IEEE 42010 and Agile Process- Create Architecture Description through Agile Architecture Framework

Shun Chi Lo and Ning Chen
Department of Computer Science, California State University, Fullerton, California, USA

Abstract - ISO/IEC/IEEE 42010 Standard, Systems and Software Engineering - Architecture Description [1] is a comprehensive guideline used to conceptualize system architecture in an architecture description. An agile architecture framework provides a context and environment in which software architecting and agile development activities are collaboratively carried out. When a small software system evolves into an enterprise-class system, the architecture can be too complex to track changes and document architectural decisions without a standard. Also, when an organization decides to apply agile development methodologies to development life cycles, a big-up-front architectural design becomes more difficult and conflicts with the lean design concept promoted by the agile communities. Therefore, an agile oriented framework is required to integrate architecture design effort and agile development practices seamlessly. This paper attempts to explore a framework incorporating agile principles for expressing the architecture in an architecture description conforming to ISO/IEC/IEEE 42010 standard.

Keyword: IEEE Std 42010, Agile, Scrum, Architecture Framework

1 Introduction

Agile development methods have proven records as efficient and effective in software development through rapid integration cycles [2]. However, it is challenging to adopt agile development in an enterprise where complex systems need to be built by multiple teams in different locations. Without an agile architecture as a shared blueprint, agile development teams can end up releasing working subsystems, with defects because agile teams are isolated from each other without appropriate system integration discussion at the architectural level.

When face-to-face communication is not practical among agile development teams across multiple functional areas and physical locations, each team may have an inconsistent interpretation of the software architecture. Lack of shared architecture views can then lead to miscommunication, isolated development results, project delays, missed requirements, and a defective software system. This paper will explore methods of documenting software architecture in agile way based on the industry standard and incorporating agile practices into an architecture framework to close the gap between agile practices, agile architecture design, and enterprise-class development. To mitigate the risk of mentioned problems, an architecture framework and artifacts can be integrated into agile development practices.

This paper starts with a review of IEEE Std 42010, agile architecture principles, and industry techniques of incorporating architecture design into agile development practices. We then propose a framework for capturing architecture information that is consistent with agile methods. The paper ends with conclusion that summarizes with our findings, insights and suggestions.

2 What is IEEE 42010?


Architecture Description (AD) - a work product resulting from executing architecture analysis within the life cycle of the system-of-interest. The following are key elements of an architecture description:

- Stakeholders and concerns: Based on ISO/IEC/IEEE 42010 standard, the system concerns are considered fundamental to the architecture of the system-of-interest [1]. The system concerns are not intended to prescribe granularly how the system can fulfill business requirements or software requirements. System concerns are identified for system designers to conceptualize system architecture by addressing architecturally significant system concerns from stakeholders.
- Viewpoints: In the architecture description, viewpoints are defined to frame one or more stakeholders' concerns [4]. Based on a viewpoint, we can apply one or more model kinds in the forms of modeling language, notations, and conventions through modeling techniques, analytical methods and/or other operations.
Views: A view is a result of applying a viewpoint and expressed in Architecture Description Language (ADL) such as ArchiMate or UML 2.0 diagrams.

Correspondence rules: Every stakeholder, concern, architecture viewpoint, architecture view, and model kind are considered as architecture description elements. A correspondence defines a relation between AD elements. Correspondence rules are rules enforcing relations.

Architectural Framework (AF) - An architecture framework establishes a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular domain of application or stakeholder community. Uses of architecture frameworks include, but are not limited to: creating architecture descriptions; developing architecture modeling tools and architecting methods; and establishing processes to facilitate communication, commitments and interoperation across multiple projects and/or organizations [1].

3 What is Agile Architecture Framework?

An agile architecture framework provides a context and environment in which software architecting and agile development activities are collaboratively carried out. A well-defined agile framework helps agile teams to capture required information of architecture description. Traditional software development methodologies such as Waterfall depend on a Big Up-Front-Design (BUFD) to start the development. This approach assumes that capturing requirements and designing detailed architecture can ensure the software product fully meets all business needs. In reality, this approach is facing many challenges in modern software development because it is impossible to capture requirements all at once in the early phase. Secondly, the planned architecture may fulfill the requirements at the time the software is released, but the defined architecture is too rigid to adapt to new circumstances or accommodate new requirements over time. The traditional software development method sees architecture design as a separate task early in the lifecycle [1].

Agile approaches, on the other hand, focus on continuous and incremental software releases to end users and require lean architecture designs. Agile approaches also encourage cross-functional teams formed to address software functionalities iteratively to minimize the project management and documentation overhead of the development process. Based on the 11th principle of the Agile Manifesto [5] about architectures: "The best architecture, requirements, and designs emerge from self-organizing teams", the goal of agile methods is allowing motivated teams to adapt quickly to changes and continuously deliver shippable product increments. In reality, it is not easy to get software or system architects and agile developers work together in the same agile team because they focus on different objectives. Developers do not need a big-up-front design during a short iteration to complete an incremental release at a project level while architects want to ensure the incremental releases align with the architectural design at an enterprise level. The solution is not easy, but the solution is clearly suggested by the Agile manifest 11th principle that a self-organized team including architect and developers need to compromise at the midpoint between Agile development and Agile architecture. The following section summarizes key agile principles and practices regarding architectural design to identify the relationship between agile project activities and Agile architecture development.

3.1 Align Architecture Design and Agile Development

In agile methods, there is no extensive planning phase when complete architecture designing takes place. In reality, many organizations have system or software architects responsible for architecture designs regardless of development methods. According to Eloranta V-P, Koskimies K through interviews with scrum teams in different organizations, there are four main software architecture practices that organizations may choose to design architecture for agile projects [6]:

1. Big-up-front design:
   This practice requires analysis, synthesis, evaluation, and the design of the architecture before the complete system is implemented through sprints. During sprints, only small changes are made to the architecture design that is usually done by architects instead of developers before developers start to write code.

2. Sprint-zero:
   This practice allows the architecture design to be carried out before the first sprint without releasing any shippable product increment. The duration of Sprint-zero is usually two to four weeks. Most of the required architecture analysis and design effort need to be done in Sprint-zero. Compared to the big-up-front design approach, the architecture design is carried out by developers during a shorter period.

3. In-sprints:
   Architecture is built during sprints. The architecture design is carried out by developers who should be very experienced in refactoring to reflect changes in the architecture design within a short period. There is not much time for architectural analysis and synthesis; therefore, the architectural design has to be lightweight.

4. Separate-architecture-team:
   A separate team is assigned to design the architecture. The team is usually composed of system and software architects who need to analyze architecturally significant concerns and requirements. They need to provide scrum teams with their design and analysis for developers to implement. They also need to evaluate the architecture
after the scrum team releases the product increment to ensure the released system aligns with the architectural design. Changes to the architecture design can be made by the scrum team or the architect during sprints, and the design artifacts should be updated.

There is no right or wrong about each design approach. As long as there is a framework or workflow integrating architecture design effort and development in agile practices, an organization can customize the framework based on their organization structure, scrum team's skill sets, and the project size. The goal is to find an effective approach to design the architecture supporting software evolution using agile methods.

3.2 Document Architectural Changes and Decisions as Architecture Knowledge

Architectural Knowledge Management (AKM) [7] is a method to support sharing, distributing, creating capturing and understanding a company’s knowledge of software architecture. The purpose of managing architecture knowledge for agile development is to improve communication between stakeholders impacted by large-scale enterprise projects and help to guide the system evolution. To align with agile approaches demanding less up-front design artifacts, AKM itself must be lightweight. Agile organizations should determine how and where to store the architectural information and knowledge obtained from Agile development iterations. Once an architectural information repository is allocated, the agile team and architects can document and codify architectural knowledge such as architectural decision memos, high-level design diagrams, UML charts, architecturally significant system concerns and requirements. The concern-stakeholder traceability could be stored and tracked by knowledge management tools.

3.3 Build Intentional and Lean Architecture

The simplest architecture is one of the principles of the Scrum SAFe approach. SAFe (Scaled Agile Framework) [8] is a web-based, and freely revealed knowledge base for implementing Lean-Agile software and system development at a web-based, and freely revealed knowledge base for Scrum SAFe approach. SAFe (Scaled Agile Framework) [8] is needed to implement an enterprise-class system:

- Different scrum teams are not synchronized and can potentially create a system with different assumptions, thus causing architectural divergence
- When there are environment changes due to business needs, scrum teams are not notified or act on these changes with different solutions
- Without a common architecture, the developers may neglect some common system attributes such as performance, interoperability, maintainability, or availability attributes
- Without a common architecture, cross-cutting concerns such as data access, security, design patterns are not in sync among all teams

Intentional Architecture, therefore, can mitigate the risks of development silos. Intentional Architecture helps designers see a bigger picture with planned architectural initiatives, which can guide multiple scrum teams to implement synchronized, integrated and interoperable systems. Intentional Architecture should only address architecturally significant concerns and requirements. Through collaborative scrum activities, emergent design, and intentional architecture complement each other. Intentional architecture sets the boundary of emergent design and allows developers to adapt the intentional initiatives into system implementations. In parallel, emergent design corrects and solidifies intentional architecture that evolves over various sprints.

To align with the concept of agile incremental releases, an Architectural Runway provides "enough" system infrastructure and technological backend services to support the incremental implementations and future business features. Architecture has value only when it provides a simple and common description of the system. Intentional architecture has to be simple with clearly defined relationships between subsystems. Simple and working architecture require design skills. Outside the scrum teams, communities of practice [9] including cross-functional teams with different skills can be formed to provide knowledge and share best practices.

3.4 "Triple-A" Framework

While the need for architecture in agile development is still being debated, Jan Salvador van der ven and Jan Bosch [10] proposed a framework composed of three subjects: Agile, Architecture, and Axes. This framework identifies three axes:

1. the person making architectural decisions in Agile
2. the way in which the architectural decisions are communicated
3. the length of the decision feedback loop (how soon the proposed architecture can be implemented)

Based on these three axes, the authors conducted multiple case studies by interviewing project team members within software development organizations. The goal is to
assess how these axes affect the project result based on the return on investment, the speed of the project and the quality of the delivered product. This framework also helps software development entities organize agile teams and determine how they should incorporate architectural design activities into their software development cycles. The authors surveyed three aspects of the architecture creation process: Architect (Who), Artifact (How), and Periodicity (When). Figure 1 illustrates different scales of process assessment used to map each case. The more toward the center, the more traditional development model the organization uses. The more away from the center, the more Agile development model the organization uses.

Based on the survey result, the authors conclude their analysis that moving away from the center; the projects have higher chance of success, which means the organization should:

- Let development teams involve architectural decision-making
- Encourage direct communication and less static architectural documentation with great details
- Establish feedback loops and perform more frequent refactoring and continuous integration

5 Our attempt on integrating IEEE 42010 to Agile Process

5.1 Identify Stakeholders and Concerns

An independent architecture team composed of systems engineers and system architects are responsible for defining the agile architecture framework. The architect team is not reporting to the development organization. The Agile architecture framework starts with a vision defined by the product and business entities in the organization for an enterprise to envision what, who and how to build an enterprise class architecture to support multiple projects. The vision can be an idea from a program portfolio to release a new product, add a new feature, or apply a solution of the existing problem. With a clearly defined vision or goal, the project initiating team mainly composed of an architect, program manager, and project manager and product owner can further identify who will be impacted or involved as stakeholders through the complete Agile development life cycle. In the meantime, the project initiating team shall identify system concerns elicited from stakeholders through assessment and analysis process. Other Scrum team members can participate in Sprint 0 as necessary. Through the Envision process described in Agile Project Management by Jim Highsmith [2], the vision of the product can be documented as overview and introduction of the architecture description. The stakeholders and system concerns should be documented in the architecture description accordingly with traceability per the ISO/IEC/IEEE 42010 standard. Figure 2 depicts the relations and flows within the sprint planning process.

In Agile terms, these envision related activities and assessment effort are carried out during a sprint pre-game planning phase or "sprint 0."

5.2 Architecture Viewpoints and Views

In addition to stakeholders and system concerns identified during the sprint planning, the product owner also provides scrum teams with product features and related user stories. Based on the preliminary analysis of product features and user stories, architecture teams can then propose architecture views governed by architecture viewpoints. These architecture views are not intended to address all details and how software components must be built. The views should address most of the architecturally significant requirements and system concerns. Based on the Agile SAFe 4.0 Model, architecture teams can also create enabler stories, which are technical stories enabling the system to fulfill business user stories or achieve system attributes, which are non-functional requirement (NFR). For example, building the infrastructure
platform for hosting application can be defined as an enabler story. System concerns can be converted to enabler stories that need to be reviewed and fulfilled by scrum teams.

Figure 3 depicts the framework of how architecture views are created before the first sprint is started, and updated through various scrum sprints until the program increment is released. The following are major practices illustrated as step numbers in Figure 3 to create architecture viewpoints and views within the agile architecture framework:

1. Architects are part of the scrum team and own the Architecture Description as an architectural artifact
2. Product features and related users stories (business, Non-functional requirements, and enabler stories) are reviewed and analyzed by scrum teams during the sprint planning (Sprint 0)
3. During Sprint 0, stakeholders and system concerns, which can be converted to enabler user stories by scrum teams, are identified and documented in the architecture description
4. Based on the analysis during sprint 0, architecture viewpoints are created to respond to system concerns and provide modeling guidelines and conventions
5. Architecture views are created accordingly to address system concerns
6. During each sprint cycle: sprint task planning, daily scrum stand-ups, sprint review, and sprint retrospective, scrum teams can provide their feedback and refine architecture views. Changes are assessed, analyzed and documented in the architecture description by the architecture team
7. The architecture team uses the architecture description as a communication method among stakeholders to ensure scrum team members understand the overall system entities and their dependencies. Architects in Agile no longer play a role as a sole authority of architectural design; instead, architect is a supporting role serving stakeholders by aligning emergent designs with intentional architecture and product visions
8. A community of practice includes a group of developers, architects, subject matter experts, business executives or consultants. They are not in an actual scrum team, but can support Scrum teams with their knowledge on development standards, design patterns, and architectural principles. They can be tasked to analyze Spikes\(^2\), industry best practices, industry trends and provide adequate recommendations to scrum teams that make the final decisions.
9. During sprint 0, the architect should identify relations between AD elements such as dependencies between architecture views and define rules for enforcing relations
10. During actual sprints, the architect is responsible for reminding scrum team members of following correspondence rules, helping impediment removal and resolving conflicts between AD elements. The correspondence rules can be adjusted due to circumstance changes and consented by the scrum team
11. In Agile, detailed architectural designs are no longer required, but the architectural knowledge expressed by architecture views, diagrams, architecture description should be stored and maintained in an AKM tool. Architectural decisions and rationale can be tracked by enabler user stories or System spikes waiting for further analysis. In the Agile Architecture Framework, architecture description per the ISO/IEC/IEEE 42010 standard is the centerpiece of architectural work product that should be managed by the architecting application and stored as an evolving asset through multiple sprints and agile projects. Figure 3 Step #11 illustrates the AKM practice.

6 Agile Architecting Best Practices

Through the agile architecture framework, architects perform architecting tasks during not only the sprint planning but also throughout the entire development lifecycle in order to continuously influence scrum teams. The following are key factors of an effective agile architecture framework:

- Address architecturally significant concerns early and define a baseline architecture first
- Architecture team should be part of the development team during the Agile development life-cycles
- Treat software architecture as a blueprint for the system under development
- Use software architecture as a means of communication
- Convert system concerns to enabler stories linked to architecture viewpoints and views, add architecture spikes to the product backlog
- Document architectural changes and decisions
- Scrum team members should get training on Agile principles and actual practices

---

\(^2\) Spike is a story requiring a time boxed investigation. The output of a spike story is an estimate for the original story.
Figure 2 Sprint Planning

Figure 3 Agile Architecture Framework
7 Conclusion

As agile development methods are widely utilized by the software industry, the necessity of architecture related documentation is questioned and sometimes even recognized as a conflict with agile principles, which encourages constant requirement changes and promotes simplicity. This study identifies potential issues if agile development is carried out without proper architectural documentation. The answer to the agile development chaos is taking an approach for capturing architectural information on a view-based architecture description document complying with industry standards such as ISO/IEC/IEEE 42010 in a way that is consistent with agile methods. Through an agile architecture framework, software or system architects create an architecture description document that evolves in parallel with agile development life cycles. The architecture description serves as a blueprint for agile software development and can be scaled to support software evolution realized by multiple agile projects. An architecture framework facilitates the development activities to support and promote agile projects by identifying new system concerns as the software system evolves. The architecture description becomes a communication tool allowing stakeholders to understand the project agility at an organizational level. The more a company applies agile practices to software development, the more important the company needs to define an architecture framework. This paper recommends various steps and best practices for capturing architectural information in an architecture description document through the agile architecture framework.

8 References


