Enabling Agile Through Teamwork

Cecilia La Place*
Chiranjeevi Ramamurthy*
claplace@asu.edu
cramamu1@asu.edu
Arizona State University
Mesa, Arizona

Abstract

For more than a decade, the Agile development process has seeped into the lives of software developers and customers, changing the way projects are planned, how teammembers and teams interact, and how customers receive their product. Agile is a teamwork heavy process, demanding superb communication and technical skills to develop projects where requirements can change the flow of the project between sprints. Similiarly, Agile teams must work together on larger scale projects to ensure project success. [Summary about importance attributes of agile teamwork here.] Furthermore, [summary about importance of design/architecture here].

Keywords agile, teamwork, design, architecture, software development process

ACM Reference Format:
Cecilia La Place and Chiranjeevi Ramamurthy. 2018. Enabling Agile Through Teamwork. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 Introduction

Agile is a software development process. It has revolutionized software development, shifting some companies and projects from a plan-first-code-after-and-repeat life cycle, to a "satisfy customers-at the time of delivery, not at project initiation" life cycle, as described by [2]. In this work, we first introduce a quick overview of Agile, delve into teamwork in agile, and then widen our scope to teamwork between teams in agile. Second, we discuss the importance of design and architecture within teams, where we highlight the ways they affect teamwork both positively and negatively.

*Both authors contributed equally to this research.

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

2 Agile

Agile development considers an element crucial to all software processes in a uniquely different manner: customers as team members. The basis of all software development is planning out a project from start to end, but requirements are prone to change when considering the fluctuating world of technology. Agile handles change by valuing the people involved, customers and developers [2]. By interfacing with customers frequently, and considering their changing requests as the project progresses, change is a byproduct. However, the success of Agile is reliant on what Cockburn and Highsmith call "responsive people and organizations" as well as "[focusing] on the talents and skills of individuals" [1]. As a result, teamwork and communication become unavoidably ensconsed in the agile process.

2.1 Teams in Agile

A single team in agile consists of the product owner, the scrum master, and the developers. In order for their team to be successful, they must "have a common focus, mutual trust, and respect," be "collarborative, but speedy [in their] decision-making process" and be adept at handling ambiguity [1]. The Agile Manifesto reinforces these attributes by valuing customer collaboration and responding to change over contracts and plans [3].

2.2 Multiple Teams in Agile

3 Design and Architecture in Agile

3.1 Design

3.2 Architecture

3.2.1 Importance of architecture

Organizations tend to fail if their agile process is not bounded by a fail-safe plan.

The fundamental reason for this is that we all operate within constraints, which can be financial, regulatory, technical or customer driven. While Agile practices have traditionally been confined to software development there is a significant push by organisations, particularly at the Enterprise end of the market, to use Agile practices to manage

traditional business functions. This new trend is euphemistically referred to as New Ways of Working. The benefits of leveraging Agile practices are numerous, with the fundamental benefit that organisations see Agile practices as a way to deliver improved outcomes for their customers and stakeholders, more efficiently and consistently.

There are numerous case studies citing the achievement of these benefits at a project level, but very few examples (to date) of successful Agile Transformations at Enterprise Scale. Proponents of Agile practices will point to the Spotify Model as proof that Agile Practices can be used to build a 13 billion USD Enterprise. Which is true, however, they didn't do it without Architecture. They did it by leveraging Architecture and its practices as an enabler and not a governing framework. The way that Architecture worked within Spotify is quite different to how Architecture currently operates within Traditional Brick and Mortar Enterprises.

It is very hard to find a clear definition of the role of Architecture in Agile. The SAFe (Scaled Agile Framework) framework has done the most to identify the role of Architecture within an Agile environment. As with all things Agile the focus is to create consistent value and Architecture is no different. In SAFe they define two distinct elements of Architecture:

- `Emergent Design`
- `Intentional Architecture`

Emergent Design provides the technical basis for development and the incremental implementation of initiatives. It helps Designers and Architects to be responsive to changing customer/ stakeholder needs to ensure the initiative continually delivers value. At this level, SAFe practitioner's see Architecture as a collaborative and interactive exercise through which the design element can emerge.

Intentional Architecture is a much more structured approach and more aligned to what many would identify as being traditional Architecture, that is a set of defined and planned Architectural initiatives which will both support and enhance the performance and usability of the initiative. In effect, Intentional Architecture is a clear recognition that we all need to operate within certain constraints such as choice of technology platform, financial budget, etc. If these constraints can be identified and incorporated into the initiative then the probability of the initiative being successful and delivering value is increased.

SAFe practitioners proport that by balancing Emergent Design and Intentionality Agile practices can be scaled to deliver Enterprise level solutions. In Safe this combination is referred to the Architectural Runway which provides the technical foundation for creating business value. Which is in complete alignment with traditional views of Architecture.

The key to the success of this approach is the level of abstraction at which the balance of Emergent Design and Intentional Architecture occur. The fundamental behaviour that will determine this is collaboration. Architects need to be able to work productively with Agile Teams to provide fast and local support to manage Emergent Design while also helping Agile Teams to appreciate and navigate the constraints defined by the Intentional Architecture. One of the key attributes of Agile Practices is the fact that Agile Teams are encouraged to provide constant feedback to their stakeholders. As emergent designs develop Architects can use this information to adapt and develop the Intentional Architecture to ensure that the overall Architecture of the Enterprise is evolving with the organization in the medium to long-term.

So does "Agile need Architecture to be Successful? " I would say the better question is "What type of Architecture does Agile need to be successful? " Agile requires Architecture that supports the way the Agile Practices deliver of outcomes (value). The type of Architecture that will do this will be a combination of a nimble reactive style of Architecture supported by a more traditional structured approach to Architecture. The challenge as with many things is to get the mix right!

3.3 Design and Architecture in Agile Teams

4 Conclusion

In conclusion...

4.1 Template Parameters

In addition to specifying the *template style* to be used in formatting your work, there are a number of *template parameters* which modify some part of the applied template style. A complete list of these parameters can be found in the *LaTeX User's Guide.*

Frequently-used parameters, or combinations of parameters, include:

- `anonymous,review`: Suitable for a "double-blind" conference submission. Anonymizes the work and includes line numbers. Use with the `\acmSubmissionID` command to print the submission's unique ID on each page of the work.
- `authorversion`: Produces a version of the work suitable for posting by the author.
- `screen`: Produces colored hyperlinks.

This document uses the following string as the first command in the source file: `\documentclass[sigconf,screen]{acmart}`.

5 Tables

The "`acmart`" document class includes the "`booktabs`" package — https://ctan.org/pkg/booktabs — for preparing high-quality tables.

Table captions are placed *above* the table.

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of

Table 1. Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material are found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed output of this document.

6 Figures

The "`figure`" environment should be used for figures. One or more images can be placed within a figure. If your figure contains third-party material, you must clearly identify it as such, as shown in the example below.

Your figures should contain a caption which describes the figure to the reader. Figure captions go below the figure. Your figures should **also** include a description suitable for screen readers, to assist the visually-challenged to better understand your work.

Figure captions are placed *below* the figure.

6.1 The "Teaser Figure"

A "teaser figure" is an image, or set of images in one figure, that are placed after all author and affiliation information, and before the body of the article, spanning the page. If you wish to have such a figure in your article, place the command immediately before the `\maketitle` command:

```
\begin{teaserfigure}
 \includegraphics[width=\textwidth]{sampleteaser}
  \caption{figure caption}
  \Description{figure description}
\end{teaserfigure}
```

References

[1] Alistair Cockburn Jim Highsmith. 2001. Agile Software Development: The People Factor. *IEEE Computer* 34, 11 (Nov. 2001), 131–133. https://doi.org/10.1109/2.963450

[2] Jim Highsmith and Alistair Cockburn. 2001. Agile Software Development: The Business of Innovation. *IEEE Computer* 34, 9 (Sept. 2001), 120–127. https://doi.org/10.1109/2.947100

[3] Arie van Bennekum Alistair Cockburn Ward Cunningham Martin Fowler James Grenning Jim Highsmith Andrew Hunt Ron Jeffries Jon Kern Brian Marick Robert C. Martin Steve Mellor Ken Schwaber Jeff Sutherland Dave Thomas Kent Beck, Mike Beedle. 2001. Manifesto for Agile Software Development. Retrieved January 18, 2019 from http://www.agilemanifesto.org/

Table 2. Some Typical Commands

Command	A Number	Comments
`\author`	100	Author
`\table`	300	For tables
`\table*`	400	For wider tables