, as well as the range of potential metadata types it can extract. The searcher program has met all requirements as well, functional and non-functional. As new requirements are introduced, we added them promptly, and if development continues so will the addition of features to the searcher program.

Maintainability

As far as maintainability is concerned, that only concerns the harvester program. Maintenance may include the adding/removal/updating of the libraries used to collect the file metadata, but in the end as long as the output remains in the same format, there should be no changes needed on the GUI side of things. Also, there is a generous amount of commenting throughout both the searcher and harvester programs, so developers not

familiar with the project would easily be understand how the code works and how to modify it if needed.
Reliability
Reliability issues have been found only in a couple of isolated cases. For the harvester program, data that is not usable by the searcher program or meaningless to a user may be harvested if the harvester is executed on any sort of system folders. The reason being that these folders can sometimes contain obscure filetypes which often have no readable metadata, or the metadata that is read is either of no use to detrimental to the program's operation. As far as the searcher program is concerned, as long as the input csv file is in the correct format, there should be no issues. The searcher program has been thoroughly tested and there has been a significant amount of bug removal to ensure as stable of a user experience as possible.
Portability
Both programs have been designed to work on Linux. Early in development however, we had working versions on windows as well, so If needed further down the track cross-compatibility implementation would not be too big of an issue. As long as the user has python 3.7.x, the Hachoir, PyPDF2 and PyQt5 libraries installed, they should be able to use these programs on any Linux distros.
Performance
Time from execution to completion of the Harvester program is linearly related with the number of files to harvest metadata from. The more files that are harvested for metadata, the longer it will take. Harvesting using a recursive search of approximately 3500 files will take around 2minutes on a mid-range laptop. The time from execution to display of the searcher GUI takes slightly more time, however, once the GUI is loaded most if not all operations are near instant. It is important to us that the user is aware of potential waiting times.

4.2 Cost of the Project

The following sub sections will discuss the time and labor costs of the project.

4.2.1 Cost of Hardware

There are no deviations to the cost of hardware from the projection provided in the initial project plan. The costing table can be viewed below.

Hardware Used	# of Items Used	Cost ($AUD)	Total Cost ($AUD)
Microsoft Surface Pro 6	5	$1,349	$6,745
MacBook Air 13" (128GB)	1	$1,699	$1,699

As the project was worked on remotely, all team members required company issued equipment. There were a total of 5 Windows machines provided to each member, with an additional MacBook Air provided to the project leader, James McKeown. There were no unexpected circumstances that required the purchase of any other pieces of hardware.

4.2.2 Cost of Software

There were multiple software solutions that were incorporated for the development of the project. There were some software solutions that were not foreseen in the initial project plan. All of the software solutions can be viewed below.

Software Used	# of Accounts Needed	Cost ($AUD)	Total Cost ($AUD)
Anaconda	2	$0	$0
Atom (IDE)	5	$0	$0
Facebook Messenger	5	$0	$0
Slack	5	$0	$0
Spyder (IDE)	2	$0	$0
Zoom Video Conferencing	5	$0	$0

In the planning stages it was decided the front-end team would use Spyder as the preferred IDE. It added more specific functionality compared to Atom in our case. Anaconda was also used for the development of the front-end GUI. Facebook Messenger and Slack were used for communication between team members. Zoom Video Conferencing was used for group meetings and screen sharing.

4.2.3 Time Cost

The project took 12 weeks to complete with each member contributing 20 hours a week. Each member therefore contributed 240 hours of work to the project with a combined team total of 1,200 hours.

4.2.4 Labor Cost

The team was split up into 2 sub teams, a front-end team and a back-end team. The front-end team consisted of Jesse Hare and Ryan Sharp. The front-end team consisted of James McKeown, Richard Dobson and Vincent Roberts.

Team Member	Role	Hourly Rate	Hours Per Week	Total Weeks Worked	Total Cost ($AUD)
James McKeown	Team Leader/Back-End Developer	$39.90	20	12	$9,576
Jesse Hare	Front-End Developer	$34.10	20	12	$8,184
Richard Dobson	Back-End Developer	$35.40	20	12	$8,496
Ryan Sharp	Front-End Developer	$34.10	20	12	$8,184
Vincent Roberts	Back-End Developer	$35.40	20	12	$8,496

The only difference between the projected labor cost and the actual labor cost was the error in how many weeks the project would take. It was initially predicated that the project would run over 10 weeks, when in fact the project ran for a total of 12 weeks.

4.2.5 Lessons Learnt About Project Cost Estimation

The major lesson learnt was that no prediction about the cost will be 100% accurate. Lots of things change in a 12 week period and there is no possible way that these can be accounted for. For example, majority of team members used Atom before this project, so it was assumed that we would use this again. However, once the front-end team researched more into what programs were out there, it was decided that Atom was not the idea choice and that the team should use Spyder

instead.

5. Professionalism and Professional Ethics

5.1 Professionalism

Throughout the project, professionalism was kept to a relatively high standard. In future projects, it would be optimal to have a set of professional standards that outlines what is expected of each team member. This would allow the team to refer back to this document should any breach of the standards occur. The following sub sections will discuss issues related to professionalism that occurred during the project.

5.1.1 Client Interaction Confusion

Client interaction is the most important part of a project, without them the project would not exist. In this project, the client was also the project supervisor who gave information on behalf of both parties. While this did not have a major effect on the outcome of the project, it was obviously difficult to differentiate between a client meeting and project supervisor meeting. As a team, we should have been clearer with the meetings we wanted to have. Upon reflection, it would have been better if we clearly communicated they type of meeting we would like with the project supervisor.

5.1.2 Lack of Client Interaction

It is important for any project to get as much client input as possible too ensure client satisfaction. While the team met the required amount of meetings with the client, it was most likely not enough. In a real world scenario it would have been best to meet with the client every 1 to 2 weeks, especially as the scrum process was used. This would have ensured that at every sprint the client had some input into the project and could see updates regularly.

5.1.3 Team Communication

Communication between team members is essential for the progress of a project. On occasion, some team members did not reply to messages

in a timely manner. This effected the progress of the project as some work had to be delayed to due other work not being completed. This was addressed via a team meeting, where all members gave their opinions on the matter. The issue was addressed and after the meeting took place, this particular issue did not happen again.

5.2 Professional Ethics

Throughout the project professional ethics was also kept to a relatively high standard. As this project was conducted in a university environment, and was therefore a mock real world project, there was only one major professional ethic concern. This will be discussed in the following sub section.

5.2.1 Sharing of Sensitive Client Information

Data privacy should be top priority for any business and associated business project. Due to financial constraints of the project is was not feasible to have a private GitHub repository. This resulted in the team using a public GitHub repository that could be accessed by any member of the public. In a real world scenario this would obviously be very bad practice. In future projects, with more financial backing, it would be essential that a private GitHub repository be used.

6. Contribution Distribution

6.1 How Work Was Distributed

Each sub team, front-end and back-end, were responsible for their own individual tasks. Meetings were held on a weekly basis to decide what tasks needed to be completed that specific week. These tasks were allocated to team members based on their previous knowledge and experience of the task. For example, Jesse had previous experience with some Python GUI elements, so he was tasked with researching and developing the initial concept.

6.2 How Effective Was The Distribution

This method of distributing tasks worked quite well and resulted in a clear structure that needed to be followed. Without this distribution, team members would be going off and doing there own thing. This way each member

knew exactly what they needed to do. While distributing tasks, there was no way to make sure everyone received the same amount of work. This was the only hurdle faced, but was mitigated as best best as possible.

6.3 Task Distribution Table

The following table represents the tasks completed in the project and the percentage of effort put it in by each team member.

Main Tasks	Richard Dobson	Jesse Hare	James McKeown	Vincent Roberts	Ryan Sharp
Research	25%	25%	25%	25%	25%
Discussions	25%	25%	25%	25%	25%
Documentation	25%	25%	25%	25%	25%
Meetings	25%	25%	25%	25%	25%
Implementation	25%	25%	25%	25%	25%

7. Conclusions

Reference List

Appendix