Distributed Agile Development Communication: An Agile Architecture Driven Framework

Yehia Ibrahim Alzoubi[1*], Asif Qumer Gill[1], Ahmed Al-Ani[2]

[1] School of Software, University of Technology Sydney, 15 Broadway/Ultimo NSW 2007, Australia.
[2] School of Electric, Mechanic and Mechatronic Systems, University of Technology Sydney, 15 Broadway/Ultimo NSW 2007.

* Corresponding author. Tel.: +61 2 9514 7938; email: yizoubi1977@gmail.com
Manuscript submitted January 26, 2015; accepted May 8, 2015.
doi: 10.17706/jsw.10.6.681-694

Abstract: Agile methods depend on active communication and effective knowledge sharing among team members for producing high quality working software systems in short releases and iterations. However, effective communication in Distributed Agile Development (DAD) can be challenging due to a number of different factors, such as physical locations, multi-cultures and time-zones. The agile body of knowledge mainly discusses some technology and non-technology solutions and strategies to mitigate the DAD communication challenges from a project management perspective. Nevertheless, it has recently been argued that there is a need to understand and analyze DAD communication from other related but different perspectives, such as enterprise strategy, enterprise architecture and service management. Due to the fact that agile EA provides a holistic view and blueprint of the whole environment in which a number of projects are developed and managed, we attempt in this study to explore the effect of agile Enterprise Architecture (EA) on DAD communication. Particularly, we propose the development of an agile EA driven approach from the architecture body of knowledge for handling the DAD communication challenges that have not been thoroughly investigated before.

Key words: Agile communication challenges, distributed agile communication, distributed agile development, enterprise architecture.

1. Introduction

Agile methods have been introduced to address a number of issues related to project development and delivery, such as over-budget or behind schedule projects, and not meeting customer's needs and expectations. These issues require adopting flexible, adaptable and short delivery cycles [1]. Agile methods have been emerged over a period of time to increasingly influence future trends in software development in both the local and distributed contexts [2].

Agile practices combined with distributed development seem to offer several benefits, such as faster time to market, the liberty of involving developers around the world, around the clock development and low cost products [3]. Despite these benefits, distributed agile or adaptive development (DAD) faces many challenges. The most noticeable challenge is the communication and knowledge sharing between dispersed teams and customers [4]-[6]. Poor communication and knowledge sharing (e.g., delivering an inadequate, inaccurate or incomplete message) are the main concerns of DAD environments [5]-[7]. As the interest in adopting DAD has been increasing, the literature on communication challenges and communication techniques/strategies of DAD has also been increasing [2]. This marks the importance of research in developing tools, techniques, strategies

and approaches to address the poor communication and knowledge sharing concerns in DAD [8], [9].

Most studies, which have addressed DAD communication and knowledge sharing challenges, recommend adopting and using the available technologies (e.g., *Wiki* and video conferencing) to enhance DAD communication. However, DAD communication needs to be looked at from more holistic perspective instead of a simple technological perspective. This paper is such one attempt and explores the impact of agile enterprise architecture (EA) on DAD communication. This study investigates that an agile EA driven approach along with the available communication technologies could possibly enhance the efficiency and effectiveness of DAD communication [10].

The structure of this paper is as follows. Section 2 provides the research background. Section 3 provides the related literature review. Section 4 presents and discusses the proposed agile EA driven DAD communication approach. Sections 5 and 6 present the discussion and conclusion respectively.

2. Background

Agile development focuses on using informal face-to-face communication among the team members. Informal communication can be defined as the personal peer-oriented communication that takes place outside the official structure and without the knowledge of management [4]. Informal communication helps in filling and correcting mistakes quickly, and is essential for agile practices and principles [1].

Since agile approaches depend heavily on informal and face-to-face active communication and coordination among co-located team members and customers, physical proximity becomes essential for participants to engage in informal communication [11]. The success of agile in small and medium environments encourages large software development organizations to adopt DAD approaches, however, these approaches may not be straightforward and possess many challenges especially communication related challenges. It has been reported that the DAD project takes 2.5 times more than the same project in the local agile context [3]. Inefficient DAD communication is perceived to be the source of most of the other DAD challenges [11]. In other words, without successful DAD communication between distributed teams and developers, DAD projects may fail. Therefore, no wonders that many researchers and practitioners are showing strong interest in investigating the issue of DAD communication.

Many solutions and strategies have been introduced to resolve DAD communication challenges through using available tools and technologies such as *Wiki*, teleconference videoconference, exchanging visits and changing developer's roles [2], [9]. However, these tools cannot resolve many challenges like difference in personal attitudes, languages and cultures, so there is still a pressing need for discovering alternative solution options [11].

The importance of the organizational communication patterns in software development has been recognized for long time (e.g., Conway's Law, which suggests that the structure of the product mirrors the organizational structure) [12]. For many years, software development has often been structured in accordance with the architecture design of the product being developed [12]. This study investigates the DAD communication issue from the perspective of enterprise level architecture instead of the local software architecture perspective. It proposes an agile EA driven approach for enhancing DAD communication. Hence, the aim of this paper is to discuss the relationship between DAD communication and agile EA.

This approach intends to provide a holistic shared vision based approach to enhance the efficiency and effectiveness of DAD communication. This approach incorporates ideas from recent research about agile and non-agile EA, and agile software development methods and frameworks. In summary, to achieve this objective, this study focuses on the following two questions:

RQ1. What are the challenges or limiting factors of DAD communication?

RQ2. How can agile EA be used to enhance DAD communication?

3. Related Literature

In this section, background literature relevant to the study is discussed. Firstly, we discuss communication challenges in the context of DAD, after which we discuss agile EA in the context of DAD communication. Secondly, we introduce the Gill Framework® -domain architecture [13] that can be used to establish an agile EA capability in the context of DAD. An agile EA capability then can be used to enhance DAD communication. The discussion about the actual development of the agile EA capability is beyond the scope of this paper. These elements as a whole provide the theoretical basis and framework for our study.

3.1. Distributed Agile Development Communication

DAD is seen as an important software delivery model [10]. However, management of co-located team, whilst still hard, is much easier than management of DAD team. This is because, for co-located team, there is an opportunity for daily face-to-face communication at any time, which helps in dealing with questions, confusions, difficulties, or doubts instantaneously (e.g., daily stand-up, showcases, reviews). This seems difficult in the case of DAD where geographical, cultural and temporal distances have been identified as the key barriers for DAD communication and collaboration [14]. The combination of these challenges decreases the efficiency of DAD communication and makes it a complex task [15].

Other authors argue that the biggest problem for DAD communication is the cross-team work, which according to Ali Babar *et al*. [16] should be reduced. Kuusinen *et al*. [17] argue that the main problems with DAD communication are the inability to have face-to-face conversation, lack of frequent feedback between users, lack of experienced designers and architect teams, and lack of synchronization between different DAD teams [17]. The ineffective communication may result in less coordination between DAD teams and lack of understanding of the customer's requirements [17]. Also, communication can be a challenge for a developer if he/she handles multiple projects at once, when he/she expects written formal documents and requirements, or when his/her responsibilities are not clear [18]. However, other studies reported that using informal communication in complex DAD projects is problematic comparing to simple co-located agile projects [5]. Moreover, too much informal communication and inadequate technology represent challenges for DAD communication especially with weak communication skills of the team members [18].

Recently, a systematic literature review study has been conducted in the context of DAD communication challenges and strategies [2]. The authors have identified seven challenging categories along with the strategies to overcome those challenges. Each category has some underlying challenges. There are 22 DAD communication challenges in total. Table 1 summarizes these challenges and recommendations [2]. These findings are used as the basis for this study. The first challenge category (C1) is "People Difference". This category has four communication challenges: cultural difference (i.e., different cultures among DAD developers), people attitude (i.e., different personal attitude of DAD developers), language (i.e., different languages used among DAD developers), and trust (i.e., the difference in trust towards DAD developers depending on their locations or countries) [2].

"Distance Differences" (C2), as the second mentioned challenge type, includes 2 challenges; different time zones and different geographical areas [2]. It has been noticed that the "Distance Differences" challenges are associated with productivity and performance, particularly when the teams are divided by space and time-zones [14].

The third challenge type (C3) is the "Team Issues". This type includes all possible challenges related to DAD teams, such as team size, team distribution (i.e., number of sites), cross-team (i.e., amount of information needs to be exchanged between DAD sites) work, cross-team communication [2]. The fourth challenge type (C4) is "Technology Issues". This type includes all issues related to technology used to establish DAD communication. It includes DAD communication tools, infrastructures, communication

bandwidth (i.e., speed of communication or internet means), and the overall cost to establish DAD communication using available technologies [2].

"Architectural Issues" type is mentioned fifth in Table 1 (C5) [2]. This category refers to architecture (i.e., difference in architectures used in DAD teams), organizational structure (i.e., differences in organizational structures between DAD teams), managerial structure (i.e., differences in management styles or structures between DAD teams), and project domain (i.e., type of project each DAD team or all teams develop). "Architecture Issues" have not been paid much attention in the literature; in fact most of the studies which mentioned this category have indirectly pointed to the "Architecture Issues" as a challenge of DAD communication [2]. Jaanu *et al*. [19] reported that the lack of appropriate architecture decreases the communication and knowledge sharing efficiency among DAD teams and team's members. Also, it represents a communication barrier to DAD due to misunderstanding or an unnecessary flow of communication as a result of insufficient definition of a system and software structure [20]. As shown in Table 1, some strategies and recommendations were introduced by authors such as using reference architecture and agreement on the high level project requirements by all DAD teams and team's members at the beginning of the new project, increasing the trust among DAD teams and members, increasing the project's transparency, promoting common interest of the project and team goals, and providing organizational chart to all DAD teams and team's members [19], [20].

Table 1. Communication Channels of Agile GSD (Adopted From [2])

Type	Challenges	Recommendations
1	People Differences (Culture differences, People attitude, Language, Trust)	Team gathering regularly, visits' exchange between distributed teams, compulsory presentations for all team's members, documenting key actions, gradual team distribution, keep interaction between distributed teams to a minimum, increasing formality, promoting "cultural liaison", promoting multiple communication modes, promoting socialization between team's members
2	Distance Differences (Time-zone, Geographic)	Using synchronous and asynchronous tools, promoting synchronized work hours, creating local teams and centralizing the experts, using project management tools and tracking systems, promoting team gathering at the beginning of each project
3	Team Issues (Size, Distribution, Cross-team communication, Team-work)	Using communication protocols, Scrum practices, knowledge transfer mechanisms, exchange visits among sites, configuration management system, starting the new project with face-to-face meeting, reducing the cross-teams' communication, team training
4	Technology Issues (Tools, Infrastructure, Communication bandwidth, Cost)	Using available tools, different communication modes, appropriate communication infrastructure
5	Architectural Issues (Architecture, Organizational structure, Managerial structure, Project domain)	Using reference architecture, promoting trust, transparency, common interest among the team, using organizational chart, promoting members' relocation
6	Processes Issues (Process, Control, Lack of commitment)	Sharing interfaces for strategic aspects, coordination with the overall strategy and local process, using documentation, standards, and monitoring systems
7	Customer Communication	Promoting customer involvement or customers' representative, rapid communication with customers

The sixth challenge type (C6) is "Process Issues". This type includes all issues related to DAD communication process. It includes the process and control applied and the commitment-level from DAD members to follow the process used in DAD communication [2]. The seventh challenge type (C7) is "Customer Communication". This type includes all issues related to customer communication (i.e., who provides the requirements). Generally, this type highlights the customer involvement and the transparency [2].

The related literature shows that there are a number of DAD communication challenges. Also, it has been noticed in the literature that the strategies and solutions being proposed to mitigate these challenges are at the very high general level and focus on using individual communication tools [11]. Moreover, the role of traditional EA has not been paid much consideration in agile studies, although it has been paid very high attention in non-agile distributed development. In agile, traditional EA is considered as an overhead since the focus is on developing working software [1]. There is a need of an agile EA approach to support the DAD. This draws our attention to the agile EA capability establishment to address DAD capability communication needs and challenges.

3.2. Agile Enterprise Architecture

There are a number of well-known industry architecture frameworks that have been developed during the last years, such as Zachman [21], Department of Defence Architecture Framework (DoDAF) [22], and The Open Group Architecture Framework TOGAF 9.1 [23]. These frameworks can be tailored to create an EA capability. EA can be defined as:

"a blueprint that describes the overall structural, behavioral, social, technological, and facility elements of an enterprise's operating environment that share common goals and principles." [24].

However, the tailoring and adoption of EA capability are not straightforward. Buckl *et al*. [25] summarizes four common challenges for agile EA; stakeholder's satisfaction, customer's requirements, stakeholders' commitment and the flexibility to requirement changes. They argue that none of the traditional EA (e.g., Zachman, DoDAF, and TOGAF) can deal with these challenges in agile software development enterprise. According to Ambler [26], agile EA has two organizational structures: (1) formal structure: this form is documented by organization such as chart, and (2) informal structure: this form is not documented, but rather used by developers to get the things done or what is called "go to guys", which means looking for other developers who have critical skills or knowledge, which is what agile it trying to be. A common problem or thread of using traditional EA is the focus on tools and processes over stakeholders' interactions [26]. To overcome the above challenges, agile EA approach is required to meet the following needs [25], [26]:

- Be people focused,
- Be as simple as possible,
- Supports delivering value in short releases and iterations,
- As a result of daily interaction between developer and customer, problems raised have to be addressed and dealt with instantaneously in agile EA,
- Works with developers in the field to gain better understanding of agile EA needs,
- Promotes the idea of self-directed and self-organized members, which helps to keep administrative overhead to lowest level.

An agile EA describes the design of an agile enterprise. An agile enterprise can be defined as:

''An entity is said to be an agile enterprise when an enterprise is responsive (scans, senses and reacts appropriately to expected and unexpected changes), flexible (adapts to expected or unexpected change at any time), speedy (accommodates expected or unexpected changes rapidly), lean (focuses on reducing

waste and cost without compromising on quality), and learning (focuses on enterprise fitness, improvement and innovation)." [24].

3.3. The Gill Framework-the Domain Architectures

The Gill Framework® [10], [13] is an agile meta-framework that provides the ADOMS (Adapting, Defining, Operating, Managing and Supporting) (see Fig. 1) approach for designing agile enterprises. The ADOMS approach can be used for adapting, defining, operating, managing and supporting agile or adaptive capabilities of an agile or adaptive enterprise.

Fig. 1. The gill framework® V 2.0 - ADOMS [13].

It incorporates concepts and practices from different enterprise architecture, strategy, project, requirements and service management frameworks. It has been developed to address the traditional EA challenges. It has its foundation in well-known agility, design, service and living service systems theories. *The Gill Framework - ADOMS* has two main layers [13]:

- The outer layer: it represents guidance for continuous adaptation of agile EA in response to continuous external and internal changes. It defines five EA adaptation capabilities (i.e., context awareness, assessment, rationalization, realization and unrealization),
- Inner layer: it has five capabilities (i.e., adapting, defining, operating, managing and supporting). (1) Adapting capability defines a generic reference description that can be tailored for a specific agile environment (2) Defining capability defines agile EA capability, (3) Operating capability operates the agile EA capability for creating agile architecture artifacts (business architecture, information architecture, social architecture, application architecture, platform architecture, infrastructure architecture, and facility architecture etc.) for supporting DAD. (4) Managing capability is focused on managing any change in agile EA capability and its artifacts. (5) Supporting capability supports other capabilities.

The domain architectures, as shown in Fig. 2, in *the Gill Framework*®, include: *interaction architecture*, *factory architecture*, *facility architecture* and *solution architecture*. *Factory architecture* includes *human architecture* and *IT architecture*. *IT architecture* includes *application architecture*, *platform architecture* and *infrastructure architecture*. *Human architecture* includes *social architecture*, *business architecture* and *information architecture*. *Social architecture* refers to social structure, culture, behavior, knowledge and opinions of community in an organization [27].

3.4. Why Agile Enterprise Architecture

EA has been used for many years by traditional software development organizations [12]. According to Bass and Kazman [28], using EA development "differs from traditional development in that it concentrates on driving design and maintenance from the perspective of software architecture. The motivation for this change of focus is that software architecture is the placeholder for system qualities such as performance, modifiability, security, and reliability. The architecture not only allows designers to maintain intellectual control over a large, complex system but also affects the development process itself, suggesting (even dictating) the assignment of work to teams, integration plans, testing plans, configuration management, and documentation. In short, the architecture is a blueprint for all activities in the software development life-cycle." However, traditional EA approaches are considered too heavy for agile development. A light-weight agile EA is required to provide the shared vision of the architecture for DAD.

Agile EA seems important to agile projects as it: (1) draws from a uniform infrastructure, platform and application, (2) leverages same design patterns and language patterns, (3) scores from same quality attributes and use a uniform scoring system, and (4) communicates the architecture value and state with all stakeholders [29]. Moreover, EA provides the basis for architecture rules, which improves implementation consistency and reduces the number of errors [30].

One goal of agile EA is to make sure that all systems fit into the whole of the existing and future environments. Agile EA aims at producing enough architecture to support the different DAD projects with minimal documentation overhead. Ambler [26] summarizes benefits of using agile EA as follows:

- Avoiding chaos that will result if team's members think that they can do whatever they want using any technology they want,
- Avoiding duplication of functionality and information,
- Achieving continuous, effective design reuse, and better integration between different systems,
- Avoiding conflict between systems, which might cause system fail,
- Decreasing development price.

Agile EA thus may facilitate communication by improving comprehensive vision through one common object of work that all DAD participants use and understand. The architecture description provides terms and concepts that serve as a common language for all DAD teams, which enables clear communication and arrangements [30]. Avritzer *et al*. [12] reported that EA enhances DAD communication, has the potential to guide task assignments and team coordination, encourages and ensures developers to identify the design rules and assigned tasks, and is more helpful to less experienced developers.

Without agile EA, road mapping, product management and integration are done through numerous teams and meetings that require travelling to headquarters locations or attending teleconferences outside work hours, in most cases [31], which needs:

- Big amount of effort and communication,
- High integration cost between DAD teams,
- High coordination and communication cost between DAD developers,
- Unintended resource allocation, which increases the coordination cost,
- Insufficient pre-iteration (up-front design) and interface specification, which leads to late delivery or manual tests,
- Interaction between DAD teams is more challenging due to different time-zones, cultures, languages, and work practices. This needs asynchronous communication or travelling to meet face-to-face with other teams or developers.

Design is a big issue in DAD environment [32], [33], as it could have negative effect on DAD communication due to longer time needed and less efficient communication as a result of less

understanding of design among DAD teams [32], [33]. To address the design issue, EA has to be introduced or shared among DAD teams at least for the minimum level [24], [32]. This study argues that to address the DAD communication issue, the design issue needs to be addressed. Therefore, to address the design issue, agile EA artifacts need to be developed and used among the DAD teams. It is anticipated that not only DAD communication will be enhanced by using the agile EA, but also the overall agility and productivity will be increased.

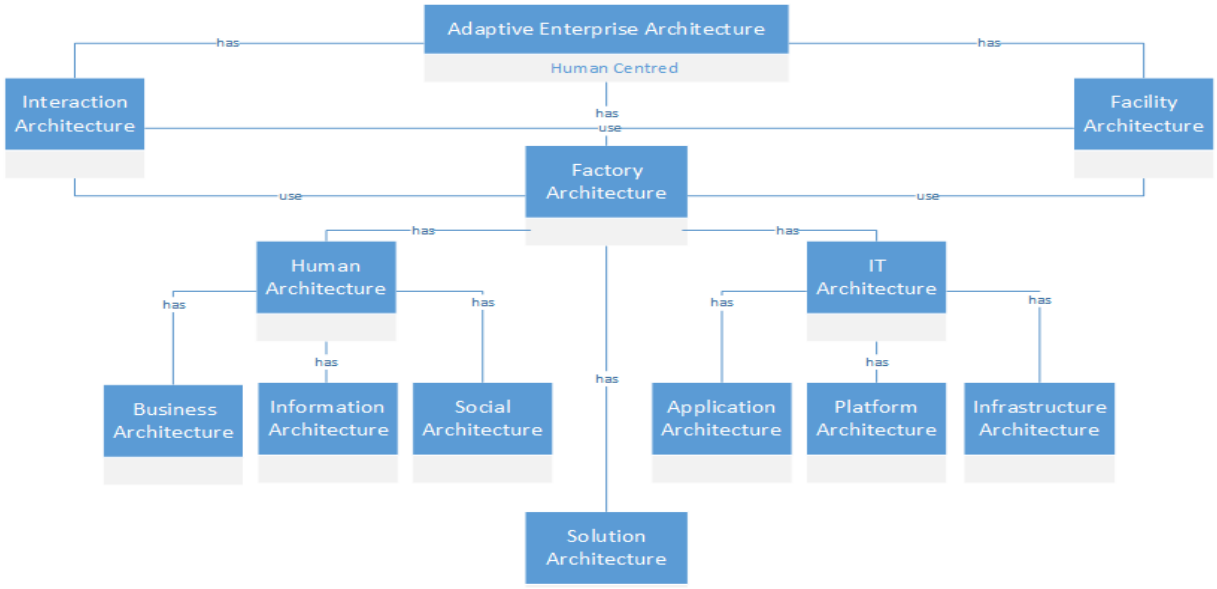

Fig. 2. The gill framework–domain architectures [27].

4. Distributed Agile Communication Approach

It is evident from the analysis that agile EA seems useful for DAD. However, it is not clear how can the agile EA actually support DAD. In this section, we propose an agile EA driven approach to support DAD communication. Firstly, we discuss the holistic framework of DAD communication incorporating agile EA as a communication tool to be used by DAD teams. Secondly, we discuss how the use of the agile EA driven DAD communication framework can enhance communication.

4.1. DAD Communication Framework

DAD communication framework is built on the notion that communication among DAD teams, among DAD team's members, or with management level can be enhanced through agile EA. Fig. 3 represents the holistic view of the DAD communication framework. This framework shows the DAD in the “program” level, where DAD program may have "N" number of DAD projects and "N" number of architecture owners. It also shows DAD in the “project” level, where DAD project may have "N" number of DAD teams and only one architecture owner (Fig. 3). In the ‘project’ level, the architecture owner share solution architecture among project’s teams and/or team’s members and update the agile EA after completing the systems or sub-systems. In the “program” level, DAD projects (i.e., architecture owners) can communicate with each other using the big picture of an agile EA. Agile EA provides the shared vision. It is a shared knowledge-base or a repository that is visible to all levels and teams. DAD teams do not use too much documentation; rather they use agile EA artifacts. These artifacts can be stored in and accessed from the shared architecture knowledge-base by the DAD teams in the form of diagrams, charts, tables, and so on.

Communication challenges affect all communication's levels (i.e., programs, projects and teams’ communication). Three main elements are used to describe the DAD communication framework [34];

people, process and technology (tools). People refer to all stakeholders (e.g., customer, developers, and managers) that are involved in DAD communication. Technology refers to tools that are used in DAD communication. Process refers to DAD organizational processes that are used in DAD communication (e.g., control, commitment, structures, and activities).

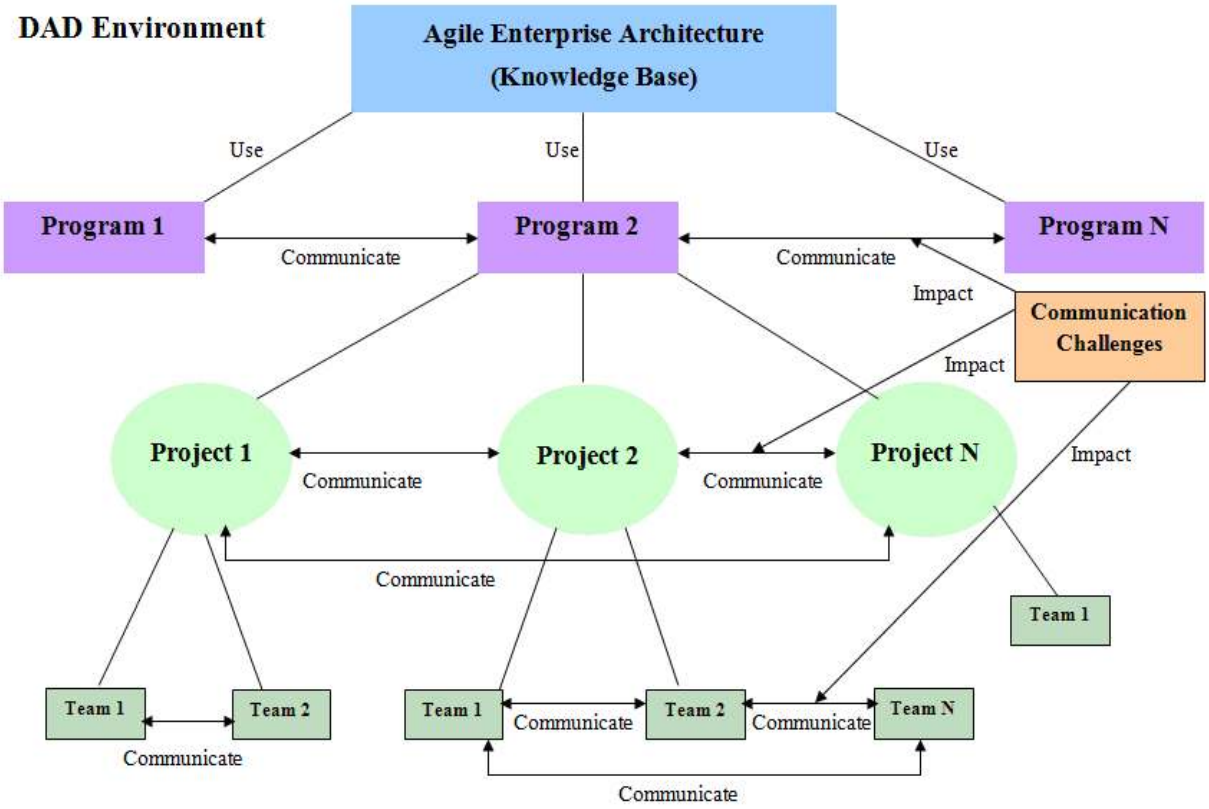

Fig. 3. DAD communication holistic framework.

4.2. DAD Communication Framework Application

As each agile method has its own practices and artifacts, this study explains five practices of agile development lifecycle building on *XP* and *Scrum* methods. These practices are: planning (adaptive), requirements practice, decomposition, sprint (small release), and working product (tested bug free product) [29]. Table 2 summarizes the impact of agile EA on DAD communication and coordination throughout the above five agile practices.

Having in mind the above five agile practices and the following scenario, the following paragraphs outline how agile EA can enhance DAD communication and mitigate the effect of challenges shown in Table 1.

Scenarios:

1) Single DAD project: the architecture owner shares solution architecture among different teams or team's members on the same project (e.g., project 2, Fig. 3). He/she also updates the agile EA, regularly, after completing the project or part of it. All teams and team's members share the team or individual artifacts through using the common vision of agile EA (i.e., knowledge base),
2) Multiple DAD projects: different DAD projects (program level) have different architectures owners; one for each project. They share different projects' artifacts through using the common vision of the agile EA knowledge base. All projects' artifacts are shared and can be accessed using agile EA knowledge base.

The role of architect (i.e., architecture owner) is important especially in the case of big teams. Architecture owner knows better than the other team members about the organizational structure, solution architecture and the trade off that may take place for the best of the business [29]. Architecture owner represent the agile EA in an agile project team. He/she plays an important role in all DAD practices. The role of architecture owner is discussed among the role of agile EA in each agile practice.

Planning (adaptive) practice: The main role of agile EA is to provide the DAD developers with the holistic architectural direction setting [29]. Agile development is done in small iterations and encourages all members to collaborate to find better solutions later during development, however; the pre-options of solutions and design patterns increase the agility, save time, and give an overall direction [26] (C6). Agile developers, liaison by architecture owner, are offered different options rather than specific solutions, design patterns, high-level diagrams, components reuse, quality and trade-off attributes (C5), and initialization to communication channels (C4, C7). Agile EA is concerned with facilitating independent development by team's members and minimizing the amount of unproductive hours (C3). Architecture owner keeps DAD teams updated with the interface changes, what backward compatibility is required, what functionality to build and the other component interfaces to develop against. As a principle, no team can initiate development on functionality that depends on the functionality being developed by another team (C3). Although this principle slows down the development process, the removal of coordination cost and the short cycles for agile team outweighs any benefit that may be achieved by current development.

Requirements practice: The main role of agile EA is to help on getting agile team members on board and structuring business and architecture needs [29]. User requirements that have significant foundational or directional influence can be identified (C7). Architecture owner can help team members to correct deviation or fine tune architecture (C5). Agile EA helps *Product Owner* to prioritize the customer's requirements with the business user requirements and build them in conjunction with the business functionality in each *Sprint* (C7). Architecture owner works with *Product Owner* to estimate the effort that an item in the *Product Backlog* will take, which means assigning the item or sub-item to the relevant developer (C2).

Decomposition practice: Agile EA helps *Product Owner* to identify boundaries of the architecture, and determine business value of each item (part of the product) (C5). Architecture owner may help in decomposing the architecture into development tasks that can be assigned to relevant developers.

Sprint (small release) practice: Agile EA helps to ensure that the *Sprint* functionality has been met and keep the *Sprint* on the track [29]. Agile team commits itself to implement specific *Sprint* goal selected during the daily meet (C1). Daily meets discuss: what has been done, what is the current status, what next to be done and if there is any problem. This helps in receiving early feedback to deal with any change or intervention needed and maintain early decisions and clear focus of agile team [25] (C5, C6). Architecture owner collaborates with the team during the *Sprint* helping in design issues and business objectives, and advising and offering consultation to team's members (C1).

Working product (tested bug free product) practice: Agile EA helps in measuring the architecture state of the product [29]. Developers measure architectural attributes; code, functionality, and refactoring, integration and testing for each *Sprint* release in order to make sure that the customer's needs were met (C7). Architecture owner may start reviewing the working product several days before the official review to ensure its functionality, then, work on code and architecture documentation.

5. Discussion

It was reported that DAD projects can be difficult to implement without sufficient leadership and support [35]. Communication and coordination are more complex in DAD projects than for the same projects in

co-located team's context due to many challenges such as time, distance, and cultural differences. Because of these challenges, most of the existing DAD teams and developers coordinate and control their work using available synchronous and asynchronous communication tools, and visiting other teams or team's members. In fact, DAD projects have two complementary communication needs; the formal communications for crucial tasks (e.g., updating project status, escalating project issues), and informal communication in order to keep team members aware of the information that would enable them to work together efficiently [36].

Addressing DAD communication challenges is conceptually simple: remove all needs for the cross-team communication and coordination. This would allow small DAD teams to develop and release independently, and increase development efficiency. However, DAD teams are still building solutions that are part of a larger system and therefore cannot be completely independent.

Table 2. Impact of Agile EA on DAD Communication and Coordination (Adopted from [29])

Practice	Impact
Planning (adaptive)	• Understand business objectives • Get input from agile team • Understand the business vision and directions
Requirements Practice	• Facilitate story session • Use user's stories to build design patterns, improve refactoring, and validate hardware and software
Decomposition	• Maintain architecture significance • Maintain business value
Sprint (small release)	• Build functionality • Support agile team
Working Product	• Review documents • Advocate refactoring

The approach, we presented here, is to move any coordination needs from the team level to the architecture [31]. DAD should have a communication system that could continuously feed DAD teams with holistic architectural and strategic information to keep them on track [37]. DAD teams need to be decoupled as much as possible regarding design rules and organizational structure. The spread of design features across sites would increase the interactions among teams for agreement on design requirements. This suggests that there could be a conflict between teams about solution design. This can be addressed by using architectural description [20] to keep all teams on the same page. This description should allow DAD teams to focus on customer's needs, avoid organizational dependencies caused by architectural dependencies, and satisfy customer's needs but keep in mind the organizational software design and include reusability as an important quality. In an empirical study, Ali Babar *et al*. [16] found that using the architectural description was very useful for the overall architectural modification and helped the new team's members to be successfully integrated in the team.

Although agile development prefers not to have architects, the role of the architecture owner can be of great influence on DAD. This role can be assumed as a "liaison" between different parties included in development (i.e., developers-developers, developers-customers, and developers-management). The architecture owner is expected to have a good understanding of what is available within the organization, what the final product has to be and what the current implementation status of the features are. He/she is

also expected to be responsible for documenting (or updating) and communicating the architecture with all agile project's participants.

6. Conclusions

It has been well-established that DAD projects face many challenges, which are mostly communication related. In fact, the cost of communication is much higher in DAD than in a local context due to the communication inefficiencies. Development that relies on significant cross-team communication performs poorly in DAD contexts. Agile EA may enhance the DAD communication, in one hand, and reduce the communication frequency, in the other hand. This paper explores the effect of agile EA on DAD communication. In the past, DAD communication was mainly discussed in the context of available social communication tools, whereas here we propose the use of a holistic agile EA as a means for supporting communication and coordination of DAD teams. This paper provides only initial investigation on the agile EA and DAD communication. In future, more empirical work on the use of agile EA in DAD will be presented to the community as an ongoing contribution in this important and timely area of research.

References

[1] Agile, M. (2001). Manifesto for agile software development. Retrieved September 20, 2014, from http://www.agilemanifesto.org.

[2] Alzoubi, Y. I., & Gill, A. Q. (2014). Agile global software development communication challenges: A systematic review. *Proceedings of the PACIS14,* Chengdu, China.

[3] Herbsleb, J. D., & Mockus, A. (2003). An empirical study of speed and communication in globally distributed software development. *Software Engineering*, *29(6)*, 481-494.

[4] Herbsleb, J. D., & Moitra, D. (2001). Global software development. *Software*, *18(2)*, 16-20.

[5] Korkala, M., & Abrahamsson, P. (2007). Communication in distributed agile development: A case study. *Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications* (pp. 203-210).

[6] Gill, A. Q. (2014). Applying agility and living service systems thinking to enterprise architecture. *International Journal of Intelligent Information Technologies*, *10(1)*, 1-15.

[7] Gill, A. Q., & Bunker, D. (2013). Towards the development of a cloud-based communication technologies assessment tool: An analysis of practitioners' perspectives, *VINE*, *43(1)*, 57-77.

[8] Gill, A. Q., Bunker, D., & Seltsikas, P. (2012). Evaluating a communication technology assessment tool (Ctat): a case of a cloud based communication tool. *Proceedings of the Pacific Asia Conference on Information Systems 2012,* Hochiminh City, Vietnam.

[9] Layman, L., Williams, L., Damian, D., & Bures, H. (2006). Essential communication practices for extreme programming in a global software development team. *Information and Software Technology*, *48(9)*, 781-794.

[10] Gill, A. Q. (2012). *The Gill Framework: Adaptive Enterprise Architecture Toolkit*, 2012.

[11] Korkala, M., & Maurer, F. (2014). Waste identification as the means for improving communication in globally distributed agile software development. *Journal of Systems and Software*.

[12] Avritzer, A., Paulish, D., Cai, Y., & Sethi, K. (2010). Coordination implications of software architecture in a global software development project. *Journal of Systems and Software*, *83(10)*, 1881-1895.

[13] Gill, A. Q. (2014b). The Gill Framework®. Retrieved December 15, 2014, from http://www.aqgill.com/adoms.

[14] Agerfalk, P. J., Fitzgerald, B., Holmstrom, H., Lings, B., Lundell, B., & Conchúir, E. O. (2005). A framework for considering opportunities and threats in distributed software development. *Proceedings of the*

International Workshop on Distributed Software Development (pp. 47-61), Citeseer.

[15] Holmstrom, H., Conchúir, E. Ó., Agerfalk, P .J., & Fitzgerald, B. (2006). Global software development challenges: A case study on temporal, geographical and socio-cultural distance. *Global Software Engineering*, 3-11.

[16] Ali, B., M., Ihme, T., & Pikkarainen, M. (2009). An industrial case of exploiting product line architectures in agile software development. *Proceedings of the 13th International Software Product Line Conference* (pp. 171-179). Carnegie Mellon University.

[17] Kuusinen, K., Mikkonen, T., & Pakarinen, S. (2012). Agile user experience development in a large software organization: good expertise but limited impact. *Human-Centered Software Engineering*, 94-111.

[18] Hummel, M., Rosenkranz, C., & Holten, R. (2013). The role of communication in agile systems development. *Business and Information Systems Engineering*, 1-13.

[19] Jaanu, T., Paasivaara, M., & Lassenius, C. (2012). Effects of four distances on communication processes in global software projects. *Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM)*.

[20] Martini, A., Pareto, L., & Bosch, J. (2013). Communication factors for speed and reuse in large-scale agile software development. *Proceedings of the 17th International Software Product Line Conference*.

[21] Zachman, J. A. (1987). A framework for information systems architecture. *IBM Systems Journal*, *26(3)*, 276-292.

[22] Department of Defence, U. S. (2010). The DoDAF architecture framework version 2.02. Retrieved December 7, 2014, from http://dodcio.defense.gov/dodaf20.aspx.

[23] The Open Group. (2014). TOGAF version 9.1 - the open group architecture framework. Retrieved December 5, 2014, from http://pubs.opengroup.org/architecture/togaf9-doc/arch.

[24] Gill, A. Q. (2013). Towards the development of an adaptive enterprise service system model. *Proceedings of the Nineteenth Americas Conference on Information Systems*.

[25] Buckl, S., Matthes, F., Monahov, I., Roth, S., Schulz, C., & Schweda, C. M. (2011). Towards an agile design of the enterprise architecture management function. *Proceedings of Computing Conference Workshops 15th IEEE International the Enterprise Distributed Object*.

[26] Ambler, S. (2014). Agile enterprise architecture. Retrieved November 25, 2014, from http://www.agiledata.org.

[27] Gill, A. Q. (2013). Defining a social architecture within the enterprise architecture context. Retrieved November 15, 2014, from http://www.orbussoftware.com/community .

[28] Bass, L., & Kazman, R. (1999). *Architecture-based development* (Report No. CMU/SEI-99-TR-007, ESC-TR-99-007). Carnegie Mellon Software Engineering Institute.

[29] Madison, J. (2010). Agile architecture interactions. *Software*, *27(2)*, 41-48.

[30] Kornstädt, K., & Sauer, J. (2007). Mastering dual-shore development – the tools and materials approach adapted to agile offshoring. *Seafood*, 83–95.

[31] Bosch, J., & Bosch, S. P. (2010). Coordination between global agile teams: from process to architecture. *Agility across Time and Space*, 17-233.

[32] Cockburn, A. (2007). *Agile Software Development: the Cooperative Game*. Harlow, Addison-Wesley.

[33] Hammad, M., Hammad, M., Bani-Salameh, H., & Fayyoumi, E. (2014). Measuring developers' design contributions in evolved software projects. *Journal of Software*, *9(12)*, 3005-3011.

[34] Hunt, B. (2013). An introduction to business process and enterprise architecture. Retrieved September 20, 2014, from http://www.orbussoftware.com/community.

[35] Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A teamwork model for understanding an agile team: a case

study of a Scrum project. *Information and Software Technology*, *52(5)*, 480-491.

[36] Lanubile, F. (2009). Collaboration in distributed software development. *Software Engineering*, 174-193.

[37] Gill, A. Q. (2014c). Hybrid adaptive software development capability: An empirical study. *Journal of Software*, *9(10)*, 2614-2621.

Yehia I. Alzoubi is a PhD research student at the School of Software at the University of Technology, Sydney. His research focuses on agile enterprise architecture and distributed agile development. He received his MSc in IT from Central Queensland university in Sydney and BSc in engineering from Mu'tah university in Jordan. He has a number of published work in scholarly peer-reviewed international scientific journals and conferences.

Asif Q. Gill is a TOGAF 9 Certified Enterprise Architect, lecturer and researcher at the School of Software at the University of Technology, Sydney. He specializes in software-intensive agile enterprise architecture and engineering practices, tools and techniques. He is an experienced author, coach, consultant, educator, researcher, speaker, trainer and thought leader. He has extensive experience in both agile, non-agile, cloud and non-cloud complex software development environments, displaying a deep appreciation of their different perspectives in a number of IT-enabled business process improvement and transformation industry projects of varying sizes.

Ahmed Al-Ani received his Ph.D. degree from Queensland University of Technology in 2003. He is currently a senior lecturer at the University of Technology, Sydney. He has published numerous journal and conference papers in pattern classification, biomedical signal processing, information hiding and information systems.