

Agile Assignment 1

by Rhys Vaughan Beckett

FILE	25296_RHYS_VAUGHAN_BECKETT_AGILE_ASSIGNMENT_1_302864_314686003.PDF (1.51M)		
TIME SUBMITTED	04-DEC-2020 04:07PM (UTC+0000)	WORD COUNT	3251
SUBMISSION ID	138354593	CHARACTER COUNT	19448

Agile Software Engineering Principles

ASSINGMENT 1

RHYS BECKETT

Table of Contents

Introduction	1
DVLA Agile Development Tools & Practises.....	1
User Stories	1
Estimation	3
Product Backlogs	4
Sprint Backlogs	6
Collaboration	7
Reflection	8
Continuous Integration/Development	8
Testing Processes	9
Summary	10
Reference:	11

Introduction

This report aims to detail and evaluate the agile development practises and tools utilised by the DVLA. It will consider the methods utilised for recording user stories, estimations, product backlog, sprint backlog and collaboration; providing examples and illustrations for each of these agile processes. Once these processes have been detailed, the report will also reflect on the importance of continuous integration and testing processes; providing an overview of the technologies and techniques utilised by the DVLA.

DVLA Agile Development Tools & Practises

User Stories

The DVLA utilise a software called Jira for most of their agile software development management tools. Jira was originally a product developed by Atlassian as a bug and issue tracker, over time it has evolved into a powerful management tool offering a range of useful functionalities. Including the tracking of user stories:

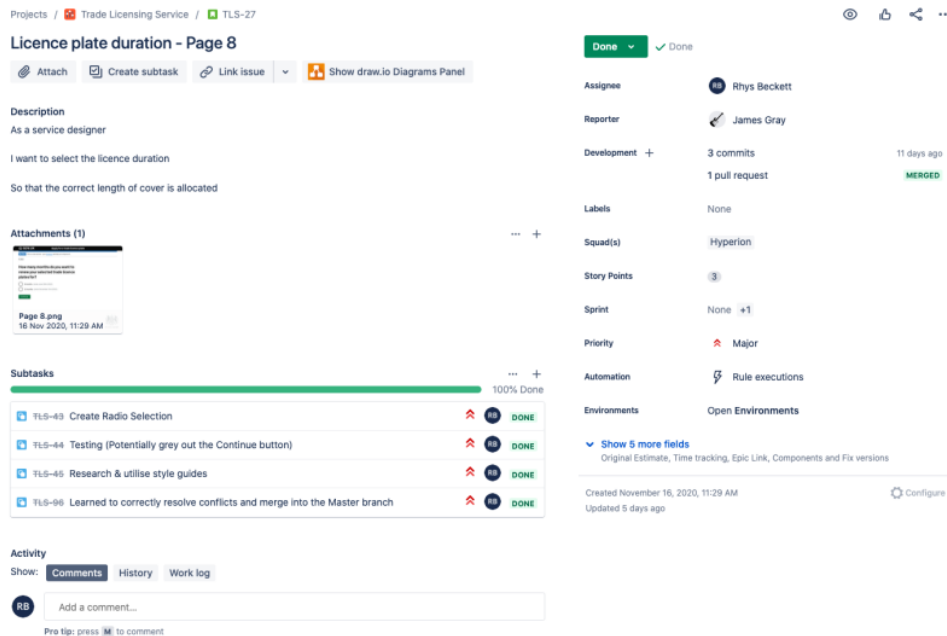


Figure 1 - DVLA User Story

The above user story was assigned to myself, in this example the user story was framed from the perspective of the ‘service designer’. The service designer is a role within the DVLA, they must design and deliver end-to-end, cross platform user experience of one or more services or parts of a service. They often act as product owners in the agile process, they will attend important ceremonies such as the planning, reviews, refinements and 3 amigos. The service designer is an elegant solution to the lack of traditional product owner within the public sector.

Jira offers a diverse set of functionalities regarding user stories. It allows for the assignment of squad & individual, attachment of documents, recording of estimation, priority & known issues, creation of sub-tasks and offers the ability to connect to an online repository.

Jeff Patton, the author of User Story Mapping: Discover the Whole Story, Build the Right Product said “Shared documents aren’t shared understanding.”¹ He further explains, that an individual’s interpretation of a written document can vary drastically from the intentions of the writer. Jira attempts to resolve this issue by providing all of these added tools and functionalities, they facilitate the growth of a shared understanding amongst the team. Being able to compare the given attachments, sub-tasks and story points provide a deeper understanding, that might even challenge previously held assumptions.

Due to the fact that we were unable to host 3 amigos ceremonies, the user stories are potentially not as refined than they could be. Most evident was the lack of input from a tester and the subsequent missing acceptance criteria. Without acceptance criteria, meaningful testing would have an awkward challenge to overcome. By definition alone, this means these stories fail the Testable characteristic of the INVEST mnemonic, since we couldn’t write a test for them.²

Fortunately, the lack of acceptance criteria did not affect our development process. I believe that this is in-part due to the emphasis the DVLA puts on extensive illustration. The UX designs attached to each user story proved enlightening, as is commonly said “A picture is worth a thousand words.” By providing these illustrations our shared understanding was immediately strengthened, achieving the same results through simple verbalisation would have taken significantly longer, wasting time and effort.

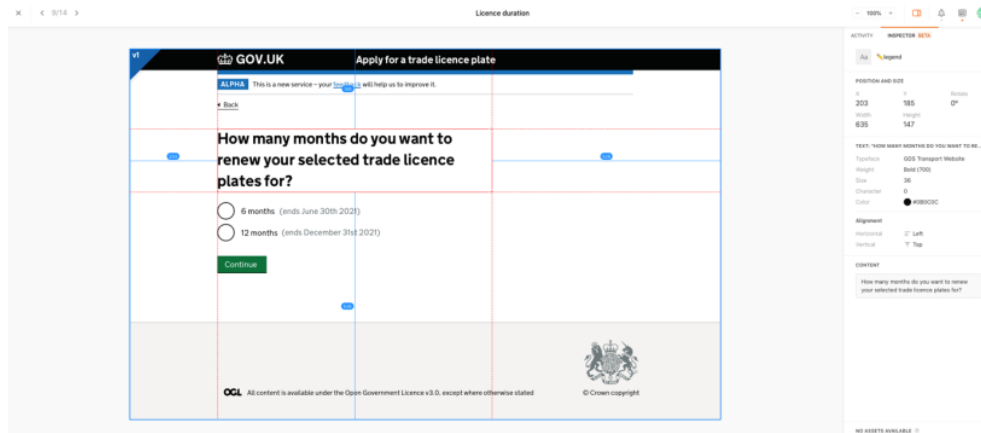


Figure 2 - Trade Licensing Service UX Sketch

Estimation

During the course of the project, we were invited to attend the Blazing Squad sprint planning session. During this session they discussed the availability of their individual developers and how this would affect the upcoming sprint. On average, they completed around 60 story points per sprint; however, due to multiple training courses occurring within the next sprint they agreed to lower the predicted completion rate to about 50 story points.

After that, the agile coach led the squad through the product backlog to estimate new user stories that had been added since the previous sprint. Once a user story was isolated and briefly discussed, the squad would estimate the story points. Blazing Squad choose to utilise the chat system available in Microsoft Teams to facilitate the planning poker method, the coach would count down from 3 and the squad would enter their Fibonacci value. Any inconsistencies were discussed until a consensus was reached.

This entire process was completed relatively quickly. I believe this is due to the maturity of the squad, their collaborative experiences provided a stronger, more immediately unified shared understanding than was evident during our estimation processes. This was most evident in the precision of their poker values, they were consistently similar; meaning very little time was needed to discuss differing opinions.

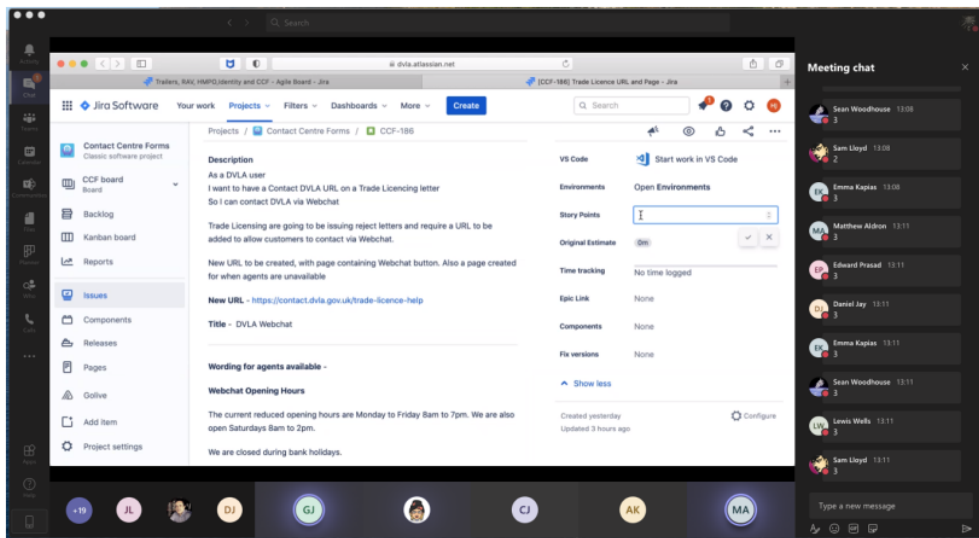


Figure 3 - Blazing Squad Planning Poker Estimations

Predicting the available capacity of the squad before the sprint proper seems to be a beneficial process, ensuring the squad is not overly burdened by the sprint backlog or left with too little to complete. Such a process is only possible with a mature squad that has multiple previous sprints to reference. This was evident in our first planning session as we underestimated the capacity/ability of the squad and needed an impromptu refinement session mid-sprint.

The estimation process utilised by Blazing Squad is similar to the method described by Mike Cohn in his book *User Stories Applied*.³ By obscuring the individual's opinion until the collective reveal, this vein of estimation methodology attempts to ensure that one individual's bias doesn't overly affect the thoughts of the whole. Following this process allows for a more diverse range of opinions as each individual considers the problem uniquely.

Though I am unaware of Blazing Squads full sprint cycle, I would suggest that the estimation process should be kept within refinement sessions. In this circumstance, they were so efficient that the overall sprint planning session was hardly affected. However, our cohort were not so fortunate, given our limited timeframe we also had to complete our estimation process within the sprint planning session. This was not ideal, since we most of the given time discussing varied opinions rather than the goal ahead.

Product Backlogs

Just as the user stories are hosted on Jira, so too are the product backlogs. The DVLA create a new backlog for each project, including our Trade Licensing Service project. These Jira backlogs provide the functionality necessary to fabricate multiple new user stories and sprints simultaneously. Since the cohort was divided into two squads, the ability to host multiple sprints from the same product backlog proved invaluable. Being able to visually confirm the user story distribution ensured a minimum of conflict and improved efficiency.

During the first sprint the squads underestimated their capabilities; meaning more user stories needed to be brought into the active sprint. This process was incredibly smooth, only requiring a short refinement session to discuss the distribution of the new tasks. Technically, Jira was even easier; simply requiring an individual to drag the chosen story into the sprint.

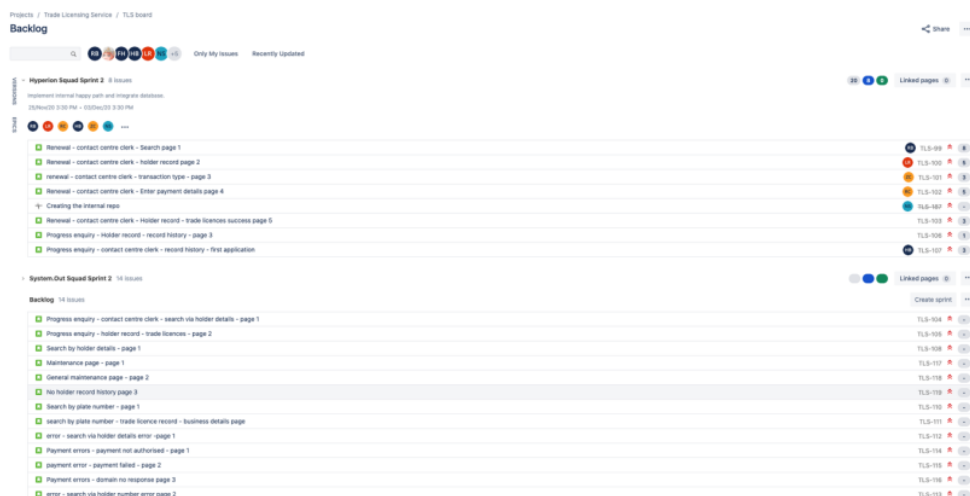


Figure 4 - Jira Product Backlog

In his book *Agile Product Management with Scrum*, Roman Pichler describes the four DEEP qualities that any good product backlog should ascribe to.⁴ Following the first criteria: the product backlog was not detailed appropriately, he quotes “The lower the priority, the less detail, until you can barely make out the backlog item,” from Schwaber and Beedle.⁵ The fact our product backlog was evenly documented goes against this mentality. In reality, this limitation was due to the limited scale and timeframe of the project, exploring the backlogs of larger projects confirmed this assumption.

As previously described in the estimation process, each and every user story is briefly discussed and estimated within either the refinement or planning ceremonies. The usefulness of this quality was made evident during the Blazing Squad sprint planning session as they were comfortably able to predict their upcoming capacity and build the next sprint backlog accordingly.

Given the flexibility of the DVLA process and Jira technology, the product backlog was incredibly emergent. As our skills and knowledge developed, the backlog evolved accordingly. This organic mutability ensured we remained efficient and effective, since each individual could reflect their capacity uniquely.

Jira always offered the ability to assign priority to any user story. Unfortunately, our squads did not initially take advantage of this feature since it seemed every user story was equally important. However, we learnt and attempted to record the priority for the second sprint, ensuring our shared understanding of the sprint goal remained more consistent.

Sprint Backlogs

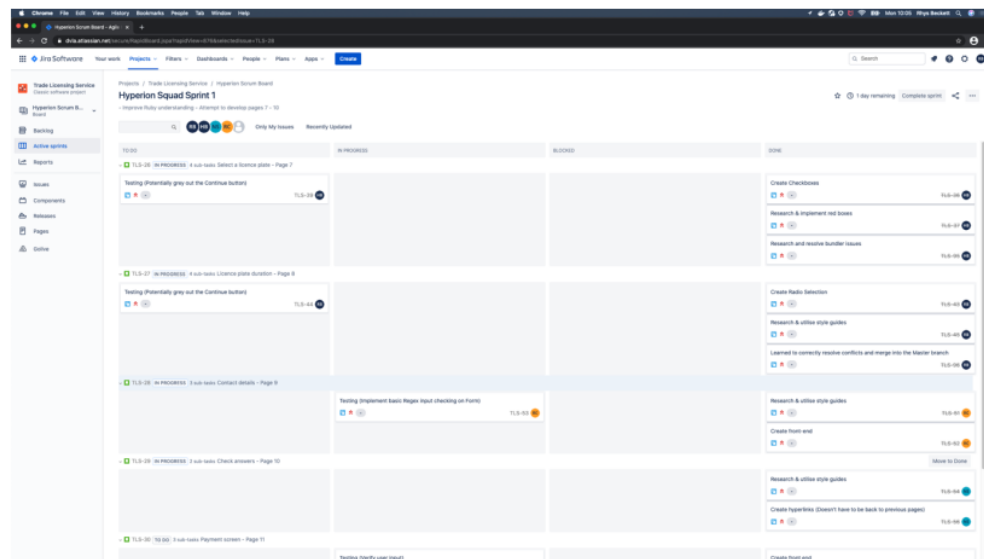


Figure 5 - Hyperion Squad Sprint Backlog

Creating a new sprint within Jira begins from the product backlog, during the sprint planning ceremony the scrum master will initiate the process. For our ceremonies the scrum master shared their screen for all the squad members to witness the procedure. Once initialised the squad discussed which estimated user stories to accept into the sprint, whilst attempting to consider the estimated capacity of the squad. After the user stories were chosen, the squad would agree on the sprint goal and officially activate the sprint.

Once a squad's sprint backlog has been populated, the squad has access to the scrum board, this scrum board is similar to a Kanban board. It is a page divided into 4 columns: To Do, In Progress, Blocked & Done. As the scrum board is generated, all the user stories added to the new sprint backlog are populated in the To Do column. Developers can select a story, then move them into any chosen column.

If the user stories within a sprint backlog are correctly formatted Jira can utilise in-built reporting tools to produce a variety of graphical visualisations of the sprint progress, these include Burndown charts, Cumulative Flow diagrams, Pie charts & more.

As described in the Scrum Guide, a sprint backlog should be composed of three main components; the sprint goal(why), the user stories(what) and the actionable plan(how). Based on the previous project, Jira offers strong functionality to answer the questions of why and what, but unfortunately how is left unexplained. In retrospect, the vague plan was probably a source of confusion and hesitation amongst the squads. Though the DVLA offered higher-level concepts for the larger potential systems, there was no actionable plan provided for the given sprints. Again, this may have been a limitation of the timeframe and lack of 3 Amigos ceremonies were such discussions may have been encouraged.

Sutherlands & Schwaber further explain “The Sprint Backlog is a plan by and for the Developers.”⁶ This ethos is directly adopted by the DVLA. Though multiple individuals were able to modify and contribute to the product backlog, the sprint backlog was created, maintained and closed by the developers; from planning to retrospective, no outside individual affected the sprint backlog. This was also true of the reporting tools available through Jira, the supporting DVLA staff repeatedly emphasised that those graphs were purely for the developers and were not utilised by the management.

Collaboration

The DVLA actively utilise most of the established collaborative agile ceremonies, during the second sprint I attended an Aztec Squad retrospective. For this process they utilise a web client called GoReflect to review their past performance. GoReflect offers a board of cards divided into three columns, squad members are encouraged to offer their thoughts by creating cards in the relevant column, usually the columns are entitled ‘What didn’t go well’, ‘What went well’ & ‘What could be improved’. Aztec have altered these to simply ‘Mad’, ‘Sad’ & ‘Glad’. Separating the negative column into both ‘Mad’ and ‘Sad’ allows more description/granularity, however it is at the cost of the improvement column, which can also encourage valid feedback of a different nature.

The retrospective began by checking the actionable items from the previous sprint’s retrospective, confirming if they’ve been completed and their potential results. They then returned to the current sprint reflection board, where the cards written by other squad members are concealed by default. This ensures each individual expresses themselves fully without bias. Once revealed the scrum master goes through each card, discussing the opinions of the original author and any comments raised by other squad members. After every card has been discussed the squad members will receive three votes to allocate importance, these are tallied, and the top three cards are considered for future resolutions/actions and brought into the next sprint.

In his book Agile Software Development, Alistair Cockburn describes ‘Shu-Hai-Ri’ in an agile context.⁷ Shu-Hai-Ri is a mentality utilised in Japanese Aikido, it describes the three stages of knowledge an individual should experience as they learn and grow. Aztec Squad’s adaption of the retrospective reminds me of the Ri stage, given their practitioner-level expertise they must adapt and test new methodologies to expand their understanding. The modified heading suited their relaxed atmosphere, by matching the mentality of the squad it encourages more engagement and collaboration.

Outside specific ceremonies, our squads repeatedly held collaborative programming sessions; specifically paired programming and mob sessions. These were encouraged by the DVLA and facilitated by utilising Microsoft Teams. In retrospect, this collaboration was vital for the success of the project. These sessions ensured a stronger shared understanding, improved relations between squad members and developed the skills & knowledge of all involved individuals.

Reflection

Continuous Integration/Development

As Kent Beck describes, continuous integration is a methodology designed to reduce the previously crippling integration issues surrounding collaborative software development [8]. Fundamentally, continuous integration is a solution that attempts to utilise the best aspects of both single programmer development and instantaneous integration. It attempts this by utilising an integration model that encourages hourly integrations. Such models like these require powerful integration platforms to reduce the time spent integrating, building and testing the new code. Fortunately, these platforms are becoming more available and affordable as the technologies improve.

Instead of more widely recognised GitHub, the DVLA utilise Bitbucket by Atlassian as their core integration platform. This decision allows them to take full advantage of the integration between Jira and Bitbucket. When these two systems are correctly linked, a developer can create a new branch from a user story. This branch will be within the relevant repository and correctly formatted to DVLA conventions. By streamlining this process, they encourage developers to create smaller branches and merge more frequently. As Martin Fowler describes in his article Patterns for Managing Source Code Branches, the DVLA encourages the 'Feature Branching' integration pattern [9]. This methodology offers the best balance between collaboration and isolation. Though developers must collaborate to succeed, they require isolation to produce functional code. By isolating each individual feature into a branch, ensures a developer can focus on their given task/user story without too much interruption; yet if necessary, the developer can pull from the main branch to remain up to date as needed.

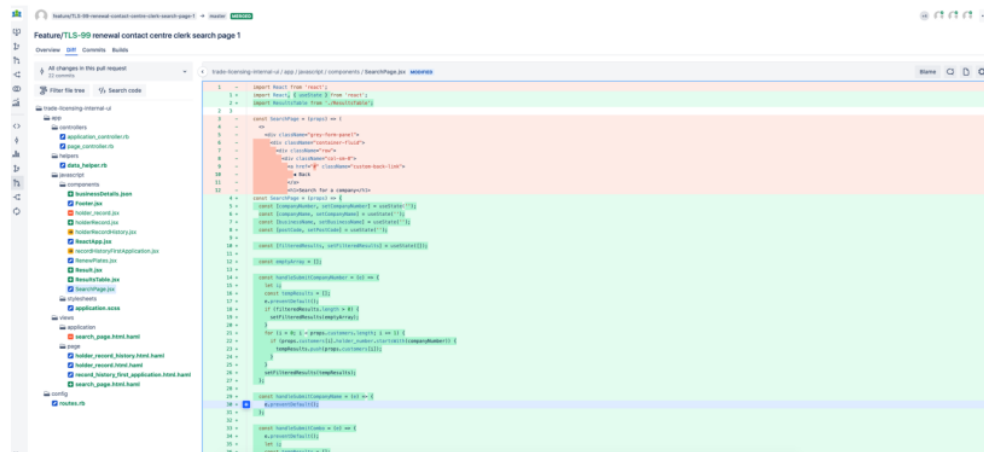


Figure 6 - BitBucket Pull Request

Once a feature is ready, the developer will attempt to merge their feature branch into the master branch. To initialise this process a developer must create a new pull request. Pull requests have to be approved by at least two other developers from the squad, this is an attempt to limit the erroneous updates made to the master branch. Bitbucket allows for commenting/discussion between individuals before approval, face-to-face discussion is

preferable however given the current circumstances surrounding COVID-19 this is an incredibly useful substitute.

Testing Processes

Frequent, robust testing is vital to a functional continuous integration environment. Every effort should be made to reduce the chance that any trivial errors are injected into the master branch. In an attempt to ensure this, the DVLA utilise a service known as Drone CI. Drone is a continuous integration platform that is primarily utilised to build, and host testing pipelines which are attached to a chosen source code repository.

As code is added to the repository it is automatically served to Drone. When Drone receives the new code, it will feed it through the relevant pipeline. These pipelines are setup by the organisation, they are basically a list of tests that need to be passed for the code to successfully build.

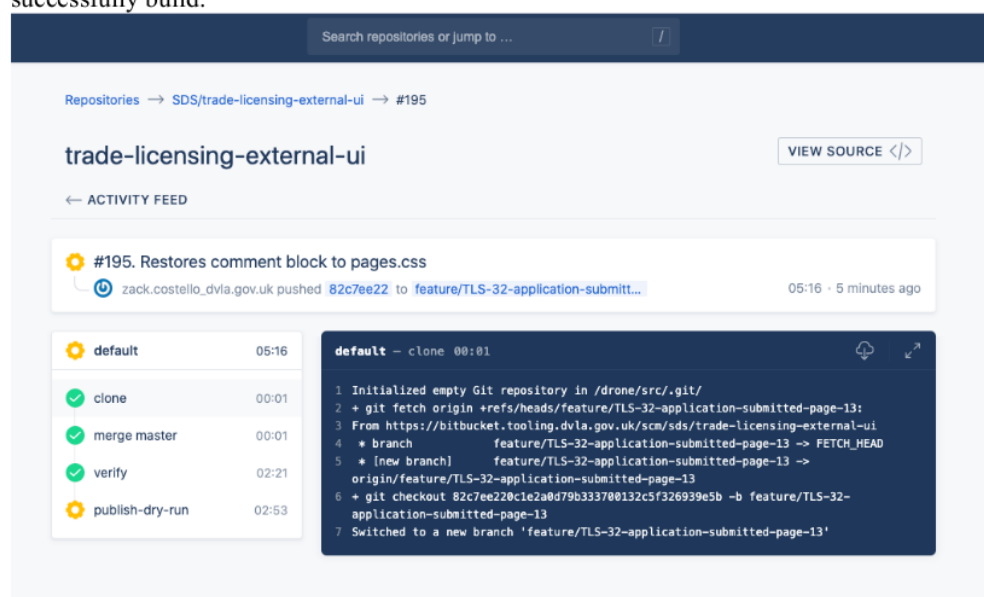


Figure 7 - Drone Continuous Testing

Firstly, it will attempt to build Docker containers against the instructions provided within the project files. This step ensures that the code is functional regardless of the environment. Once the containers are produced and the environment is stable, Drone will then begin to test against the code itself. These tests include some that are chosen to confirm that the developed code matches the DVLA coding standards, including: Rubocop, StyleLint and ESLint.

The DVLA utilise Behaviour-Driven Development to guide the software development process. As explained by Dan North “Programmers wanted to know where to start, what to test and what not to test, how much to test in one go, what to call their tests, and how to understand why a test fails.” [10] BDD is an attempt to answers this situation concisely by humanising the testing language and utilising real sentences. Expressive language helps the developer understand when a test is burdensome, unnecessary or erroneous. To produce and handle these tests, the DVLA utilise tools like Cucumber and RSpec.

Unfortunately, the timeframe given for the Trade Licensing Project meant testing was reduced to an absolute minimum. Without the initial acceptance criteria, producing test to code against was impractical. Only the automated formatting tests were actually implemented since those pipelines were pre-generated within the project. Though the DVLA seem to have implemented strong testing tools and processes, some of them were lacking during this particular project.

Summary

In conclusion, the DVLA utilise a wide range of processes and tools to facilitate the agile software development principles. Every project has a unique environment with multiple variables, including different squads, technologies and timeframes. Just as the projects differ, so too does the solution. The DVLA attempts to apply the best solution to each given scenario. Some squads limit the size of acceptable user stories, others prefer larger stories that can be broken into smaller acceptance criteria.

Overall, I believe that these solutions are very successful, with perhaps only a few small implementation issues as the agile solution does not match the project perfectly. As was evident in our Trade Licensing Service project, some ceremonies were disregarded due to the given timeframe and our inexperience. Though I would have preferred to witness those ceremonies I believe that this solution was best for the given environment. Hopefully as the DVLA continues to develop and learn, their agile solutions will become even more accurate and effective.

Reference:

1. Patton, Jeff; Economy, Peter. User Story Mapping: Discover the Whole Story, Build the Right Product. O'Reilly Media, Inc, 2014.
2. Wake, Bill. INVEST in Good Stories, and SMART Tasks. 2003. <https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/> [30 November 2020]
3. Cohn, Mike. User Stories Applied: For Agile Software Development. Addison-Wesley Professional, 2004 p. 88.
4. Pichler, Roman. Agile Product Management with Scrum: Creating Products that Customers Love. Addison-Wesley Professional, 2010, p. 48.
5. Schwaber, Ken; Beedle, Mike. Agile Software Development with Scrum. Prentice Hall, 2002.
6. Sutherland, Jeff; Schwaber, Ken. Scrum Guide. 2020. <https://www.scrumguides.org/scrum-guide.html> [01 December 2020]
7. Cockburn, Alistair. Agile Software Development: The Cooperative Game. Pearson Education, 2006.
8. Beck, Kent. Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, 200, p. 97.
9. Fowler, Martin. Patterns for Managing Source Code Branches. 2020. <https://martinfowler.com/articles/branching-patterns.html#integration-patterns> [02 December 2020]
10. North, Dan. Introducing BDD. 2006. <https://dannorth.net/introducing-bdd/> [03 December 2020]

Agile Assignment 1

GRADEMARK REPORT

FINAL GRADE

74/100

GENERAL COMMENTS

Instructor

Agile Process Evaluation (40): 33

Examples (30): 23

Testing (30): 18

Discussion:

Nice referencing and good discussion of DVLA practice.

Avoid using the third person except in specific reflection sections.

Examples could have illustrated more of the interaction that takes place, such as moving stories on the Kanban board.

The discussion of continuous integration testing lacked detail and you could have investigated the topic in more detail.

More referencing required.

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12
