

AGILE Approaches To Software Development

ACCA7001

Introduction

This report aims to describe the processes involved in an Agile Development methodology and evaluate their impact on the product developed. A case study of the UK Government project, Gov.UK has been chosen to demonstrate the stages in an Agile software development process and the arguments presented. Gov.UK is 'the Government as a platform' service that was initiated in 2012 and is continuing to include more services to the portfolio. The Government Digital Services team adopted Agile methodologies and have seen tremendous success.

Rapid developments in technology and evolving business models influence quality, cost, time and productivity of software development process [1]. Organisations aim for product delivery within short time scales and high returns on investments. Methodologies such as waterfall methods involving heavy documentation and longer delivery result in the product being too late for the purpose intended [2]. Demand for methods that counter the drawbacks of traditional approaches has resulted in a new generation of methodologies, shaped by the Agile manifesto.

The Agile approach:

The Agile approach is built on the principles of reflective practice. Key priorities include a focus on continuous improvement while maintaining technical excellence, simplicity and delivery. This is influenced by reviewing practices, collaboration, communication and feedback. It requires an effective cross functional team with all stakeholders including customers, business

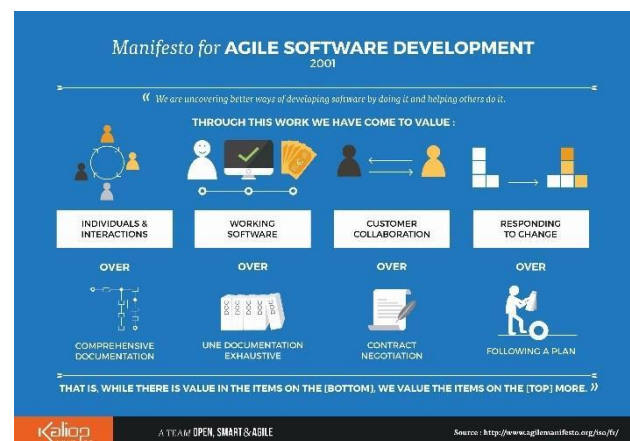


Figure 1[16]

analysts, developers and quality assurance. Equal engagement is important throughout the process with shared responsibilities in a motivated environment. The primary aim is customer satisfaction by delivering high quality of working products at short intervals. Direct contact with users is essential for success. Changing requirements is a constant factor and embracing change throughout the production life cycle is imperative.

Methodologies

Software development life cycles go through five stages. Recurrence of these stages at time bound intervals of two to six weeks involves delivering a minimum viable product after every iteration. Functionalities are added as increments in successive iterations. The team reviews the work at the end of each iteration. User input drives development at all stages.

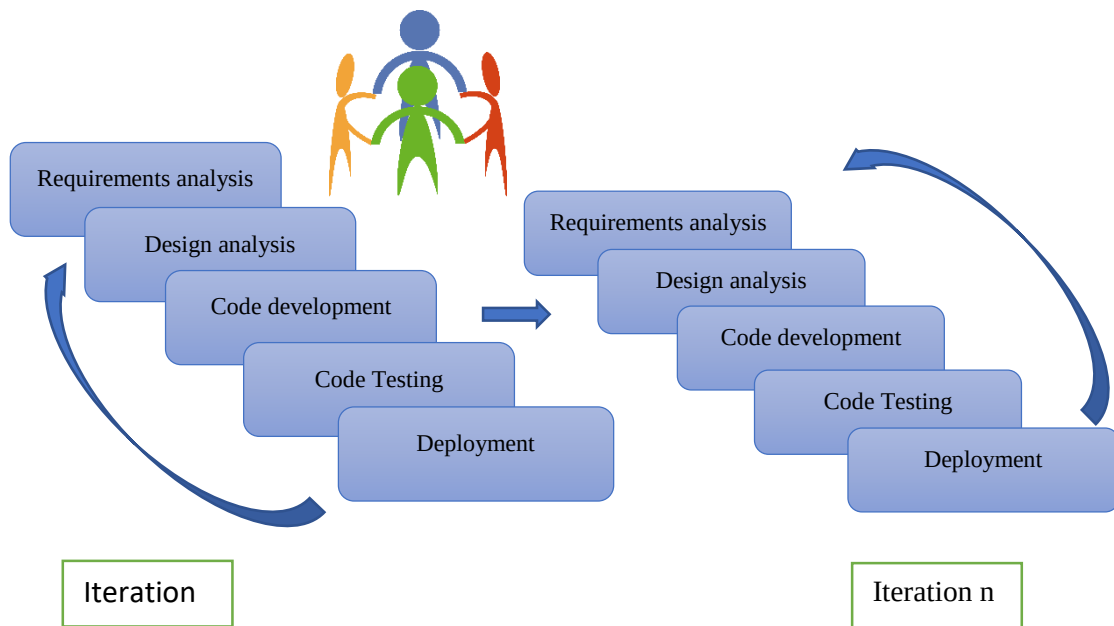


Image2:

Several methodologies have since evolved in the Agile processes family and have gained popularity. Industry favourites include [3]:

- Extreme Programming (XP),
- Scrum
- Feature Driven Development
- Agile Rational Unified Process
- Dynamic System Development Method
- Scaled Agile Framework
- LeSS (Large scale Scrum)
- Crystal Families

- Adaptive Software Development and
- Open Source Software Developments.

Flexible developments have also evolved such as Scrumban, an amalgamation of Scrum and Kanban. A survey conducted by Scrum Alliance [4] reports that 89% of respondents named Scrum as the Agile approach used in their organization. A report by Version One [5], published in 2017, identified Scrum and Scrum-XP-Hybrids as the most common Agile methodologies, used by 68% of respondents.

Traditional vs Agile

Parameters	Traditional Development	Agile Development
Priorities	Focus is on the process of the development	Focus is on people and engagement with stakeholders
Scope and Requirements	Requirements captured at the beginning. Change management process is essential.	Requirements evolve throughout. Business value is associated to the teams.
Work plan	Sequential. Tasks allocated by Project manager.	Iterative and incremental. Self-organising teams select their own tasks.
Releases	Schedules and features are decided at the beginning	Based on user story priority. Planned by team and product owner
Performance and reporting	Detailed metrics	Agile metrics and traditional metrics from testing.

Digital by default – Gov.UK.Alpha

In 2010, the existing UK government website was reviewed which culminated in a vision for the government services to serve the citizens through a Single Government Domain [6]. This

was titled as the Gov.uk.Alpha project which aimed to demonstrate that Agile works. Objectives include:

- Developing a prototype of a single UK Government website and test in public.
- Adopting Agile, multi-disciplinary team of developers, content editors and designers using development techniques driven by user experience [7, 8].
- Achieving within a budget of £261000 [7] .

Scrum was the chosen Agile methodology [7]. Following the success of Alpha, the Government Digital Services (GDS) and Gov.UK were established in 2012. Originally, 140 people worked across 11 teams. Now there are 168 people working amongst 24 teams. GDS delivered its first product, an e-petition service, in 8 weeks [9]. A year after its launch, the project expanded with content from 330 organisations from 24 ministerial departments. Currently all 25 ministerial departments and 385 other agencies and public bodies offer their digital services through the Gov.UK website.

The Government mantra of 'digital by default' is driven by the policy which mandates Agile methods. All GDS projects have adopted the Scaled Agile Framework with strict service standards and approval process for releases. Most teams used Scrum for the management of the development process powered by Test driven development for implementation. Typical one to two week sprints, embedded acceptance tests and quick releases characterised the development process.

The UK GDS has devised a 10-point design standards based on the Agile principles and classifies the development cycle into 4 phases [10].

- Discovery - Research on identifying high level requirements that are dictated by user needs and policy constraints.
- Alpha – Development of prototypes, continuously integrated and tested with small groups of stakeholders with reliance on feedback.
- Beta - Private and public beta involves the testing in a live environment. This is the largest period in the development phase.

- Live - Open to the public in full production, still entails developers to continuously iterate feature improvements.

Discovery - User Stories

Software projects with a focus on automation of routines and transactions are a thing of the past. In a traditional development process, there is no expectation to understand the complexities of the requirements of users and organisations. User engagement is only required during the initial stage of the project. This has now evolved into a more sociotechnical approach [11]. Agile as compared to the traditional methods is a user driven development. The user is the most important actor and has the freedom to change or add new requirements. Active involvement of users helps to identify realistic expectations with a high-quality end product. Despite this identifying risks and issues at an early stage, some would argue that the continuous user presence can give a false sense of confidence, especially in the XP environment [1].

User Research

User requirements are continuously gathered through research via GDS. Research facilitates the identification of potential services that require development. Each service team has a dedicated researcher who joins in the sprints. Designers, developers, business analysts and product managers are also involved making it a typical 'everyone is responsible' Agile method [10]. At the initial inception of the Gov.UK project in 2011, user researchers identified and prioritised the top 100 user requirements [8].

Ongoing user research results in the backlog being appended continuously. Data from the use of live services provide for changes in the stories that are catered for in the next iteration. This results in feature creeping but ensures high user satisfaction. As an example, discovery on carers allowance application found that users preferred summaries. This was incorporated into the UI, increasing the number of applications [9].

Alpha – Design and Development

Agile practices involve a quick and dynamic transition from requirement analysis to development phase. Planning is continuous, based on a 12 months roadmap that ensures planning for scope and release deadlines. Prototypes are built and tested with users in this phase. Prototypes help in ascertaining the technical feasibility of the mission. It also enables risk analysis for beta and allows cost estimates. Alpha usually last up to 8 weeks with 3 stages [10].

- Inception - set goals
- Iterations - build and test prototypes, learn and adapt.
- Conclusion – move into Beta.

User research identifies needs that fall under a variety of situations which are initially used to build prototype for several user journeys. This is narrowed down to the one that would deliver optimum user experience. GUI designers work with the DevOps team. Cost implications are identified and the prospect of going into Beta is forecast.

Sprints

Agile methods are centred around the concept of minimum viable product. A sprint is an iteration that last between two weeks and a month of development. Unlike traditional development at the end of each sprint in lieu of an entire product handed for testing, an MVP is readied for deployment. Each Sprint involves a collection of ceremonies and artefacts which includes planning, daily scrums, reviews and retrospective of sprint backlogs and increment. A typical example of a weekly sprint [7] would be that of the Gov.UK teams working model that focused on determining stories and workload in conjunction with extensive communication and a single focus. According to Alan Eccles, two week sprints enable to measure success of outcomes and its impact on customers [12].

Daily Scrum

In Agile, 15 minute standups of team discussions on progress, planning and identification of blockers replace extensive documentation as means of knowledge sharing in traditional development. In Agile, informal and frequent communication between all parties concerned is

the norm [10]. In a larger projects such as Gov.UK, communication lines, project structure and diversity of teams may present some barriers which are overcome by daily scrum [13].



Figure 1 Burndown chart [13]

As an example, GDS had 14 member teams which experienced strained communications in the initial stages. To overcome this, program level stand ups attended by cross team delivery managers were introduced which helped to improve visibility and collaboration [14]. Some features of Agile are:

- Establishing working groups for each program
- Development of Epics and user stories to arrive at releases which are tracked
- Dashboards display of progress based on burndown figures and velocity
- Capturing of Metrics at sprint, release and project level

In Agile risks and blockers are identified and addressed in the standups. A digital record of risks and dependencies (RAID) is maintained, although this is not strictly an Agile practice, this provides opportunities to plan mitigation. Agile works at its best in small organisations and a combination of methodologies maybe required in a larger context that best suit the environment[40].

Testing

Users test the functional aspects of the software while the developers test the functional and non-functional aspects of the software. In a test-driven development, all tests are written prior to the development code. Fail first unit tests ensure that code is developed only for required goals. Feature codes that are reviewed and tested move on to integration. Integration testing

involves high level scripts for critical functions followed by scripts for all other test conditions and exception handling and non-functional performance testing.

As an example, in Gov.Uk the top 10 services used were tested and data collected from these tests were used to modify or add new features[36]. Blackbox user testing of interface and system is not just the QA team's responsibility. Everyone within the team gets a two-hour exposure to user testing every six weeks. A deeper understanding of user requirements results in better designs in successive iterations. Prototype testing assists in the maintenance. Smaller sprints result in a stable code base by identifying bugs quicker and earlier.

Review:

A review involves demonstration (1-2 hours) to all stakeholders of the product developed so far which includes brief information on plans for the immediate future. In a scaled environment, this opens the doors to cross teams and facilitates sharing of vision and knowledge. Multiple teams working on a common goal have a single session. The Product Owner indicates the customer value created. DevOps related user stories are also part of the review. The Team demonstrates completed Sprint Backlog items. Architecture related items could be showcased by walking through the components using mock objects or stubs.

Retrospective:

Agile is self-reflective practice and retrospective is a process improvement exercise of 60-90 minutes with discussions on positives and negatives in the completed iteration. Overall team performance, quality, and communication within the team and with external stakeholders, are reviewed. Improvement actions are prioritised and logged.

A developer's perspective of a feature might be different to that of the user. The developer makes decisions on managing effort against feature completeness[17]. Any compromises made on the quality of the product over faster delivery might result in a technical debt. As an example, to tackle such issues, GDS derived the following four principles.

To increase flexibility, no work that would require more than 3 months was committed to unless the scope could be trimmed. Teams provide evidence of delivered value with metrics. To develop a creative and motivating environment, the team was allowed a week firebreak

working outside the project. 11 week missions to increase sustainable building [14] with Smaller groups resulting in better self-organising teams.

Deployment:

Projects using Agile methods may be ready for deployment early and frequently at the end of each sprint; therefore, the release into production should be more robust. High Intervention tasks require some amount of human judgement as opposed to low intervention tasks. Spending more time on high intervention tasks and automating lower ones result in quality output. Technical decisions such as choice of server demand more human time than repetitive tasks. Continuous Integration and continuous D(CICD) facilitates to maintain the product in deployable state at all times using Git repositories [15]. In a Government as a platform service, it is imperative that the services are released quickly and bug free. Multiple teams also warrant for a coordinated attempt on maintaining the pace and CICD merges all code constantly providing an up to date version. A deploy release phase with a high degree of automation results in better business value to the customer as in the case of GDS where users are utilised for testing.

Beta

When services to Beta are pushed at this stage, an end to end beta prototype of working version of the service is built. This is first launched in private beta for restricted users and furthered as public beta to capture user experience. This forms part of the overall system and acceptance testing. All approvals and assurance testing with regards to service standards are obtained [10].

Live

In our case study, GDS policy states that the service assessment should be satisfied for the public beta to go live. Research and iterations continue to improve services. The system is continually tested for performance, vulnerability and regression. Status monitoring for availability is vital.

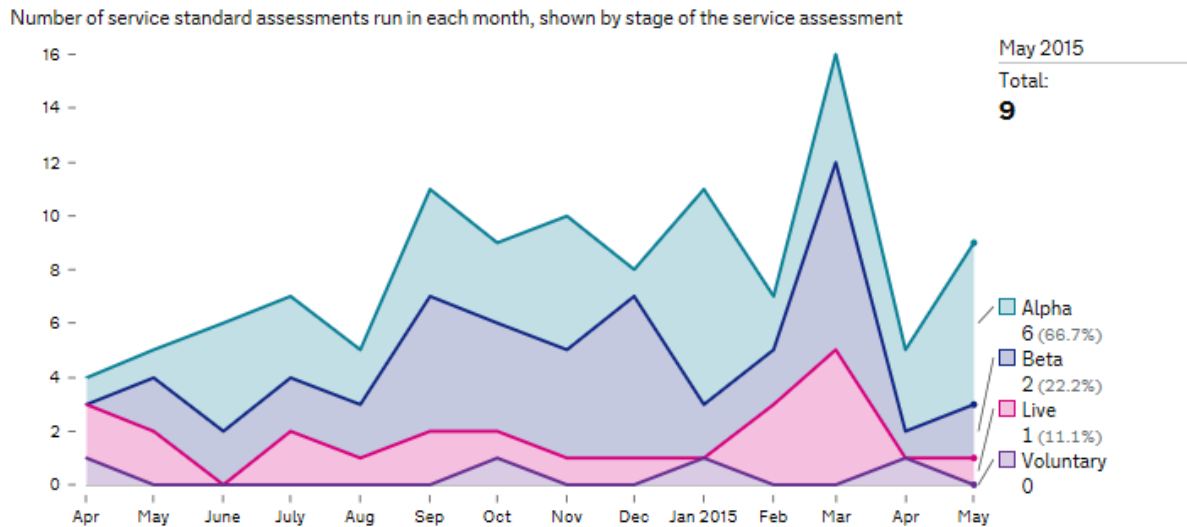


Figure 2Gov.uk

Conclusion

Agile Software development works on the principle of user satisfaction, flexibility, collaboration and efficiency. Feature scope might change during development. However, efficient sprint planning and distributed development ensure deadlines are met. Quality is implicit. Definition of Done confirms the quality of deliverables. Agile does not measure the processes but the impact of the process. Metrics are captured at sprint, release and project level improving quality. Testing, review and retrospective improve quality of product as well as the team performance. CICD facilitates continuous deployment of the product with high efficiency and quality. CICD reduces risks by picking up defects and impediments at an early stage and produces bug free product. The myth that Agile loses control due to self-organising teams is disproved by building control into the process. Scaled Agile framework has more governance and control. The Government Digital service with its team of 140 people is a testimony to how Agile works. They have successfully launched several digital services for 25 ministerial departments and 385 agencies using Agile methodologies.

References

- [1] S. V. K. R. P. & R. P. Kuppaswami, "The effects of individual XP practices on software development effort.," *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 6, p. 6, 2003. [Online] Available: 10.1145/966221.966239 [Accessed 01 12 2018].
- [2] "“What is lightweight methodology?”," [Online]. Available: <http://www.xprogramming.com/light1.html>. [Accessed 01 12 2018].
- [3] O. S. J. R. a. J. W. Pekka Abrahamsson, "Agile Software Development Methods: Review and Analysis".[Online] Available: <https://www.vtt.fi/inf/pdf/publications/2002/P478.pdf> [Accessed 01 12 2018]
- [4] Scrum Alliance.org, "2017-18 State of Scrum Report -Scaling and Agile Transformation". [Online] Available:<https://www.scrumalliance.org/learn-about-scrum/state-of-scrum/2018-state-of-scrum> [Accessed: 01/12/2018]
- [5] VersionOne, "The 11th Annual State of Agile Report. Technical Report" [Online] <https://explore.versionone.com/state-of-agile/> versionone-11th-annual-state-of-agile-report-2 [Available:11/12/2018]
- [6] GOV.UK, "Directgov 2010 and Beyond: Revolution Not Evolution," 23/11/2010. [Online]. Available: <https://www.gov.uk/government/publications/directgov-2010-and-beyond-revolution-not-evolution-a-report-by-martha-lane-fox>. [Accessed 11 12 2018].
- [7] J. Arnold, "Agile does work in government 13 May 2011," GOV.UK - GDS Blog, 2011. [Online] Available: <https://gds.blog.gov.uk/2011/05/13/agile-does-work-in-government/> [Accessed: 05/01/2018]
- [8] GDS Blog, "Towards a Single Government Domain," GOV.UK DGS Blog, 2011.
- [9] GOV.UK, "A GDS story," 2011. [Online]. Available: <https://gds.blog.gov.uk/2018/12/07/a-gds-story-2018/>. [Accessed 11 12 2018].
- [10] GOV.UK, "Agile Delivery - Service Manual," [Online]. Available: <https://www.gov.uk/service-manual/agile-delivery>. [Accessed 12 12 2018].
- [11] A. C. V. B. P. B. Sridhar Nerur, "Towards an Understanding of the Conceptual Underpinnings of Agile Development Methodologies," DOI 10.1007/978-3-642-12575-1_2, Springer Verlag Berlin Heidelberg 2010, 2010.
- [12] Government Digital Services, "The Public Guardian on agile development". [Online] Available:<https://gds.blog.gov.uk/2014/09/26/the-public-guardian-on-agile-development/> [Accessed: 12/12/2018]
- [13] T. Loosemore, "Alpha.gov.uk wrap-up," 29 July 2011.[Online]Available: <https://gds.blog.gov.uk/2011/07/29/alpha-gov-uk-wrap-up/> [Accessed: 08/01/2019]

- [14] N. T. a. A. W. Jennifer Allum, "GOV.UK: a journey in scaling agile," 26 April 2018. [Online]
] <https://gds.blog.gov.uk/2018/04/26/gov-uk-a-journey-in-scaling-agile/> [Accessed: 08/01/2019]
- [15] S. A. I. B. S. Arachchi and I. Perera, "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management,," in *Moratuwa Engineering Research Conference (MERCon)*, Moratuwa, 2018. [Online]
Available:<http://ieeexplore.ieee.org.ezproxy.uwtsd.ac.uk/stamp/stamp.jsp?tp=&arnumber=8421965&isnumber=8421884> [Accessed: 19/12/2018]
- [16] Image 1: <https://blog.kaliop.com/en/blog/2015/04/02/draw-me-an-agile-manifesto/>
- [17] J. Arnold ""What we've learnt about scaling Agile" GDS blog, 2012 [Online]
Available:<https://gds.blog.gov.uk/2012/10/26/what-weve-learnt-about-scaling-agile/>
[Accessed:12/12/2018]

Bibliography

1. Beck, K.: Embracing change with extreme programming. IEEE Computer, 70–77 (1999)
2. Koskela, J., & Abrahamsson, P. (2004). On-site customer in an XP project: empirical results from a case study. In Lecture Notes in Computer Science (pp. 1 - 11). Heidelberg: Springer. https://doi.org/10.1007/978-3-540-30181-3_1
3. Roderick Chapman, Neil White, and Jim Woodcock, *What can Agile bring to High Integrity Software development?* october 2017 | vol. 60 no. 10 communications of the acm 41 [Online]Available:DOI.10.11.5/31332333 [Accessed: 11/12/2018]
4. VTT technical centre published 15/01/2010
<https://www.vttresearch.com/media/news/user-driven-software-development-becomes-more-efficient-through-social-media>