Scaling Agile at the Program Level in an Australian Software Vendor Environment: A Case Study

Ramesh Lal
School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
Email: ramesh.lal@aut.ac.nz

Tony Clear
School of Engineering, Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
Email: tony.clear@aut.ac.nz

Abstract

Set in the context of an Australian software vendor for a global enterprise product, this paper explores the roles and practices that have evolved as the company scales from its already well established agile software teams, to agility at the enterprise level. With a focus on roles and practices at the program level within a Disciplined Agile Delivery framework, this study adds to the limited body of research into the process and impact of scaled agile approaches in software vendor environments.

Keywords Scaled Agile Frameworks, Program Management, Disciplined Agile Delivery, Software Vendor, Software Process Improvement.

1 Introduction

Since its inception in 2001, an agile approach has enabled software vendors to undertake software process improvement (SPI) activities through *agile adoption* and *adaptation,* and now through adopting a *scaled agile framework*. The focus of this paper is on adoption of the *scaled agile framework*, helping software vendors to realize the benefits of agility at enterprise level without being overwhelmed by regulations and rigidity (Fitzgerald et al., 2013).

Factors that drive agile adaptation in a software vendor environment have been captured through an agile approach adaptation framework proposed by Lal (2011) based on his investigation of two major software vendors. His investigation reveals that for successful SPI the agile approach adaptation must be mutual and supported by cross-functional effort in a software vendor environment.

For software vendors driven by an agile approach the scaled agile frameworks appear to be the next option to enhance their agile product development environment for large-scale software development. The scaled agile framework aims to expand throughout the organisation the agile mindset and agile ways of working. There are a number of consultant-driven scaled agile frameworks such as Scrum of Scrums (SoS), Large-scale Scrum (LeSS), Scaled Agile Framework (SAFe), Disciplined Agile Delivery (DAD), Large Scale Scrum (LeSS), Spotify, Nexus, and Recipes for Agile Governance in the Enterprise (RAGE) (Alqudah & Razali, 2016). Although practitioners are active in scaling agile methods (Ambler & Lines, 2012; Kniberg & Ivarsson, 2012), academic research into scaled agile frameworks has been relatively limited (Paasivaara, 2017; Fitzgerald et al., 2013).

This research investigates a major software vendor, based in Box Hill Melbourne, and their adoption of the Disciplined Agile Delivery (DAD) framework (Ambler & Lines, 2012) to provide an understanding of the reasons for adoption; and to highlight the organisational and functional changes required at the program management level to successfully adopt a scaled agile framework. Such understanding is topical, as practitioners experiment with scaling agile across the organisation, with Freudenberg and Sharp (2010) having noted that "*the agile practitioner community would like to look at complex multilayered questions*" among them agile methods in distributed settings. This study addresses these questions through the challenges of scaling agile methods within an organisation, and may benefit other software vendors seeking to identify and plan strategies for successful adoption.

2 Literature Review

At present, there is limited academic research on scaled agile methods (Paasivaara, 2017), although the practitioner literature is quite active in the area (Ambler, 2012; Kniberg & Ivarsson, 2012; Leffingwell, 2016). However, it is expected that interest in academia will grow since some of the scale up methods such as SAFe method have started to become popular (Stojanov et al., 2015). Paasivaara provides a case study highlighting adoption issues and challenges with the SAFe framework in a globally distributed software vendor environment. Whereas, Stojanov, Turetken, & Trinekens, propose a SAFe maturity model, which identifies five maturity levels that from the lowest to highest are collaborative, evolutionary, effective, adaptive, and encompassing. Both these studies highlight the challenging nature of the adopting the scale-up framework. In Contrast Fitzgerald et al., (2013) show the successful adoption of an agile method they termed "Regulated Scrum" by an Irish software vendor.

According to Laatni (2017), the external pressure to be competitive together with the impact of digitalization are two main reasons for organisations to adopt large scale agility. In her study she states that SAFe, LeSS and a few other frameworks are useful starting points to bring discipline to large scale software development. She points out that scaled agile methods are just a "recipe for transformation" and it's the level of agile transformation which leads to success. Her study provides an agile transformation model which is based on Team, Program, and Portfolio levels and which goes through five levels of transformation- beginner, novice, fluent, advanced, and world-class.

Alqudah & Razali (2016) through their research provide a list of scaled agile methods, identifying phases, practices and roles for each. In their list are the following scale up methods. Disciplined Agile Delivery method (DAD), Large Scale Scrum (LeSS), Scaled Agile Framework (SAFe), Spotify, Recipes for Agile Governance in the Enterprise (RAGE). Most of these scaled agile methods are consultant developed and driven. However, 'Spotify' is the only scaled agile method that appears to have been developed by practitioners (Kniberg & Ivarsson, 2012).

While Alqudah & Razali do not include the authors of these methods, http://davidfrico.com/agile-quagmire.pdf provides the author including year of publication in regards to these scaled agile methods, shown in Table 1.

Table 1 Scaled Agile Methods (ex. Alqudah & Razali, 2016, & http://davidfrico.com/agile-quagmire.pdf)

	Who	When
DAD	Scott Ambler	2012
LeSS	Craig Larman	2008
SAFe	Dean Leffingwell	2011
Nexus Method	Ken Schwabe	2011
RAGE	Kevin Thompson	2013
Spotify	Henrik K Kniberg & Anders Ivarsson	2012
Enterprise Scrum	Ken Schwaber	2007
ScrumPLoop (Scrum Pattern Langaues of Program)	Jeff Sutherland	2008

In their review of scaled agile frameworks Alqudah and Razali (2016) identify key DAD 'roles' which they break into primary and secondary and DAD 'practices' across a three phase lifecycle. This provides a starting point for our investigation.

3 Research methodology

This investigation collected data directly from practitioners of a scaled agile approach based on predefined research questions (Paré & Elam, 1997). This research sought to explore the evolution of roles and practices (Alqudah & Razali, 2016) within a scaled agile setting. That initial focus was used to formulate the research question for this study. It emerged from the study that the company had chosen to adopt the Disciplined Agile Delivery (DAD) methodological framework (Ambler & Lines, 2012) to aid them in scaling their agile delivery. As the study proceeded, the emerging roles and practices proved to range across all levels of the broader DAD framework. Given the limited focus in the literature to date on agile above the team/project level, the research question was then narrowed for this paper to focus on those roles and practices emerging at the DAD program management level.

While scaled agile approaches are relatively new phenomena, it is critical that the DAD method, like any other scaled agile method, is also well understood since agile has become a mainstream development approach. With limited prior academic research, a qualitative research approach was adopted to investigate the nature of DAD adoption from a software vendor's direct experience. Hence, we adopted a case study method, using open-ended interviews as a data collection technique (Patton, 1990). A case study can benefit from a prior developed *theoretical proposition* (in this case at an intermediate stage the DAD framework was selected, with a focus on the program management level) providing a scientific basis for the research (Yin, 1994). Software Engineering has been criticised for paying "*little or no attention to theory development, [with] very few studies…based on existing theories*" (Stol et al., 2016). DAD is a practitioner developed framework but it does provide a useful grounding for a "*knowledge seeking*" (ibid) study such as this, which hopes to "*form the basis for a middle-range theory*" (ibid.).

The main research question for this study then became:

> How does adoption of DAD work at the program management level in a software vendor environment?

The case study method was determined to be the most appropriate and best suited qualitative method for this study, since we deemed the case study to match the Runeson and Höst (2009) categories of "*Exploratory—finding out what is happening, seeking new insights and generating ideas and hypotheses for new research [&] Descriptive—portraying a situation or phenomenon".*

For this research data collection activities took place over a period of 12 months with the first interview held in November 2015 followed by eight interviews in April 2017.

3.1 Case study selection & context

The selection criterion was that software vendor(s) must have adopted and have delivered at least one multi-project program of work using a scaled agile method. Upon our request, a software vendor from Box Hill Melbourne agreed to be part of this research. This vendor organisation had adopted DAD methods in 2016, having more than a year's experience in delivery of projects with this scaled agile method. It had adopted an agile approach for software development since 2003, including having significant experience in adapting their organisational structures and agile development process and practices over the years to maintain its status as a global market leader with their software products.

3.2 Study protocol

The study was guided by a protocol as recommended by Runeson & Höst (2009), which involved creating procedures for data collection and organisation to facilitate analysis and report writing.

3.2.1 Data collection

A request was made with the Director of Software Engineering to conduct the interviews with the candidates, which we identified based on the various software engineering roles they have. We ended up conducting eight interviews over a week period in April 2017. The roles we interviewed were Principal Engineer, Senior Software Engineer, Team Leader, Engineering Manager, Product Owner/UX Engineer, Quality Assurance Manager and Director of Software Engineering (did two interviews).

All the study participants were part of a DAD transformation from their previous agile approach. Senior business executives and the newly appointed program manager were not selected for interviews because of their busy schedules, but the Director of Software Engineering, a senior executive who had until recently occupied the program manager role, covered for this gap. The interview instruments and key themes based on which individual interview questions were formulated were guided by: reasons and approach for switching to DAD method and on its adoption - focussing on the roles and practices adopted for DAD implementation. These roles and practices were identified primarily at team/project and program levels.

Each interview session was planned for an hour. Interview sessions were held in the meeting places at the production lab of the participants. Questions were formulated as single questions. Interviewees were informed of their right not to answer any particular question they were not comfortable with. Participant anonymity was also ensured. Agreement was made on recording of interviews and individual consent was taken before a session began. These recorded interviews were later transcribed. Recording enabled accuracy for data collection and to be more focused on the interviewee (Patton, 1990).

3.2.2 Data Analysis

Interview data was analysed by searching through each script and identifying instances of key practices and roles in operation. This data mapping produced the sets of key roles and practices presented in section 5 below, demonstrating the evolution of roles and practices as the case study organisation implemented the program management level of DAD as a scaled agile approach.

4 Case study background

The software engineering vendor based in Box Hill Melbourne, was a world market leader with its product till 2010 when it was acquired by another major software vendor. With this partnership, they became one of the fastest growing global software vendors and recognized in the software industry as a market leader and a provider of highly intelligent asset management software. The clients for their software products are mostly Fortune 500 Companies, including 80,000 global companies in different sectors. Their head office is based in USA and have software engineering units in USA and India.

The engineering department was set up in 1990 with a small team developing on-premise (packaged) software. The company underwent challenges with resourcing, estimating for a very complex product, consistency of delivery and quality of releases, prior to a migration to agile methods driven by the software engineers in 2003. Since that time, agile methods have been embedded in the organisation with a hybrid approach including adopting software as a service delivery model (SaaS). These developments occasioned changes in structure with the introduction of "squads" (Kniberg & Ivarsson, 2012) into their development teams.

The SaaS approach posed the challenge of how to provide an excellent user experience (UX), with UCD (user-centred design) driving the product development, including hiring a senior UX person. A UX mindset was a big transition in their software engineering unit at Box Hill, Melbourne. As a result, they have identified different personas and written scenarios on how the users based on these personas will use their product in their client organisations. While their product development had been driven by the business objectives, considering the user objectives was also critical and frequently in conflict when it came to UX design for their product.

4.1 Adopting a scaled agile method- DAD (2016)

With this software vendor, DAD (Disciplined Agile Delivery) adoption was driven partly due to quality issues experienced by their clients with their SaaS offering. The DAD adoption was driven top-down through their Senior Vice President Engineering, proposing DAD as an approach that might help the entire organisation to grow quality and enhance their ability for future product innovation. Importantly, DAD actually brings the business level planning (business strategy and product planning) and software engineering planning (program, project and development) together, aligning the two with each other.

> *We didn't make that decision, it was made for us. But having said that, we still have a degree of team independence, being able to decide how they do things. Sort 50-50, some things are mandated, but we do get to make decisions. (Principle Engineer/AO)*

The approach taken here for DAD transition was that the consultants from USA were brought to train and coach the engineering unit at Box Hill. However, in the true DAD fashion of self-organisation, the structural change in the engineering unit became a bottom-up process, which went smoothly with significant structural changes adopted within the engineering unit.

> *About openness and honesty, and doing everything together … we did this transition into discipline agile stuff, we got Scott Ambler here to train us and we got his agile coach.*
> *Self-organized the team structure, so got into a big room, and we had a blank white board, and we gave everyone a post-it with their name, and we said "let's build the team structure form the bottom up … discussion just kept happening on the whiteboard until we arrived at a new team structured and filled those*

5 Program management

The data analysis highlighted a number of practices and roles, across many levels of the three level DAD framework (Portfolio, Program and Project/team). The company's prior focus on agile practices had been largely at the development team level, as has been the majority of the academic literature (Paasivaara, 2017). Therefore, we chose to focus on the relatively unexplored area of program level practices and roles. The practice mapping framework of Kirk and Tempero (2012) is chosen to depict the practices identified from our study, in Table 2 below. These practices are elaborated in section 5.1 ff.

Table 2: Agile Practices Identified at the DAD Program Level

Practice Category	Practice Sub-Category	Practice in Use by sub-category	Practice In Use Across Categories
Define	Roadmap	Product planning (portfolio and release planning) Feature Funnel Practice	Program and knowledge management – software tools Self-organising teams T skilled engineers Definition of Done
	Scope	Work item list	Program and knowledge management – software tools Self-organising teams T skilled engineers Definition of Done
Make	Design		Program and knowledge management – software tools Self-organising teams T skilled engineers Definition of Done
	Implement	Program daily tactical huddles	Program and knowledge management – software tools Self-organising teams T skilled engineers Definition of Done
	Integrate	DevOps Practices	Program and knowledge management – software tools Self-organising teams T skilled engineers Definition of Done
Deliver	Release	Program Delivered through quarterly release plan Program iteration release show/tell practice DevOps Practices	Program and knowledge management – software tools Self-organising teams T skilled engineers Definition of Done
	Support	DevOps Practices	Program and knowledge management – software tools Self-organising teams T skilled engineers Definition of Done

In table 3 we repeat this mapping with a focus on the roles involved in the practices, again the roles are further elaborated in section 5.2 ff.

5.1 Program Level Practices

5.1.1 Product planning (portfolio and release planning)

Their DAD adoption brought the product planning closer to the engineering. Previously, their product planning team met every six months in USA to make implementation decisions, resulting in a portfolio of high-level requirements. Now, with DAD adoption a separate program management team has been setup and is co-located with the engineering to do product planning and to set product direction in each development team. The **program management team** consists of the **program manager** (PM), **product owners** (PO) and **enterprise architects** (the DAD roles that they now have adopted.

[Product managers] have an input into product management. Product management … direct the product vision. Embracing the DAD framework, we put a product management role in every local team … they are the owner of the product direction… works with the team… the product management team, based at the US, doing more of the strategic looking out.

Based on the DAD method, this software vendor now has their business level planning driving their software engineering work. Their strategic planning, done at the business level, provides up to a three-year vision for their product. The vision plan is the basis for their portfolio and release planning done by the program management team. The portfolio has a collection of prioritised business value high-level requirements for up to a one-year period. Release planning, covers a quarterly release, and produces a storyboard with user stories based on specific high-level requirements picked from their portfolio.

5.1.2 Feature funnel practice

A feature funnel practice is performed at program level that allows program management to collect, measure, and understand feature requests and enhancements that come from the product management team. They can then build and enhance their product based on business and customer needs, and classify a new feature or an enhancement based upon its business value.

Got this feature funnel and twice a week, the PM and the key people are meeting, looking at the feature funnel, saying what's the next most important thing, who is going to do the fleshing out of that storyboard, who's going to look at enterprise architecture, and that's forming a road map. Every 6 months, we have the session where all the key senior people, all come together in Melbourne, and we take a look at our overall road map and say where are we with that, what are the commitments we're making with that at the big picture.

The funnelling work on business value features is done well ahead of the release cycle the features will be in. Hence, make sure enough work is done and understanding is there, meaning less time and effort spent during the inception phase to learn and minimise risks. Now, the UX and UI come out through the funnel practice just like any functional requirements, captured through user stories. Some requests are directly from the customers for a new feature or for an enhancement. This requires the UX consultant (now one of the POs) to investigate and learn from people who fit their persona and write users story that will into the storyboard.

5.1.3 Program- delivered through quarterly release plan

Since the DAD adoption, their development teams are structured into four separate DAD teams. These teams may work on any user story in their current storyboard. Prior to the DAD adoption, their teams were based on the functional lines of their product, which limited each team's ability to be able to delivery user stories relating to different functional areas of their product.

5.1.4 Program- daily tactical huddles

The daily tactical huddle is an important practice for their program management. They get a project status report from each team and identify fixes if any to help a team if there is any issue, support another team to catch-up if they have fallen behind or even help solve an implementation problem etc. The key stakeholders for this meeting are the three leaders from each of their DAD teams and program management team.

5.1.5 Program – iteration release show/tell practice

This practice requires POs of their DAD development teams to present the completed user stories in a demonstration. These are the done items, which now require reviews and reflection upon the risk items and results of test runs from their program management and stakeholders.

5.1.6 Program and knowledge Management – software tools

The entire program is managed using Jira and Confluence. These tools allow for effective planning, execution and management of each quarterly release. These tools provide the ability to create and manage information on programs including their code base. Knowledge management is critical for reliable plans and learning of their code base by new engineers. Jira allows the program management to create visibility in their program by organising and coordinating the four DAD teams and their schedules, allocating and prioritising user stories. Risk management, tracking data on releases, information on program and project retrospectives are kept with Confluence software.

5.1.7 Self-organising teams

Their development teams have adopted the DAD philosophy to be enterprise aware and have a "people-first", approach. They are required to be self-disciplined, self-organizing and learning oriented. Hence, each team is empowered and expected to constantly strive to achieve the best possible results. However, each team has to embrace DAD practices, deliver iteration releases based on their definition of done and must use Jira for tracking their project. However, DAD adoption experience with this software vendor also suggests that tools such as Jira may also prevent individuals to talk face to face since everything is logged into Jira.

5.1.8 T Skilled engineers

T Skills concept brought another major shift in their engineering department with DAD adoption. Now engineering individuals must have in-depth (specialist) skills in a particular technical area the business needs, including a wide breadth of engineering skills to able to perform analysis, design, code, test (QA tasks), document (technical writing), and deployment knowledge. When this software vendor adopted cloud deployment, they shifted from generalist to squad based development teams to develop a variety of specialist skills in-house. With DAD adoption, the QA and UX engineers require adaptation to be T Skilled. Here, the goal is to integrate all the skills of a software, QA and UX engineers into a single technical role. Here, technical writer role is seen having unique skills, specific to a writer.

5.1.9 Work item list (product backlog)

DAD has brought a change in how the analysis and design work is now done to create the team's work item list. Now, most of this work is done at the program level. The feature funnel helps to identify business value features which go on to the storyboard, where their program management team will do an appropriate level of architecture planning and flesh out the user stories, creating work item list. The other items could be bugs experienced by customers and requests for enhancement. Hence, with a DAD approach the product backlog is referred to as a work item list since it has more than just user stories relating to new features. The program management based on business needs directs priority setting in the work item list.

5.1.10 Definition of done (DOD)

Their DAD teams must identify and log in their DOD criteria, applied to every user story in the work item list to get a tick for their completion. Here, DOD criteria and acceptance tests must be applied and met for a user story to be labelled as complete. If not, then the program sees that user story as not done.

> *Defects have been found and fixed. All testing have been done, and full automation of that testing to the extent of what we say needs to be automated, manual testing have been done, performance testing [done], documentation is updated, so the online help is updated and ready to go to release,. If not then the story is not done.*

5.1.11 DevOps practices

Their DAD teams have adopted a DevOps culture requiring an awareness of operations practices, having a complete picture of the entire lifecycle, from design to production. Hence, the goal is that they can also build a continuous deployment capability.

5.2 Program Level Roles

Table 3 Agile Roles Identified at the DAD Program Level

Practice Category	Practice Sub-Category	Practice in Use by sub-category	Roles Involved
Define	Roadmap	Product planning (portfolio and release planning) Feature Funnel Practice	Product Owner Enterprise Architect Program Manager

	Scope	Work item list	Product Owner Enterprise Architect Program Manager Team Lead/member Architecture Owner
Make	Design		Enterprise Architect Architecture Owner Team Lead/member Product Owner
	Implement	Program daily tactical huddles	Team Lead/member Product Owner
	Integrate	DevOps Practices	Team Lead/member
Deliver	Release	Program Delivered through quarterly release plan Program iteration release show/tell practice DevOps Practices	Program manager Product Owner/ Team Lead/member /Architecture Owner/Independent QA Team Lead/member
	Support	DevOps Practices	Team Lead/member

5.2.1 Program delivery team: primary roles

Based on the DAD philosophy, their development (delivery) teams have primary and secondary roles. In each team, they have three leadership roles- an **architect owner** (AO), a **product owner** (PO) and a **team lead**. Other nine individuals in the primary roles are mostly software engineers but also include a technical writer and a QA engineer. Here, all team member must participate to carry out the spikes to minimise risks, plan product and sprint backlogs, design the architecture, implement user stories, carry out the quality assurance tasks, implement tests, fix bugs and broken builds, and do documenting, throughout the project.

Earlier I was doing mainly development, ... I know the entire process ... worked earlier with agile ... , it's kind of the mindset, it's kind of roles and responsibilities ... work from the start to end, development, testing, documentation, getting all that done ... you're responsible to what you're doing and to be transparent to other .. different of mindset in managerial, they have roles to facilitate us, PO for what, AO for how

5.2.2 Program manager (PM) role

This is a new senior engineering role based on the DAD paradigm, which is to ensure strategic goals are achieved through a program. Hence, this role is responsible for planning and governance to oversee successful delivery of features for every quarterly release. Here, the program manager works with senior engineering individuals such as the POs, enterprise architects and team leaders to manage the program. Currently, this role has been filled by an individual with almost two decades of local (this vendor) project management and product development management experiences. Here, the program manager requires strong leadership skills to meet quarterly release goals.

Program manager is helping ensure that as an overall program, we're collaborating together, are we having a daily coordination meeting with the PO, AO, and team lead of each team in the program, are we doing program level retrospective, are we managing our program level risks and issues.

5.2.3 Product owner (PO) role

One of the primary roles in their DAD development teams is that of product owners. This role is also part of program management. A senior role and represents product management team, helping to deliver the programs through development projects. Individuals appointed in this role have substantial local (software vendor) product knowledge and most of them have worked for an extended period in software engineering or product management teams. Hence, POs help with program management through portfolio and release planning, and getting the work item list ready for their development teams before the start of the inception phase.

5.2.4 Enterprise architect (EA) role

The enterprise architects work at the program level. They are responsible for formulating an architectural vision, which provides direction for their DAD development teams. Their aim is to flesh out enterprise architecture, which is adaptable and easily grown. This role is different from the architect owner role that is part of their DAD teams. Here, it is a rotating role, whereby senior engineers are given, the opportunity to work in the enterprise architect role.

5.2.5 Team leadership roles

Having three clearly defined leadership roles within their DAD development teams is a big shift from the single leadership (product development manager) role they had prior to DAD adoption. Here, these three leadership roles are work related. Their **team lead** facilitates practices and tasks to deliver their iteration release goals. Their **architecture owner** guides their team to identify and mitigate the technical risks. The **product owner** represents the product and program management teams to get the features they want implemented. Here, they have a clear separation of their functional management from management of their DAD teamwork. While training was provided, some of their software engineering managers found it challenging to adapt to their new leadership roles. They had to learn new management skills whereby they had to step back to allow the team to step up and take initiatives. Another challenge was learning not to step on each other's toes when in these new leadership roles.

5.2.6 Team lead role

The DAD team lead is the person who is responsible for ensuring that the team is collaborating and communicating, and following the DAD processes. The team leads with other AOs and POs attends their daily tactical huddle, where all the program leaders get together and discuss their cross team collaboration issue. After their tactical huddle, the team lead must ensure that there is a team daily stand-up meeting. The team lead also has to track the team's progress using Jira and report that to the program manager including presenting to the program the team's progress and issues.

5.2.7 Architecture owner role

Architecture Owners (AO) works closely with the Enterprise Architects at program level. The main responsibility is to identify all user stories, which have risks and run spikes with the team. The AOs also get heavily involved in the construction phase- sometimes, as new information emerges it requires changing the architecture as well. Here, AOs have ended up doing more coding due to the lack of in-depth technical capability in teams.

5.2.8 Secondary roles

The secondary roles in each team is filled by a quality assurance engineer, introduced temporarily from their separate QA team to carry out independent testing that validates the team's work throughout an iteration cycle. This testing by the independent QA engineer is in addition to testing done internally by the team and the embedded QA engineer. The independent tester is also part of iteration demonstrations.

6 Discussion

We posed the research question "How does adoption of DAD work at the program management level in a software vendor environment?"

This case study tells us that scaled agile method adoption in a software vendor environment must have executive support. Scaled agile methods bring significant structural and process change, but for organisations with a prior commitment to and experience in agile approaches, scaled agile practices and roles can swiftly become a norm without resistance. Executives must champion the adoption and organisational transformation with an appropriate change management plan. The goal through any agile scaled method is an organisational shift to better align the technical activities with the enterprise direction, "*with the ultimate goal of enterprise agility*" (Ambler & Lines, 2014).

This case study shows that the DAD method can integrate the software vendor organisation for a collective effort by having a program management approach, which aligns the business function with software engineering tasks. Past criticisms of agile methods were in its limitations in going beyond the project level to provide support for long-term product development (Fogelstrom, Gorschek, Svahnberg, & Olsson, 2009).

We have learnt that the DAD method provides a systematic (program) approach to deliver software products to the marketplace. This includes modern and enhanced processes and practices with structural changes (units and roles). Hence, at program level there are clearly defined procedures, roles, and practices in relation to program management. Importantly, our investigation shows that at program level, roles are at senior level and individuals filling in these roles are mostly local (highly) experienced software engineering personnel. Increasingly the need for 'T Skilled Engineers' was being seen, with squad structures enabling specialisation in specific areas, while demanding breadth across the lifecycle. Overall, a program management team requires highly skilled and capable individuals.

Through program management, the DAD method brings balance by considering every other stakeholder's needs. We have also learnt that with the DAD method high quality features are deployed. The work done at program level ensure business value features are identified, and the complexity and variety with features are minimised resulting in shorter implementation time.

Our investigation shows that the DAD is plan-driven, in fact it clearly identifies planning practices at product level. The product owner role is the glue between program and projects, providing the daily working relationship and visibility between program and project levels. We have also learnt that with DAD method tools are useful to manage a program and projects, but they can have a negative impact on face-to-face interaction and collaboration. This side effect needs to be carefully managed.

DAD teams deliver programs through self-contained teams, which have key individuals and stakeholders for spontaneous collaboration and swift decision-making. The DAD method defines three leadership roles in a team but these roles all are task and project related. We did not observe specific practices related to people management as recommended by DAD (http://www.disciplinedagiledelivery.com/agility-at-scale/people-management/). Hence, this type of setup could also have a negative impact on team member's motivation since HR needs are dealt with by a manager who is not part of their project team-therefore a comfort factor may be lacking (Beecham et al., 2008).

In devising a "*middle-range theory*" from these insights, we considered the applicability of a broader framework from the literature as the study progressed, observing that a primary goal for agility across the enterprise lay in engendering "dynamic capabilities" within a firm (Wang & Ahmed, 2007). In the latter model we noted that *adaptive*, *absorptive* and *innovative* capabilities were identified as key elements. But how precisely the key mechanisms of roles and practices in software vendor organisations might operate was not obvious. Therefore, we hypothesise that the "*dynamic capabilities*" model may be adapted to accommodate the realities of a software vendor embarking on a programme of scaling agile across the enterprise. Extending the dynamic capabilities model will be the subject of future work.

6.1 Limitations, validity and reliability

As a single case study, generalisability of these findings can be argued. However, the stage of development of the case study organisation appears to us to echo the experience of many similar software vendors going through the change process of introducing a scaled agile framework. Therefore, we believe these findings may be of value to other practitioners and lead the way for future research.

To establish evidence in the report, citations were used when quoting the actual data as collected from the interviews. Review of the report was done to get feedback from the case study organisation and peers.

Internal validity was ensured through researchers having discussion with their peers, comparing the findings with key literature (identified prior research studies on the SAFe method), and having peer review of the research report, including reading practitioner reports. All these protocols helped to make sure that *assumptions* were being made correctly, various *alternatives* and *possibilities* were well thought-out (Yin, 1994). Further, as the purpose of the case study was revelatory and exploratory rather than explanatory, it has been argued (Fitzgerald, 2013) that an internal validity assessment could be omitted.

The process we have used for this investigation of adoption of DAD method, all the artefacts used and produced such as the protocol, *process notes, raw data, transcribed data, categories for data reconstruction and synthesis*, can easily be audited (Lincoln & Guba, 1985).

7 Conclusion

Through this exploratory and descriptive case study we have illustrated how one large global software vendor has made a transition to the Disciplined Agile Delivery Framework, and the changes in roles and practices that have occurred at the program level. This paper adds to the limited body of knowledge about scaled agile methods, and the implementation of agile above the software development team level, through an empirical study. We found a rich set of new roles and practices and a complex interrelationship which was in operation to enable alignment of business and software engineering perspectives, needs and priorities. We conclude that the relative maturity of the case study organisation in agile approaches, the commitment to agility at all levels of the organisation and the adaptability and skill of those involved enabled a successful transition to what is but one more step on a longer journey.

8 References

Ambler, S. W. and M. Lines. 2014. "*Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*", IBM Press.

Alqudah, M. & Razali, R. 2016. “A review of Scaling Agile Methods in Large Software Development”. *Internal Journal on Advanced Science Engineering Information Technology*. 6: 828-837.

Beecham, S., et al. 2008. "Motivation in Software Engineering: A systematic literature review." *Information and Software Technology*. 50(9): 860-878.

Fitzgerald, B., et al. 2013. “*Scaling agile methods to regulated environments: An industry case study*”. Software Engineering (ICSE), 2013 35th International Conference on, IEEE: 863-872.

Fogelstrom, N. D., Gorschek, T., Svahnberg, M., & Olsson, P. 2009. “*The impact of agile principles on market-driven software product development*”. Software Process: Improvement and Practice.

Huck, S. W., & Sandler, H. M. 1979. “*Rival hypotheses: "minute mysteries" for the critical thinker*”. London: Harper & Row.

Kirk, D. and E. Tempero. 2012. "A lightweight framework for describing software practices." *Journal of Systems & Software*. 85: 582-595.

Kniberg,H.and A. Ivarsson (2012). "Scaling agile@ spotify." https://ucvox.files.wordpress.com/2012/11/113617905-scaling-agile-spotify-11.pdf. Retrieved 16 Aug, 2017.

Lal, R. 2011. “Strategic factors in Agile Software Development method Adaptation: A Study of Market-Driven Organisation”, (Doctoral dissertation, Massey University, Auckland, New Zealand). Retrieved from http://mro.massey.ac.nz/handle/10179/2496.

Laanti, M. 2017. “*Agile Transformation Model for large software Development Organisations”*. Workshop presented at XP’17 Conference, May 22-26, 2017 Cologne, Germany.

LeCompte, M. D., & Goetz, J. P. 1982. “*Problems of reliability and validity in ethnographic research”*. Review of Educational Research, 52, 38-54.

Leffingwell, D. 2016. “*SAFe® 4.0 Reference Guide: Scaled Agile Framework® for Lean Software and Systems Engineering*”, Addison-Wesley Professional.

Lincoln, Y. S., & Guba, G. E. 1985. “*Naturalistic Inquiry*”. Beverly Hills: Sage Publication.

Maxwell, A. J. (2005). “*Qualitative research design: an interactive approach*”. (Vol. 41). London: Sage Publications.

Paasivaara, M. (2017). “*Adopting SAFe to scale agile in a globally distributed organization*”. In: Proceedings of the 12th International Conference on Global Software Engineering. pp. 36-40. IEEE Press, 2017.

Paré, G., & Elam, J. J. 1997. “*Using case study research to build theories if IT Implementation*”. In S. A. Lee, J. Liebenau & I. J. DeGross (Eds.), information systems and qualitative research (pp. 542-568). London: Chapman & Hall.

Patton, Q. M. 1990. “Qualitative evaluation and research methods (2nd ed.)”. London: Sage Publications, 1990.

Freudenberg, S. and H. Sharp. 2010. "The top 10 burning research questions from practitioners." *IEEE Software*. **27**(5): 8-9.

Runeson, P. and M. Höst. 2009. "Guidelines for conducting and reporting case study research in software engineering." *Empirical Software Engineering*. 14(2): 131-164.

Stojanov, I., Turretken, O., & Trienekens, J. 2015. “*A maturity Model for Scaling Agile Development*”. Paper presented 41st Euromicro Conference on Software Engineering and Advance Application. IEEE Computer Society (pp. 446 – 453).

Stol, K.-J., Goedicke, M., & Jacobson, I. 2016. Introduction to the special section—General Theories of Software Engineering: New advances and implications for research. *Information and Software Technology, 70*, 176-180.

Wang, C. L. and P. K. Ahmed (2007). "*Dynamic capabilities: A review and research agenda."* 9(1): 31-51.

Yin, K. R.. 1990. “Case study research: Design and methods” (2nd ed. Vol. 41). London: Sage Publications, 1994.

Acknowledgements

The support from the case study organisation for this research and the participants who generously gave their time to be interviewed for the study are gratefully acknowledged.

Copyright