Agile vs. Plan-Driven Perceptions of Software Architecture

Irit Hadar
Department of Information Systems
University of Haifa
Haifa, Israel
hadari@is.haifa.ac.il

Sofia Sherman
Department of Information Systems
University of Haifa
Haifa, Israel
shermans@is.haifa.ac.il

***Abstract*-The use of agile methodologies in industry has increased significantly over the past decade, promoting the value of human-centric software development process. This growing use derives the need to adjust agile methodologies to bigger, more complex system development projects, where architecture plays a significant role. However, many believe that an essential conflict exists between the requirement of minimalism in agile methods and the need for well-defined and documented architecture in complex systems. This paper presents an exploratory study aimed at understanding the software architecture related activities as perceived by architects with and without experience in agile methodologies. The findings indicate that while architects practicing only plan-driven methodologies perceive architecture activities as being related only to the first phases of the development process, architects involved in agile projects perceive architecture activities to be related to most or all phases of the development lifecycle. The latter perceptions are consistent with suggestions expressed in the literature regarding architecture in general and in agile methodologies in particular. Based on these findings we suggest that agile methods not only lead architects to adjust their behavior to the agile philosophy, but also improve architects' perceptions and practice of architecture in general.**

Keywords-software architecture; development methodologies; agile methodology; qualitative research

I. AGILITY AND ARCHITECTURE

The role and importance of software architecture have gained considerable recognition over the past two decades, as software systems become increasingly complex. There are many definitions of software architecture in the literature; however, there is no universally accepted definition of what software architecture is. Eeles summarizes the definitions by their commonality: "most definitions indicate that an architecture is concerned with both structure and behavior, is concerned with significant decisions only, may conform to an architectural style, is influenced by its stakeholders and its environment, and embodies decisions based on rationale"[1].

Architecture activities and the architects' roles and responsibilities must be adjusted to each organization or project [2]. Kruchten lists the architecture activities including, among other, technical activities associated with design and requirements analysis, reviews and guidelines throughout the development process, risk assessment and mitigation, consulting design, implementation, and integration teams, and problem-solving activities that are beyond solving strictly architectural issues [2] .Looking at these activities, we find architects' involvement throughout the lifecycle of the development process.

When referring to architecture processes or activities, researchers usually compare plan-driven software development with agile methodologies. Plan-driven approaches are traditional methodologies that include extensive planning, codified processes, and rigorous reuse making development an efficient and predictable activity that gradually matures toward perfection [3]. In plan-driven approaches, the work begins with the elicitation and documentation of client requirements, followed by architectural and high-level design development and inspection [4]. Known examples of such approaches are the waterfall, incremental, and iterative methodologies. In these methodologies, the architecture phase is performed after requirements specification and before the detailed design, and is accompanied by a great deal of documentation. This approach is also called BDUF – big design upfront.

Agile methodologies (XP, Scrum, etc.) emphasize rapid and flexible development. These methodologies are based on the Agile Manifesto (http://agilemanifesto.org), which transforms the development process from being process-centric to being human-centric. The main characteristics of agile development are short releases, flexibility, and minimal documentation [5]. At a first glance this seems to be essentially contradictory to architecture, or as Booch puts it: Agile architecture is an oxymoron [6]. Blair et al. declare that "The architect's role in agile development remains at best ill-defined and at worst simply ignored." [7, p.26]. Boehm claims that the values of agile development may be interpreted in many ways, some of which are problematic because they ignore architectural features that do not support the current version of the system [3]. He claims that this approach works well when future requirements are unpredictable, but may cause the loss of valuable architectural support.

The question whether agility and architecture can coexist has been addressed recently at conferences and in the literature. A panel and a workshop, both entitled Architecture in an Agile World, were presented at OOPSLA 2009, and a special issue of IEEE Software on Agility and Architecture was published in March-April 2010, all aimed to understand

and resolve this contradiction between agility and architecture. Abrahamsson et al. state that companies supporting architecture's vital role in the development lifecycle doubt any development approach that does not pay sufficient attention to architecture [8]. Falessi et al. found that non-agile developers are pessimistic regarding agility and architecture coexistence [9]. However, more optimistic opinions exist as well. While Abrahamsson et al. claim that the tension between agile approaches and architecture lies on the axis between the adaptation approach of agile methodologies and the plan-driven approach that pays sufficient attention to architecture, they also suggest that this tension between agility and architecture might be a false dichotomy that stems from the misuse of agile methods by users who believe that agility is about ignoring architecture and jumping to refactoring [8]. They note that according to Kent Beck, architecture is just as important in agile (specifically, XP) projects as it is in any software project. Moreover, Booch [6] states that all good software-intensive architectures are agile, and Spinellis notes that architecture is always important in the case of large and complex projects regardless of their development methodology: "Look at a large successful software system and beneath it you'll find an architecture that's kept its evolution on track." [10, p.11].

Nord and Tomayko claim that software architecture is part of product quality and is not tied to a particular process, technology, culture, or tool [11]. Abrahamsson et al. strongly believe that a healthy focus on architecture is not antithetic to any agile process [8]. Falessi et al. conducted a survey among 72 developers and found software architecture to be relevant in the context of agile development; however, they claim that new methods and special training to integrate architectural practices within agile development are required [9]. Blair et al., for example, offer a framework for improving architectural design from an agile perspective by identifying architecture activities at different times during the release cycle as well as identifying different roles that architects fulfill [7].

In light of the debate in the literature as to if and how architecture can co-exist with agile development methodologies, we aimed to learn how architecture is actually practiced and perceived by architects working with different development methodologies. The research question we set out to explore was: How do architects working with different development methodologies perceive architecture and its related activities? More specifically, we were interested in identifying what differences, if any, exist between the perceptions of architects with and without experience in agile methodologies.

II. THE EMPIRICAL STUDY

As an exploratory research, the empirical study was conducted using the qualitative grounded theory approach, which is an effective approach for studying a phenomenon in its natural settings [12, 13]. During their search, software architects performed tasks designed according to the research objectives; the process of performing these tasks as well as its outcomes were documented and analyzed. The data analysis included coding the data obtained, and characterizing and classifying it to categories, as will be described later.

The research was conducted at one of the world's largest IT management software providers, whose name and other identifying information will not be mentioned due to confidentiality reasons. The study was conducted during an architecture conference held by that firm, to which architects were invited from different business units and different projects developing different products. During the conference, architects attended a variety of lectures and workshops according to their preferences. The study described here took place at a workshop entitled The Architecture Process, which was developed by the authors of this paper and whose goals were to:

- Understand the various architecture activities
- Identify the core skills needed to be an architect
- Relate architects' skills to architecture activities

The workshop started with a short introduction on the workshop's objective and program after which participants were divided to groups based on a questionnaire they filled out before the session. The participants then worked in their groups to achieve the above stated objectives (based on detailed instructions they received). In this paper we focus only on the architecture activities.

Forty-two architects took part in the described workshop. As mentioned, participants were asked to fill out a questionnaire, based on which they were divided into groups of six participants each, according to their experience in development methodologies. Seven groups were formed: four in which all participants had both agile and plan-driven development experience and three in which all participants had only plan-driven development experience. In addition to the development methodology criterion, we made sure that the groups were heterogeneous, i.e. all groups had a similar number of highly experienced architects and architects with relatively little experience, and all members of each group working in different units and on different projects.

Focus groups are a known and acceptable tool of qualitative research [14, 15]. As a qualitative method for gathering data, focus groups bring together several participants to discuss a topic of mutual interest to themselves and the researcher and serve as a source of data [14]. In addition, in order to triangulate and further validate the results obtained via the focus groups, we conducted a follow-up study with five individual architects who did not participate in the original study (two with only plan-driven development experience and three with both agile and plan-driven development experience). These architects underwent the same procedure as the groups; however, conducted the task individually.

Participants were asked to identify and list different software architecture activities within the software development process. They were also given a general

development process flowchart provided by the firm (due to confidentiality reasons, we cannot present the flowchart here but its components will be described later). During the workshop, the groups discussed the architecture activities and summarized them in a table with the following headings: activity name, activity description, and people (roles) involved in the activity. All forms were collected from the participants at the end of the workshop.

Data from the forms underwent qualitative analysis for each group separately, and results were then considered based on the participants' development experience in the different methodologies, noting identical and different architecture activities listed by the different groups. Each activity was mapped to its appropriate category in the development process flowchart and categories were added according to need. All architectural activities suggested by the participants are presented below, in the Findings section.

Limitations: The following limitations regarding the empirical study should be noted. The tasks of the main study were conducted in groups, thus the data collected does not reflect perceptions held by each individual, but rather by the majority or the more dominant members of each group. This limitation was mitigated by a follow-up study with individual architects. Another limitation is that all participants work in the same company. However, this company is a large and heterogeneous enterprise, with many of its projects and units recently purchased; thus its architects illustrate different organizational cultures, contributing to the diversity and generality of the results of this study.

III. FINDINGS – DIFFERENT PERCEPTIONS

A. Group Characteristics

Participants were divided into two types of groups based on the questionnaire analysis: Agile experience (A) and non-agile experience (NA).

Agile experience groups included only architects who had experience with agile (as well as plan-driven) development methodologies during their work as architects whereas the non-agile experience groups included only architects who had no experience with agile development. Table 1 summarizes the experience in the two types of methodologies that the participants of each group had as software architects.

TABLE I. PARTICIPANTS' EXPERIENCE IN AGILE AND PLAN-DRIVEN METHODOLOGIES

Group	Experience (in years)	
	Agile	***Plan-driven***
NA1	-	2-10
NA2	-	2-10
NA3	-	2-10
A1	1-3	1-8
A2	1	1-8
A3	2-3	2-8
A4	1-3	4-10

Workshop participants were asked to list architecture activities, after which we mapped each activity to the relevant category in the development lifecycle. The lifecycle described in the flowchart included the following general phases: Top Level Design, Detailed Design, Code Development, and Testing. All four phases were defined as categories in our data analysis. In addition, and based on the data collected, we added two more phases to our categories: Requirements Analysis (at the beginning of the lifecycle), and Support (at the end of the lifecycle).

B. Non-Agile Experience

The three non-agile groups included six participants each, all of whom had experience with plan-driven development methodologies only (see Table 1). During the groups' discussions, participants listed various architecture activities, which we mapped to the relevant development phases as detailed in Table 2.

TABLE II. ARCHITECTURE ACTIVITIES LISTED BY NON-AGILE EXPERIENCE GROUPS

	Group		
Category (development phase)	**NA1**	**NA2**	**NA3**
Requirements	• *Understanding the needs* • *Market analysis update* • *Studying and understanding the features*	• *Preparing requirements document*	• *Eliciting requirements*
Top Level Design	• *Designing* • *Proof of concept*	• *Designing reference architecture* • *Preparing top level design documents* • *Reviewing documentation* • *Benchmarking, scalability, and performance*	• *Identifying key modules* • *Preparing reference architecture documents* • *Designing key interfaces*

In addition to indicating the architecture activities, participants were also asked to list the people, or roles, involved in each activity. All three groups indicated the roles of Development Manager, Product Manager, and QA Manager.

C. Agile Experience

The four agile experience groups each had six participants with experience with both agile and plan-driven development methodologies (see Table 1). Like in the other groups, these groups were also asked to list architecture activities, which we mapped to the relevant development phases, as presented in Table 3.

Here too, participants were asked to list the people, or roles, involved in each of the activities. All agile experience groups indicated the roles of Product Manager, Development Manager, QA Manager, Customer, Support team, QA team, and Developers.

TABLE III. ARCHITECTURE ACTIVITIES LISTED BY AGILE EXPERIENCE

	Group			
Category (development phase)	**A1**	**A2**	**A3**	**A4**
Requirements	• *Understanding the business (interacting with the Product Manager and Customer)*	• *Understanding requirements (discussion with product management)* • *Planning and prioritizing requirements. Providing this information as feedback for management/end users*	• *Identifying business needs* • *Gathering requirements and studying product feasibility* • *Prioritizing requirements*	• *Requirement analysis (collaborating with all those involved in the development lifecycle)*
Top Level Design	• *Baselining requirements*	• *Identifying modules and flows between modules. Providing top level design* • *Reviewing the design* • *Researching technologies available for design implementation*	• *Defining system and application architecture* • *Defining technologies to be used* • *Top level design documentation*	• *Defining architecture goals* • *Delivering top level design documents*
Detailed Design	• *Interacting with team to finalize the detailed design specification* • *Test plan review*	• *Providing detailed design for every module and reviewing it*	• *Documenting detailed design*	• *Reviewing detailed design (feedback and correction)*
Code Development	• *Writing critical code* • *Reviewing code*	• *Providing coding guidelines, unit testing and code review*	• *Coding* • *Code review*	
Testing	• *Interacting with beta customers*	• *Creating and reviewing QA test*	• *Testing system performance*	• *Integrating and reviewing test plan*
Support	• *Training, supporting and sustaining* • *Helping resolve critical customer issues*	• *Configuration management and build management*	• *Sharing knowledge with teams*	

IV. DISSCUSSION

One observation stands out when comparing the two tables – the non-agile groups listed activities related only to the requirements and top level design phases, while all agile groups (namely architects with agile experience in addition to plan-driven experience) listed activities that span the entire development lifecycle, including activities that are not traditionally considered architectural (e.g. code development and testing activities). The agile groups also listed more stakeholders that they interact with as a result of their active involvement in these additional activities throughout the development lifecycle (see Table 4).

TABLE IV. PEOPLE ARCHITECTS INTERACT WITH

Agile experience	Non-agile experience
Product manager, Development Manager QA Manager Customer Support team QA Team Developers	Product Manager Development Manager QA Manager

The results above are further reinforced by the fact that the participants were not asked to refer to a specific development methodology when performing the tasks, but rather to describe architecture activities in general. This may suggest that the architects' perceptions about architecture have been altered due to their experience with agile development.

The data collected in the follow-up study, in which individuals were asked to perform these tasks, was aligned with the results presented above. Moreover, in the individual tasks we explicitly asked the architects to specify the architecture activities at each phase, presenting them with the full list of categories identified in the main study (see Table 3). The architects with experience in agile development specified activities in all categories, similarly to the agile groups of the main study. The architects with no agile experience specified activities in the requirements and design phases, and in answer to the other categories they repeatedly wrote phrases such as "typically little to no activity", "minimal involvement" and "only when needed", never specifying what these activities (in case they are in fact needed) may be.

V. CONCLUSIONS

Agile methodologies have been found to be of great value to software development, improving quality and shortening time-to-market of software products. However, while the benefits of agile methodologies to different development activities, e.g. coding and testing, have been illustrated and discussed, when it comes to architecture, the literature mostly discusses the possible collisions.

The findings of this research illustrate benefits software architecture may draw from agile methodologies. Specifically, architects with experience in both agile and plan-driven methodologies tend to assume responsibility over, and be involved in, more development phases than architects who have experience only in plan-driven development. This is in line with Abrahamsson et al., who describe the architect's role in agile methodologies as being, among other things, a mentor, prototype, and trouble shooter, concentrating more on code and focusing on internal coordination [8], and with Blair et al., who assert that the architect, specifically in agile development, should be a facilitator who serves the development team in a collaborative, hands-on way, ensuring the building of architecturally sound applications [7]. This may also imply that agile architects tend to leave some of their decisions to later stages of the development lifecycle, as many of the above mentioned researchers propose. Booch, for example, suggests that architecture is doing just enough design to advance development – not so much that you prematurely and unnecessarily bind decisions, yet not so little that you avoid tackling the big problems [6]. Architects with agile experience also indicate a wider range of people who they interact and collaborate with during their work activities. This is also in line with Blair et al., who describe the architect in agile development as a facilitator who works among developers and stakeholders and enables direct interaction among them all [7].

Based on the architects' perceptions reflected in this study, it seems that while non-agile architects consider their job done after the requirements and top level design phases are over and tend not to become involved in the other development phases, architects with agile experience perceive architecture activities as spread throughout the development lifecycle, including interaction and collaboration with additional stakeholders accordingly. One reason for this difference in perception may be that architects with agile experience reduce the effort they invest in up-front architecture decisions, and so have time to spare for other activities that promote the development of the software and maintain its architectural integrity. Following this finding, it would be interesting in future research to investigate directly how architects with experience in both plan-driven and agile methodologies perceive the differences in these matters between the two methodologies.

We conclude that practical experience in agile methodologies raises the awareness of architects to the importance of their involvement throughout the development lifecycle, as recommended by the literature. Thus, this research reveals an additional strength of agile methodologies and reduces some of the concerns regarding the tension between agility and architecture, illustrating that architecture may be adapted, and possibly improved, in agile environments. Future quantitative research may be conducted to further validate and verify this hypothesis, with

the participation of architects from different firms and domains.

REFERENCES

[1] Eeles, P. What is a Software Architecture? 2006, http://www.ibm.com/developerworks/rational/library/feb06/eeles/, accessed December2011.

[2] Kruchten, P. WhatDo Software ArchitectsReallyDo? *The Journal of Systems and Software,* 81, 2008, 2413–2416.

[3] Boehm, B. Get Ready for Agile Methods, with Care.*Computer35(1)*, 2002, 64-69.

[4] Lindvall, M., Basili, V., Boehm,B., Costa,P, Dangle,K., Shull, F.,Tesoriero, R., Williams, L., and Zelkowitz, M. Empirical Findings in Agile Methods. In Wells, D. and Williams, L., editors, *Extreme Programming and Agile Methods — XP/Agile Universe*, 2418(19), 2002,81-92. Springer Berlin Heidelberg.

[5] Ambler, S.W. Lessons in Agility from Internet-Based Development.*IEEE Software19(2)*, 2002, 66-73.

[6] Booch, G. An Architectural Oxymoron. *IEEE Software27(5)*, 2010, 95-96.

[7] Blair, S., Watt, R., and Cull, T. Responsibility-Driven Architecture. *IEEE Software 27(2)*, 2010, 26-32.

[8] Abrahamsson, P., Babar, A.M., and Kruchten, P. Agility and Architecture: Can They Coexist? *IEEE Software*,*27(2)*,2010, 16-22.

[9] Falessi, D., Cantone, G., Sarcia', S.A., Calavaro, G., Subiaco, P., and D'Amore, C. Peaceful Coexistence: Agile Developer Perspectives on Software Architecture.*IEEE Software 27(2)*, 2010, 23-25.

[10] Spinellis, D. Tools of the Trade: Software Tracks. *IEEE Software27(2)*, 2010, 10-11.

[11] Nord R.L. and Tomayko, J.E.Software Architecture-CentricMethods and AgileDevelopment.*IEEE Software 23(2)*, 2006, 47-53.

[12] Strauss, A. and Corbin, J.Basics of Qualitative Research; Grounded Theory Procedures and Techniques,1990, Newbury Park: Sage.

[13] Strauss, A. and Corbin, J. Grounded Theory Methodology: An Overview. Chapter 1 in Denzin, N.K. and Lincoln, Y.S. (eds.), *Handbook of Qualitative Research*, 1994, Thousand Oaks: Sage, 273-285.

[14] Morgan D.L. and Spanish. M.T. Focus Groups: A New Tool for Qualitative Research. *Qualitative Sociology 7(3)*, 1984, 253-270,

[15] Stewart, D.W., Shamdasani, P.N., and Rook, D.W.*Focus Groups: Theory and Practice.* 2nd ed., 2007, Newbury Park, CA: Sage Publications.