SOFTWARE SOLITIONS
SOFIA

Iterative Process Review

Review of the Agile Iterative Processes Used Within sofia

Document Control

Editor	Version	Date	Update
Alex Cash	1.0	02/06/2015	Created document
Alex Cash	1.1	02/06/2015	Updated titles

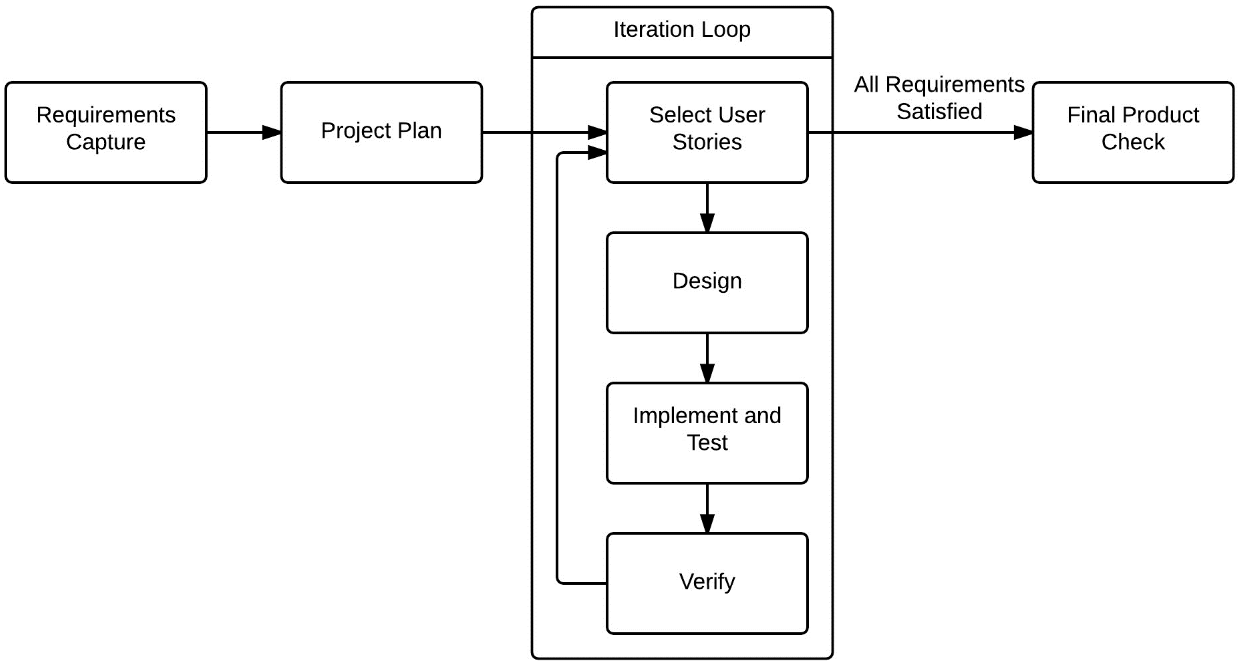
Table of Contents

Iterative Process Review

1.0 Introduction

1.1 Agile Processes Within sofia

As can be seen in the sofia QA Manual, we have operated using an agile iterative approach to software development. The process that was adopted, as shown in the QA Manual, uses the following system:



This system allows us to confirm at a number of points throughout the project that the work completed is both up to standard and fits with the customer's requirements. As can be seen above, the first step of the process was to capture the requirements of the customer, which were presented in the form of user stories, these shall be discussed later in this document. The next step was to create an overall project plan which can be seen in the Project Planning Review document. Following on from this, the development process entered the iteration loop, which was to be repeated a number of times until all customer requirements (user stories) were met. Finally, a check was carried out to ensure that all customer requirements were completely met. This check was performed on the product as a whole, rather than any subsection or module.

2.0 Customer Requirements As User Stories

2.1 Defining the Overall Requirement

One of the first steps in this project was to decide on what type of product the customer wanted. From this we could then analyse what would make the product a success and as such realise tangible tasks in the form of user stories.

The overall customer requirement we were working to was that the software product would be a worthwhile investment for the customer and thus be the basis of a successful business enterprise. The product would be a successful standalone product that customers would want. It was also required that the product should:

- Run in a Windows PC environment
- Be written in the Java programming language
- Display many forms of multimedia (image, graphic, text, audio, video)
- Be able to use online media sources
- Work by using the XML format

2.2 The Requirements

Taking the requirements above into consideration, the team came to the decision to create a suite of products based around the teaching industry. The suite is made up of the teacher product known as TeachEasy, and the student product known as LearnEasy. The full description of the requirements can be seen in the Functional Specification document.

From the Functional Specification document, the requirements for the TeachEasy program were summarised by:

> "As a teacher, I can create an interactive lesson for a student of mine to complete independently"

This can then be broken down into the following:

> "As a teacher, I can provide any number of my students with access to a lesson I created previously"
>
> "As a teacher, I can create a lesson comprised of a number of discrete pages, each of which I can customise"
>
> "As a teacher, I can pick a page category from a number of pre-defined templates (e.g. video, quiz, etc.)"
>
> "As a teacher, I can include multimedia objects in the lessons I create"
>
> "As a teacher, I can save lessons I am working on then access them again and edit them at a later date"
>
> "As a teacher, I can assign marks to exercises on each page, where necessary"

Also from the Functional Specification document, the requirements for the LearnEasy program were summarised by:

> "As a student, I can complete an interactive lesson created for me by my teacher, in my own time"

This can then be broken down into the following:

"As a student, I can view a lesson created by my teacher"

"As a student, I cannot edit a lesson"

"As a student, I can choose the lesson I want to work on from a selection of lessons provided to me by my teacher(s)"

"As a student, I can pause and resume lessons"

"As a student, I can view all forms of multimedia that my teacher has included in the lesson"

"As a student, I can interact with all suitable forms of multimedia (e.g. pause a video)"

"As a student, I am provided with a record detailing my lesson completion and level of achievement, which I can choose to print out"

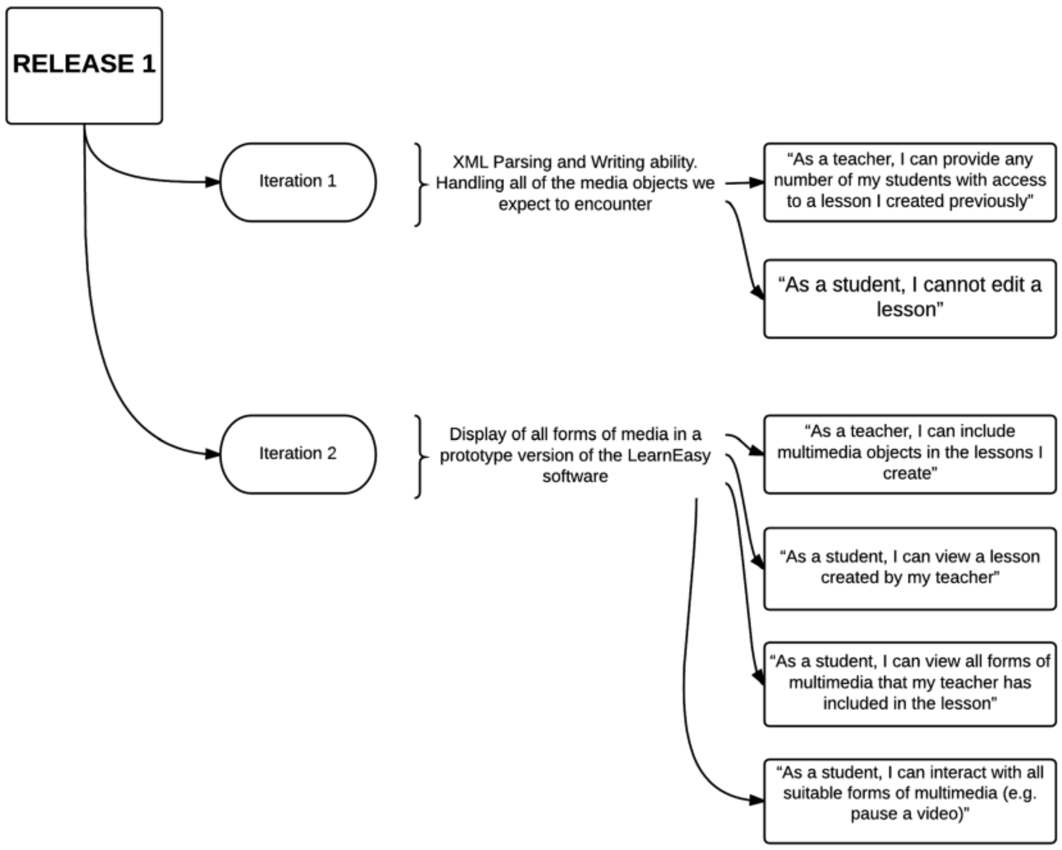
The user stories above were a good starting point for the development process as they clearly defined the requirements for the whole project without being overly constraining on how the final product should function or look. They were flexible enough to allow creative decisions to influence the implementation process and the addition of any features we felt were necessary or would improve the resultant products.

3.0 Iteration Planning

3.1 Creating the Iterations

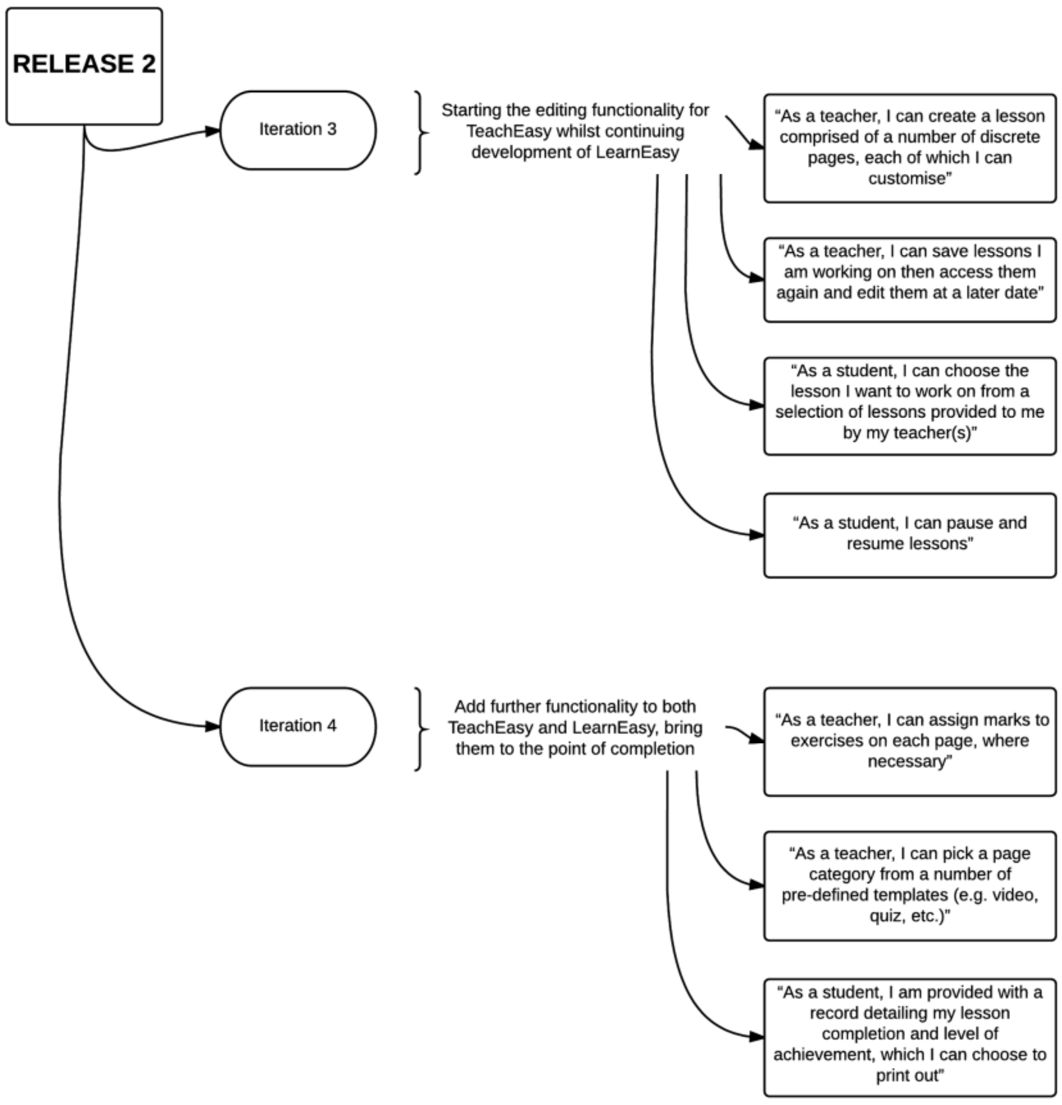
The iterations were divided up so as to focus heavily on first creating a working version of LearnEasy before focussing on the TeachEasy work. This was because it was decided that we would want the teacher to be able to preview a version of the lesson created in TeachEasy within the program. This would require a stripped down version of LearnEasy to be available before TeachEasy would work correctly. It would also mean that if something went terribly wrong, we would be able to support the minimum requirements of the customer (the product would open and correctly display XML files).

The customer required two distinct iterations to be used, however we decided as a team that we would like to use more. This would enable us to ensure we were meeting the requirements correctly and verify we were staying on the correct track. For this reason, we decided to divide the project into two releases made up of iterations, rather than just two iterations. The first release was to include two iterations, which were arranged as follows:


RELEASE 1
Iteration 1
XML Parsing and Writing ability. Handling all of the media objects we expect to encounter
"As a teacher, I can provide any number of my students with access to a lesson I created previously"
"As a student, I cannot edit a lesson"
Iteration 2
Display of all forms of media in a prototype version of the LearnEasy software
"As a teacher, I can include multimedia objects in the lessons I create"
"As a student, I can view a lesson created by my teacher"
"As a student, I can view all forms of multimedia that my teacher has included in the lesson"
"As a student, I can interact with all suitable forms of multimedia (e.g. pause a video)"

As you can see, the first iteration was designed to almost exclusively handle the backend XML reading and writing. This ensured we would have a strong basis to work from moving forward into the next iterations. The second iteration focussed on handling all of the required media types both within LearnEasy and in our XML reader and writer.

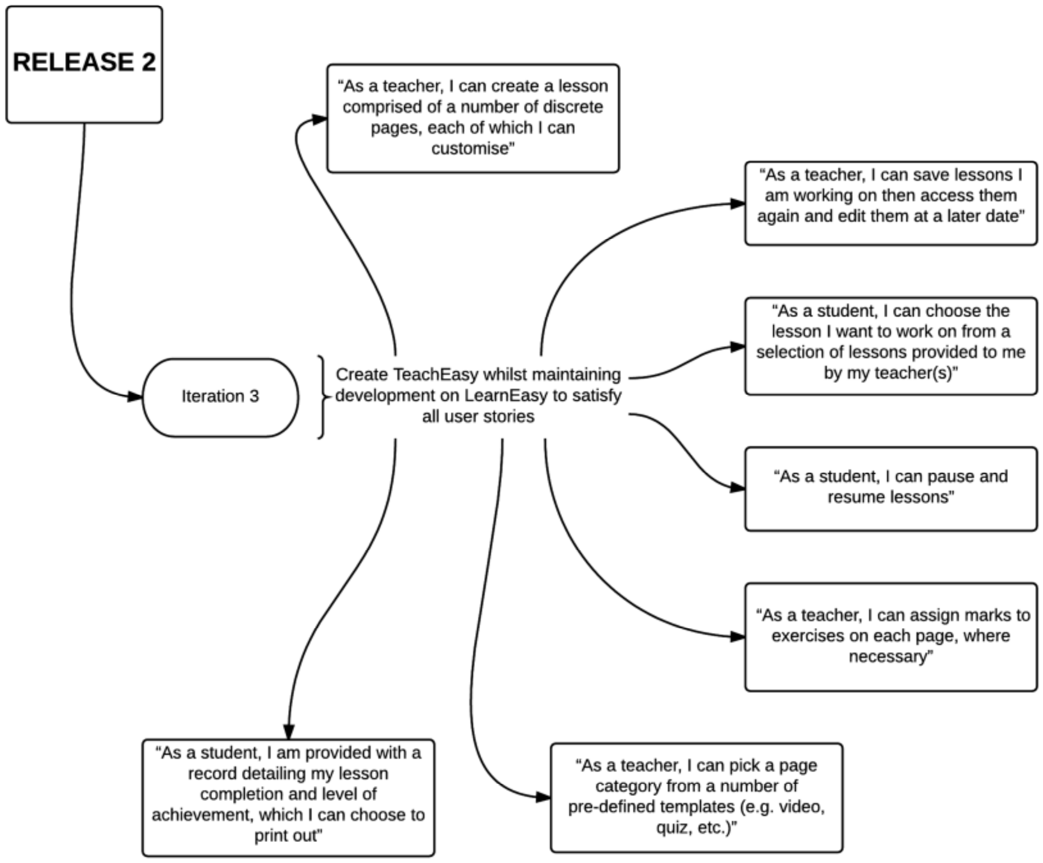
Initially, the second release was designed as follows:


RELEASE 2
Iteration 3
Starting the editing functionality for TeachEasy whilst continuing development of LearnEasy
"As a teacher, I can create a lesson comprised of a number of discrete pages, each of which I can customise"
"As a teacher, I can save lessons I am working on then access them again and edit them at a later date"
"As a student, I can choose the lesson I want to work on from a selection of lessons provided to me by my teacher(s)"
"As a student, I can pause and resume lessons"
Iteration 4
Add further functionality to both TeachEasy and LearnEasy, bring them to the point of completion
"As a teacher, I can assign marks to exercises on each page, where necessary"
"As a teacher, I can pick a page category from a number of pre-defined templates (e.g. video, quiz, etc.)"
"As a student, I am provided with a record detailing my lesson completion and level of achievement, which I can choose to print out"

The third iteration was intended to create the basis of the TeachEasy software. The plan was to implement a client that enabled the user to create a lesson comprising a number of pages that also handled the standard Windows file open and save methods. This required the XML writer (created in iteration one) part of our software to be operational and working in a way that produces XML files of the same format as the XML reader. Additionally, further improvements to LearnEasy were planned.

The fourth and final iteration was planned to add all of the editing functionality along with marking features in both LearnEasy and TeachEasy. This iteration also planned to add page templates to LearnEasy to make it even easier for the customer to use.

At the point that would have been considered the end of iteration three, it was decided that the features implemented did not amount to enough code to require a full iterative review and verification period. For this reason, it was necessary to combine iteration three and four into one larger iteration. Additionally, due to time constraints it was necessary to save time somewhere, and instead of removing intended features, reducing the number of iterations was a much more satisfactory solution. The revised iteration plan can be seen in the following diagram:


RELEASE 2
Iteration 3
Create TeachEasy whilst maintaining development on LearnEasy to satisfy all user stories
"As a teacher, I can create a lesson comprised of a number of discrete pages, each of which I can customise"
"As a teacher, I can save lessons I am working on then access them again and edit them at a later date"
"As a student, I can choose the lesson I want to work on from a selection of lessons provided to me by my teacher(s)"
"As a student, I can pause and resume lessons"
"As a teacher, I can assign marks to exercises on each page, where necessary"
"As a student, I am provided with a record detailing my lesson completion and level of achievement, which I can choose to print out"
"As a teacher, I can pick a page category from a number of pre-defined templates (e.g. video, quiz, etc.)"

Here we can see that all of the user stories for the third and fourth iteration have been moved to all be part of the third iteration. In addition to re-organising the iteration plan, it was decided to separate the team into two sub-teams, one focussing on TeachEasy and one focusing on improvements to LearnEasy. This enabled us to streamline our development efforts and have concurrent work ongoing. It also simplified the work for the two development managers as they had less staff to manage and provide work for. The team working on the TeachEasy client was roughly double the size of the team working on LearnEasy, which was decided on based on the amount of work remaining for each.

4.0 Iteration Review

4.1 Iteration One Review

Iteration one was a fantastic success, the XML reader and writer were both written and functioned as intended. They were later refactored due to changes in the XML format being used, but this was not an issue as they were initially written well and to the desired specification. The second user story which reads "As a student, I cannot edit a lesson" did not require implementation in code directly. This story was satisfied by not allowing any editing functionality in the LearnEasy client and as such was a result of the product design rather than implementation.

4.2 Iteration Two Review

Iteration two was also very successful. The implementation produced successful features that worked as intended, whilst the testing helped to stamp out early defects. The media handlers created worked as desired and two were sold (audio and video handlers) to another company as can be seen in our contract documentation. This iteration enabled us to handle all the desirable media (stored either online or locally) and display it correctly. At the end of this iteration, and release, we had the basics of the LearnEasy product in place and were able to demonstrate its functionality to the customer. All parties were happy with the progress and as such the iteration was verified successfully. The code was also tested very thoroughly throughout, ensuring confidence in the code for later work.

4.3 Iteration Three Review

Iteration three was by far the largest iteration and saw the completion of the products. The LearnEasy client was improved upon from its state after iteration two – both functionally and aesthetically. Mark tracking, achievement awards (certificates upon completion of lessons), the ability to choose which lesson to open from a home screen, and a new GUI were all implemented. Sadly it was not possible to implement a system for students to pause and then resume lessons at a later date. This was because our mark tracking system required the progress to be stored in runtime data and did not save the information upon closing the program. This would be a future feature to implement should the project progress. Other than this however, LearnEasy was completed very successfully and ended up meeting specifications correctly with no unintended behaviours. This was demonstrated to the customer who expressed satisfaction with the product, verifying the success of the work on LearnEasy.

The TeachEasy client also changed a huge amount during iteration three, most of the editing functionality was implemented along with quality-of-life changes such as the addition of page templates. The ability to click and drag objects around the page, delete using the DEL key, copy and paste using CTRL+C and CTRL+V, resize objects, add online media sources, assign marks to questions, provide questions in multiple formats (string or checkbox for example), add and remove pages, add page templates, save and reopen files, and other features were all implemented in this iteration. Because there was so much code being added it required extensive testing throughout the implementation and testing phase, which was completed well. Many tests failed and required fixes to be implemented; these can be seen in our bug fix and test report documents.
This whole iteration was demonstrated to the customer who verified that the work met expectations and thus approved the entire project as a success.

Splitting the team into two turned out to be a very good decision, as the work for the third iteration did not require much cross-development between TeachEasy and LearnEasy. It enabled the development managers to focus in on their tasks better and devote more time to coding rather than to management tasks. It proved very effective and resulted in two excellent products being produced.

5.0 Summary

Although the iteration plan had to be changed, the team adjusted well to continue with the work and maintain high quality output. The testing processes implemented worked well to ensure the code functioned as intended and integrations were smooth and successful. The iterative model we adopted worked very well to allow us to evaluate progress throughout the project and helped to keep us on track. Without the process we used, it is more than likely that we would have had numerous large problems to deal with, slowing progress and possibly causing us to miss the deadlines set.