2020

Aligning an ISO/EIC 42010 System Architecture Model and Agile Practice

Ronald Beckett
Swinburne University of Technology, rcb@reinvent.net.au

Rajyalakshmi Gaddipati
Charles Sturt University, rgaddipati@studygroup.com

Aligning an ISO/EIC 42010 System Architecture Model and Agile Practice

Ronald C Beckett
Swinburne University of Technology
Melbourne, Australia
Email: rcb@reinvent.net.au

Rajyalakshmi Gaddipati
Charles Sturt University
Melbourne, Australia
Email: rgaddipati@studygroup.com

Abstract

The ISO/EIC 42010 system architecture description standard evolved over a number of years with substantial practitioner inputs. It presents a high level, top-down view of requirements that may be interpreted as needed for different applications. Agile system development methods have proved effective in practice, but represent a bottom up view drawing on user stories. The question considered in this paper is how they might be harmonised. Experience from using these tools over several years in practical masters degree student projects has been used to explore this question. We suggest a logical compatibility lies in their core themes: stakeholder needs (who) frame architecture descriptions (what) and the associated rationale (why). A particular interpretation of ISO/EIC 42010 and a model outlining the evolution of architecture in an agile environment are presented. Several suggestions for future research are made.

Keywords Agile, architecture description, user stories, models

1 Introduction

As smart devices and processes become part of everyday life, new ways of working and doing business continue to evolve. There is pressure for enterprises to become more agile to cope with changing environments (e.g. Nijssen and Paauwe, 2012). Fast learning and an ability to rapidly reconfigure assets is observed in the responses. Systems engineering, computer science and information systems professionals need to work together in developing and sustaining such practices, and role flexibility and boundary spanning is needed (e.g. Sheard, 1996; Prifti et al, 2017). Tools that help reach common understandings and rapidly develop solutions to challenges are needed. For example, Ovaska et al (2003) described the role of system architecture in providing a coordinating mechanism to help understand a system's functionality. Qumer and Henderson-Sellers (2008) provided an overview of the adoption of agile methods that may help rapidly develop software solutions. However some researchers suggest there are potential shortcomings with such top-down or bottom-up approaches. Thummadi and Lyytinen (2020) compared actual practices and outcomes in different development teams using traditional (requirements engineering /waterfall) and agile project management practices in a large company. They observed process and team knowledge and skill gaps, we whish explore further in this paper. Our research question is: *how might architecture and agile perspectives be harmonized?* In this paper, we draw on experience from two knowledge and skill development cases to compare similarities, differences and needs. Literature relating to architecture description frameworks and agile practices is first briefly reviewed, followed by a description of the research approach and findings. This leads to the presentation of system architecture description and agile architecture development models, and suggestions about how they can be harmonized.

2 Some Observations from the Literature

From a study of goal congruence in a multi-site development environment, Ovaska et al (2003) noted the benefits of defining and describing the system architecture plus requirements for an associated development methodology. In broad terms, the latter involved coordination via a focus on the linkages between architecture functional components. In this way, all the component designers had a common understanding of interdependencies and had well-defined interfaces that supported development of their own components. Youngs et al (1999) described the need for and benefits from the use of an architecture description standard at IBM. An emphasis on a minimal set of shared concepts supported

practical use and facilitated teaching a broad range of practitioners with different skills. Working in a similar style across the organization facilitated flexible work assignment and the documents produced could form part of the enterprise knowledge base. A little later, Maier et al (2001) outlined architectural description practices reflected in an IEEE standard, which was subsequently adopted as an ISO standard. A key idea of IEEE 1471 was the introduction of *architecture viewpoints* to codify best practices for the creation and use of architecture views within an architecture description. Emery and Hilliard (2009) provided background to the deployment of the ISO standard in a paper titled "Every architecture description needs a framework: Expressing architecture frameworks using ISO/IEC 42010".

The following briefly presents this framework and discusses some aspects of its application. Then a review of the application of architectures in agile development projects is presented, where some researchers have suggested there may be coordination issues. This is followed by a brief discussion of the utility "user stories" that are at the core of agile architecture development.

2.1 The ISO/EIC 42010 Architecture Description Framework

The elements of this framework are shown in figure 1. The way we view the framework is as follows:

1. The rounded boxes at the top of the diagram (mission, environment, system, and architecture) represent context. The ISO 42010 statement relating to mission is: *A system exists to fulfill one or more missions in its environment. A mission is a use or operation for which a system is intended by one or more stakeholders to meet some set of objectives.* The operating environment may be considered from multiple perspectives: engineering and operational considerations combined with the digital and application environment as shown. In thinking about the systems element of the ISO 42010 framework it is first necessary to declare the boundaries – what functionality is required within the system to achieve the desired mission and what linkages with the broader ecosystem (e.g. internet connection) are needed to make it work. The architecture is intended to show high level functionality - *Architecting is concerned with developing satisfactory and feasible system concepts, maintaining the integrity of those system concepts through development, certifying built systems for use, and assuring those system concepts through operational and evolutionary phases. Detailed systems engineering activities, such as detailed requirements definition and interface specification and the architecting of major subsystems are tasks that typically follow development of the system architecture. (ISO 32010).*
2. The central shaded boxes show interactions framing the architectural description. The ISO 42010 standard indicates that: *The principal class of users for this recommended practice comprises stakeholders in system development and evolution, including the following:*
    - *Those that use, own, and acquire the system (users, operators, and acquirers, or clients)*
    - *Those that develop, describe, and document architectures (architects)*
    - *Those that develop, deliver, and maintain the system (architects, designers, programmers, maintainers, testers, domain engineers, quality assurance staff, configuration management staff, suppliers, and project managers or developers)*
    - *Those who oversee and evaluate systems and their development (chief information officers, auditors, and independent assessors*
    - *A secondary class of users of this recommended practice comprises those involved in the enterprise-wide infrastructure activities that span multiple system developments, including methodologists, process and process-improvement engineers, researchers, producers of standards, tool builders, and trainers.*

    ISO 42010 defines an architecture description as *a collection of products to document an architecture. Multiple viewpoints are to be represented, and any inconsistency between viewpoints noted. It is suggested the rationale for the architectural concepts selected should be represented, along with evidence of the consideration of alternative architectural concepts plus the rationale for the choices made. If describing an adaptation of a pre-existing system the rationale for the legacy system architecture should be presented*
3. The smaller lower boxes represent ways that multiple viewpoints may enrich the architectural description. These boxes will be discussed further later.

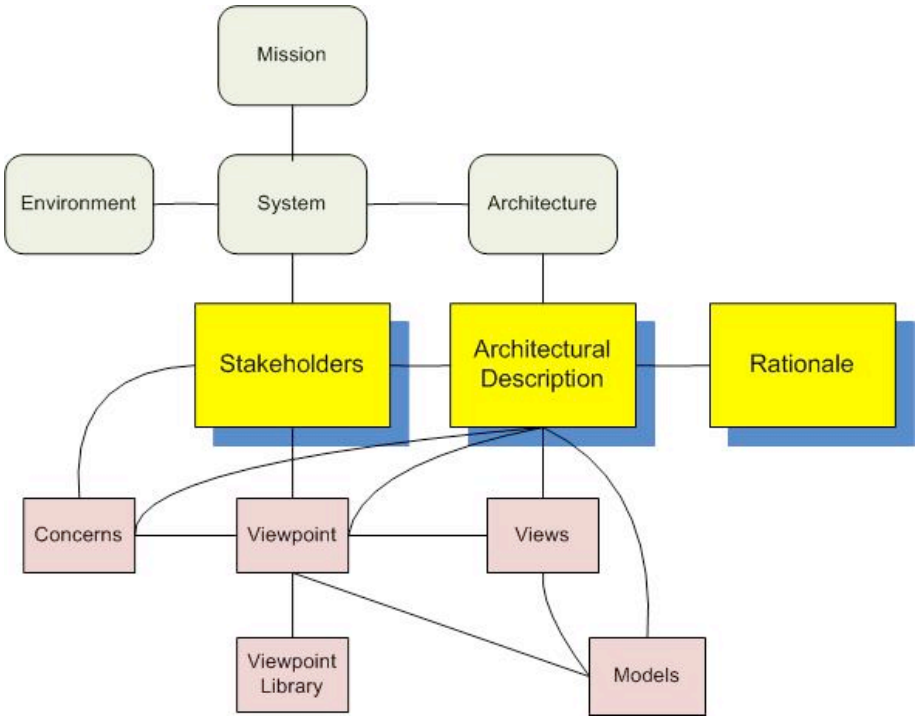

Figure 1. Elements of the ISO/EIC Framework

		To											
		Stakeholders	Architectural description	Rationale	Mission	Environment	System	Architecture	Concerns	Viewpoint	Viewpoint library	Views	Models
From	Stakeholders	X	I	I			I		I	I			
	Architectural description	I	X				I					I	I
	Rationale		I	X									
	Mission				X		I						
	Environment					X	I						
	System	I	I		I	I	X	I					
	Architecture						I	X					
	Concerns		I						X	I			
	Viewpoint	I	I							X	I	I	I
	Viewpoint library									I	X		
	Views		I							I		X	I
	Models		I							I		I	X

Table 1. An ISO 42010 element interaction matrix

Mo and Beckett (2019) made the following comment on the interactions shown between the elements in figure 1. The standard states that *"in the figure, boxes represent classes of things. Lines connecting boxes represent associations between things. An association has two roles (one in each direction)."*

Within the body of the standard there is reference to other interactions, e.g. "*the rationale shall address the extent to which the stakeholders and concerns are covered by the viewpoints selected*." Mo and Beckett (2019) captured these ideas via an interaction matrix, shown as Table 1. The shaded 'I' cells represent the linkages shown in Figure 1. Mo and Beckett noted that whilst the diagonal (black X) cells were shown a null, there may be interactions within the cells, e.g. between stakeholders.

2.2 The Evolution of an Architecture Description in an Agile Project

Gill (2015) contends that whilst agile development practices focus on developing and delivering working software systems in small iterations with minimal documentation, locally project focused agile practices overlook the need for holistic enterprise architecture. This is seen as a gap in practice. Paetsch et al (2003) suggested that requirements engineering and agile approaches may seem incompatible, with the former relying on a document-centric orientation towards coordination and the latter on a people-centric orientation (e.g. Eberlein and Leite, 2002). However potential synergies were seen in the areas of: customer involvement, interviews, prioritization, modeling, documentation, validation, configuration management, brainstorming, and handling non-functional requirements. Schön et al (2017) conducted a systematic literature review of agile requirements engineering and also suggested there was a need for more empirical studies. They noted that continuous communication and collaboration was the most common approach to user engagement and the most commonly utilized artifacts were user stories, prototypes, use cases, scenarios and story cards. The definition and delivery of non-functional requirements was seen as a particular challenge. Gayathri and Theetchenya (2016) also suggested that lack of attention to non-functional requirements at an early stage of development may compromise software projects. Costa et al (2014) observed that in relatively small agile projects where there is good customer involvement minimal formal modeling or documentation may be needed. That may not be the case in other projects where critical issues and architectural diagrams may have to be formally identified, e.g. in the management of multiple scrum teams. They suggested a practice where user stories relating to architecture interface, data and control elements might be linked to a use case. They suggested templates that could be used for this purpose. The concept was successfully trialed in a large industry project.

2.3 The Utility of User Stories

Wautelet et al (2016) argued that user stories in isolation do not present a coherent representation of the intended system 'to-be' functions and proposed an approach to map user stories to a use case diagram. They proposed associating tags with stories to support their clustering in various ways. Liskin et al (2014) observed that sometimes user stories were too coarse, potentially leading to misunderstandings This was seen as important to development practitioners, particularly where a large volume of work was associated with one story, making it difficult to fit into sprint schedules. Daly (2015) discussed a related practitioner approach using features of an agile project management tool (Jira) which supported the representation of a large issue as an 'Epic' that may have associated lower level user stories and sub-tasks.

The idea of user stories may seem simple –who needs what and why, but creating appropriate detail and using them may not. Cohn (2004) suggests that the scope of a user story should allow delivery on it to be completed in one sprint. He suggests a user story should have three attributes: (a) a description used for planning and as a reminder, (b) conversations about the story helping to flesh out details, and (c) information that supports testing to determine when the story is complete. An initial goal-level story can be replaced with more detailed ones when that becomes necessary. He suggests: *A team can very quickly write a few dozen user stories to give them an overall feel for the system. They can then plunge into the details on a few of the stories and can be coding much sooner than a team that feels compelled to complete an IEEE 830–style software requirements specification.* In relation to such requirements specifications, Cohen comments: *Unfortunately, it's effectively impossible to write all of a system's requirements this way. A powerful and important feedback loop occurs when users first see the software being built for them. When users see the software, they come up with new ideas and change their minds about old ideas. When changes are requested to the software contemplated in a requirements specification, we've become accustomed to calling it a "change of scope."* From this point of view, it might be argued that the architecture 'evolves' in an agile project.

3 Research Approach

With the foregoing in mind our research question was: *how might architecture-oriented and agile-oriented practices be harmonized?* We drew on case studies representing the two viewpoints to compare similarities, difference and unmet needs. We utilized an action research strategy (Coghlan,

2003) in the case studies where we were directly engaged in delivering theory to inform practice and learning from that experience. In both cases the aim was, in a structured hands-on way to simulate a professional workplace, introducing Masters degree students to practices they were not necessarily familiar with. Learning was to be student team driven with academic facilitation, and at the end review and feedback was provided on what was delivered. Observations collected regarding the tools used were made via the formal assessments carried out and via student reflections required at the end of each course. We (the researchers) had a number of face-to-face meetings to discuss our respective observations, and used an on-line Slack project space to share information and record our discussions in the development of this paper.

In one case study called Support Solutions Architecture (SSA) students taking a capstone Masters of Systems Engineering subject were observed. The bespoke industry subject was oriented towards innovative and economical long-term support arrangements for complex engineering assets. The primary business driver was a move towards private-public partnerships in establishing such arrangements (see Beckett, 2017). Both operational and business outcomes had to be considered from the viewpoints of a diverse range of stakeholders. The subject utilized a blended learning approach. It was run over 6 weeks in an intensive delivery mode. Over several years student cohorts of experienced engineering support practitioners from different companies tackled 15 projects. ISO/EIC 42010 was used as framework to capture and compare attributes of multiple case studies across a number of industry sectors using a pre-configured wiki tool. Consideration was then given to long-term changes in the operating environment, and student teams had to propose consequent adaptations to the as-is situation.

The other was called Masters Project (MP) and was a capstone WIL subject in enterprise systems and in software engineering programs. Over several semesters, some 40 industry-initiated enterprise systems projects were tackled by teams of international students. This subject was organized as a series of sprints. Student teams were given client firm briefs and contacts and were asked to draw out and work with user stories. They had to deliver three reports and associated artifacts endorsed by the client over the subject duration.

4 Findings

4.1 The SSA Case

This bespoke defence industry course built on experience with a similar course offered in another country. The blended learning format involved both initiating and final face-to-face week-long workshops and on-line interaction using university LMS tools in between. A trial version of the course was first delivered to a cohort of very experienced system support practitioners. Many members of this cohort had prior experience working with some form of enterprise architecture system. The first version of the SSA subject contained a lot of theoretical content it was not regarded as appropriate. A proposal for content and team-based activities built around ISO/EIC 42010 was accepted. Subsequent cohorts consisted of individuals nominated by their companies, and all were regarded as IT literate.

The upper parts of the framework shown in figure 1 were tackled in chunks in the style of agile sprints to define an as-is situation. Outcomes were progressively documented in a multi-page wiki, and that content formed the basis of formal assessments with feedback. As we were dealing with economically viable, knowledge-based socio-technical systems, supplementary tools reflecting business model and knowledge architecture views were also introduced. This could be interpreted as models stimulating particular views. The students only had access to published case material, so the collection of user stories was not practical. As an alternative, the teams were prompted with a number of questions that could be interpreted as a viewpoint template:

- What are the activities/physical processes at work? Who does what to make them happen?
- What information is needed to undertake each physical process?
- What background knowledge is needed to support each physical process?
- What kinds of technical data have to be managed in the data system?
- What kinds of management/control data have to be managed in the data system?
- What kinds of background knowledge are associated with managing this data?
- What kinds of decisions have to be made, and how is this process supported?

- What kind of data is needed to make decisions?
- What kinds of background knowledge support the decision system?
- What overarching architectural knowledge is needed to integrate the physical/data/decision systems and understand linkages between this and other functional systems?

Many teams noted more interactions between elements of the framework than shown in figure 1, and when one team added these linkages from the point of view of architecture sub-components, the resultant diagram looked like a bowl of spaghetti (see figure 2). The team referred to this representation as 'an ecosystem' that included expectations of the broader community and the influence of competitors. The first learning here was that an interaction matrix (Table 1) view drew out additional discussion, particularly when sub-element aspects were considered. By way of example, there were interactions between the business model and stakeholders, rationale, viewpoints and views.

In the first day of the final workshop, teams had to develop a presentation on their project 'as-is' situation and present it to the other teams, academics and an industry representative. This illustrated the extent to which individual teams may have adapted the ISO/EIC 42010 framework. A second learning was that, along the way, most teams had started to identify concerns to be addressed. This generally reflected a risk management perspective common in their home industry. A third learning was that reframing of explanation of the boxes shown in the lower part of figure 1 was needed as their utility was questioned. Treating them as a set representing operational views seemed to make more sense than considering them individually. A fourth learning related to team member interaction. Curiously, some teams thought they functioned better on-line than face to face. This may be the subject of a separate study.

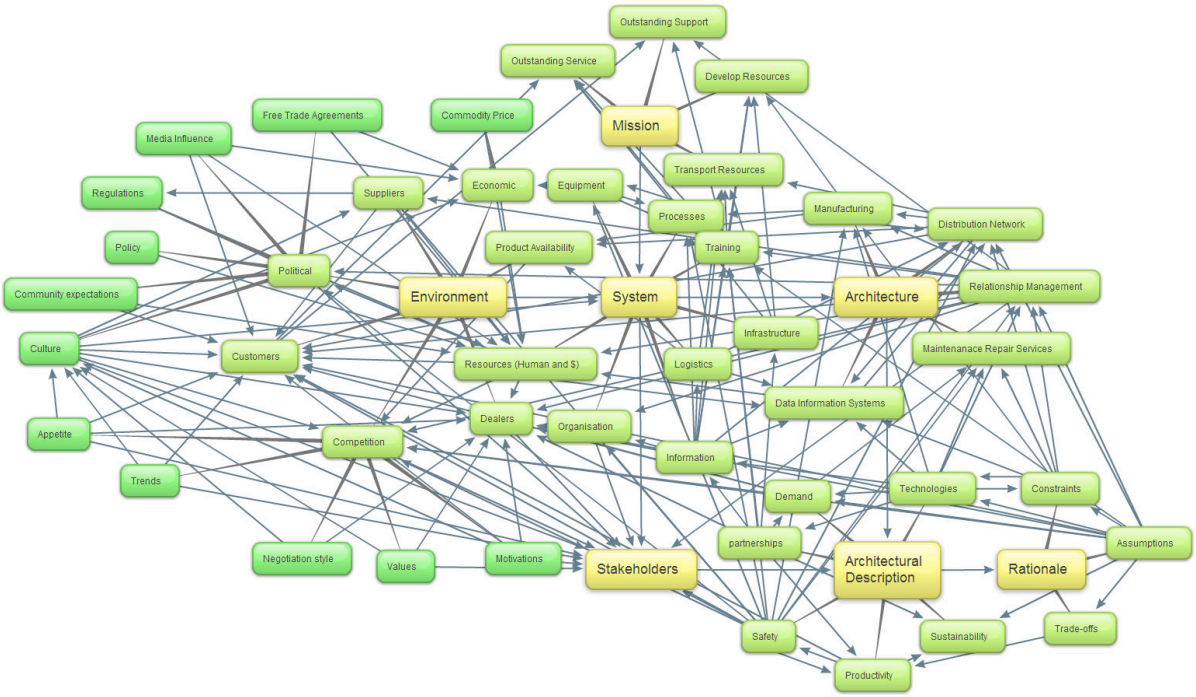

Figure 2. Complex interactions between ISO 42010 elements.

The next step was to identify global challenges that might impact all or particular stakeholders (who), and for each team to develop a 'to-be' response to one of them in their project (what). Whilst many parts of an organization may be impacted, the task was to focus on one aspect of operations to propose an innovative change. A form of template was provided to facilitate consideration of views from the perspective of different stakeholders. The value proposition, what was required to deliver on this and who would be responsible for what taking potential risks into account were represented in this viewpoint template (why). At the end of this final workshop the cohort was invited to reflect what had been learned, what might be improved in delivering the subject, and present this to a panel of academic and industry representatives. A fifth learning was that the ISO 42010 framework had to be

supplemented with complementary models and viewpoints at appropriate stages of case analysis to facilitate its use in the particular application.

4.2 The MP Case

The MP subject was organized as face-to face lectures with scheduled workshop time. But as with all university courses, students were expected to do additional work outside the classroom, and that meant both assigning tasks within the team and organizing meetings. Very few students had worked in self-managed teams before, and it took some time for students to adapt. The variety in team dynamics could be the subject of a separate paper. Teams were offered access to an online project management tool (Basecamp) and could also invite clients into this workspace, but usage was not consistent. This was later changed to a mandated requirement to use Microsoft Teams. Our first learning - make team interaction protocols clear and provide training as needed. The projects fell into two categories: (a) 'research' projects exploring the potential utility to a client business of new tools like block-chain, data analytics and cloud-based special purpose software, and (b) 'requirements' projects where the team had to develop software requirements and change management proposals for later implementation. In the first two weeks the teams had to develop a project plan, learning something about the client, the project requirements and identifying the types of activities to be considered in forthcoming sprints. The project clients were generally small firms that were not all familiar with the user story concept, so could be confused when students asked them for their user stories. In addition, clients would express their requirements in their own professional language, which had to be interpreted by the students. Our second learning was - appreciate the need for boundary-spanning activities. Client user stories were mostly pitched at a fairly high level, and the teams needed a finer level of detail to progress their sprints. This was addressed by going back to clients with specific questions or developing proposed user stories in conjunction with a Supervisor, and taking these back to clients for discussion. Our third learning was - understand granularity considerations when working with user stories. The collection of user stories had to be supplemented with additional views, e.g. use case diagrams and sequence diagrams. Our fourth learning - user stories need to be supplemented with some form of architecture description. Some matters of timing were problematic. Getting all team members synchronized, finding time windows to work with clients and get their signoff on proposals / work were frequently problems. Our fifth learning: both the scheduling of internal team activities and external interactions need to be considered in maintaining fast-paced iteration cycles.

5 Discussion

Our literature survey described two potentially misaligned approaches to system definition: top –down requirements engineering and bottom-up agile development. The criticism of top-down approaches was that if taken to fine levels of detail the task can be burdensome and the result difficult to change. The criticism of bottom-up agile methods was that, whilst there is some flexibility offered, system elements may not link well in an overall structure. We drew on experience in two case scenarios – one top-down oriented and the other bottom-up oriented. Following the lead of the Thummadi and Lyytinen (2020), cases were compared in terms of development activities: understanding the project, identifying stakeholders, identifying customer requirements, organizing functional requirements, adding non-functional requirements, client feedback. From a top-down architecture perspective, this showed some consistency if the following adaptations to the ISO/EIC 42010 framework shown in figure 1 were made:

- The combination of mission, system and environment may be viewed as a set framing operational context. We have added a link between environment and stakeholders reflecting a potential broader community or government interest in a particular system.
- The core activities are slightly reframed reflecting the user story elements: stakeholders = who, architectural = what, rationale = why. Stakeholders are those with specific requirements or those who may be impacted by current issues. The architecture description should include structured collections of user stories. The rationale will include why a particular feature is needed and why a particular implementation choice (how) was made. Consistent with the literature (e.g. Capilla et al, 2016) we have viewed decision / rationale information as representing knowledge architecture.
- Recognizing that concerns have multiple dimensions (e.g. Lago et al, 2010) these were clustered under the categories of potential risks and non-functional requirements. This is a topic for future research.

- Views, viewpoints, viewpoint libraries and models have been clustered together to represent a set of operational views from an analyst perspective. One item in a viewpoint library could be a user story template. Following the lead of others, this will be a topic for future research

The outcome is shown in figure 3, which also notes parallels with stakeholder theory reflecting a business perspective (e.g. Lucke and Lechner, 2011). This could help the stakeholders align with associated change management activities.. This will also be a topic of future research.

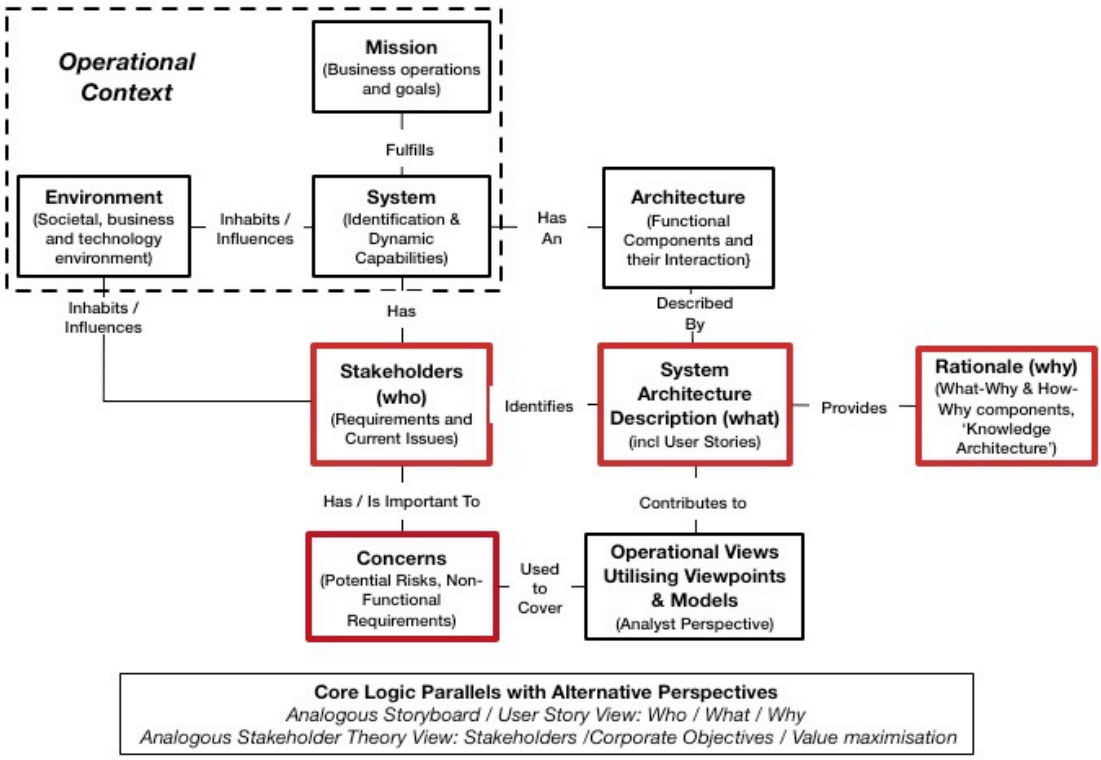

Figure 3. An alternative representation of the ISO/EIC 42010 Architect Description Framework

From a bottom-up Agile perspective, it is suggested user stories can be utilised at every stage of an agile project to identify and evolve an architecture as follows:

- **Requirements Gathering**: Initially, Business defines strategies, policies and procedures. Business Client must provide the detailed user stories on the expected system functioning. These user stories must be reviewed and tested by the stakeholders, operational, strategic and tactical level of employees and employers.
- **System Analysis**: In this phase, Business Analyst and Software Analyst must define the user stories together. The complex system can be broken down into simple components to understand the functional features. Then define the user stories for all the components and functional features.
- **Architectural Design**: During this phase implement the design based on the above defined user stories. However, the design may not be compatible for all the user stories. Eliminate or redefine the user stories which fit into the proposed design. Redefinition or elimination stage needs approval from the business client. This phase must consider the Hardware and software used by the business client. Then write review and rewrite the user stories if required.
- **Development and Testing**: This phase includes writing the code based on the defined User stories so far. Re-define the user stories if coding cannot be generated for the specific defined user story or a certain functional feature cannot be developed. This phase needs the

microscopic level of defining the user stories as they can be reflected in test cases, which are utilised as follows:

- *Testing Activity 1:* Starts after the requirement gathering and before the design/development begins to design component test protocols
- *Testing Activity 2:* starts after and during the design and development phase to ensure integration of components.

The process requires continuous monitoring, reviewing defining and redefining the user stories as experience is gained from progressive testing and customer feedback. The proposed evolutionary process is summarised in figure 4 and may be viewed as an experiential learning cycle (e.g. Kolb, 2014). This supports development in two ways. Firstly, there is fast feedback from clients to verify (or otherwise) their requirements have been met. Secondly, as clients begin to appreciate what is practical, they may wish to modify the functional requirements of the system. An agile architecture-building practice could be aligned with ISO/EIC 42010 if that were to be the architecture design description vehicle to provide a bridge between different stages of agile development and help a business adapt to a dynamic IT environment.

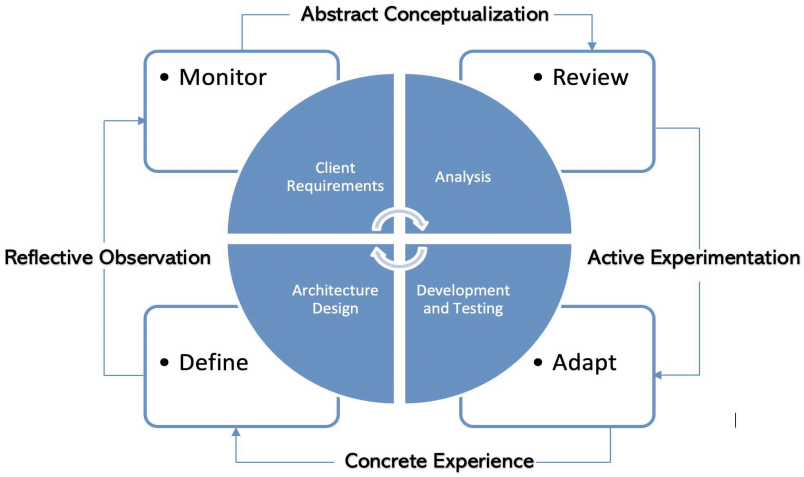

Figure 4. Agile architecture building cycle

We consider there is a logical compatibility between the two cases presented even though the details may be different. Both started by understanding something about the project and its context. Both drew out requirements, albeit in a different way, reflecting who needed what functionality and why. Both considered non-functional requirements/concerns, and both drew on some form of supplementary model.

Schön et al (2017 had suggested there was a need for more empirical studies of 'Agile Requirements Engineering', and that in 93% of the studies they examined. the use of suitable artifacts was required. Drawing on our empirical studies, we further these suggestions in two ways:

- By proposing two artifacts that may offer complementary views of complex systems and their evolution (figures 3 & 4), and
- By proposing how they might be linked together to bridge stakeholder and system perspectives.

In the SSA case, architecture elements of figure 3 were documented in an evolutionary way using a wiki tool. In some MP case projects, a storyboard having similar elements to figure 3 was used as a project starting point. The common theme was some form of architecture design to reflect client requirements.

6 Concluding Remarks

In this paper we considered the utility of two tools in practice: an architecture description framework (ISO/EIC 42010) and an agile project management methodology, drawing on experiences of Masters degree students over multiple practice-oriented projects in two different cases. We started with a

literature review of architecture description frameworks and agile practices. This indicated there might be some issues to be managed with each tool, and they might be incompatible. With the foregoing gap in mind our research question was: *how might architecture-oriented and agile-oriented practices be harmonized?* We observed that the architecture tool required the integration of multiple views of a system, identifying stakeholders (who), an architecture description (what) and the underlying rationale (what). This was seen as the same logical structure as the user stories used in agile practice. We make two contributions to theory and practice. A particular interpretation of the ISO/EIC framework presented in the paper (figure 3) reflects experience from the analysis of 15 large company systems engineering practical projects. An agile architecture evolution model presented in the paper (figure 4) reflects experience from 40 small company information systems practical projects.

A limitation of the research may be that we drew on student projects. From this perspective, the utility of the tools developed (reflected in figures 3 & 4) remain to be tested in an industry setting. However, they did have a practical foundation. In the architecture case, the students were practitioners with more than 10 years experience who were sponsored by their employers to learn something to be applied in current workplace projects. In the agile case, candidate projects were negotiated with industry or government clients who were hoping to learn from the student projects.

Three areas for further research have been suggested in relation to ISO/EIC 42010: one relating to the treatment of the concerns element, one relating to the treatment of the view / viewpoint / model element, and one relating to the application of stakeholder theory to architecture considerations. It was also suggested the agile architecture evolution model be further tested using the ISO/EIC 42010 framework as the architecture design vehicle at the architecture definition stage (see figure 4)

7 References

Beckett, R. C. (2017) A Service-Dominant Logic view of Private-Public Support Services Partnerships. 15th ANZAM Operations Management, Supply Chain and Services Management Symposium, Queenstown, New Zealand, 13 – 14 June.

Capilla, R., Jansen, A., Tang, A., Avgeriou, P., & Babar, M. A. (2016). 10 years of software architecture knowledge management: Practice and future. Journal of Systems and Software, 116, 191-205.

Coghlan, D, (2003) Practitioner Research for Organizational Knowledge: Mechanistic- and Organistic-oriented Approaches to Insider Action Research. Management Learning 34(4): 451 - 463

Cohn, M.: User Stories Applied. Pearson, Boston (2004

Costa, N., Santos, N., Ferreira, N., & Machado, R. J. (2014, June). Delivering user stories for implementing logical software architectures by multiple scrum teams. In International Conference on Computational Science and Its Applications (pp. 747-762). Springer, Cham.

da Costa, A. A. Jnr, Misra, S., & Soares, M. S. (2019, July). A systematic mapping study on software architectures description based on ISO/IEC/IEEE 42010: 2011. In International Conference on Computational Science and Its Applications (pp. 17-30). Springer, Cham

Daly, L (2015) Break it down: decomposing user stories in Jira

https://www.atlassian.com/blog/jira-software/break-decomposing-user-stories-jira

Eberlein, A., & Leite, J. C. S. P. (2002). Agile requirements definition: A view from requirements engineering. In Proceedings of the International Workshop on Time-Constrained Requirements Engineering (TCRE'02) (pp. 4-8).

Emery, D., & Hilliard, R. (2009). Every architecture description needs a framework: Expressing architecture frameworks using ISO/IEC 42010. In 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture (pp. 31-40). IEEE.

Gayathri, S., & Theetchenya, S. (2016). Requirements Engineering For Eliciting Non Functional Requirements Using Sprints In Agile Environment. Journal of Recent Research in Engineering and Technology, 3(2)

Gill, A. Q. (2015). Adaptive enterprise architecture drivenagiledevelopment. In International Conference on Information Systems Development, ISD 2015

ISO 42010 (2007) Systems and software engineering — Recommended practice for architectural description of software-intensive systems. International Standard ISO/IEC 42010, IEEE Std 1471-2000 https://www.iso.org/standard/45991.html

Kolb, D. A. (2014). *Experiential learning: Experience as the source of learning and development*. FT press.

Lago, P., Avgeriou, P., & Hilliard, R. (2010). Guest editors' introduction: Software architecture: Framing stakeholders' concerns. IEEE Software, 27(6), 20-24.

Liskin, O., Pham, R., Kiesling, S., & Schneider, K. (2014). Why we need a granularity concept for user stories. In International Conference on Agile Software Development (pp. 110-125). Springer, Cham.

Lucke, C., & Lechner, U. (2011). Towards an Approach for Stakeholder-Oriented Elicitation and Identification of Concerns in EA. In IFIP Working Conference on The Practice of Enterprise Modeling (pp. 147-161). Springer, Berlin, Heidelberg.

Maier, M. W., Emery, D., & Hilliard, R. (2001). Software architecture: Introducing IEEE standard 1471. Computer, 34(4), 107-109.

Mo, J. P., & Beckett, R. C. (2018). Architectural knowledge integration in a social innovation context. In Transdisciplinary Engineering Methods for Social Innovation of Industry 4.0. Proceedings of the 25th ISPE International Conference on Transdisciplinary Engineering (pp. 763-772). IOS Press.

Nijssen, M., & Paauwe, J. (2012). HRM in turbulent times: how to achieve organizational agility? The International Journal of Human Resource Management, 23(16), 3315-3335.

Ovaska, P., Rossi, M., & Marttiin, P. (2003). Architecture as a coordination tool in multi-site software development. Software Process: Improvement and Practice, 8(4), 233-247.

Paetsch, F., Eberlein, A., & Maurer, F. (2003, June). Requirements engineering and agile software development. In WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. (pp. 308-313). IEEE.

Panunzio, M., & Vardanega, T. (2014). An architectural approach with separation of concerns to address extra-functional requirements in the development of embedded real-time software systems. Journal of Systems Architecture, 60(9), 770-781.

Prifti, L., Knigge, M., Kienegger, H. and Krcmar, H. (2017) A Competency Model for" Industrie 4.0" Employees. Wirtschaftsinformatik, University of St Gallan, Switzerland, February 12 - 15 https://aisel.aisnet.org/wi2017/track01/paper/4/

Qumer, A., & Henderson-Sellers, B. (2008). A framework to support the evaluation, adoption and improvement of agile methods in practice. Journal of systems and software, 81(11), 1899-1919.

Schön, E. M., Thomaschewski, J., & Escalona, M. J. (2017). Agile Requirements Engineering: A systematic literature review. Computer Standards & Interfaces, 49, 79-91.

Sheard, S. A. (1996). Twelve systems engineering roles. In INCOSE International Symposium (Vol. 6, No. 1, pp. 478-485).

Thummadi, B. V., Shiv, O., & Lyytinen, K. (2011). Enacted routines in agile and waterfall processes. In 2011 Agile Conference (pp. 67-76). IEEE.

Wautelet, Yves ; Heng, Samedi ; Hintea, Diana ; Kolp, Manuel ; Poelmans, Stephan (2016) . Bridging User Story Sets with the Use Case Model. The International Conference on Conceptual Modeling (Gifu, Japan, du 14/11/2016 au 17/11/2016). In: Sebastian Link and Juan Trujillo, Advances in Conceptual Modeling - {ER} 2016 Workshops, AHA, MoBiD,MORE-BI, MReBA, QMMQ, SCME, and WM2SP, Gifu, Japan, November 14-17,2016, Proceedings, 2016, p. 127–138 http://hdl.handle.net/2078.1/181121

Youngs, R., Redmond-Pyle, D., Spaas, P., & Kahan, E. (1999). A standard for architecture description. IBM Systems Journal, 38(1), 32-50.