Principles of Digital Biology - Agile Strategies

Stefan Grigore
Student ID: 9970931

Principles of Digital Biology
COMP60532

Contents

1 Analysis and scope of the problem **2**
- 1.1 Current issues . . . 2
  - 1.1.1 Organisational structure . . . 2
  - 1.1.2 The principles of the NHS Constitution . . . 2
  - 1.1.3 Enabling the planned digital expansion . . . 2
  - 1.1.4 Staff morale . . . 3

2 Strategies **3**
- 2.1 Example over persuasion . . . 3
- 2.2 Failing fast . . . 3
- 2.3 Iterative and incremental design . . . 4
- 2.4 Storyboarding sessions . . . 4
  - 2.4.1 User stories . . . 4
  - 2.4.2 The storyboard . . . 4
- 2.5 Scrum . . . 5
  - 2.5.1 Roles in the team . . . 5
  - 2.5.2 Scrum ceremonies . . . 6
- 2.6 Kanban . . . 6
- 2.7 Extreme programming . . . 6
  - 2.7.1 Test Driven Development . . . 7
  - 2.7.2 Behaviour Driven Development . . . 7
  - 2.7.3 Pair programming . . . 7

3 Recommendations **7**

4 Conclusion **7**

Appendices **7**
- Appendix 1: Scrum workflow diagram . . . 8
- Appendix 2: Kanban board . . . 8
- Appendix 3: Test Driven Development Workflow . . . 9

References **10**

1 Analysis and scope of the problem

Software has played an essential role in coping with the ever-increasing and diversifying workloads that healthcare systems face. Software, data and regulations differ across a vast range of health ecosystems, which raises the need for a certain level of adaptability in the approach to developing these applications.

The goal becomes the implementation of a product development workflow that would enable healthcare systems to provide affordable healthcare with the use of software that can be delivered *"just in time"*, minimising waste while maintaining the high standards that safety-critical systems require.

This report presents such a product development strategy in the form of the Agile project management process, as applied to the case of a genomics team in the NHS which constantly requires pieces of software being developed for new analysis. The scope of this report is to provide a framework that could be gradually extended to other teams in the organisation, with the hope of providing a standardised way of developing and managing projects, while maintaining the adaptability that each service and team requires.

1.1 Current issues

1.1.1 Organisational structure

The NHS is a collection of services, each subject to their own varying individual incentives (The Strategy Unit, 2019). They all adhere to the NHS Constitution (Department of Health & Social Care, 2015) and are regulated by the Health and Social Care Act 2012 (HM Government, 2012), however, they all have a high level of autonomy in the way they operate, which makes it difficult to ensure that there is a high standard of operation and collaboration between the services.

To mitigate this, as part of the NHS Long Term Plan, *"every area is expected to become an integrated care system by 2021"* (NHS, 2020), meaning that commissioners will issue contracts to providers, each responsible for their respective NHS service in the area. The standardisation of the structure and operation of the NHS services does not need to be implemented only by regulations and commissioners and could be possibly driven organically by the results of individual services or even teams within services implementing prototype workflows for incentives and product development.

1.1.2 The principles of the NHS Constitution

The NHS puts the patient at *"the heart of everything that it does"* (Department of Health & Social Care, 2015). Given the complex organisational structure however, it could be difficult to assess whether the way in which the individual services develop projects and incentives does not stray from this principle.

A workflow that puts the end-user (i.e. the patient) at the centre of how project plans are formulated would ensure this principle and the commitment to *"providing the best value for taxpayers' money"* (Department of Health & Social Care, 2015). An analysis of the NHS found that its main weakness is healthcare outcomes, performing *"less well than similar countries on the overall rate at which people die when successful medical care could have saved their lives"* (Dayan, 2018).

1.1.3 Enabling the planned digital expansion

The NHS Long Term Plan specifies that technology will play a central role in stimulating research and helping clinicians reach their full potential (NHS, 2020). It is crucial, however, that the development of projects and incentives enables this close collaboration between clinicians, research groups and the teams developing the software. The NHS should implement this workflow in a fairly standardised way across the services to reduce costs related to onboarding, induction and context switching as departments, projects and teams evolve.

1.1.4 Staff morale

Project management methodologies usually follow one of two schools of thought. One approach is breaking down projects into sequential linear steps, where each phase depends on the deliverables of the previous one (i.e one phase flows from the other). This methodology, appropriately named *"waterfall"*, is very convenient for projects that require extensive review and approval, as the project plan can be laid out in its entirety ready to be reviewed and signed off. Naturally, any modification to the plan while the project is ongoing can be bothersome as the review process needs to start again and the documentation of the plan needs to be modified.

Often, performing maintenance and making adjustments to the plan ends up taking more time than implementing the project itself (Sutherland, 2014). It is often the case that by the time the project goes to market, the application has become irrelevant or the requirements changed in such a way that the development up to date has little value. All these aspects have a great impact on team morale, and the amount of time and resources that end up being wasted conflict with the NHS principle of *"providing the best value for taxpayers' money"*.

Another approach is breaking down projects into parallel, end to end deliverable *"slices"* of work. The goal of this approach is to get a working, minimum viable version of the product in a short amount of time in order to get feedback from the end-users and stakeholders as early as possible throughout development. This approach ensures that the project stays relevant as it is being developed, and gives a high amount of flexibility regarding changing requirements. This has a positive impact on team morale, as working prototypes motivate the team to go forward rather than have them hoping that everything will fall into place at the end of the long development cycle which could take months or years. This approach, appropriately named *"agile"*, follows the advice that when faced with two or more alternatives that deliver roughly the same value, we should take the path that makes future changes easier (Thomas, 2015). Agile techniques and strategies should follow this principle.

2 Strategies

2.1 Example over persuasion

Taking into consideration the structure of the NHS that was described in the sections above, it would be very difficult and costly to enforce a standardised workflow across all NHS services. Perhaps the most difficult obstacle to overcome would be at a psychological level. Change involves risk, which in turn mandates a thorough analysis that is most likely not feasible.

It is difficult to persuade people with the theory of what good project development workflows are, especially when there are existing methods implemented within the organisation, but if you show them how effective they can be through means such as prototypes or quick demonstrations you will likely be able to have a meaningful impact while minimising the costs. For the NHS, teams implementing prototype workflows and strategies would make surrounding teams and organisations adopt the same strategies if they prove to be highly effective for patient care. This would provide a guideline and drive change in the NHS organically, leaving room for the adjustments that each team and service requires.

2.2 Failing fast

Most projects and incentives fail, and the people implementing them end up wishing that they knew that the project was not viable sooner. To minimise the loss of taxpayer money on projects that are not viable, a very short feedback cycle needs to be implemented which will require a transition from extensive long-term plan documentation to minimal short term planning and prototyping.

2.3 Iterative and incremental design

To facilitate a short feedback loop between the people developing the products and the end-users and stakeholders, the project needs to be built in relatively short development cycles. At the end of each cycle, a working iteration of the project can be demonstrated to the users and stakeholders to figure out whether the project is on the right path. This improves morale, as working pieces of software give the motivation to move forward. These development cycles are usually short (i.e. 2 weeks), and are appropriately named *"sprints"*.

2.4 Storyboarding sessions

In line with the NHS principle of putting patients at the heart of all operations, projects and incentives should be organised in the form of a *"story"* centred around the behaviour and needs of the patient and the tasks that would fulfil these needs. Even if the end-user of the project is not necessarily a patient (i.e. software used by a clinician), the end-user should always be put in the centre of how work is organised.

Requirements can be split into two categories: functional requirements, which outline behaviour and functionalities, and non-functional requirements, which specify architectures or ways in which systems should be built. *"Based on the unique functional aspects a software system is targeted to address, the nonfunctional ones often align themselves"* (Chen, 2013).

A storyboard puts these functional requirements at the forefront, and a session involving developers, stakeholders and end-users such as clinicians or even patients would be a good strategy for coming up with a relevant project plan to meet these goals and requirements.

2.4.1 User stories

"Stories use non-technical language to provide context for the development team and their efforts" (Atlassian, 2020). To provide a standardised way of expressing these stories to convey the user, the feature and the value of the task, the Connextra template (Lucassen, 2016) can be followed:

As a <role> I can <capability>, so that <receive benefit>

The advantage of organising work like this is that non-technical people can be involved in the process. The template makes us think about the value that the story would bring, enabling us to figure out whether the task is useful or not in an attempt to minimise waste and maximise the value to the end-user.

User stories need to describe work that can be completed by the end of a sprint (i.e. by the end of 2 weeks). They should be *"thin end-to-end slices of the system"*. When a user story describes a process that is too complicated to be completed in a sprint, for example, *"As a clinician, I want to see a DNA variant summary report for a patient so that I can detect potential risks of diseases"*, it becomes an *"epic"* body of work which can be broken down into user stories that could be completed in a single sprint. User stories belonging to this DNA variant summary epic could be things such as *"I want to see the samples with their number, name and single-nucleotide variant"* or *"I want to see a bar chart of the aligned bases against the samples"*.

2.4.2 The storyboard

The start of each project is the establishment of goals. To illustrate the storyboarding session with a simpler example, let us take the goal or story of *"I'm hungry and I want a sandwich"*. The project that would meet this goal can be split into parallel end-to-end deliverable chunks of work (i.e. the *"epics"*). Some epics could be getting the ingredients, preparing the sandwich and so on. Tasks or user stories can be assigned to these epics (i.e. getting bread, ham, cheese, toasting the sandwich, packing it etc.). Figure 1 shows how such a storyboard could be constructed.

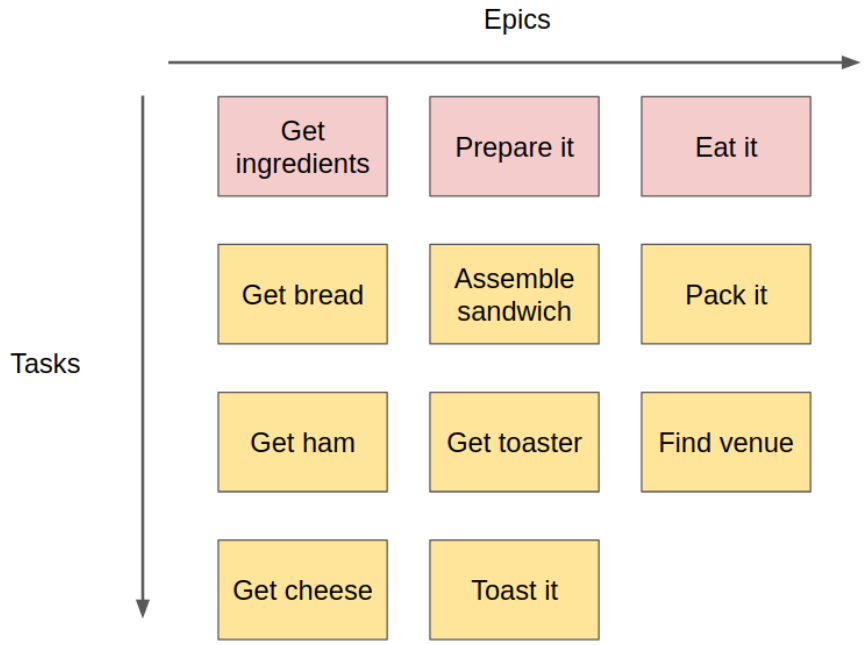


Figure 1: Very basic storyboard example, where the priority of epics goes from left to right and the priority of tasks goes from top to bottom

These stories and epics can be produced and discussed throughout the session with the use of post-it notes. Things can be switched around or removed, making the whole process adaptable and highly efficient for the goal of reaching a consensus. This plan layed out as post-it notes on the wall of the office space can act as an "information radiator" that people can just glance at and that would facilitate a tight feedback loop.

The advantage of organising the project in this way is that each *"epic"* can be delivered at various levels of completeness, but with the aim of delivering a working, minimal viable prototype of the project at the end of each development cycle. The key is to improve each iteration of the project by adding new features and improving existing ones incrementally. This also gives the agility of switching the direction of the project while minimising losses when certain bits of work start to look like they would not bring much value.

2.5 Scrum

Scrum is a workflow that aims to deliver the minimum viable product at the end of each sprint with the use of several *"rituals"*, namely the daily stand-up, the sprint planning, review, demonstration and retrospective (Atlassian, 2020).

2.5.1 Roles in the team

There are three roles in a scrum team: developer, product owner and scrum master (Atlassian, 2020). The term of developer changes in the context of a scrum team, and transitions from the general idea of a developer as an engineer. Developers can be people bringing skills from all kinds of backgrounds such as software engineering, marketing, design, business etc., all working together towards the goal of the project. The product owner is the person orchestrating the various tasks and stories following the wider business and stakeholder context. They provide the direction of the project that will bring the most value. Finally, the scrum master ensures that the team is applying the workflow correctly, monitors the performance and makes adjustments to ensure smooth operation. Scrum teams are usually between 5 to 7 people. In some teams, the roles can sometimes overlap, for example by having a product owner that is also a developer, or by having a developer that is also a scrum master etc.

2.5.2 Scrum ceremonies

Sprint planning: At the beginning of each development cycle the product owner comes with a draft of the sprint plan in the form of user stories or tasks that they think should be completed by the end of the sprint. The tasks are discussed within the team, new tasks are added or existing ones are removed. User story points are assigned to each item via a team consensus based on the relative difficulty of tasks and the performance of the workflow is measured in the number of story points completed per sprint (also known as *"velocity"*). This amount would also serve as a guideline during planning which would indicate whether the team is picking too much or too little work. The scrum master monitors these statistics to ensure the smooth operation of the team.
Daily stand-ups: To make the feedback loop even more frequent, scrum teams have a daily 5-minute meeting where people talk about what they worked the day before and about what they will be working on the current day, and mention whether they have any bottlenecks. Besides identifying bottlenecks, this practice motivates people to deliver something valuable each day that they can talk about in the stand-up the following day.
Sprint review: Midway through the sprint, the team meets to check whether they are still on track to complete the scope of the sprint and make adjustments accordingly.
Sprint demo: At the end of the sprint the team can meet the stakeholders, end-users or the people interested in the project and demonstrate new functionality, get feedback and take notes for future tasks.
Sprint retrospective: Before the next development cycle begins, the team meets to discuss what went well, what could be improved and based on these insights produce tasks that will be picked up in future sprints.

A diagram of the Scrum workflow can be found in Appendix 1.

2.6 Kanban

Scrum tackles the challenges of long-term planning by providing a workflow for producing short term iterations of the project which are improved incrementally. One important point of Scrum is to not alter the scope of the sprint (i.e. by adding more tasks to the sprint that were not planned while the sprint is ongoing). This follows the reasoning that if we alter the sprint scope we will not be able to complete the work that we planned, and possibly not even the work that we didn't plan for either. How do we deal, however, with urgent and important work that comes up such as application breaking issues appearing throughout the sprint?

Kanban is an alternative workflow which does not involve sprints. The tasks are organised as a list of items that are ordered by priority and maintained regularly by the product owner. Tasks pass through various stages of completion across a *"kanban board"* (i.e. stages such as *"to do"*, *"in progress"*, *"done"*), and developers pick up the most appropriate tasks for them from the top of the ordered list of tasks. The goal is to get these *"cards"* across the board through the stages as quickly as possible. Because of the lack of sprints, we cannot talk about performance in terms of completed story points per sprint or *"velocity"*. The measure of performance becomes the *"cycle time"*, which is the amount of time it takes for a card to pass across the board from the moment it is picked from the prioritised *"to do"* column until the moment it is placed onto the *"done"* column.

An example Kanban board is shown in Appendix 2

2.7 Extreme programming

One important issue concerns how we could introduce this incremental design in the production of safety-critical systems that require extensive plans and documentation to ensure stability. In the case of gene sequencing, a wrong analysis could lead to the prescription of medication to patients that could be fatal to them. Various practices can ensure a high standard of test coverage and safety. These practices move away from the traditional rigid specifications in favour of more collaborative approaches focused on iterative and incremental implementation, also known as *"extreme programming"*.

2.7.1 Test Driven Development

Test Driven Development (Beck, 2000) is a programming workflow consisting of 3 steps. First, a test is written based on specifications. When run, the test will fail as the implementation is yet to be put in place. The second step is the modification of the system that would enable the test to pass. The key is to implement something as simple as possible, without doing premature optimisations. Now, the test will pass when run. The third step is improving the code that we just wrote while also keeping the test passing and not doing unnecessary premature optimisations. When no more improvements can be made, the process begins again with the next bit of specification. This workflow ensures that there is no regression (i.e. previous tests will cover existing functionality and the developer does not need to worry about unknowingly breaking it when implementing new features). A diagram of the Test Driven Development workflow is shown in Appendix 3.

2.7.2 Behaviour Driven Development

Behaviour Driven Development (North, 2006), just as Test Driven Development, is a test-first approach where the tests are produced by directly following logic specifications written in the form of user-centric scenarios formulated using templates such as the *given-when-then* pattern (i.e. *"Given a patient, when they navigate to their profile, then they can see their test results"*). These scenarios can be written by management and can be parsed automatically to produce test placeholders, which will become the basis for testing and implementation.

2.7.3 Pair programming

Pair programming (Williams, 2001) is a technique in which two programmers develop software collaboratively. One programmer is the "driver" that writes the code, while the other is the "navigator" that provides insights and reviews about what is being written. The roles can switch during a pair programming session, and the technique maximises the quality of the design and implementation. Combined with the techniques above, this style of programming can provide reliable, high-quality software that could meet the specifications of safety-critical systems.

3 Recommendations

For our genomics team, the recommendation would be to form a team of people coming from various relevant backgrounds across the service, and to organise storyboarding sessions with the relevant stakeholders, clinicians, patients etc. for the various pieces of software that are required.

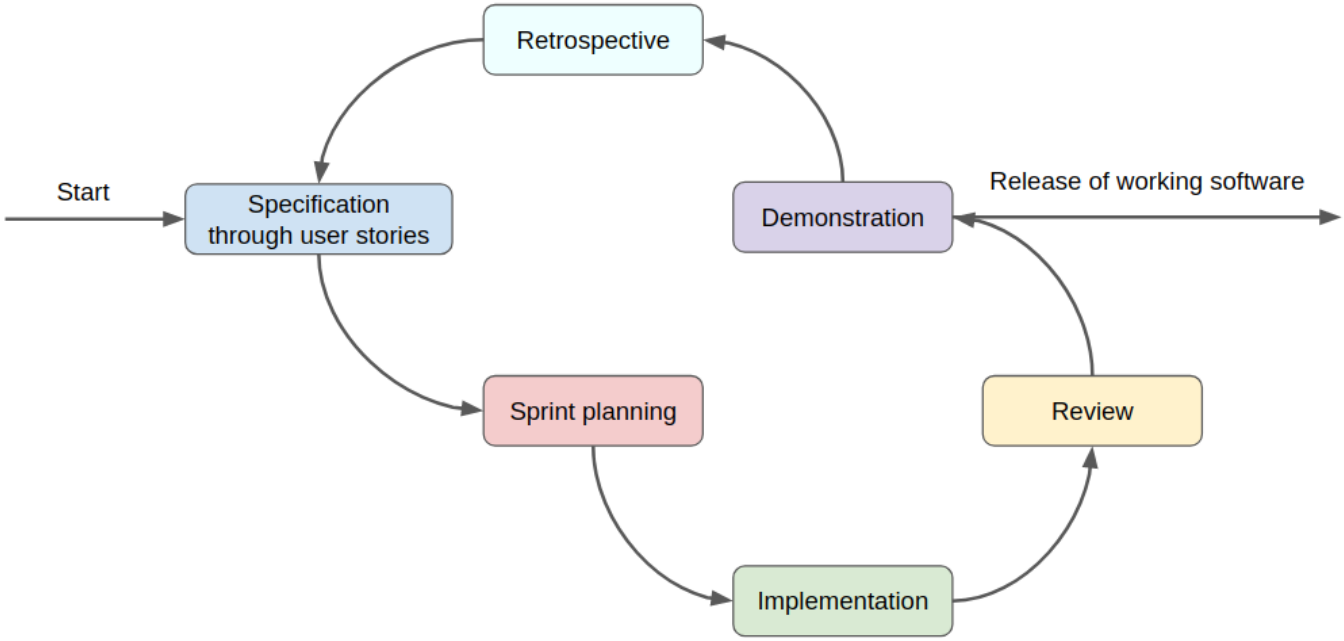
In the initial stages of the project, a Scrum workflow would be most appropriate to deliver working iterations of the projects fast, and these projects can be gradually switched to a Kanban workflow as they mature and the majority of the work moves from planning and implementing new features to maintenance and solving various issues that arise. Working iterations of the software should be demonstrated to clinicians frequently (i.e. at the end of each sprint), and a close feedback loop should be maintained to ensure the quality and relevance of the implementation. Extreme programming methods such as Test Driven Development, Behavior Driven Development and pair programming would ensure the high-quality of the implementation while putting the end-users at the forefront, coupled with the *"user story"* based specification and planning.

As the team becomes more Agile and its effectiveness, velocity, cycle time and morale improve, other teams within the genomics service would follow the example. This would produce a change across the service in an organic way that would hopefully spread to other services across the NHS.

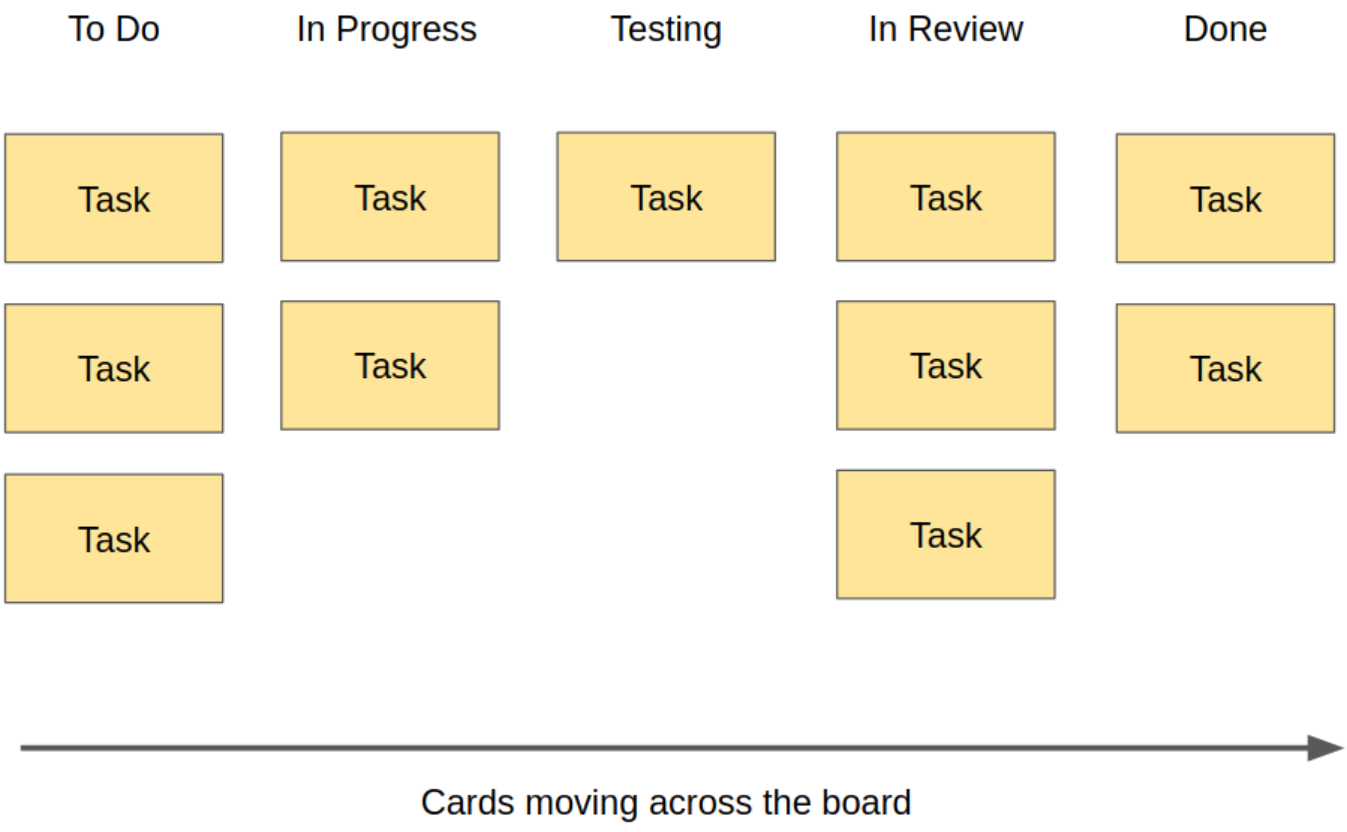
4 Conclusion

The measure of success is the fast delivery of working software that improves healthcare outcomes. This report provides strategies for how projects can be developed in a manner that puts the patient at the centre of operations, maximises adaptability and minimises waste (i.e becoming Agile), from the initial team building, planning and specifications to the implementation and testing. These can all be implemented in parallel and in a continuous manner throughout development, moving away from traditional workflows such as the *"waterfall"* methodology which might take longer and end up wasting taxpayer money.

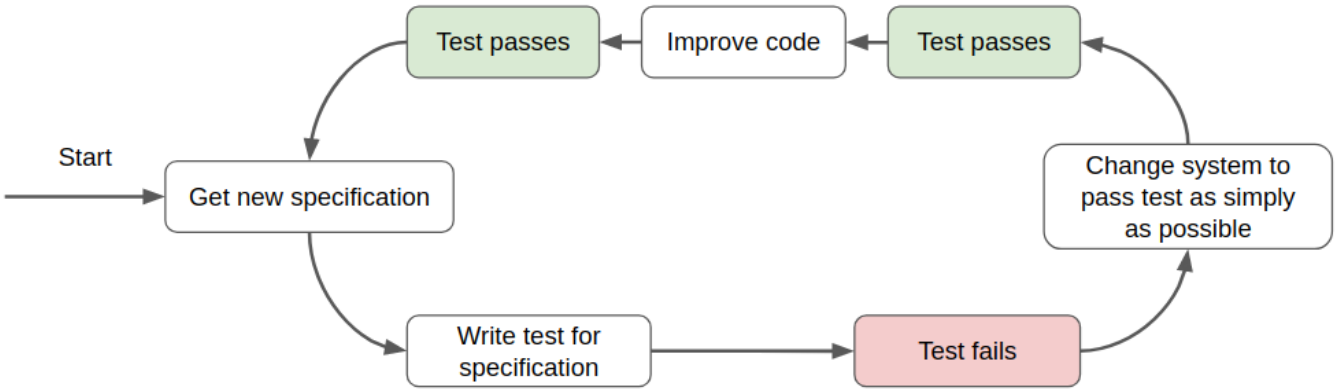
Appendix 1: Scrum workflow diagram



Appendix 2: Example Kanban board



Appendix 3: Test Driven Development Workflow



References

Atlassian (2020) *https://www.atlassian.com/agile/project-management/user-stories [accessed March 2020]*

Atlassian (2020) *https://www.atlassian.com/agile/kanban [accessed March 2020]*

Atlassian (2020) *https://www.atlassian.com/agile/scrum/ceremonies [accessed March 2020]*

Atlassian (2020) *https://www.atlassian.com/agile/scrum/roles [accessed March 2020]*

Beck, K. (2000) *Test-Driven Development by Example*

Dayan, M., Ward, D., Gardner, T., Kelly, E. (2018) *How good is the NHS?*

Department of Health & Social Care (2015) *The NHS Constitution for England*

Chen, L., Ali Babar, M., Nuseibeh, B. (2013) *Characterizing Architecturally Significant Requirements*

HM Government (2012) *Health and Social Care Act 2012*

Lucassen, G., Dalpiaz, F., Werf, J. M. E. M. van der, Brinkkemper, S (2016) *"The Use and Effectiveness of User Stories in Practice"*

NHS (2020) *Integrated Care Provider Contract*

NHS (2020) *NHS Long Term Plan*

North, D. (2006) *Introducing BDD*

Sutherland, J. (2014) *Scrum: The Art of Doing Twice the Work in Half the Time*

Thomas, D. (2015) *GOTO 2015 - Agile is Dead*

Williams, L. (2001) *Integrating pair programming into a software development process*