Discussion of Large-Scale Agile Development

East Carolina University
Department of Computer Science
Greenville, NC

Abstract—This paper discusses the definition of agile development and proceeds to detail research and studies in large projects and companies. The merits to large scale agile development are mainly business driven: being able to adapt to developing customer needs and producing a value-add product for a company. Many projects focused on functionality end up creating a plethora of mediocre functionalities that are of no use to actual consumers as releases continue. However, the methods discussed in this paper focus on optimizing software for best use of the customer, including eliminating functionalities that are of no use. In addition, by integrating business-oriented team members the focus becomes on creating a product the customer needs and will use and therefore creates value for the project. The difficulties encountered namely include that large projects do not have the contact methods or communication abilities needed, have high levels of complexity of product ownership in a large project, and have problems related to development on a single product while many people have access to it. The research included several models and accommodations to Agile Software Development that would help to counterbalance and overcome the complications changing and volatile, concepts and projects become obsolete before their completion. The goal with scaling agile development is to continually produce market appropriate projects and encourage the willingness to scrap projects that no longer hold value.

II. BACKGROUND

A. Traditional Agile Software Development (ASD)

Understanding why companies would want to use agile development over traditional methodologies is best described with including an excerpt from the manifesto: “...we have come to value: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, responding to change over following a plan.” [10] Though all qualities are vital for the success of Agile Software Development, the fourth concept is of particular importance for business, considering the market, being able to respond to change encourages companies to focus on agile development.

One of the most used Agile methodologies is scrum, which is necessary to discuss with regard to this project. Scrum emphasizes three roles within a development team: product owner, development team, and the scrum master. The product owner, here, is not only accountable for the success of the project, but would act as a “proxy to customers to make sure that the functionality that is developed is aligned with their needs” [4]. Agile methodologies, and scrum in particular, insist on daily meetings and assignments to make the team gel and produce a great product. Part of this is having time-limited “sprints” in which an instance is created and a backlog of functionalities is considered. This backlog determines the functionalities to be produced in the next sprint [10].

This functionality, utilized by ASD, that is important to discuss in broader context. Noted later, much of the research done involves reprioritizing the backlog or making better use of the backlog to ensure functionality necessity. The backlog consists of functionalities either denoted in the project requirements or noticed during production that would be necessary or useful in order for the final product to meet the business need or goal. The concept of the backlog may not be new, but having it serve more of a central purpose to the

Keywords—agile development; large scale development; business-value; software engineering; scrum; GORE model; SAFe Model; QFT; QCD;

I. INTRODUCTION

The need for and use of software engineering has exploded in the past fifteen years. Effectively everything people interact with in their daily lives to improve efficiency is software. One of the main goals of agile development is to create a product that is not only useful, but meets the business requirements that were outlined at the project start. However, agile development has shown to be more effective in smaller projects and that to be effective in larger projects its principles would need to be adapted and modified. Identifying these modifications is crucial for creating better products to bring to consumers as well as bringing the right products to the businesses that contract them.

One paper goes so far as to state that “the most significant challenge facing 21st-century organizations will be [organizations’] ability to adapt to rapid and unpredictable change in more rapid and appropriate ways that their competitors.” The software development market is ever-
overall project rather than having a continuous selection of pieces to be worked on is important to ASD and to scaling ASD.

More broadly, agile incorporates such principles as delivering working software frequently, collaboration between business people and developers, and regular reflection. All of these activities are essential to the Agile process between and during sprints to make the process effective. Comprehending this context is important for understanding why large companies would want to embed these principles into their projects. In theory, there are frequent iterations of working software, developed using the best minds, technical expertise and collaborative methods, and integrated feedback that produces the most successful implementation of a piece of software [10].

B. Traditional Large Scale Development

Large-scale development traditionally has struggled with becoming agile. The obstacles encountered include but are not limited to lack of daily communication, project ownership complications on large projects, incorporating customer feedback to eliminate redundancies, and lack of definable stakeholder requirements to create the desired product. Large scale development, opposite of agile development, is very tuned in to market needs and trying to create a product to meet a specific market or global market pain point. However, in defining one system requirement or multiple system requirements and creating a product to meet these can create an overly complicated product, overloaded with functionality, or can create a product that takes too long as misses the market window.

Large projects typically have a lot of oversight due to the cash flow into the project and the importance of the deliverable. One of the major problems with implementing agile development into this environment, is that those same majors desire the same level of oversight and want to institute major governance systems and reporting mechanisms which are exactly the opposite of agile development. The result is a failure of the project and the rejection of agile development continuing and in future projects [1].

III. LARGE-SCALE AGILE DEVELOPMENT

Overall the benefits described when companies were successful using agile development were “accelerated time-to-market, increased productivity and quality, reduced risks and project costs.” One of the greatest benefits is team morale, with collective ownership in a project every contributor feels responsible for the deliverable and proud of their contributions to the overall success of the project. There is great push to develop and research better methods of implementing Large-Scale Agile Software Development, in fact it was voted the “top burning research question” at the XP2010 conference. This paper discusses the business values attributed to large scale development, the complications encountered when implementing large-scale development, and some of the recommendations for implementation found when researching.

A. Business Value to Large-Scale Development

One of the biggest issues is that many times stakeholders in a large project are unable to define all requirements at a project’s start. Moreover, the business value of a project can range depending on the user and can be intangible. The goal for large-scale development to continually address business needs of customers is that value is delivered not just at implementation but that it is still relevant as time goes on. This can be difficult to deliver on, however agile development goals and methodologies can help with this because instances continuously evaluate effectiveness and can throw out old functionalities if they are deemed ineffective.

In addition, the user stories aspect of agile is very useful when considering large-scale implementations. Rather than using explicitly defined and lengthy requirements, the user simply describes the needs, or business pain points, the product needs to deliver on. A business team member is present when all major decisions are made to describe what the product ideally should do and determine if what the developers are picturing matches what the project goal is. If at any given time it is not going to succeed in that goal, this is easily identified through evaluation in the context of the user story. One suggested implementation of user stories was to have user stories at each phase of development. Rather than focus on the whole product, when functionalities are pulled from the backlog that functionality should be defined by a user story and pursued [5].

One of the biggest issues with large scale development is the concept of ‘software bloat.’ This is where many functionalities are developed because they were initially described as useful within the project. Then, because more functionalities are identified for the next instance, the functionality is either underdeveloped and not useful, or the goal of the project has been re-described and the functionality is now obsolete but left in the system. Under many of the models and projects researched, this would be eliminated because the functionality would be discussed and omitted with a business decision maker present and a project owner continuously evaluating the usefulness of the project. A related, but perhaps equally important value-add component of ASD is the integrated customer feedback. Using frequent iterations to ensure that the requirements being developed are those that are used requires the use of studies of end-user viability.

Another with large scale development as compared to smaller projects is timeline and streamlining. As a project gets larger with more contributors and a given team gets bigger, keeping with deadlines can be increasingly difficult. However, this is incredibly important in software development as missing a market window can actually force a product to be outdated before it even gets released and all costs related to the project are effectively sunk.
One of the “business reasons” for not developing large-scale projects in the necessity of clear and achievable ROI, which is simply not always the case when developing software. In most cases huge amounts of risk are involved with whether the product will be first to market and achieve greatest value or if the product will not meet the business need it was developed for or even if the business need thought to be met wasn’t a pain point for any other companies. Developing a clear business case can sometimes be the biggest hindrance in getting a large-scale project going. Though ASD provides a faster route-to-market, this risk can sometimes prove insurmountable[5].

Many documents tout the value of ASD as having faster time-to-market, representing more accurate customer requirements, and having flexibility to adapt to changing business needs and customer desires [9]. The product owner is vital to representing these needs and ASD meets these requirements immensely well by actually integrating and embedding a business person into the project and development.

B. Complications to Large Scale Agile Development

There are various levels of complexity attributed to large scale ASD. Not the least of which is the added layer of additional departments in large corporations such as marketing, human resources, legal and operations. It would not be efficient to include an individual from each of these departments to interface with the development teams at all times or even necessarily regularly, however each of these is vital to company success.

A primary consideration of large-scale agile development is that no single product owner can represent all requirements and factors of a project[6]. Similarly, if a project has multiple intended audiences or a large customer base a single owner cannot represent all of these interests either.

One study had some interesting findings when conducting a systematic literature review of Large-Scale ASD that found the categories of challenges are change resistance, lack of investment, agile difficult to implement, coordination challenges in multi-team environment, different approaches emerge in multi-team environment, hierarchical management and organizational boundaries, requirements engineering challenges, quality assurance challenges, and integrating non-development functions in the transformation [6]. An interesting note about these challenges is that the majority of them relate purely to the scaling problem. Adding to many minds to these projects tends to force deviation from the goals of ASD. Also, there are generically issues with implementation related to people’s misunderstanding of ASD and its usability.

This same team did a study of the challenges and success factors of ASD and found the top three challenges of implementation to be:

1) other functions unwilling to change (31% cases),
2) lack of guidance from the literature (21%),
3) misunderstanding agile concepts (19%),

4) reverting to the old way of working (19%)" [7].

These findings are interesting in that it is not consistently that ASD is not conducive to large-scale projects but rather that the implementations are difficult and sometimes prove to be greater than the benefits ASD provides for.

Some additional challenges, recognized in a separate study are: the 'open loop' problem, requirements being seen as truths, lack of validation of feedback and large amounts of (useless) data. The open loop problem represents an inability to retrieve accurate customer data. The problem that arises when requirements are seen as truths is that there is a great hesitancy to stop development on a feature if it was originally described by the customer, even if it ultimately does not fit the goal of the project. As a result, great amounts of time may be wasted developing a feature that ultimately provides no benefit to the project. Finally, the push for big data has provided large amounts of analytical data. However, if the push for this data related to these projects is not asking for the correct questions about these projects that data is rendered not only useless but it takes up team members' time and energy that could be better spent elsewhere [9].

Implementations also noticed that the level of customer involvement required by Agile was simply not possible in large projects. To this point, without accurate customer involvement teams were typically shooting in the dark and guessing and requirements resulting in potentially great products that simply were not market necessary [6].

A final issue is that of distributed teams. Typically, large organizations are either multi-state or multi-national organizations with members telecommuting or working independently of one another. This makes the goals of weekly or daily interaction and updates slightly more difficult and in some cases insurmountable. Without these team members’ input, others make assumptions as to what needs to be accomplished and more often than not they guess wrong [6].

IV. RECOMMENDATIONS FOR IMPLEMENTATION

Though these recommendations are not comprehensive and will not solve every problem associated with large-scale agile development they do attempt to overcome some of the major issues. Research was done within one company applying some of these methods to two different projects within the same company: one with agile and one without. It was noticed that in the agile project the iteration planning meetings gave insight and greater knowledge to what other teams were working on and created a bigger picture view of the project than what the individual teams understood, this was seen as a great benefit and made the work easier. In addition, they felt the cross-functional understandings and communications made for a more seamless integration and allowed team members to feel empowered to take on different jobs. Going along with that was that overall project visibility was greater with agile development. Finally, the pressure felt by team members was greater in non-agile development due to
the lesser visibility. These benefits make the exploring of adapting agile principles to large-scale projects worth researching [3].

A. The SAFe Model

There were several suggested implementations that would make large-scale agile development successful. One of these was the Scaled Agile Framework Maturity Model SAFe MM). SAFe is effectively an organizational method that integrates business units into a development-oriented organization. The three main functional areas are the team level, the program level and the portfolio level. Each is aligned with the strategic vision of the overall project and utilizes ART (Agile Release Train) to produce iterations synchronized with business unit functions. The SAFe Model recognizes the conflicting goals between business needs and development needs which are:

- “maximizing the financial value of the portfolio
- linking the portfolio to the strategy of an organization through investment themes
- ensuring that the scope of activities is feasible by measuring appropriate metrics, and
- balancing the portfolio on relevant dimensions by defining and managing business and architectural epics, which run across value streams” [2]

One major pitfall of large-scale ASD that SAFe Maturity Model also encompasses is the sometimes inaccurate representation of customer requirements. The suggestion for rectifying this situation was having story owners that were not required individually during all phases of development but rather having a single story owner that changed during development. This story owner would represent the current requirements being worked on while the other owners could go back to their other positions within their respective organizations [2]. This benefits everyone involved because the company is not wasting business talent sticking them in a development team for extended periods of time and the development team is confident in the project they are working on because the appropriate person is on the team at the time any given requirement is being worked on. Having an “owner” of a user story and requirement makes this possible.

In addition, these same researchers suggested that Agile perhaps did need more comprehensive documentation to correctly implement and build upon another’s’ improvements. Though less documentation is a highly regarded principle in ASD, when creating such a large project with many contributors, such as an open source project, it is just not realistic to not have comprehensive documentation.

B. The QCD Model

Another recommendation, depicted above, is described is the ‘QCD’ (Qualitative/Quantitative Customer-Driven Development) Model [9]. What the image is showing is the ultimate hypothesis of the project which is integrating big data into feedback testing and backlog prioritization. This is rather remarkable in its ability to assess business value of functionality. What it attempts to conquer is integrating the customer feedback into development before alpha/beta testing.

Its ultimate goal is that features are re-prioritized in the backlog as a limited number of users are allowed access. It views requirements as a hypothesis of functionality rather than a full description. In following with this, the hypothesis is tested through multiple iterations with continuous user feedback to produce an ideal product. Though a great model, it can be difficult to implement, particularly in large scale projects. However, with business users involved in development they can become such testers to provide feedback.

Integrating user feedback early enables a project to be sure that it is accurately meeting business pain points. As discussed, many projects are hesitant to abandon functionalities even when they are deemed unnecessary. A system more often is overloaded with functionalities that can make it more complicated to use than streamlined and effective. By reprioritizing the backlog and abandoning projects deemed ineffectively early the QCD model assists the ASD process.

C. Applying Sociocratic Principles

One researcher suggested conquering organizational discrepancies associated with large-scale ASD by applying sociocratic principles to software development. The ideas suggested in this study were: using consent for team decisions, aligning teams with functionality, and connecting agile development to the organization. The idea is that the team itself is reflected well to the larger organization, and the goals of the larger organization are in turn reflected well to the team.
Having a counterpart or point of contact in the larger organization is imperative to double linking the development team to the larger organization. This method was intended to be implemented to combat both methods of utilizing ASD: when a large company is transitioning to using ASD and when ASD was the primary process for software development in a large organization. In both, it was found that using consent for team decision making allowed the product owner to share responsibility with the team overall. If the team "bought-in" to the project and the ASD methodology, the project had a higher likelihood of success. One of the typically used, but ill-defined, procedures outlined was that of double-linking and aligning teams depicted above. Rather than having a single team member from the business unit be involved in the development process, multiple members were double-linked as appropriate to act on behalf of their particular team. From this perspective, someone from marketing, sales, or operations that have field and analytical experience could be involved. Though this could create more bureaucracy, it ultimately proved a more efficient method of keeping the organization connected and the project connected to market feedback.

D. The GORE Model

A final suggestion was that of GORE (Goal-oriented requirements of engineering). The main benefit that this model suggests is not having a single business member integrated in the team, but consistent communication and support from upper management. It goes so far as to advocate for a "B-scrum," which would act as a steering committee that works with the various instances and reevaluates business needs and works with the development the create value out of the product. Some of the described benefits listed, not comprehensive are: "achieve requirements completeness, avoid irrelevant requirements, explain requirements to stakeholders, explore alternatives, and relate requirements to organizational and business context". [3]

As compared to the double-linking suggested by the previous researcher this method advocates having an entire team dedicated to connecting the development teams to the overall organization. This B-Scrum team meets with members from the management and other business units to be able to better understand and relay business needs and market developments to the development teams.

E. Providing and Accommodating Agility in Context

All of the suggestions researched and described include altering ASD to be able to better accommodate a larger projects' needs and organization. The main pain point seems to be reflecting business value. A development team can develop a great product, and there be no market for it. Conversely, there can be a great market for a product but no development process to accurately meet these needs. However, including members from the larger organization and keeping open lines of communication of feedback seem to counter the negative aspects that ASD can have on a big project.

These observations were noted in the systematic literature review that surveyed current research in the large-scale ASD field. The top factors that enabled success for companies that implemented ASD were:

- "Ensure management support (29%)
- Coach teams as they learn by doing (29%)
- Customize the agile approach carefully (26%)" [7].

There are two main things to note about these success factors. They ultimately are comprehensive, meaning that these truly were the main problems that people discovered when implementing projects using ASD, but also that ASD on the surface probably isn't ever going to work large scale. However, by utilizing some of the methods outlined in the research and customizing them to the project goals or team it is likely that ASD can provide the benefits seen in small groups.

V. SUMMARY

Overall, there are decidedly some drawbacks to choosing to start a large-scale ASD project or attempting to implement ASD elements into a large-scale project. However, over the course of
the research studied, it became increasingly evident that the main issues were with the implementation and with misunderstanding of the underlying principles. Typically, the principles were not the disrupting factors in Agile. Human error, inability to continue communication as required with Agile, or inability to utilize customer feedback data effectively were the main consequences of these problems with implementation.

Moreover, several methods of adapting Agile to large-scale have been researched and presented that would fix these problems. By utilizing team members involved in both the development and larger organization needs could be effectively met. In addition, using the big data typically generated in a larger organization on customer feedback. In addition, Agile presents a more efficient and business value-add proposition than traditional large-scale development.

REFERENCES


