Unpacking Agile Enterprise Architecture Innovation work practices: A Qualitative Case Study of a Railroad Company

Full Paper

B Veeresh Thummadi
Centre for Enterprise Architecture,
Pennsylvania State University, USA
veereshthummadi@psu.edu

Vishal Khapre
Centre for Enterprise Architecture,
Pennsylvania State University, USA
vdk4@psu.edu

Rosalie Ocker
Centre for Enterprise Architecture,
Pennsylvania State University, USA
rocker@ist.psu.edu

Abstract

Agile EA is the process for managing enterprise architecture modeling and redesign efforts with principles of agile methods. However, very little work has been done till date on how organizations adopt these methodological innovations such as integration of agile methods with enterprise architecture. This is problematic, because we know that organizations face stiff challenges in bringing new innovations that fundamentally disrupt their enterprise architecture. Hence we ask: How does agile EA get adopted in practice and what are the underlying mechanisms through which teams self-organize and adapt? To this end, we studied a large-scale agile EA development effort to modernize the legacy systems at a top railroad company referred to as “Alpha” (a pseudonym). Our qualitative analysis shows how multi-teams self-organize and adjust the pace of the development efforts by strategically (1) choosing different type of agile methods and (2) embedding resources across teams for increasing communications.

Keywords

Agile Enterprise Architecture, routines, complexity, self-organization, scaled agile.

Introduction

Agile EA (or simply “agile EA”) is the process for infusing and managing enterprise architecture modeling and redesign efforts with principles of agile methods for faster development times (Bloomberg 2013). Although increasingly prevalent, little research has been done on how organizations adopt these methodological innovations1 such as integration of agile methods with enterprise architecture for bringing change within an organization (Fitzgerald et al. 2013).

This is problematic, because we know that organizations face stiff challenges in bringing new innovations that fundamentally disrupt their enterprise architecture systems (Richardson et al. 1990; Tyre et al. 1994). It is for this reason organizations rely on external consultants and organizational change management teams to internalize and institutionalize the concepts that are non-native to the actors (Birkinshaw et al. 2008). While past studies on agile EA have provided insights on how organizations can build enterprise

---

1 The practice of infusing agile methods into the EA process is quite novel and we argue that it constitutes a methodological innovation. Further, the concepts of lean thinking and just in time are fundamentally oppositional to traditional EA principles that require extensive documentation using a sequential process.
architecture in increments through institutionalization (Isham 2008; Ross et al. 2006), there is little emphasis on how organizations adopt and self-organize through agile EA methods (Feldman et al. 2003; Laanti 2014; Leffingwell 2007, pp 95; Vidgen et al. 2009a).

Hence we ask:

- **RQ1: How does agile EA get adopted in practice and what are the underlying mechanisms through which teams self-organize and adapt?**

Recently scholars have suggested the use of complexity theory to study how organizations adopt new innovations in agile methods (Dingsøyr et al. 2013; Vidgen et al. 2009b). We use this line of argument to develop a mid-range process theory about the enactment of agile EA. To this end, we conducted a field study in a large and long-standing railroad company based in the United States referred to as “Alpha” (a pseudonym), which transitioned to agile EA from the years 2011-2016. During this period, Alpha adopted agile EA method during a corporate initiative and modernized their services for improved data analytics. Hence, it became a reliable setting for us to conduct our study.

Our study develops three key insights. First, we develop here a process theory about complexity of agile EA work practices. We identified how agile and enterprise architecture, that have fundamentally oppositional principles, are married and adopted to bring change in the way software development is carried out in the organization. Second, we show how agile teams self-organized and strategically embedded resources across teams to increase the communication paths. Third our study provides insights to the practitioners on how to manage and organize large-scale agile development efforts.

To date, very few companies have successfully adopted agile EA, however there is increasing interest in scaling agile concepts to the enterprise level (Fitzgerald et al. 2013; Laanti 2014). Hence this study will benefit organizations that are considering adoption of agile EA. The limitations of the study include reduced generalizability, as the interview data is collected from a single organization. However, this study is first of its kind and should provide inroads for Information Systems (IS) scholars to better understand the agile EA work practices.

The remainder of the paper is organized as follows. In the next section, we review literature on enterprise architecture and large-scale agile development to identify research gaps. Then, we discuss how complexity theory can be used to understand the agile EA adoption and change. Next describe our data collection procedures, site description and outline the ways through which we analyzed data. Lastly, we discuss our research findings and provide discussion and conclusions.

**Background**

The central body of literature in enterprise architecture deals with the idea of creating an organizing logic for organizations to survive on a long-term basis without sacrificing the short-term needs of the organization (Ross et al. 2006; Zachman 1987). It was originally inspired from the principles of civil engineering, where “blue prints” are necessary to design complicated structures. Consequently, enterprise architects have explored various ways for designing blue prints for effectively handling organizational changes through innovations in digital tools and frameworks (Cameron et al. 2013; Lapalme 2012). Enterprise architects postulate that to build an effective architecture, a series of steps need to be followed for evaluating all possible alternatives, and thus recommend heavy the use of modelling (i.e., diagramming) artifacts. For most of the organizations, aligning to this set of stringent EA processes to design artifacts has been daunting and problematic and is generally considered to be a roadblock (Buckl et al. 2011). Recent methodological innovations in large-scale agile development methods have enabled enterprise architects to explore the promising dimension of integrating EA with agile methods (Isham 2008).

In 2001, agile methods were formally introduced through Agile Manifesto and outlined twelve guiding principles for developing software faster in an iterative fashion (Beck et al. 2001). Despite agile methods wide-scale success, it has garnered little attention in the space of large development projects and enterprise architecture that require coordination across multiple teams (Dingsøyr et al. 2013; Dingsøyr et al. 2014). Here, organizations face two types of uncertainties in introducing agile EA: 1) how to adopt agile EA? And 2) how to organize the teams for reducing the frictions between teams? As the concept of agile EA is both new and of practical value, it is important for scholars of information systems to conduct
studies that focus on how agile EA is process gets enacted in organizations. Our work specifically aims in this direction.

Organizational scholars suggest usage of complexity theories to study complex situations in agile methods (Vidgen et al. 2009a). Studies in the past have used Complex Adaptive Systems (CAS) or chaos theories to understand how small agile teams self-organize to increase the agility (Vidgen et al. 2009a). However, there is limited understanding on how large-scale development efforts can be organized using agile methods and its relation to enterprise architecture. To explore this phenomenon, we use complexity theory to understand how agile EA gets enacted.

**Complexity theory perspective**

Complexity theory originally evolved from cybernetics and systems theory and is widely used in the fields of organization studies, strategic management and information systems to study uncertainty and non-linearity in a system or an organization (Mitleton-Kelly 2003). For example, in an organization adopting agile EA, few rules and principles exist, making the system more non-linear and chaotic. To understand this complex phenomena, we outline here three key principles: (1) initial triggers (2) push to the edges and (3) self-organization, which succinctly characterize complex systems and are suitable for studying agile EA (Anderson 1999; Mitleton-Kelly 2003).

(1) **Principle of initial triggers:** The principle of initial triggers here refers to shocks that a system receives due to internal or external disruptions caused either by natural or artificial corrosions (Kaufman 1993). A system in its natural state exists at the state of equilibrium. When triggered by stimuli the system can move out of the initial state and this process is generally referred to as symmetry breaking as the order of the system is broken down. In the case of agile EA, there can be many events that can occur within an organization or in the industry sector that can trigger natural drift in how the organization functions. For example, a major process innovation or a regulatory standard or redesign of tools can create these initial triggers in the organization.

(2) **Principle of push to the edges:** The principle of push to the edges here refers to the act of pushing a system to an unstable or difficult situation for creating new order. Scholars also refer to this principle as far from equilibrium state or the edge of chaos showcasing the chaos and disorder that gets generated in complex situations when a system is moved from equilibrium conditions or status quo (Mitleton-Kelly 2003). Furthermore, new innovations in organizations are generated when agents are at the tipping point or pushed to the edges. In the case of agile EA, the agents in the systems can be pushed away from the comfort zones due to the new routines, ideas and policies. For example, in agile EA, the enterprise architects can be expected to create amenable architecture for 1 or 2 program increments (PI) [a program increment is typically 3-4 months] and is iterated over every PI to add fidelity, which is completely unlike traditional EA, where in the architecture is created for 10-15 years. Hence in agile EA, actors can be pushed to the edges with new policies and routines.

(3) **Principle of self-organizing:** The principle of self-organizing here refers to the act of a system reorganizing without any external forces due to natural evolution. Agents in the systems are assumed to be making autonomous decisions (i.e., managers provide little guidance in team routines), thus giving raise to emergent properties. Research on self-organizing agile teams has purported to have higher speeds and efficiency than the ones with command and control philosophy. Furthermore self-organizing can reshape the roles of manager, who is traditionally known to control the pace and trajectory of the software development activities. In the case of agile EA, there is very limited understanding of how multi-teams self-organize and what are the emergent properties that evolve through such interactions (Hoda et al. 2013; Vidgen et al. 2009a).

In summary the above principles act as a theoretical lens to study empirically how agile EA gets enacted in practice. In the next section we describe our case study, data collection and data analysis procedures used in this study.
Research design

Case study description

We collected data from “Alpha” (a pseudonym for a company which is an industry leader in railroad industry in United States). Alpha established a Transportation System (TS) in 1992. TS is based on legacy technologies based on mainframe system to support daily operations, short term and long term planning such as shipping, inventory and workforce management. When a customer sends a shipment request for shipping freight, TS receives it. The clerk then may need to amend the shipment request via data entry behind the terminals. The system creates a work plan for moving freight, which is continuously edited by operations personnel based on tribal knowledge before assigning the job to crew for moving the freight. Alpha's technical and clerical workforce who are primarily running these operations and TS are close to retirement and hence Alpha decided to modernize their legacy systems through 'service modernization' project during 2011 for process automation i.e., automation of distribution of work to crew based on demand and supply. The project also envisions on creating self-services applications (mobile apps) for customers through their digital phones for improving the efficiency in shipment delivery by cutting the data-entry jobs. Currently the scope of the project includes service planning and freight movement, which includes operations such as waybilling (a waybill here refers to information exchanged between customers and carriers on shipment on consignment of goods), train scheduling, train tracking and crew management. Alpha is currently investing to create advanced features such as Geographical Information System (GIS)/Self-support for improved data analytics that can support 30 million lines of code and 3.6 billion database queries per day.

To support the existing mainframe a system, Alpha has been using home grown architecture practices for managing, designing and modeling enterprise architecture for over 25 years now. The existing EA frameworks and processes generate heavy documentation, which were stored in binders. The internal experts and external consultants recommended a move towards integrating agile development methods in enterprise architecture. Alpha is modernizing Transportation System (TS) by analyzing their competitors, benchmarking the company's performance and building a roadmap. With top management support, Alpha made the transition to home grown agile enterprise architecture practice based on The Open Architecture Group Framework (TOGAF) and inspired from SAFe (Scaled Agile Framework), a well-known codified framework for scaling agile methods to an enterprise level (Laanti 2014). SAFe relies on principles of agile development, lean thinking and systems thinking. Alpha’s architecture and delivery practice has three levels: portfolio, program and feature team level. We now discuss our data collection procedures.

Data collection

We conducted 28 interviews from top management to software developers and vendors (see Table 1). Following an interview protocol developed prior to the interviews, the average interview took around 45 minutes. All interviews were recorded and transcribed. We collected additional information through source documents (three newsletters, two roadmaps and two presentations). Specifically, we interviewed the team members of Shipping Process Management (SPM) and Workforce Utilization Management (WUM) projects (or release trains) to understand how the agile EA evolved and discussed the inter-team interactions and routines followed. Each of these divisions had 3-4 teams underneath that either followed scrum or scrumban, a hybrid of Scrum and Kanban methodologies, or kanban depending on the nature of the developmental effort needed. SPM projects manage the workflows of shipping requests of the customers while WUM manages the daily operations of the workforce such as crew management. We discuss next our data analysis procedures.

Table 1: List of Interviews at Alpha Company and the roles

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Role</th>
<th>Total Number of interviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Director</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Enterprise Architect</td>
<td>5</td>
</tr>
</tbody>
</table>
Data Analysis

We conducted qualitative data analysis in an iterative fashion by travelling back and forth between the data, emerging constructs and theory (Turner 2012; Locke 2001). We used grounded theory approach to analyze the qualitative data in a systematic manner (Miles and Huberman 1994). First few rounds of analysis were carried out during the familiarization stage when we interviewed the developers from the top management. These were later revisited as we began to collect more data. Given our focus in understanding adoption and complexity in agile EA we analyzed the data and performed first order analysis to discover themes and patterns in the events described by the actors. Second order analysis was later carried out to develop deeper theoretical understanding of the patterns discovered by integrating and relating the first order findings (see Table 2 for data structure, see more details about the method in Gioia et al 2013, we cannot describe our complete list of codes given the space limitations) (Gioia et al. 2013).

Table 2: Data structure

<table>
<thead>
<tr>
<th>1st order concepts</th>
<th>2nd Order Themes</th>
<th>Theoretical constructs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation of enterprise architecture</td>
<td>Assessment of enterprise architecture practice</td>
<td>Initial triggers</td>
</tr>
<tr>
<td>Heavy usage of documentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marrying agile and enterprise architecture</td>
<td>Rough edges between the methods</td>
<td>Push to the edges</td>
</tr>
<tr>
<td>Rubs between fast turn around times and maintaining structure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Little more time in managers hands</td>
<td>Ambiguous roles and responsibilities</td>
<td></td>
</tr>
<tr>
<td>Managers doing more enterprise architect roles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Increasing communication across teams</td>
<td>Porous communications across modules</td>
<td>Self organizing</td>
</tr>
<tr>
<td>Embedding resources in portfolio to feature teams</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teams to be able to remove their own obstacles</td>
<td>Empowerment of teams in local decision-making</td>
<td></td>
</tr>
<tr>
<td>Developers should feel ownership about tools, methods and security</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Data structure
Findings

In this section, we address our original research question on: How does agile EA get adopted in practice and what are the underlying mechanisms through which teams self-organize and adapt? We now describe and narrate our findings based on the data structure that emerged from the data.

Initial triggers

Alpha modernization efforts dates back to the 1990s when the company built the legacy systems for supporting the train operations and services. Recently, Alpha has started a new initiative in 2011 called service modernization to modernize the legacy systems for supporting advanced analytics and mobile interface development to the customers, which created initial triggers. One common theme that emerged from the data was the assessment of enterprise architecture practices from external consultants, which triggered the implementation of agile EA.

Assessment of enterprise architecture practice:

Assessment of enterprise architecture practice here refers to the evaluation of enterprise architecture based on the needs of the organization. Alpha engaged two top-consulting companies and found that the enterprise architecture was lacking agility and the traditional way of developing systems through waterfall may not be fruitful option in the long run. One of the Directors, quoted that Actors [developers] who join the teams need to develop the software in much faster way than what was traditionally been done in the organization, indicating the need to change existing software development practice.

"Yeah, the triggering event, which we engaged as a consulting company to come in and look at our whole IT operation and say, are we ready, is the organization ready to be successful in delivering something so large? They came in and looked at it, and said, one of the areas that we are really suffering is enterprise architecture from the standpoint of we need to really connect the work we’re doing back to the business capabilities that are involved, and really have traceability from lines of code”

This realm of events and consulting reports created initial triggers pushed Alpha away from the comfort zone (or equilibrium) and led to initiation of change management team primarily geared towards amalgamating enterprise architecture with agile methods, dubbed agile EA, for synchronizing dependencies across business, enterprise architecture and product development teams. According to the Director, the long-term goal of agile EA factory [a concept of creating software like factory] is to deliver products at a much faster rate and hence he quoted:

"We’ve been building what we call the factory, which is getting all the processes set up to start really delivering. I think when delivery starts we’re still going to have to face that battle with all right, we’re going to budget for this much money.”

Push to the edges

The initial triggers pushed Alpha to the edges and created uncertainty about the modernization effort. Specifically, we observed two themes emerging from the data, namely (1) rough edges and (2) ambiguous roles and responsibilities.

Rough edges between the methods

We define rough edges as objects [or concepts] having conflicting views. In Alpha organization, there was uncertainty of how enterprise architecture needs to be meshed with existing agile methods and how to marry the two concepts together. This has created rough edges between the two. For example one of the Directors mentioned that there were rubs in terms of how to retain faster turn around times while maintaining the structure in enterprise architecture.

"I see that we move very early on that enterprise architecture and agile delivery there were some rubs there with the fast turnaround that teams need to make, and agile, and the structure that you want in EA...there are still rubs with an enterprise decision needs to get made, and the team needs to move right now. I’m still seeing a tendency to make a self-optimal decision for the sake of the team in order to make things go faster, and long
term that is going to cause pain. I would tell you that I don’t think we have this all figured out yet. We will figure it out, but it’s not all figured out.”

In addition, one of the Enterprise Architects mentioned about the difficulties in marrying the two [agile methodology and enterprise architecture], hence he mentioned that, “Then marrying the two together [agile methodology and EA] when they are traditionally I think there’s more of what you’re looking for, these are ... traditionally you would see these at odds with each other. With enterprise architecture being very large, slow moving, want to understand, broad and deep across domains. How do you ... that’s where we are right now is, how do we find that balance?”

**Ambiguous roles and responsibilities**

We define ambiguous roles and responsibilities as the uncertainty in actors’ functions and routines that needs to be carried out. As Alpha was new to scaled agile, allocation of resources was rather challenging. For example, the role of Project Manager role was gradually consumed by the roles and responsibilities of Release Transportation Engineer (RTE) and Scrum Master, and the actors were slowly adjusting and sinking into new roles by taking up new responsibilities that cut across the roles of managers and enterprise architects. Hence, one of the Directors mentioned that:

“We’re seeing managers with maybe a little more time on their hands, and having evolved that role of the manager as we go forward. What we see is in some cases have to be what we call the release training engineer, which is sort of this crowd master for the really high-level group”

Furthermore, evidence of ambiguous roles and responsibilities concept became apparent when actors described the concept of chicken and pigs for allocating resources. This concept emphasizes that a resource is called ‘chicken’ when he/she commits only part-time or few hours for the project and is refereed to as ‘pig’ when he/she is dedicated or committed for full-time to a project. Hence, one of Enterprise Architects mentioned how his time got distributed depending on his role, hence he affirmed:

“For an example a team needs my dedicated attention, mu dedicated time for that one sprint or there is a one story that they are playing, I can be a pig for that sprint for the team but I would not be there for all their entire 10 sprints. I am a chicken. They can always call me, consult me, get my time on ad-hoc basis but I am not committed to them. I am not required to be with them all the time.”

**Self-organizing**

Self-organizing here represents how actors at various levels of Alpha enacting scaled agile rearranged their engagement and operational model/contracts to expedite delivery of software architecture, design, and management activities with minimal or no guidance from direct managers and upper level management. Specifically, we observed two themes emerging from the data, namely (1) porous communications across modules and (2) team empowerment, that showcased the self-organizing nature of scaled agile. We describe the themes more in detail next.

**Porous communications across modules**

We define porous communication across modules as the ability to communicate and interact and see through different modules or groups of the organization to synchronize software development activities with overarching business needs. This type of ability to synchronize posed new challenges to actors as they began planning across multiple groups with intention to address requirements both from the standpoint of individual units and business units as a whole. As one of the RTE mentioned, there was some apprehension:

“I think the biggest challenge I see within the teams is breaking down the silo. You’re not just planning your own work, you are planning across a bigger organization. You have to take that into account. I think teams really want to go and do their own thing.
Adopting Scaled Agile versus just doing Agile approach has been challenging for ... The concept of planning for all areas has been a big work in progress for us.”

One of the earliest examples of porous communications across modules concept arose from the interview data when actors began describing about technical debt (a term used by extra development teams to account for work that needs to retrofitted to align to enterprise standards) and how important it was to communicate and manage. A software developer described that as they began collaborating with other teams and solution architects they realized that they need to maintain low levels of technical debt to reduce friction across teams and he quoted:

“If you are moving a lot quicker than other teams and you have dependencies within each other, you can get in some really hard places. I think, probably one of the biggest things I’ve noticed is managing technical debt... You have to maintain your levels of technical debt. You have to keep some kind of ratio where every sprint you’re taking on hopefully is small on technical debt.”

Perhaps the strongest form of porous communications across modules was expressed in the interviews when actors indicated the frequency of interactions with other teams and how each group tactically organized members in different units of organization to increase communication paths. One of the RTE mentioned how communication was increased across various levels as follows:

“We actually have team members embedded at the portfolio level and down to our feature team level. Their work may cross. That’s the communication path between our areas of portfolio work all the way down to the feature team. That’s where we’re utilizing our scrum on scrum. We actually have scrum masters assigned at our portfolio level work as well as that feature team level. To get that cross communication, our scrum of scrums is where we give up dates across the program to everybody. Then, scrum masters are responsible for communication that back to their respective areas.”

Empowerment of teams in local decision-making

We define empowerment of teams in local decision-making as the ability to make independent decisions within team based on the intuition and judgment in orchestrating software development activities. Top management at Alpha was extremely favorable to teams and always empowered teams. One of the Directors mentioned, how top management empowered teams in removing their own roadblocks and hence he quoted:

“The old rule of management is to be the obstacle remover. To really be successful at agile, the teams need to be able to remove their own obstacles, and we have to be able to empower the teams to remove their own obstacles.”

One of the earliest examples of empowerment of teams in local decision-making concept arose from the interview data when actors began describing about their freedom to select the type of agile method and decision to choose open source collaborations tools. One of the RTE described that different teams pick their own agile based on the needs, hence she cited,

“They [other team] picked more of Kanban approach because the work itself does not necessary fit within a 2-week sprint. We do have Kanban, we have a little bit of a scrumaban, and then we have true scrum running within our lower team.”

Furthermore, the teams also included open source versioning and hence one of the developer mentioned:

“We recently moved from SVN [Subversion, a software revision control technology] to Git [a distributed version/revision control technology]. We’ve been setting up our Git flow and our methodology as a team for what we want to do for collaborating on projects and just how we set up our branches and how each individual ... The responsibility of every developer is really what we’ve been trying to establish. We’ve gotten to a point where we are still learning, still growing, but we have, we’ve ironed out some of the kinks for collaboration on that using Git.”

The strongest form of evidence of empowerment of teams in local decision-making came from the interview data when security issues were being discussed. The Security Architect mentioned how security
became an enabler than a hindrance when ownership to design security was distributed across agile teams. Hence he quoted:

“They [developers] have to feel ownership because if they don't feel ownership we continue to have that kind of contentious relationship. Where security doesn’t enable, security hinders. In giving them or making them owners of the security requirements and compliance within their projects, within their teams, gives them that ownership.”

**Discussion & Conclusions**

Our findings indicate that agile EA was conceived during initial EA assessment and later implemented by creation of Agile Enterprise Architecture Framework and Process. However, the method was substantially adapted to suit the local conditions of the participating agile teams due to self-organizing over time. Thus, we might conclude that agile EA method follows similar type of self-organizing principles to that of smaller agile teams (Vidgen et al. 2009a). We now highlight some of our important findings below.

Our first finding illustrates how the organization was pushed to the edges in the pursuit of agile EA innovation. In the initial periods, actors struggled in unifying two diverse concepts such as agile and enterprise architecture. This in turn created rough edges and pushed the actors away from comfort zones. Furthermore, the articulation of roles as per agile EA was minimal which gave significant degrees of freedom for actors to modify their roles and responsibilities, thereby creating ambiguity in who should be doing what. Such ambiguity created perfect incubation conditions for self-organizing teams to create roles and responsibilities that most suited for their needs. In both the projects/agile release trains we studied, some parts of managers’ responsibilities were replaced by RTE. The primary responsibility of an RTE is to facilitate interactions across teams and monitor the project health. Further, he/she had to take additional responsibilities like delivery and alignment of the enterprise architecture in the bounded context of the agile train. This finding is similar to earlier studies by Vidgen and Wang (2009) who found that the role of managers evolved from the role of leader to a facilitator in agile teams (Vidgen et al. 2009a). Second, our findings suggest that multi-team interactions in agile teams have led to unusual characteristics that are not traditionally associated with small-scale agile teams. We found that through self-organizing agile teams embedded team members across different levels (from portfolio to feature teams) of the organization to increase communication about requirements and dependencies. Thus, we might conclude that modules in agile EA are more porous than traditional small-scale agile teams. Our third finding indicates that small agile teams in agile EA were empowered for team-level decisions. This is surprising because some critics of large-scale development agile methods suggest that small teams are not sufficiently empowered and we found the opposite². What was interesting was the agile teams were empowered on the choice of tools (such as open source vs. proprietary) and design methods. Despite this, we have to caution that agile teams were not given complete freedom in terms of designing their own requirements because of dependencies between functional and architectural requirements in the service modernization effort.

Our study contributes to theory by illustrating how CAS theories can be used to understand agile EA. Unlike, previous studies (Vidgen et al. 2009a), we use CAS to explain large scale agile and self-organization across different agile teams. Further this study provides guidance for organizations in understanding the grand challenges of integrating enterprise architecture with agile methods.

Our study has several limitations. First, the interview data was collected from just one organization and hence is less generalizable. Second, the findings are based on qualitative data collected from informants and hence the study would benefit from additional analysis using quantitative methods to increase validity. Third, our study considers agile EA as complex and we cannot practically prove given our data (Kauffman 1993). However conventional wisdom on organization science and information systems consider sociological systems as complex (Anderson 1999). Future research should consider expanding the study to include diverse organizations. To conclude, we believe our study should provide new inroads about emerging agile work practices in organizations.

---

² [www.infoq.com](http://www.infoq.com) (Refer to the article on “Has SAFe Cracked the Large Agile Adoption Nut?” for more details)
References


Bloomberg, J. 2013. The agile architecture revolution: how cloud computing, rest-based SOA, and mobile computing are changing enterprise IT, (John Wiley & Sons.


Vidgen, R., and Wang, X. 2009b. "A coevolving systems approach to the organization of agile software, ").