



# Going Agile? Beware of Architecture-related Changes and Challenges!

M. Ali Babar

Agile approaches have undoubtedly been gaining popularity an increasing number of companies as a mechanism for reducing cost and increasing ability to handle change in dynamic market conditions. However, there is also a significant concern about the role and importance of architecture-related issues when using agile approaches. Many practitioners of agile approaches view software architecture in the context of the plan-driven development paradigm. That is why they appear to consider upfront architecture design and evaluation of very little value to the customers of a system. On the other hand, software architecture researchers and practitioners appear to be sceptics of using agile approaches for developing large scale software systems.

Despite these opposing views about the role and importance of software architecture in two camps, there is a growing interest in identifying the mechanics and prerequisites of bridging the gap between agile and architectural approaches. One way of bridging this gap is to gain a good understanding of architecture-related changes and challenges when using agile approaches. The objective of this article is to shed some light on how the adoption of agile approaches may impact on architecture-related practices and the kinds of architectural challenges faced by software development teams. This article is based on observations gathered from extensive interactions and discussions with dozens of practitioners developing large scale software systems using agile approaches. The exploration of the role of architecture and architecture-related practices in agile teams have been driven based on the activities (i.e., architectural analysis, architectural synthesis, and architectural evaluation) of a general design model, architecture documentation, and sharing of design decision practices.

**Role of architect:** When companies introduce agile approaches, there are usually several changes in the role and responsibilities of software architects. The projects using agile approaches usually have a role called solution architect, which is assumed by previously called software architects. However, the solution architects take on more management oriented responsibilities, which may also include being

a Scrum Master. A new role called "Implementation architect" can also be introduced. This role is usually responsible for getting the User Stories implemented and providing technical mentoring to developers. The implementation architect can also be responsible for deciding about the timing and amount of re-factoring to be carried out.

**Architectural analysis:** An agile team usually pushes most of the tasks related to architecture analysis phase (e.g., examining context and defining problems) towards customers, who are responsible for providing Users Stories for which design decisions are made and implemented. Most of the design decisions made by the solution and implementation architects are based on the features to be delivered. That means there is no attention paid to quality attributes. Rather, the quality attributes may not get any attention if their achievement is not considered a measure of success.

**Architectural synthesis:** The agile projects that I usually come across apply two stages for designing solutions: software architects working with customers draw a very high level architectural roadmap; then solution and implementation architects make the potential design decisions considering the User Stories and their priorities, budget, time, and other technical constraints. The designers in agile teams usually consider a limited number of well known solutions.

**Architectural evaluation:** The architecture evaluation is considered relatively less important in agile projects. A project manager may invite developers from another project to look for serious design flaws at a very high level. We may not observe any change in this practice as a result of adopting agile approaches because many projects evaluate architecture in normally even before adopting agile approaches. Architecture evaluation is usually carried out for quality attributes. Agile followers claim that re-factoring can help achieve quality attributes. However, it is a common observation that re-factoring, both at the code level and architecture level, help achieve the desired quality attributes to a certain extent such as improving maintainability by fixing the structure.

**Architecture documentation:** Agile approaches advocate "*Working software over comprehensive documentation.*" That is why there can be several changes in software architecture documentation practices as a

result of using agile approaches. However, these changes appear to be on the positive side of the practices as it helps in reducing the unnecessary documentation. There is usually drastic reduction in the amount and detail of architectural documentation. Companies usually report 30-40% reduction in resources required for documentation. Table 1 summarizes the architecture-related artifacts used by agile teams compared with architecture-centric teams for devising and implementing design solutions.

**Communicating design decisions:** One of the key architecture-related practices is communicating design decisions and rationale underpinning them to all relevant stakeholders. Most of the agile teams use Wikis and design meetings for sharing design decisions. The Wikis are also used for similar objective by software development teams irrespective of software development methodology. However, agile teams use Wiki more frequently and intensively without any formal templates or structure. Design decisions are also shared and explained on teams' whiteboards, which keep the design until it got implemented. Wikis are also used for communicating design decisions and their rationale to customers and maintenance team, who usually get the Wiki with the final release of the software.

## Architecture-related Challenges

Software architecture plays a vital role in developing and evolving any business critical system. Not paying sufficient attention to architectural aspects usually results in several issues that can potentially have negative impact on architectural practices, artifacts or design decisions. Following paragraphs briefly describe the most commonly observed architecture-related difficulties development teams experience when using agile approaches.

**Incorrect prioritization of User Stories:** One of the key architecture-related challenges commonly experienced by agile teams is that User Stories may be prioritized without taking the technical considerations into account. If critical interdependencies among User Stories are discovered later on, it usually requires significant re-factoring with consequences for the whole structure of the software. One way of dealing with this challenge is to involve architects and

developers in prioritizing User Stories in a group session, which is called "Feature Analysis Workshops.

**Lack of consideration for alternative design choices:** When the emphasis is on achieving the required features within time and budget, development teams are forced to focus on a limited number of solutions. One risk of this approach is that architects usually miss out on a possibly better design choice by not paying sufficient attention to upfront design. Developers may also find themselves in a dilemma as they are neither given upfront design nor time to come up with proper design. Since developers need to justify everything they do, so they tend to skip the consideration for alternative designs and implement whatever is known solution.

One potential solution to avoid this situation is to have an iteration for doing architecturally focused work – Zero iteration, which can easily be combined with the "Feature Analysis Workshop." Or there can be allocated time in each iteration for developers to think about different design choices.

**Lack of focus on quality attributes:** Lack of focus on quality attributes for making design decisions usually results in architectural structures that can hardly meet quality requirements later on. Such systems need

huge budget and time for fixing quality attributes during maintenance projects. Since the satisfaction of quality attributes is usually not a measure of success that is why development teams normally do not pay any particular attention to achieving quality attributes. One strategy to deal with this situation is to make the satisfaction of quality attributes a measure of success so development team has clear incentive for achieving quality attributes during the development project rather than leaving it for the maintenance project.

**Unknown domain and untried solutions:** Most of the time it is a common observation that new domain and untried solutions can present quite challenging situations. That is why agile approaches may not be appropriate when working in unknown domain, with new clients, or with untried solutions. If the domain and technologies are very well known and understood, it is relatively less risky to start developing features without giving much consideration to the architectural aspects in a particular domain and technologies being used. Hence, when working in an unknown domain with untried technologies, a hybrid approach may be more suitable to apply rather than pure agile. The agile approaches

may be more suitable to use when clients and solutions are well understood.

Other architecture-related challenges commonly experienced are: 1) lack of skill set as agile is more suitable to really good developers who know the system very well and are able to craft sophisticated design while implementing User stories with out having an upfront design activity; 2) Ad hoc and unplanned documentation of design decisions on Wiki usually result in several difficulties in finding the required information about the key design decisions. Such search can consume a lot of time and effort.

Dr. M. Ali Babar is a Senior Researcher with Lero, the Irish Software Engineering Research Centre, where he leads the research projects on Software Architecture and Evidence-Based Software Engineering. He is one of the lead researchers on A-Cube (Architecture-Centric Agile Approaches) initiative, which aims to bridge the gap between agile approaches and architectural practices. Prior to joining R&D, he worked in ICT for several years in Australia.

Table 1: The artefacts used by the agile teams compared with architecture-centric approaches

Activities	Artifacts	Artifacts used by the agile teams
Architectural analysis	Context	Platforms, fixed cost, fixed duration
	Requirements, Quality requirements	User stories focusing on features to be delivered. No particular focus on quality attributes
Architectural synthesis	Candidate architectural solutions	Limited number of solutions known by the team
	Architectural design (views, perspectives, prototypes).	Architectural Infrastructure plan (AIP), company policy and procedures, and User Stories
	Rationale	Rationale
Architectural evaluation	Quality attributes	Focus on features described by User Stories
	Architectural assessment	No particular focus on quality attributes Inter team cooperation for design assessment
Overall process driver	Architectural backlog	Product backlogs and Sprint backlogs