

Ridwan Chowdhury 2012479

by Ridwan Mamun Chowdhury

FILE	25306_RIDWAN_MAMUN_CHOWDHURY_RIDWAN_CHOWDHURY_2012479_302864_34417349.DOCX (1.03M)		
TIME SUBMITTED	04-DEC-2020 03:07PM (UTC+0000)	WORD COUNT	3416
SUBMISSION ID	138343347	CHARACTER COUNT	18943

Evaluation of the Agile development practices and tools used at the DVLA

Ridwan Chowdhury (2012479)

Agile and Software Engineering Principles (ACCA7001)

Lecturer: Mike Darcey

University of Wales Trinity Saint David

Contents

- 1. Introduction
 - 1.1 History
 - 1.2 What is Agile software development?
- 2. Agile Development at the DVLA
 - 2.1 Scrum Framework
 - 2.2 User Stories
 - 2.3 Estimation
 - 2.4 Product Backlogs
 - 2.5 Sprint Backlogs
 - 2.6 Collaboration
- 3. Continuous Integration
 - 3.1 Testing Processes
- 4. References

1. Introduction

This report will look at how the DVLA incorporates Agile development practices, based on our experience as a trainee-engineer working on a curated project. The report will consider the methods used for recording User Stories, Estimation, Product Backlogs, Sprint Backlogs and Collaboration. The report will also reflect on the importance of continuous integration and testing processes, providing an overview of the technologies and techniques used at the DVLA.

1.1 History

The need for agile development arose from decades of technological advancement, and the difficulties organisations faced in adapting to market changes and their subsequent changing business requirements. In the early days of software, the lack of standardised development practices resulted in chaotic and unplanned approaches, this brought about the initial foundational theories and software engineering methodologies such as the Waterfall model [1]. The ideas behind these early practices took inspiration from physical engineering, an engineering discipline which has been known to humankind for many millennia. These earlier methodologies placed higher importance in processes and documentation over delivering working code. This proved to be a very rigid structure making it difficult and expensive for developers to go back and incorporate changes to their requirements once they have progressed through the development process. As a result, projects would be cancelled part way through as it would be too expensive to change requirements, or projects would be completed as per the original objective yet miss out the most current business requirements. During the 1990's when PCs started to become common amongst the general public and within workplaces, demand for software products grew immensely, and businesses were unable to react fast enough to meet customer demands, this is where the term 'application delivery lag' was coined; where the time to deliver working software after validating a business need would take on average, three years. Physical engineering projects such as bridges and buildings are designed to last many years, decades even, this is not the same with software projects where there are often changes in the external environment prompting innovation, thereby leading to improvement in computing power, development tools available, and the way in which projects are carried out [2].

1.2 What Is Agile Software Development?

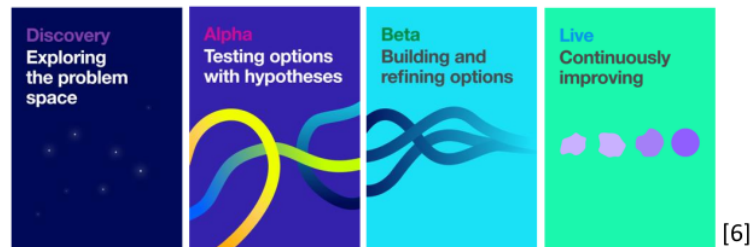
Agile software development refers to a set of frameworks and practices based on the principles laid out in the agile manifesto [3]. The manifesto includes 12 core principles [4] and it is up to self-organising cross-functional teams to incorporate these ideas and values as they best see fit. With Agile software development, teams prioritise continuous delivery of valuable software through iterative development. At each iteration, teams review their requirements, plans and results with feedback from users. Incremental development allows teams to focus on short-term goals and be able to test continuously as they progress, allowing for flexibility in the long-term goal of a project. In this way teams can maximise value by ensuring that every version of a product's release is usable and newer versions build on previous versions with the additions of new features. Essentially, software

development following agile principles is about working collaboratively, building in increments, working on smaller chunks and focusing on specific issues, whilst learning from feedback and adapting to change, this includes changing the way we work on a project.

2. Agile Development at the DVLA

Agile development at the DVLA can be broken down into four separate phases [5]:

- Discovery
- Alpha
- Beta
- Live



The Discovery phase [7] is there to gather information and establish whether a service is effective enough to meet a specific business or user need. At this stage, the goal is to understand what the problem is, what users are trying to achieve, what the constraints are, and whether there are any opportunities to improve services. The discovery phase lasts approximately 4 to 8 weeks.

The Alpha phase [8] lasts 6 to 8 weeks and builds on the knowledge gained during the discovery phase by focusing on creating a proof-of-concept. A proof of concept only requires implementation of some functionality of a service and provides an opportunity to experiment and explore new approaches. Prototype services are built iteratively and tested with users in order to understand the viability of a project going live. At the end of this phase there will be a clear understanding of exactly what is required in order to build a fully functional, end-to-end service, and whether it would be feasible to do so.

The Beta phase [9] uses the knowledge acquired during the Alpha phase to build a fully a functional end-to-end prototype, which is to be tested with the public. Feedback is gathered in order to improve the service and understand how users are using the service.

During the Live phase [10], the service is continually improved with iterative and incremental development based on the data gathered during the beta phase through user feedback and analytical data.

Over the past few weeks the MSc students have worked on a project at the DVLA in it's Alpha phase using the Scrum framework. At this stage we were required to build a proof of concept of a particular service addressing a business need, which was identified during the projects discovery phase.

2.1 Scrum Framework

According to a report by Digital Ai [5] where they surveyed over 40,000 Agile executives, practitioners and consultants, Scrum was by far the most widely practiced framework with

more than 75% of respondents having reported incorporating Scrum or a hybrid of Scrum in their Agile development process.

Scrum is based on two core principles; empiricism and lean thinking [11]. Empiricism refers to the importance of learning by doing and making decisions based on what you have learnt. Lean thinking highlights the importance of reducing waste by only focusing on matters that will provide immediate value to the user.

Scrum values are a determinant of whether the framework is being successfully used or not, and these are: commitment, focus, openness, respect and courage. As team members become confident in demonstrating these living values consistently, they will begin to see the benefits of Scrum through its three pillars: transparency, inspection and adaptation. Transparency ensures that the emergent work can be inspected accurately, and any issues or constraints are identified as early as possible. Low transparency can hinder the entire development process with misinformed decisions. Inspection is carried out frequently within Scrum events, this prompts developers to solve problems early on and be open to change. Adaptation refers to teams acting on problems they have identified during the inspections, Scrum accommodates adaptation through self-organising teams, allowing for teams to prioritise on their own account and be reactive.

Scrum is designed to optimise predictability and manage levels of risk with an iterative and incremental approach. Teams working with Scrum benefit from its highly collaborative nature allowing for groups of people to share and obtain the skills and knowledge required to do the work. Teams usually consist of around 6 to 9 people, it is important for teams to be organised in smaller numbers, this is to ensure that the scrum events are as effective as possible and each team member is able to participate and communicate with each other.

A Scrum team is made up of multiple Developers, a Product Owner and a Scrum Master. The Scrum Master is responsible for implementing Scrum within the team by organising and facilitating Scrum events, providing support when issues arise, and fostering and encouraging Scrum. Teams work in Sprints which usually last 2 to 4 weeks, each sprint contains 5 Scrum events. The events include: Sprint Planning, Daily Scrums, Sprint Review, and Sprint Retrospective. Sprint Planning is the first event of a Sprint, this is where team collaborate on making a plan for the Sprint. The team decide on a Sprint Goal, this is something that they could potentially present to stakeholders to demonstrate the value created at the end of a Sprint. To achieve the Sprint Goal the Product Owner and Developers choose User Stories from a Product Backlog to work on for the Sprint. Developers estimate each user story in terms of difficulty and time required, this allows teams to forecast their Sprint and break down User Stories into smaller sub-tasks.

During a Sprint, teams also partake in a Daily Scrum, this event gives Developers the opportunities to express any difficulties they may be facing, make quick decisions, identify blockers, and track progress towards the Sprint Goal. The Daily Scrum is timeboxed to 15 minutes and is only there for the team to highlight issues, any updates or discussions on specific topics are to be arranged outside of the Daily Scrum.



The Sprint Review takes place at the end of the Sprint, this is a working sessions where the Scrum Team present and discuss their work to the customer. The Sprint Review allows the customer to be involved throughout the development process, providing feedback and instructions on what to do next. The Scrum Team are also able to adjust their Sprint Backlog to meet new objectives.

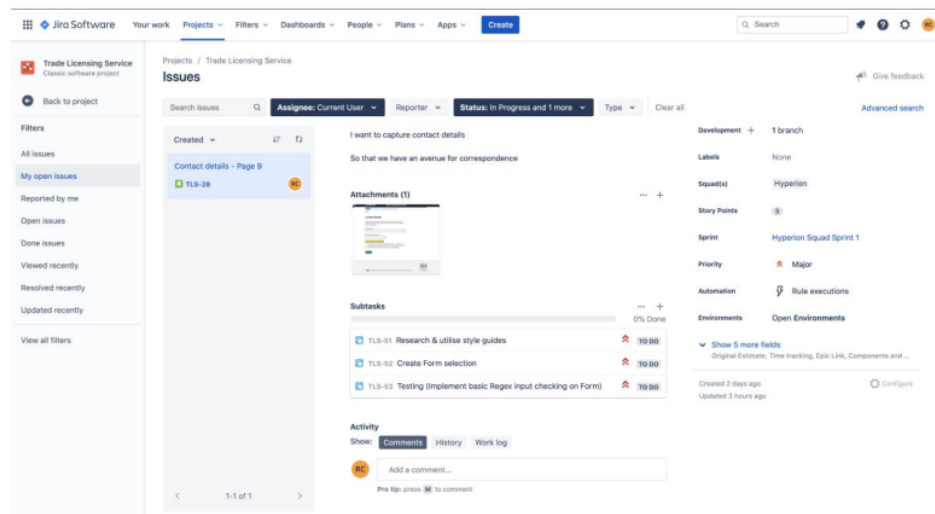
The Sprint is concluded with a Sprint Retrospective, where the Scrum Team reflect on issues they had faced, successes they have had, and what they could do to improve on for next Sprint. The Sprint Retrospective gives the Scrum Team an opportunity to share any new knowledge they have learnt over the course of the Sprint and come to a working agreement for future Sprints. Sprints are repeated until the business need is met or becomes obsolete.

Alternative popular Agile methodologies include: Kanban, and Lean. Kanban is designed to improve efficiency and increase workflow. Lean is designed to improve focus by reducing waste.

2.2 User Stories

Agile Methodology is all about prioritising the user's needs and this starts with User Stories. Developers build services around the description of a user and what they would like to do with a particular service. A user story is made up of: a description of the person using the service (e.g. 'as a business customer'), what the person needs the service to do (e.g. 'I want to renew my business trade plates online'), and the reason the person needs this service (e.g. 'So that my renewal can be processed quicker.'). User Stories are also created with a set of acceptance criteria that developers can check off to and confirm that the need has been met.

The below screenshot demonstrates how User Stories are used at the DVLA.



Projects are posted to Jira - an issue and project tracking software, and User Stories are assigned to developers, broken up into smaller sub-tasks if required and placed on a Sprint Backlog. The Sprint backlog contains 4 columns, starting with 'TO DO' developers move their assigned task to 'IN PROGRESS' once they start actively working on a task. Once a task has been completed it is moved over to 'DONE', and if there are any issues preventing a task from being completed then it can be placed in the 'BLOCKED' column to be highlighted in the Daily Scrum.

2.3 Estimation

Estimation is done by the Scrum Team by assigning a figure ranking the difficulty of a given task. In our project at the DVLA we estimated User Stories in our Sprint Planning session based on our cumulative understanding of a problem, we then would reveal a figure ranking its difficulty at the same time. The most commonly occurring figure is then selected as the estimation for a User Story.

2.4 Product Backlogs

The Product Backlog is owned by the Product Owner. The Product Owner is a person who represents the needs of stakeholders and is responsible for developing a Product Goal. The Product Owner is also responsible for the ordering of items on the Product Backlog, these are items which will eventually be actioned on in a Sprint and transferred to a Sprint Backlog which Developers will work on. Any items on a Product Backlog can only be removed or amended by the Product Owner, it is the Product Owners aim to maximise value from the work done by the Scrum Team. Items could include: User Stories, bugs, and spikes of knowledge acquisition.

2.5 Sprint Backlogs

Sprint Backlogs consist of items selected from a Product Backlog for a particular Sprint.  Sprint Backlogs belong to the Developers and they are committed to turning User Stories into incremental value. The Sprint Backlog acts as a forecast for the Sprint, giving stakeholders an idea of what features will be implemented in the service and how the service will look at the end of a Sprint. Improvements in processes highlighted in the previous Sprint Retrospectives are also included in the Sprint Backlog to encourage continuous improvement. As the Sprint progresses, the Sprint Backlog is modified by the Developer team until all items are considered done as per the Definition of Done set out by the DVLA, at this point the Sprint Goal would have been achieved.

2.6 Collaboration

Working Agile and collaboration goes hand in hand. Scrum cannot be implemented without a team that trusts each other and commit to doing the work. For our project at the DVLA, we were placed in teams of 6 and were required to achieve our Sprint Goal through collaborative work. Collaboration allowed us to achieve our Sprint Goal through the

cumulative effort and knowledge of our team. We used several methods to share knowledge and collaborate on pieces of work, such as using version control, pair programming, mob programming, and daily meetings.

The DVLA uses Bitbucket – a Git based code repository, designed to allow seamless collaboration on development projects. Git allows multiple members of a team to work on a project together, it works by hosting code on a remote repository and tracking its contents. Developers can clone the repository to their local machines and create branches of a master branch. Git keeps track of changes made to the source code and allows you to see what changes have been made by other developers working in the same repository. Developers are then able to commit, push and merge their changes back into the master branch without affecting any code written by other developers. Version control also allows developers to identify merge conflicts and fix breaking code by reverting commits.

We also used Pair Programming and mob programming as a means to collaborate. Pair programming is an Agile software development technique where two people work together on the same code at the same time using the same machine [13]. During the pair programming session one person takes control of peripherals and writes code whilst the other guides and inspects for mistakes as they are made. Developers can switch roles to keep productivity up and help drive motivation. Pair programming also produces a sense of shared responsibility and discipline, where developers would need to avoid distractions and not take long or unnecessary breaks. However, Pair Programming can also be seen as doubling costs as it would now take double the man power to complete a task, although costs can be mitigated in the long run if the session is successful in preventing bugs, increasing code quality, and upskilling developers within the team.

Mob Programming is another Agile software development technique we used at the DVLA during our Sprint. In Mob Programming the team work together on the same code at the same time [14]. This allows the team to benefit from the cumulative knowledge and skills available amongst each other. If there are skill gaps in the team, the more experienced developers are able to take lead whilst the lesser-able team members inspect for mistakes, share their approaches to problems, and learn from the rest of team. Mob Programming ensures team members, of all skill levels are able to follow on at the same time and keep on track with the rest of the team. Having the entire team present allows you to make decisions and discuss important matters quickly.

3. Continuous Integration

Continuous integration is a process which automates the testing and integration of code from multiple developers with a shared repository [15]. The purpose of CI is to improve code quality through automating tasks, and identifying problems in the code early on, thereby preventing larger more expensive problems to build up later on in the project, and increasing velocity. In our case at the DVLA we used Drone for continuous integration, which works by triggering an automated workflow on a server hosted by Drone, once a developer commits changes to the source code on Bitbucket. Drone runs a series of commands based on what was specified in the CI pipeline. The below screenshot shows the steps taken within Drone when committing code.

Repositories → SDS/trade-licensing-internal-ui → #114

trade-licensing-internal-ui

[VIEW SOURCE](#) [RESTART](#)

← ACTIVITY FEED

✓ #114. fix authenticity error
ridwan.chowdhury_dvla.gov.uk pushed f2b82e7e to master 08:34 · 2 days ago

✓ default	08:34
✓ clone	00:01
✓ verify	02:25
✓ publish	06:08

```
default - clone 00:01
1 Initialized empty Git repository in /drone/src/.git/
2 + git fetch origin +refs/heads/master:
3 From https://bitbucket.tooling.dvla.gov.uk/scm/sds/trade-licensing-internal-ui
4 * branch      master       -> FETCH_HEAD
5 * [new branch] master       -> origin/master
6 + git checkout f2b82e7e92a446729baa81a8b562be62f5457b5c -b master
7 Already on 'master'
```

Depending on the configuration of the CI pipeline the steps taken can vary. In our project, Drone was configured with three stages: clone, verify, and publish. Clone, simply initializes an empty Git repository, fetches code from Bitbucket, and then switches to the master branch. Verify, fetches and installs the required dependencies to build the project and then runs automated tests to verify the installation has completed successfully. Finally, publish creates a new Docker container containing the Docker Engine and applications to build. If the commit passes all stages of the CI pipeline then it is marked as a successful build and can be pulled into the master branch. The CI pipeline can also contain extra steps which were not configured in our project, such as performance testing, unit testing, functional tests and so forth. Continuous delivery can also be implemented after this point where the code is then directly deployed as a live service, this is to improve velocity.

3.1 Testing processes

The DVLA applies testing at multiple points of the development process in order to identify problems early and minimise mistakes. Whilst developers write code, they also write unit tests, functional tests, and performance tests. Unit tests are automated tests that run on a unit of code to ensure that it behaves as expected. The benefit of using Unit Testing is that tests are run automatically, so if a test against a unit of code has passed but then future works somehow breaks the code, the developer can quickly identify what broke the code. Performance testing is another process the DVLA use which focuses on the speed, responsiveness, and stability of a service. Functional testing is when functions of code are tested to ensure that the function works as expected. Besides tests written by developers themselves, the practices I outlined in previous paragraphs such as, Continuous Integration, and Git, also ensure that the quality and stability of a service is maintained throughout the development process.

4. References

- [1] P. Varhol, "The complete history of agile software development", TechBeacon, 2020. [Online]. Available: <https://techbeacon.com/app-dev-testing/agility-beyond-history-legacy-agile-development>. [Accessed: 26- Nov- 2020].
- [2] Keegan, Anne & Turner, Rodney. (2002). The Management of Innovation in Project-Based Firms. Long Range Planning. 35. 367-388. 10.1016/S0024-6301(02)00069-9.
- [3] "What is Agile Software Development?", Agile Alliance, 2020. [Online]. Available: <https://www.agilealliance.org/agile101/>. [Accessed: 29- Nov- 2020].
- [3] "Manifesto for Agile Software Development", Agilemanifesto.org, 2020. [Online]. Available: <https://agilemanifesto.org/>. [Accessed: 29- Nov- 2020].
- [4] "AGILE: Understanding what agile project management is and how it helps to deliver projects and services | Digital passport", Engage.dhsc.gov.uk, 2020. [Online]. Available: <https://engage.dhsc.gov.uk/digitalpassport/agile/>. [Accessed: 30- Nov- 2020].
- [5] "14th Annual State of Agile Report", Digital Ai, 2020.
- [6] Government Digital Service (GDS), Stages of service design posters. 2017.
- [7] "How the discovery phase works", GOV.UK. [Online]. Available: <https://www.gov.uk/service-manual/agile-delivery/how-the-discovery-phase-works>. [Accessed: 01- Dec- 2020].
- [8] "How the alpha phase works", GOV.UK. [Online]. Available: <https://www.gov.uk/service-manual/agile-delivery/how-the-alpha-phase-works>. [Accessed: 01- Dec- 2020].
- [9] "How the beta phase works", GOV.UK. [Online]. Available: <https://www.gov.uk/service-manual/agile-delivery/how-the-beta-phase-works>. [Accessed: 01- Dec- 2020].
- [10] "How the live phase works", GOV.UK. [Online]. Available: <https://www.gov.uk/service-manual/agile-delivery/how-the-live-phase-works>. [Accessed: 01- Dec- 2020].
- [11] K. Schwaber and J. Sutherland, "Scrum Guide | Scrum Guides", Scrumguides.org, 2020. [Online]. Available: <https://www.scrumguides.org/scrum-guide.html>. [Accessed: 30- Nov- 2020].
- [12] "What is a Sprint Backlog?", Scrum.org. [Online]. Available: <https://www.scrum.org/resources/what-is-a-sprint-backlog>. [Accessed: 03- Dec- 2020].

[13] "Pair Programming: Does It Really Work?", Agile Alliance. [Online]. Available: [https://www.agilealliance.org/glossary/pairing/#q=~\(infinite~false~filters~\(postType~\('~page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video\)~tags~\('~pair*20programming\)\)~searchTerm~'~sort~false~sortDirection~'asc~page~1\).](https://www.agilealliance.org/glossary/pairing/#q=~(infinite~false~filters~(postType~('~page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~('~pair*20programming))~searchTerm~'~sort~false~sortDirection~'asc~page~1).) [Accessed: 03- Dec- 2020].

[14] M. Hammarberg, "Remote Mob Programming", CodeBetter. [Online]. Available: <https://www.remotemobprogramming.org/>. [Accessed: 03- Dec- 2020].

[15] "What is Continuous Integration | Atlassian", Atlassian. [Online]. Available: <https://www.atlassian.com/continuous-delivery/continuous-integration>. [Accessed: 04- Dec- 2020].

FINAL GRADE

57 / 100

GENERAL COMMENTS

Instructor

Agile Process Evaluation (40): 26

Examples (30): 16

Testing (30): 15

Discussion:

I wanted more reflection on the use of agile within the DVLA.

More examples should have been provided to illustrate the process.

Clearly some work has been done in testing, but the evaluation presented in this report is very limited.

More references required at various points throughout the report.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

 **Comment 1**
references

 **Comment 2**
references

 **Comment 3**
references