

SOFTWARE ENGINEERING: CONTEXTUALISING COMPUTER SCIENCE

Software engineering is an important and highly relevant topic for students to study at high school. Many students, along with the wider public, believe that developing software is all about writing code. This leads to the stereotypical misconception that jobs in the information technology sector are reserved for “geeky” programmers, and that those jobs have a narrow focus lacking in social interaction and purpose. Females especially want to pursue career pathways where they see relevance and connection to real world issues. According to Rane Johnson-Stempson (2014), *Microsoft Research, Education & Scholarly Communications Principal Research Director*:

As I speak with young women around the world, I continue to find that their disinterest in computing stems from a lack of familiarity with its exciting and impactful career possibilities. The students I talk to almost invariably voice a desire for a vocation, not just a career. They want work that is socially meaningful...We need to show young women that computer science is the field to be in if they want to make an impact on society and help people. We need to make them realize that the latest advances in improving healthcare, protecting the environment and upgrading manufacturing have come from technological innovations. (para. 3, 6)

So how can the study of software engineering help students (especially females) overcome these misconceptions? Software engineering situates programming within the larger context of problem analysis, design and testing of a software product to ensure it meets the needs of users and wider stakeholders. Through studying software engineering, students can see that the actual goal of software development is to bring benefit to the end users and the wider community stakeholders. Furthermore, the study of software engineering highlights the critical role that interaction with users and clients, as well as other team members, plays in software development and thus dispels the misconception that is a socially isolated field.

As suggested in the Software Engineering Chapter of the *Computer Science Field Guide* (Bell & Morgan, 2014), the concepts are difficult to teach without real world case studies. This topic presents the ideal opportunity for guest speakers from the information technology sector to visit the classroom and discuss their software development process, or for students to visit a company to gain first hand experience. In either case, students are exposed not only to software engineering as a topic for an external report, but also real life examples of software development in context and possible career pathways.

Finally, software engineering fits very well within a program of teaching and learning at Level 3 NCEA that has a project development and NCEA Scholarship focus. Students can model software engineering practices within their own project development, which supports internal assessment opportunities within the NCEA Generic Technology Achievement Standards Framework and the Technology Scholarship Portfolio Report. There is a strong correlation between software engineering

and the NCEA Generic Technology Internal Achievement Standards, thus this topic underpins a level 3 Digital Technologies programme with a focus on developing stakeholder-oriented projects:

- Brief development – determining **functional requirements** from different stakeholder groups and **non-functional requirements** from the context considerations of the project including the social and physical environment.
- Project management – researching **project management methodologies**, applying a project management schedule
- Technological modelling – **risk mitigation** through determination of requirements, balancing **competing and contestable factors** and **iterative** development
- Prototyping – trialling, **rigorous testing** and **iterative** development

OVERVIEW: RESOURCES AND IMPLEMENTATION

In order to develop both my own and my students' understanding of the topic of software engineering, two main resources were utilised. The *Software Engineering* chapter of the *Computer Science Field Guide* (Bell & Morgan, 2014) was used to provide the topic framework and explanation of the key concepts. *Head First Software Development* (Miles & Pilone, 2008) was used to provide further examples with diagrams and illustrations of the key concepts as well as review activities. Various other Internet sites were used for reference or clarification of terminology, which are referred to throughout this report.

In addition to these resources, a three-day immersion field trip to visit software development companies and learn about their processes first-hand was a key experience for understanding the theory and terminology in context. The students were able to spend a day at Orion Health in Christchurch learning about Agile methodology and working with their cross-functional SCRUM teams. This experience was crucial to understanding the theory presented in the other resources.

The students were inspired by the field trip and could see the benefits of implementing an Agile methodology in their own project development. Thus, the result of the field trip was a 'team decision' to turn our Level 3 Digital Technology class into an 'Agile Classroom'. The goals for implementing an 'Agile Classroom' are as follows:

- Students will develop their project briefs and final outcomes in an iterative and authentic manner, following the Agile methodology, which will improve the quality of the projects and NCEA Scholarship portfolio.
- Students will manage their development timeline more effectively via the use of SCRUM boards and visual planning.
- Students will be able to write authentic reports for the NCEA computer science external, reflecting on their process in conjunction with what they have seen implemented in the real world.
- Students will be motivated to pursue software engineering as part of their tertiary study.

In the remaining sections of this paper, the key concepts of software engineering will be discussed, resources analysed and details of our classroom implementation will be highlighted.

SOFTWARE ENGINEERING: KEY CONCEPTS

MANAGING COMPLEXITY

Software projects in the real world are large and complex and are generally developed by teams, as it is impossible for any single person to understand the entire problem or code and test the solution. A structured methodology for development is necessary to manage the complexity of the software and to ensure it meets the requirements of the stakeholders, is delivered on time and within budget. Without a structured process for development of software, the probability and severity of risk for project failure and bugs in the software increases dramatically.

The introductory material of the Software Engineering chapter of the *Computer Science Field Guide* (Bell & Morgan, 2014) provides excellent examples, including two video segments, to help students understand the complexity of large software projects and the scope and severity of the loss that can occur when software projects fail. The *Field Guide* suggests that students can learn a great deal as to why it is crucial to have sound software engineering practices from looking into projects that have failed. Website links are provided to relevant articles about software failures that students and teachers can explore for discussion. In addition to the examples suggested within the chapter, the NovaPay failure is a relevant example for New Zealand students (and their teachers). There are many available on-line discussions and articles that focus on this topic, such as, *Eight ways Novapay failed* (Gauld, 2013) or *Novopay 'failure' reduced risk hopes* (Gibb, 2013).

It is important for students to look at these large-scale examples of software development projects, both successful and failed implementations, as the scope of their own development is quite limited. However, many of the key principles of software engineering apply no matter the scale of the development and students can adapt these principles to their own practice. An example from *Head First Software Development* (Miles & Pilone, 2008), in relation to stakeholder consultation and iterative development, illustrates this point:

You might only be a development team of one, but when it comes to your project there are always, at a minimum, two people who have a stake in your software being a success: your customer and you. You still have two perspectives to take into account when making sure your software is on the right path, so iteration is still really helpful even in the smallest of teams.
(p. 12)

SOFTWARE DEVELOPMENT STAGES

Software engineering models comprise of the same general stages: *analysing requirements, designing the software, coding/implementation, and testing*. The software process or methodology that is followed to work through the stages varies, however there are two main categories of methodologies: linear and iterative. Linear methodologies work through each stage until completion, whilst iterative methodologies work through the stages in cycles. It is generally accepted that a purely linear approach is not responsive enough in software development and that iterative approaches help to mitigate the risk of project failure. Two specific implementations of methodologies will be discussed later in this report.

Requirements Analysis

A crucial aspect of the software engineering process (or any engineering process) is to ascertain the functional and non-functional requirements of the project before launching into the development of the outcome. However, it is often difficult for the stakeholders to fully communicate their requirements. Furthermore, different stakeholder groups may have conflicting requirements that need to be negotiated so that the software suits a range of needs. The hardware that the software is to run on and the physical environment it is situated within also need to be determined as part of the analysis. However, the introduction of new technology during the software development may lead to changes in these requirements. *Section 16.2 of the Computer Science Field Guide* (Bell & Morgan, 2014) discusses these key concepts in further detail with examples and contexts that high school students would relate to and understand.

16.2.1. PROJECT: FINDING THE REQUIREMENTS from the *CS Field Guide* (Bell & Morgan, 2014) would be a useful guideline for students who are developing their own project brief and need to research the requirements of a variety of stakeholders. NCEA Achievement Standard 91068, *Undertake brief development to address an issue within a determined context*, requires students to develop a project brief and specifications that reflect on-going consideration of a range of stakeholders' views and needs. As part of the brief development process, students are also required to analyse the social and physical environment of the intended outcome and the wider context considerations. Analysing the stakeholders' needs helps the students to develop the functional requirements; whilst analysing the social and physical environment along with the relevant context considerations helps the students to determine the non-functional requirements.

In our classroom implementation, as part of the brief development assessment, students were given the task of researching the project context and stakeholder issues to develop an initial set of project requirements.

An excerpt from the task and sample student responses are shown below:

Research (through interviews, questionnaires, on-line research, etc.) the context and issue in more detail.

- 1. What is the current situation within the context? What issues are there that need to be resolved? What needs or opportunities for the users reside in this issue?*
- 2. What is the social environment in which the solution will be situated -- the school, the hostel, a sports club, etc.?*
- 3. Who are the key and wider stakeholders within this environment? Think not just of the main "clients" but also the users and community who will be affected by the solution you develop.*
- 4. What is the physical environment? What hardware/software will be used to create and access your solution? Will it be public on the WWW or local to an intranet within the school or hostel?*

*Create an A3 poster for your SCRUM board that summarises your project idea. Include a 'conceptual statement' that clearly communicates **what** will be developed, for **whom**, **where** it will be situated, and **why** it should be developed. You may break this down for your poster into headings "What", "Who", "Where", "Why".*

Figures 1 & 2 - Each student is using SCRUM board as visual project management method. The A3 posters were a suggestion from Jan Behrens, Product Development Director Orion Health, during our visit.

ATC-BRIEF


Who	What are the issues/what they need	How they currently deal with these issues	Why these methods could and should be improved
Who are the stakeholders <i>The organisers of No.42 Squadron ATC</i>	<ul style="list-style-type: none"> More effective method of advertising the organisation to the public. A way in which the public can contact the organisers with any personal queries about the group. 	<ul style="list-style-type: none"> Squadron advertises once a year in the newspaper. Posters are also put up in local schools. Currently there is no way that the public can contact the squadron organisers or group. 	<ul style="list-style-type: none"> Advertising through Newspaper and on posters is bad for the environment as it uses a lot of paper, and is not easily accessible for all members of the public. The squadron needs to be easily contactable by the public.
<i>Cadets who attend No.42 Squadron ATC</i>	<ul style="list-style-type: none"> Accessible information about upcoming events/camps, regarding to when and where the events are being held, and the gear list for any camps. 	<ul style="list-style-type: none"> Paper copies of information about upcoming events, camp gear lists and camp permission slips are handed out to cadets to take home. Some messages regarding important upcoming events are also put on the No.42 Squadron ATC Facebook page. 	<ul style="list-style-type: none"> Wastage of paper, and inefficient use of time as leaders have to hand type and print all information slips for every single camp/event. Also handing out information on slips to cadets has proven to be an unreliable method at getting gear lists, camp dates etc. home to parents. (liable to lose hand printed paper copies) Not all parents of cadets uses Facebook and so find it difficult to acquire information through this method of online communication from the organisers.
<i>Parents of Cadets</i>	<ul style="list-style-type: none"> Efficient access to information concerning when parents association meetings are being held. 	<ul style="list-style-type: none"> Emails are sent to parents of cadets to remind them of parent association meetings. 	<ul style="list-style-type: none"> Inefficient use of time for the organisers having to hand type and send emails reminding parents of the next up coming meeting.

Why
Creating an updateable dynamic website that advertises No.42 Squadron to the public as well as allows the public to contact the organisers through an email link/in built system, and also displays information about upcoming events for both cadets and parents of cadets, is beneficial to the stakeholders because:

- It will minimise the amount of time and effort that the organisers would spend on tedious tasks like typing and printing camp/event slips and hand typing and sending reminder emails about upcoming meetings for parents.
- As printing letters to be handed out requires money for the ink and paper - which is an administration cost of the squadron - it is more cost effective to have information displayed on a website accessible at home rather than on paper sheets to be taken home. Furthermore advertisement in newspapers also comes at a cost to the squadron's finances.
- Printing information on paper and advertisement on posters around schools is not only costly for the squadron's administration finances but also bad for the environment with a large amount of paper usage. Advertisement and communicating information on a website is a better alternative particularly when taking into consideration the impact that printing and wasting paper has on the environment.
- Having advertisement with contact details and displaying information slips on a website is an efficient and reliable method of communication, due to the ease and accessibility that people in our society have to the Internet. So a website would be more accessible to a wider range of audience than having advertisements in newspapers or posting information on a Facebook page. This will make advertisement for the squadron more effective, and also communication between the members (and parents of members who may not have Facebook) of the squadron more efficient.

Where
Social Environment: The community that will be using this outcome will be the people directly involved with No.42 Squadron - particularly the cadets, parents of cadets, and organisers for the Squadron - and also the wider audience, which includes any members of the public who are seeking to find information about ATC No.42 squadron.
Physical Environment: Hardware: The outcome will be situated on the World Wide Web due to this being easily accessible for directly stated stakeholders (cadets, organisers etc) as well as members of the public.
Software: Coded in HTML, CSS, PHP, and MySQL giving the website a dynamic aspect that enables it to be easily updateable by the manager of the site.


When
From the stakeholders there is no specific deadline for the completion of this technological outcome, however the faster the outcome is produced and put into use, the quicker the stakeholders can start managing and advertising the communication of information in an efficient time-saving less costly manner. I am aiming to have this proposed website up and running for Squadron No.42 ATC before the end of term 3 this year, as this project must be completed before this deadline in order to be able to be submitted as an assignment that counts towards NCEA.



Who
We are creating this eBook for Mrs. Anglin who is in charge of the Columba College Centenary. This eBook is going to be used by anyone who is attending the Columba College Centenary in 2015. This makes it a wide audience as anyone can come to the events at the Centenary and anyone who wants to download it can. To make sure it is easy to use for everyone, we will test it on many age groups. Because it is a centenary, people of any ages may want to look at the eBook, such as old girls of the school who could be elderly or have just left school or younger children in the junior school. This means we have to make the book easy to use for all ages, this means we will make it as uncomplicated as possible while still making the design interesting.

How
We will create this eBook using iBooks author, which is an application on our laptops. This application will allow us to make a book with a main screen and lots of different pages of the decades. We will be able to create a format for the whole book for continuity through out the decades. The App. allows us to put text and pictures on the pages so the book can just be read or put in audio and video for a more interactive eBook. We can use In-Design to create the design formats for the different pages so we will be able to create our own designs that will best fit the book. For the posters we will also use In-Design and Photoshop to create the designs we want.

When and Where
This eBook is to be finished before the end of term 3 along with the posters. The Columba College Centenary is in March 2015 so any final changes have to be made well before then. This eBook will be available at the Centenary for people attending to look through and also will be available online to buy from Amazon.



Columba College Centenary eBook

Regardless of whether or not students are developing their own brief and project specifications, it is useful to introduce the concept of 'User Stories', to assist students in understanding how stakeholder requirements are determined and provide a clear format for articulating these functional requirements of the software. Chapter 2 of *Head First Software Development* (Miles & Pilone, 2008) provides practice exercises and straightforward criteria for developing user stories, such as, "a requirement should be written in the customer's language and read like a user story: a story about how their users interact with the software you're building" (pg. 30). User stories do not define methodologies for implementation, as the developer determines details of implementation in the design phase. The user story deals with required functionality from a stakeholder's perspective. An illustration from the chapter 2 of the book that the students found very helpful is shown below:

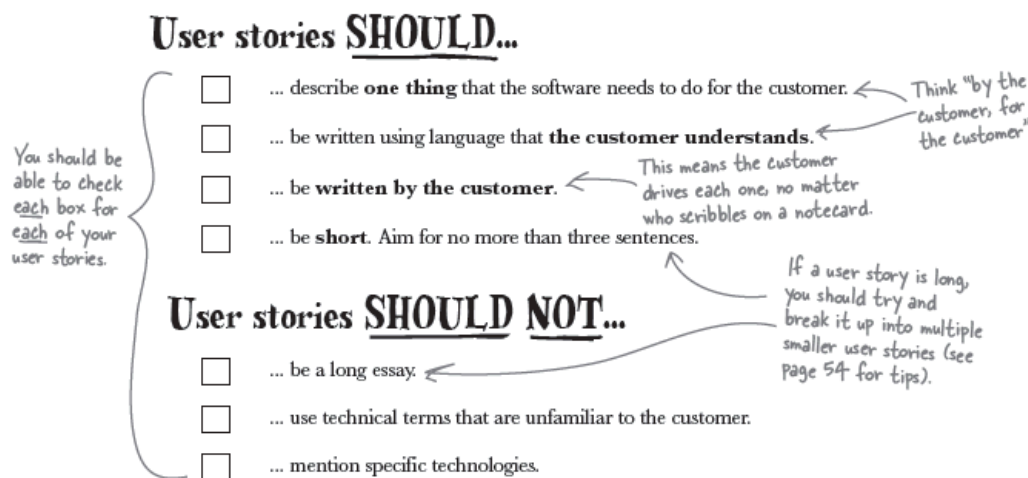


Figure 3 - User Stories (Miles & Pilone, 2008, pg. 39)

A typically used format of a user story is:

"As a <type of user>, I want <some goal> so that <some reason>". (Cohn, 2014, section 1)

This was the format that we were shown at Orion Health and the girls' first introduction to the idea was through developing their own user stories on the day of our visit. The user stories described what they wanted to achieve from the day's session and we used those to develop the day's tasks. In order to reinforce this learning, and gain first-hand experience with developing user stories for a software project, the students have implemented this methodology in developing their project briefs. After completing the research described above, the students developed 'user stories' for their various stakeholders.

An example is shown below from a student who's project focus is developing a dynamic website for a local restaurant:

As a customer of the restaurant
I want to be able to narrow down dietary requirements (for vegetarian and gluten free options)
So that I can determine if there is food that would agree with my diet.

When presenting the topic of requirements analysis, it is beneficial to introduce the concept iterative development. When all requirements are gathered up front and no further stakeholder consultation or research is completed until presentation of the final product, it leads to software failures or time and budget overruns, as discussed previously. Iterative development is a key to mitigating risk of failure. In chapter one of *Head First Software Development* (Miles & Pilone, 2008) clear examples are provided of what can go wrong using the 'Big Bang Approach' to requirements analysis. Review questions are also included that can aid students to think about how failure to communicate clearly with stakeholders to obtain requirements iteratively can derail project success. The chapter also provides explanations of iterative development processes with illustrations and diagrams that are helpful for student understanding. Iterative development involves analysing requirements, designing the software, coding, testing and obtaining stakeholder feedback on one discrete segment of the software before proceeding on to the next. This provides a means for ensuring feedback is obtained at regular intervals along the way in order to keep the software on track with the client's expectations.

In our class implementation of software engineering practices, the next step after determining the user stories was to develop initial specifications and project tasks. Below is an example where the student has broken one user story into overarching development tasks relating to that particular story. She has built in tasks to iteratively consult with her stakeholders. This is only one portion of the entire website, so a similar iterative process will occur for other user stories, until the entire site is built and tested.

<p>As a student user I want to use an interactive app which calculates nuclear fission/fusion reactions based on input So that I can earn a better understanding of nuclear fission/fusion equations</p>	<ul style="list-style-type: none"> ● Sketch design and format ● Define input format and parameters ● Design interactions ● Consult stakeholders and edit ● Create high-fidelity sketch of outcome ● Consult stakeholders ● Pseudocode outcome ● Consult Mrs. McMahon, edit ● Code JavaScript ● Test outcome
---	---

The student has also analysed other factors to determine non-functional requirements:

One of the most important things to test for is the functionality and usability of the outcome on a variety of different platforms and environments, including hardware, software and browsers. Testing must be repeated on as many different combinations of hardware, platform and browser as possible, so that the reliability and functionality of the product can be ensured for as wide a range of audiences as possible. If impossible to resolve, at least an acknowledgement of this failure should be present in the form of error handling procedures, so that the users are aware of the state they have reached and its cause.

- **Testing**
 - ✧ Internet browsers on my laptop, e.g. Safari, Chrome, Firefox
 - ✧ Online testing e.g. W3C markup validation service, Sauce labs
 - ✧ Different hardware, e.g. Windows computers at school, Apple tablets, non-Apple tablets, smartphones
 - ✧ Uploading outcome to a temporary and restricted URL

Software Design

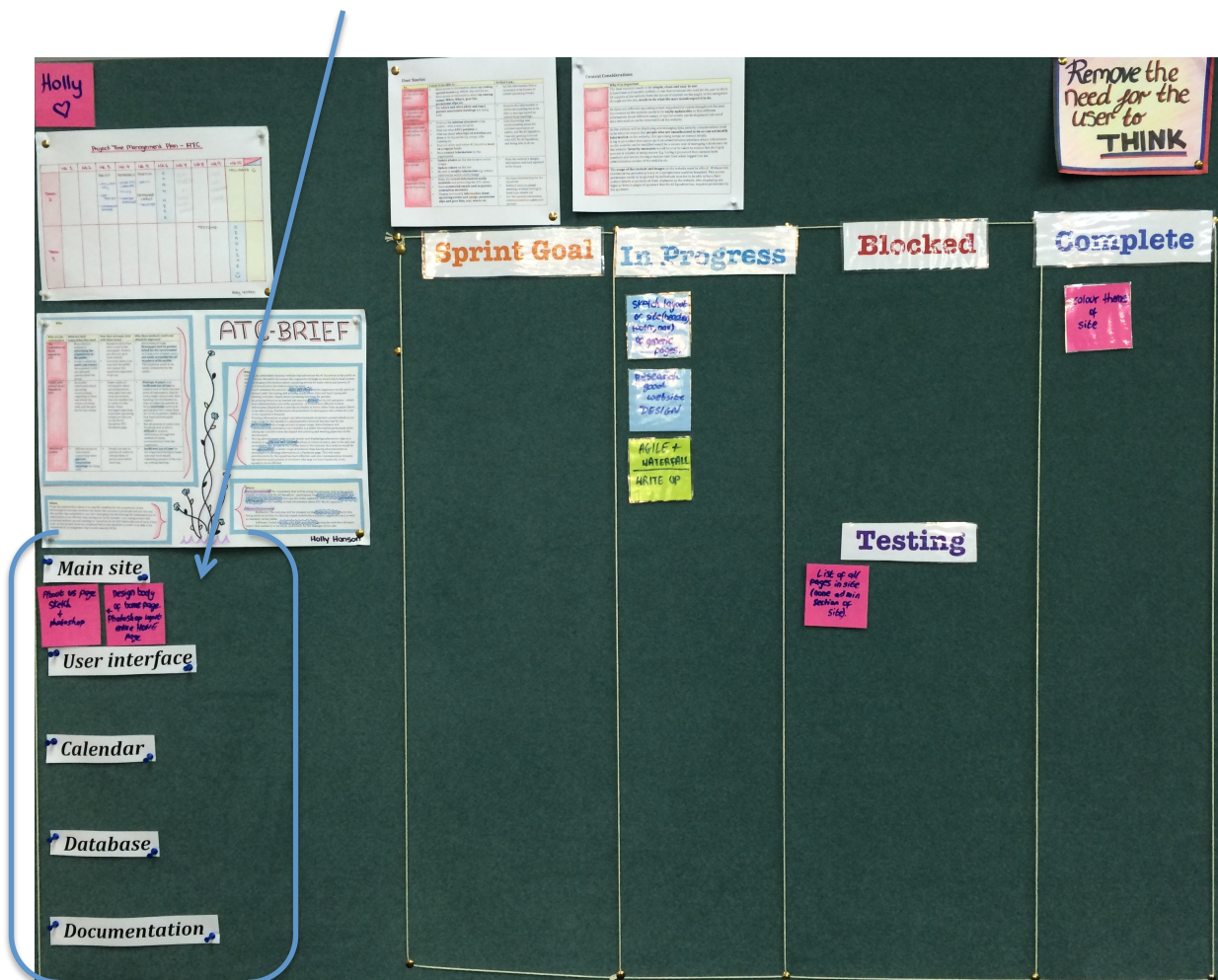
Once you have decided what your software needs to be able to do, you can actually build it. But just blindly starting to program is likely to get you into trouble; remember that most software is huge and very complex. You need to somehow minimise the amount of complexity in software, otherwise it will become impossible to understand and maintain for other developers in the future. Software design is all about managing this complexity and making sure that the software we create has a good structure. (Bell & Morgan, 2014, Section 16.3, para. 1-2)

One of the key concepts in designing the structure of the software is *subdivision* or breaking the software into smaller parts that can be built independently. Depending upon the size of the project and the software process methodology, subdivision may occur within a team and/or across teams. For example, within a cross-functional team, one person may be working on the user interface, while other team members are working on the programming code, while yet others are working on the database structure (with the testers working in tandem with the other members of a team). In very large-scale projects, various teams may be working on different parts of the software in parallel.

The other key concept is *abstraction*, which refers to breaking the software into layers where each layer only needs to know how to communicate with the layer contiguous to it, but not how it works. Two excellent examples to illustrate abstraction are provided in the *Computer Science Field Guide, section 16.3. DESIGN: HOW DO WE BUILD IT?* (Bell & Morgan, 2014). One example relates to the computer's layered system (hardware -> operating system -> application software) and another to the layered system of Facebook (database-> logic -> user interface). Students who have completed the NCEA Level 1 Digital Information external assessment, will be familiar with the computer's layered system whereby application software communicates with the operating system, which in turn communicates with the hardware. The application layer does not know how to control to the hardware as requiring all applications to know about every piece of hardware available would make application software far too complex and cumbersome. Students who have created database driven websites will be able to relate to the Facebook example. They would have created an interface using HTML/CSS, programming logic using a language like PHP to access the data via queries using MySQL or similar. Another means to explain the notion of abstraction to students is through simple API's, for example Google Maps. Many students may have used the Google Maps' API to embed a map into a website. They will understand that they access the functionality of Google Maps, by only understanding how to utilise the API, without understanding the logic behind how it works. In the terminology of the Technology curriculum, abstraction deals with "black-boxed systems".

16.3.1. PROJECT: DESIGNING YOUR SOFTWARE from the *Computer Science Field Guide* (Bell & Morgan, 2014) could be used with the Generic Technology AS91610, *Develop a conceptual design considering fitness for purpose in the broadest sense*. When developing a conceptual design, they need to refer to the project specifications (requirements analysis) and design a conceptual model of the different parts of their software. For example in a website with a database back end, students should design the user interface, the database model, the logic layer for accessing data and presenting it via the web interface.

Although students don't generally work in teams, they can still break the development into *subdivisions*; e.g. creating a database back-end for a website, creating a menu system, designing the user-interface, and programming the logic. The student's SCRUM board pictured below shows that she has planned to manage her project development in subdivisions. As our class is following an Agile methodology, the student is using an iterative process of analysis, conceptual design, implementing and testing for each layer.



Testing

Software must be tested before being released as clients and end-users wouldn't be happy with a product that is full of bugs. Some bugs can be very severe and cause financial disaster, harm or even death. Thus testing must be thorough, iterative and planned. Students can refer back to the software failures discussions to appreciate the severe nature of risk that can occur through flawed testing procedures.

There are a range of testing types that are carried out during the testing phase. *Automated testing* can be programmed to run repetitively to find the probability of a bug occurring. Automated tests can also be run after each programming update to determine if that update to the software has caused a bug to occur. *Manual testing* is also necessary. It may be exploratory in nature and completed by a member of the development team, or it may be carried out by an end-user.

Unit Testing is another key type of testing, meaning that as each discrete portion of the software is being developed, it is tested as a unit to reduce the complexity of finding a bug in the final code. *Integration testing* takes place once all the units have been tested to determine if all the parts work together properly as a whole.

User Acceptance Testing is important to determine if the software is meeting user/stakeholder expectations. Whilst unit/integration testing may reduce bugs, it doesn't guarantee that the software is what the user actually needs. Acceptance testing left until the last stage before product release can be disastrous to software projects.

The activity provided in *section 16.4 of the Computer Science Field Guide* (Bell & Morgan, 2014) on black-box testing is worth using with students, especially if they are completing a programming implementation achievement standard. This will assist the students in thinking about some of their own test cases, which they can implement during their own programming. Chapter 7 (pp. 237-242) of *Head First Software Development* (Miles & Pilone, 2008) provides a good summary of box-box, grey-box and white-box testing procedures. There is also an activity in which students can practice writing black-box and grey-box testing procedures.

The group of students involved in this implementation project have a very sound foundation in understanding the role, nature and importance of testing through their studies at Level 2 NCEA. In our Digital Technology curriculum at Level 2, we focus on the external topic, *Demonstrate understanding of how technological modelling supports risk management* (AS919358). Through studying this topic, the students have participated in software testing run by local software development companies and those companies have explained the vital role of testing to their process, as well as their testing methodologies. One of the companies studied, ADInstruments produces software for many hospitals, research institutes, and pharmaceutical companies, thus incorrect results produced by the software could be devastating. Having students study the risk management topic at NCEA Level 2 is an excellent foundation for understanding software engineering practices at Level 3. If students have not studied this topic at Level 2, then a visiting a local software company or having a company visit the class to make a presentation is invaluable.

When the class visited Orion Health this year, they had a workshop with one of the software testers who explained their processes and led them through some sample testing exercises. The students

were also able to sit with a tester as she ran some automated testing procedures on an updated piece of code.

Finally, we were provided with an example of unit test conditions within the Orion process (which links directly to the user story):

As a Collector (Phlebotomist / Nurse)
I want to be able to easily search for collections
So that I can easily locate those collections I want to reprint labels for

Scenario: Collected orders are listed in the recent collections screen sorted by date
Given a patient has had two orders collected
When I open the recent collection screen for that patient
Then the collections for those orders are displayed
And the most recent collection is displayed first

Scenario: Labels are not available for reprint once the specimens have arrived at the lab
Given a patient has had an order collected
And the specimens from that collection have arrived at the lab
When I open the recent collections screen for that patient
Then the collection is not displayed

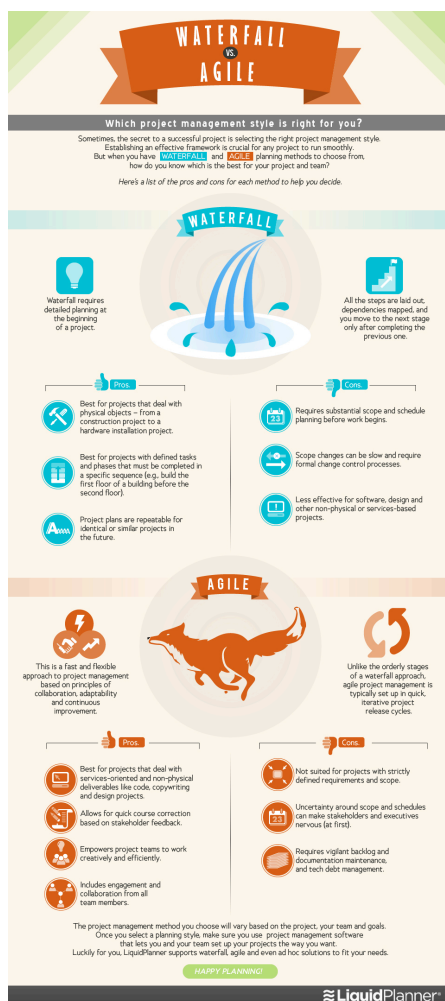
The next step for the students within our class implementation is to develop unit tests based upon this model. Thus far, any ‘testing’ the students have done has been in the form of obtaining stakeholder feedback on conceptual designs. Having completed the initial requirements analysis and conceptual designs, most of the students are about to embark on sprint iterations that will involve implementation and testing.

SOFTWARE PROCESSES

As discussed previously, the phases in software development (analysis, design, implementation and testing) can be carried out in a linear or iterative process. Linear processes, usually termed Waterfall or Big Bang approaches, were adapted from more traditional engineering disciplines, where it is not always feasible or practical to build in an iterative fashion. However, in software development, where flexibility and responsiveness are important, an iterative or Agile process is the usually the better option. From our class visit to Orion and various readings on the topic, I developed the table below to provide a summary of the differences between Waterfall and Agile methodologies.

<i>Linear verses Iterative Development</i>	
Waterfall (Linear)	Agile (Iterative)
Each phase in the process is completed before moving on to the next.	Each phase is completed iteratively in a short sprint of 2-weeks.
There is little change once the project design phase is complete.	Flexibility is the key, so that the software can be changed if there is a design issue or change to requirements.
Software development work is completed in “silo” environment.	Software development is completed in cross-functional teams.
Coders and testers work in opposition.	Coders and testers work in tandem.

As stated previously, the students decided as a “self-organising team” to become an ‘Agile Classroom’ after our trip to Orion Health. The students were able gain insight, in only one day, into the benefits of using an Agile methodology in the classroom for their Level 3 projects. They had previous experience at Level 2 in creating a website for a real client under the Waterfall methodology, which they have all now termed the “Waterfall of Tears”. This is due to the fact that they experienced first-hand what happens when the final outcome is not shown to the client until the end and the client then adds/modifies the requirements. Although the client was happy with the final outcome and the students gained invaluable experience, they were determined not to repeat the mistakes of the past.



To follow-on from our visit to Orion, I developed a task for the students to help them organise their thoughts on software processes. This task supports the students toward providing evidence for the following assessments:

Internal - Brief Development (3.1) – Context Consideration of Software Development Methodology

Internal - Project Management (3.2) – Undertake project management to support technological practice

External - Software Engineering (3.44) - Demonstrate understanding of areas of computer science

In the task, student were asked to perform further research into Agile methodology and visual planning, as well as to reflect upon their own experience developing a project using Waterfall methodology and what they had learned at Orion Health. I was very pleased by the level of understanding the students have shown in their responses and I can ascertain from these responses that the students are engaged with the topic and are on-track for success in both in their internal assessment projects and their external assessment reports. I have included the task and three sample student responses as appendices to this document.

The infographic at the right was a resource that the students found helpful in evaluating Waterfall verses Agile methodology. (Sussex, 2013)

CONCLUSION: THE AGILE CLASSROOM

The students have set up SCRUM boards around the classroom for visual project management, which keeps them accountable to me and the other members of the class. They cannot hide behind a computer and feign progress. They have created user stories to serve as their project backlog. They are breaking down user stories into a series of tasks and are working iteratively to develop their projects over 2-week sprints to complete those tasks. Each Friday, half of the class must do a 'stand-up' and report on their progress. As the overall 'team leader', I conference with each student daily as to their progress toward their sprint goals.

As this is an on-going project, the final conclusion as to the success of our Agile Classroom implementation will have to be evaluated at the end of the school year. However, as of the present time, both the students and I feel it has been successful. The students are engaged with their learning, and I am seeing quality and authentic requirements analysis and project management. Their written work toward the external topic thus far demonstrates a solid understanding of the principles. Furthermore, the visual nature of their SCRUM boards is intriguing to the other students in different year levels and hopefully will motivate the younger students to continue on with Digital Technology.



BIBLIOGRAPHY

Bell, T., & Morgan, J. (2014, Jan 23). *Software Engineering*. Retrieved March 1, 2014 from Computer Science Field Guide:

<http://www.cosc.canterbury.ac.nz/csfieldguide/teacherguide23012014/ComplexityTractability.html>

Cohn, M. (2014, NA NA). *User Stories*. Retrieved May 16, 2014 from Mountain Goat Software:

<http://www.mountaingoatsoftware.com/agile/user-stories>

Gauld, R. (2013, March 19). *Eight ways Novopay failed*. Retrieved June 1, 2014 from TVNZ:

<http://tvnz.co.nz/seven-sharp/eight-ways-novopay-failed-5373279>

Gibb, J. (2013, October 25). *Novopay 'failure' reduced risk hopes*. Retrieved June 1, 2014 from Otago Daily Times: <http://www.odt.co.nz/campus/university-otago/278467/novopay-failure-reduced-risk-hopes>

Johnson-Stempson, R. (2014, May 22). *Misconceptions Inhibit Talented Women from Tech Careers*.

Retrieved June 2, 2014 from Microsoft Citizenship Asia Pacific:

http://blogs.technet.com/b/microsoft_citizenship_asia_pacific/archive/2014/05/23/misconceptions-inhibit-talented-women-from-tech-careers.aspx

Miles, R., & Pilone, D. (2008). *Head First Software Development*. Sebastopol, CA, USS: O'Reilly Media.

Sussex, T. (2013, October 2). *Agile vs. Waterfall: Which Project Management Style Is Right for You? [Infographic]*. Retrieved June 13, 2014 from LiquidPlanner:

<http://www.liquidplanner.com/blog/agile-v-waterfall-which-project-management-style-is-right-for-you/>

