The Journal of Systems & Software 184 (2022) 111111

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

# Architecture evaluation in continuous development☆

S. Magnus Ågren [a,*], Eric Knauss [a], Rogardt Heldal [b,a], Patrizio Pelliccione [c,a], Anders Alminger [d], Magnus Antonsson [d,1], Thomas Karlkvist [d,2], Anders Lindeborg [d,3]

[a] Chalmers | University of Gothenburg, Sweden
[b] Western Norway University of Applied Sciences, Norway
[c] Gran Sasso Science Institute (GSSI), Italy
[d] Volvo Cars, Sweden

ARTICLE INFO

*Article history:*
Received 6 October 2020
Received in revised form 3 August 2021
Accepted 26 September 2021
Available online 19 October 2021

*Keywords:*
Architecture evaluation
Continuous software engineering

ABSTRACT

**Context** In automotive, stage-gate processes have previously been the norm, with architecture created mainly during an early phase and then used to guide subsequent development phases. Current iterative and Agile development methods, where the implementation evolves continuously, changes the role of architecture.
**Objective** We investigate how architecture evaluation can provide useful feedback during development of continuously evolving systems.
**Method** Starting from the Architecture Tradeoff Analysis Method (ATAM), we performed architecture evaluation, both in a national research project led by an automotive Original Equipment Manufacturer (OEM), and at the OEM, in the context of continuous development. This allows us to include the experience of several architects from different organizations over several years. Using data produced during the evaluations we perform a post-hoc analysis to derive initial findings. We then validate and refine these findings through a series of focus groups with architects and industry experts.
**Findings** We propose principles of continuous evaluation and evolution of architecture, and based on these discuss a roadmap for future research.
**Conclusion** In iterative development settings, the needs are different from what typical architecture evaluation methods provide. Our principles show the importance of dedicated feedback-loops for continuous evolution of systems and their architecture.


## 1. Introduction

In traditional and stage-gated systems development, a software architecture is a high-level abstraction of the system that guides its implementation. A software architecture provides structure and balances different, potentially conflicting, stakeholder concerns. What the architecture should address is informed by high-level product requirements and business goals. Current software-intensive systems are, however, increasingly expected to evolve continuously. This drives the automotive domain towards more exploratory ways of working (Hohl et al., 2017; Stupperich and Schneider, 2011; Ågren et al., 2019). Furthermore, current, predictive ways of requirements engineering are difficult to combine with open-ended work (Ågren et al., 2019). With a more dynamic approach to requirements, the requirements that form the basis for a system architecture will be changing more frequently. Thus, for systems intended to continuously evolve, there is also a need for more continuous approaches to software architecture. Woods describes the need for architecture as a stream of decisions, provided as they are needed (Woods, 2019). For large-scale software development, architecture has also been described as an enabler of agility (Nord et al., 2014). This point of view is also supported by the hypothesis from practitioners, that "when developing large and complex systems a clear and well-defined architecture facilitates and enables agility" (Pelliccione et al., 2017, p. 6). Therefore, it is important that the architecture is of "good quality". For instance, it is important to answer the question: *Do architectural design decisions of an (envisioned) system fulfill the stakeholder concerns and satisfy the business goals?*

We should also consider that an architecture description easily becomes obsolete, an effect which is increasingly evident when architectural erosion and architectural drift start to be visible,

☆ Editor: Neil Ernst.
* Corresponding author.
*E-mail addresses:* magnus.agren@chalmers.se (S.M. Ågren), eric.knauss@gu.se (E. Knauss), rogardt.heldal@hvl.no (R. Heldal), patrizio.pelliccione@gssi.it (P. Pelliccione).
1 Present affiliation: GE Additive.
2 Present affiliation: CMON Consulting.
3 Present affiliation: Malama AB.

https://doi.org/10.1016/j.jss.2021.111111

and in general when the "quality" of an architecture description decreases (Pelliccione et al., 2017; Wohlrab et al., 2019). This opens a different understanding of quality, which is related to technical debt (Cunningham, 1992; Kruchten et al., 2012) and aims to answer questions like: *Does the implemented system conform to its architecture description?* A third understanding of quality is related to the architecture description, rather than to the architecture, and it aims to answer questions like: *Is this architecture description complete?* (ISO/IEC, 2011) intending that all the important aspects, according to the stakeholders' concerns, are properly described.

It is then important to properly specify the intended notion of quality and to make use of a systematic approach to continuously reason about the quality of an architecture. For this, we turn to existing evaluation methods. Architecture evaluation, as described by Knodel and Naab (Knodel and Naab, 2016), has "[the mission] to determine the quality of the (envisioned) software system and the quality of the (auxiliary) artifacts created during architecting or derived from the architecture". However, existing methods are limited in their support for continuous architectural evaluation (Buchgeher and Weinreich, 2014). Our interest here is not the evaluation per se, but, since we believe that the architecture cannot be defined upfront once for all, in how to make an argument that the architecture is good at a certain point in time during the system development and evolution. We take into account all three interpretations of quality discussed above: does the architecture fulfill its stakeholders goals, do architecture and implementation conform, is the architecture complete.

We investigate how architecture evaluation can provide useful feedback during development of continuously evolving systems. We operationalize this as the following research questions:

**RQ1:** How can architectural evaluation in a continuous setting provide timely feedback on whether a specific capability is supported by the current architecture?

**RQ2:** What information should be provided to support architectural evaluation in a continuous setting?

**RQ3:** How in principle can existing frameworks for architectural evaluations be made fit for continuous evaluation?

The motivation for the analysis comes directly from an Original Equipment Manufacturer (OEM) in the automotive domain, which also brings the architecture, processes, and the context in which the evaluation should be performed. The investigation was performed in the context of a national project led by the OEM and involving many suppliers. Among the project aims was deriving principles for continuous reasoning about the quality of architectural decisions and the quality of architecture in continuous system development. To this end, the project setting had the following properties of interest: (i) a concrete and real architecture and product, together with defined processes and organizational constraints, (ii) the freedom to investigate the problem in a financially independent research project with its own management and time-schedule, and (iii) access to experts and consultant selected by the OEM and the other project partners.

*Paper outline.* The rest of this paper is structured as follows. Section 2 gives background on architecture evaluation. Section 3 describes related work and anticipates our findings. Section 4 describes our method, including the studied case. Section 5 describes the individual principles we derive, and Section 6 answers our research questions by describing the interplay of the principles. Section 7 then discusses the implications of our findings. Lastly, Section 8 concludes the paper.

## 2. Architecture evaluation

This section covers related topics from literature, to provide a background of concepts utilized throughout the paper.

Architecture evaluation is the activity of evaluating the architectural design decisions of an (envisioned) system to build confidence that the system can fulfill the stakeholder concerns. Typically, evaluation techniques actually evaluate what is documented in an architecture description. Using definitions from the ISO/IEC/IEEE 42010 standard (ISO/IEC, 2011), *stakeholders* are individuals, groups, or organizations holding concerns for the system of interest. A *concern* is defined as any interest in the system. The term derives from the phrase "separation of concerns" as originally coined by Edsger Dijkstra. Examples of concerns include (system) purpose, functionality, structure, behavior, cost, supportability, safety, and interoperability.

Most widely used architecture evaluation methods are scenario-based (Babar, 2014). Scenario-based methods express quality attributes (for example flexibility and maintainability) as scenarios, describing the response of the studied system to a certain stimulus. For a large-scale system, the architecture typically needs to balance many, potentially conflicting, quality attributes. The quality attributes themselves are, however, rarely concrete enough to be directly used for evaluation. For example, aiming to evaluate whether a system is safe as a plain yes/no question offers little guidance on how to actually concretize safety. *Quality attribute scenarios* are here an approach to make quality attributes concrete (Kazman et al., 2012). A quality attribute scenario consists of six parts:

**source** an entity that generates a stimulus

**stimulus** a condition that affects the system

**artifact** the part of the system that was stimulated by the stimulus

**environment** the condition under which the stimulus occurred

**response** the activity that results because of the stimulus

**measure** the measure by which the system's response will be evaluated

The following example scenario, for the quality attribute maintainability, draws on the work done in our case project:

**source** An automotive development organization

**stimulus** has new software ready for deployment.

**artifact** For affected product variants vehicles

**environment** that are stationary and connected

**response** the update is deployed and confirmed

**measure** to 90% of the fleet within 6 h

### 2.1. Architecture tradeoff analysis method

The Architecture Tradeoff Analysis Method (ATAM) was developed by Kazman et al. (2000). As the name implies, the method identifies tradeoffs, between quality attributes, resulting from the evaluated architecture decisions. An evaluation in ATAM takes several inputs, that can be grouped into two separate streams: business and architecture. The business stream consists of the *business drivers* of the system to be evaluated (e.g., high

availability or time to market or high security), the *quality attributes* needed to fulfill the business drivers (e.g., performance, availability, security, modifiability, usability), and *quality attribute scenarios* concretizing the quality attributes, covering what the system is intended to achieve. The architecture stream consists of an *architectural plan*, *architectural approaches*, styles, or patterns used that contribute to achieving the quality attributes, and *architectural decisions* that cover how the system will achieve its goals. From the two input streams, the evaluation produces a number of outputs; notably *tradeoffs*, but also *sensitivity points*, *risks*, and *non-risks* of the architecture decisions. A non-risk denotes a good decision that rely on assumptions frequently implicit in the architecture. A sensitivity point denotes a property critical for achieving a particular quality attribute; for example, security may be sensitive to encryption strength (Kazman et al., 2000, p. 22). A tradeoff is when an architectural decision affects a sensitivity point for more than one quality attribute. Continuing the security example (Kazman et al., 2000, p. 23), changing the encryption strength might increase security, but at the cost of performance. Prioritization between quality attributes depends on the business drivers of the system, and the quality attribute scenarios thus links business goals to the architecture of the system.

The activities comprising ATAM can be broadly grouped into four phases, numbered 0–3.

**Phase 0 – Presentation** deals with preparation before the evaluation itself starts. During this phase, all stakeholders to be involved in the evaluation are brought on-board. This phase proceeds informally, and thus has no set duration.

In **Phase 1 – Investigation and Analysis**, an initial evaluation is done, procedurally consisting of six steps. First, (1) the evaluation method of ATAM itself is presented, along with (2) the business drivers, and (3) the intended system architecture. Architectural approaches (4) are then identified, but not analyzed. Then, the quality attributes (5) of the system are elicited, specified down to the level of scenarios, annotated with stimuli and responses, and prioritized. A given quality attribute may optionally consist of several sub-attributes. Eventually, at the most detailed level, a quality attribute is expressed as one or more scenarios. The number of levels is not prescribed by the process; the focus is on reaching the concrete form of scenarios. As the sixth step (6) architectural approaches are analyzed against the quality attribute scenarios identified in step 5. Phase 1 is typically run over 1–2 days.

Before **Phase 2 – Testing** can begin, all information that is necessary for the evaluation needs to be extracted. Therefore, there is, typically, a period of a few weeks between phases 1 and 2, when the outcome of Phase 1 is extended and refined. In Phase 2, the main evaluation is carried out, in two additional steps. First, (7) based on the scenarios from step 5, additional scenarios are generated through brainstorming with the entire group of stakeholders. To keep the scope of the evaluation manageable, in discussion with stakeholders the scenarios are then prioritized, and those deemed most important are kept. In step 8, step 6 is reiterated; therefore, architectural approaches are evaluated, but mostly taking into account highly ranked scenarios from step 7. Running Phase 2 typically takes 2 days.

Finally, in **Phase 3 – Reporting** (9) the evaluation results, along with any proposed mitigation strategies, are presented to the assembled stakeholders and documented in a report.

Although architecture evaluation is well established, we note that existing approaches typically target architecture as a finished output of an early project phase (Bashroush et al., 2004; Babar and Capilla, 2008; Salger, 2009; Zalewski and Kijas, 2013; Choi and Yeom, 2002; Barber et al., 2003; Jeong and Kim, 2006; Li et al., 2007; Eloranta and Koskimies, 2010; Ovaska et al., 2010; Scheerer et al., 2017). Many of these works also rely on the availability of architectural models. The finished artifacts that describe the architecture can thus be evaluated before a project proceeds to implementation and further phases. In contrast, we are interested in architecture evaluation during development of continuously evolving systems, where also the architecture needs to evolve throughout development.

## 3. Related work

This section relates our findings to previous and parallel research.

Knodel and Naab (2016) provide an approach to architecture evaluation that goes beyond what ATAM covers, also evaluating whether business drivers are agreed on, the quality of the architecture documentation, consistency between architecture and implementation, and the quality of the implementation (source code). While Knodel and Naab emphasize the usefulness of early and regular architecture evaluation, they do not, however, provide specific instructions for how to integrate architecture evaluation in a continuous workflow. They integrate ATAM as the way to check if the architectural approaches at hand are adequate to address the business drivers. Some changes are made to textbook ATAM, notably not requiring the participation of all stakeholders for the entire evaluation. This complements our finding to not evaluate all decisions in one sitting. They also give a wide range of examples of mistakes frequently made in practice during architecture evaluation. A number of these correspond to our findings. *Losing the good atmosphere due to the evaluation* compares to the most negative reactions we experienced during our evaluation efforts. Securing buy-in remains important also in a continuous setting. *Losing overview over the number of drivers and decisions* matches the problems we faced of concluding evaluations within the available time. *Lack of a clear goal for an evaluation* corresponds to the need we identify to evaluate on demand when a tradeoff is needed. Having a clear goal is important for architecture evaluation in general. For continuously evolving systems, what is particular is the need to be able to trigger an evaluation on demand.

Galster and Avgeriou (2014) propose an approach to achieve variability-handling architecture, in which architecture evaluation has a prominent role. To make the approach work in an agile setting, they divide it into two phases, executed at different times of a product life cycle. The first phase is executed once, creating an initial architecture. Here, effort-intensive methods such as scenario-based evaluation can be used. The second phase, which is intended to be run continuously (e.g. once per sprint) uses a more lightweight checklist-based evaluation. Although their approach specifically targets variability, the suggested division of efforts could be one mitigation for the problems we experienced, of completing evaluations within the available time.

Buchgeher and Weinreich (2014) cover architecture analysis, both in general and for continuous settings. They observe that most approaches have been developed for plan-driven processes, where a finished architecture is available for analysis at a specific point during the process. Similar to our setting, they note that in agile processes architecture will need to be evaluated incrementally, as the architecture is continuously evolving and thus inherently incomplete. In line with our findings to limit the evaluation scope, they suggest that evaluation methods should be able to focus on specific parts of an architecture. They suggest that increased automation would facilitate integration of architecture evaluation with continuous ways of working; while also observing that evaluation focusing on semantics rather than structure cannot easily be automated.

Erder and Pureur (2016) consider ATAM "appropriate for conducting an in-depth architecture review at a critical phase of a

**Table 1**
Project partners, providing staff with specific competencies as needed by the project. For each partner, a few persons served as core participants.

| Partner Id | Roles of core participants | Expertise provided to the project |
|---|---|---|
| | Original Equipment Manufacturer (OEM), *project lead* | |
| P1 | Senior architects | Architecture and vehicle-level development competence. |
| | Suppliers: electronics and software | |
| P2 | CEO, Architects | Adaptive AUTOSAR and Continuous integration |
| P3 | CTO, Technical expert | Networking technologies (CAN, TCP/IP). |
| P4 | Researcher, Architect | Safety-certified components. |
| | Suppliers: technology consultants | |
| P5 | Security consultant | Security |
| P6 | Architect, Developer | Continuous integration, Testing |
| P7 | Business consultants, Senior architects | Architecture, Agile methods |
| P8 | Safety experts | Safety cases, Architecture |
| P9 | Business strategist, Architects | Architecture, Continuous integration |
| P10 | Technology consultant | Continuous integration |
| | Universities | |
| P11 | Researchers | Architecture, Empirical research, Requirements, System-of-systems |
| P12 | Researchers | Networking technology, System-of-systems |
| | Research institutes | |
| P13 | Safety experts | Safety cases, Architecture |
| P14 | Researchers | System-of-systems, ATAM |
| P15 | Researchers | Continuous integration |
| P13 | Researchers | System-of-systems, Test equipment |

project", but note that the time and resources required may prevent ATAM from being used repetitively. For periodic evaluations, they recommend using more lightweight approaches instead. Furthermore, for continuous settings, they suggest supplementing qualitative evaluation techniques with automated testing.

## 4. Method

Our research approach was to conduct a case study (Runeson and Höst, 2009). A case study is a suitable method for studying a phenomenon in its natural context, especially when the phenomenon is difficult to study in isolation. For this study, we had two such settings: (1) a research project with industry collaboration, where architecture evaluation would be one task; (2) an automotive company interested in architecture evaluation. The case study method is appropriate since in both settings it is difficult to clearly delimit the architecture evaluation work from other activities in the context, and from the context itself. The settings are different in the type of observations that can be made and allow triangulation if carefully considering their differences.

### 4.1. Overall study design

In our study, we had the opportunity to learn about continuous architectural evaluation as participant observers. The struggle to make architectural evaluation work was observed in two settings: within a research project and within the architectural teams of company partners, in particular P1, of the same research project (see the next section on Case Description for details).

To derive our principles, we started with a post hoc analysis of the data resulting from our case. Two of the authors separately went through the data and wrote down their reflections. We then unified these reflections to a set of principles, which served as interview guide for a series of focus group meetings. The focus group participants were architects from project partner P1, who brought their experience from introducing architecture evaluation in the company setting. Through the focus group series, we then refined the principles to their current form, where industry experts and academic researchers agreed that they covered the most important lessons learnt in a small set of distinct principles.

### 4.2. Case description

Next Generation Electrical Architecture (NGEA)[4] was an automotive research project running from January 2015 until May 2019. The project was a joint research effort between automotive companies, universities, and research institutes. Table 1 lists the project partners, including their roles and areas of expertise. The project was led by an OEM (partner P1) in the process of building a new electrical and software architecture for its vehicles, and experiencing an increase in the amount of software and the adoption of agile ways of working (Pelliccione et al., 2017). This study has been conducted in the context of both the NGEA project and at P1. In total ten (mostly international) companies, two universities, and four research institutes participated.

The project was of strategic importance for the participating industry partners, and each company provided experienced experts, including lead architects and decision makers. For each partner, as shown in Table 1, a few persons served as core participants, with additional specialists attending depending on the topic of each activity. In particular, four different lead architects from P1 participated in the project, often at the same time. A consultant at P14 with significant ATAM experience provided training, but at the time, key persons from several partners had built up a good level of expertise and the training turned out to be mostly confirmatory. During the validation round in the focus groups, we had access to those participants at P1 that were responsible for architectural evaluation at this large automotive manufacturer.

The project was part of an overall goal of building the next generation of electrical software architecture for the OEM. The focus within this project was to create new knowledge, as well as experiment with new instruments and processes. The project investigated future needs from automotive electrical architectures and, specifically, the following three areas: the future architecture of a car in the form of topologies and support software, ways of working and architecture to support continuous integration, and the car as a constituent in a system-of-systems. The project thus explored continuous development rather than the concept being

[4] Project details at https://www.vinnova.se/en/p/next-generation-electrical-architecture/ and https://www.vinnova.se/en/p/next-generation-electrical-architecture-step-2/

**Table 2**
Project Workgroups.

| Workgroup | Topic |
|---|---|
| WGA | Evaluation of the project outcomes |
| WGB | Architecture topologies |
| WGC | Cars as constituents in a system-of-systems |
| WGD | Adaptive AUTOSAR |
| WGE | Transparency between collaborating organizations |
| WGF | Patterns and strategies for continuous integration |

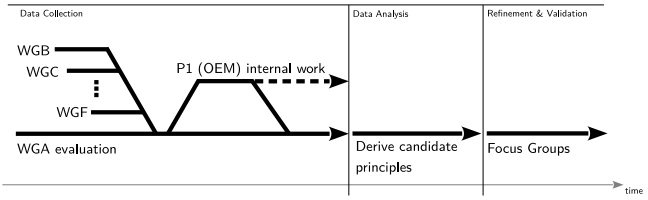


**Fig. 1.** Case Timeline.

defined beforehand. Van Der Valk et al. (2018) also studied continuous development in this setting. The project was organized in workgroups, as shown in Table 2. As mentioned above, depending on the workgroup topic, and the specific session withing each workgroup, the project partners sent participants with topic expertise.

In order to make this new architectural knowledge as actionable and useful as possible, the project aims included reasoning about the quality of architectural decisions and the quality of architecture in continuous system development. Consequently, one of the workgroups (WGA) had the task to evaluate proposals from the other workgroups and analyze tradeoffs between these. We highlight that the rest of the workgroups were not defined for the purpose of conducting the evaluation; they were defined for the purpose of the project, which, as explained above, also had additional goals. The workgroups were to identify architectural approaches for their respective topics, and disseminate these to the project partners together with a tradeoff analysis. For this reason, workgroups WGB to WGF received guidance throughout the project, from WGA, on how to perform the evaluations.

### 4.3. Data collection

Each partner including the OEM selected experts and/or consultants for performing the evaluation activities of WGA. Based on the tradeoffs, WGA would derive recommendations for future electrical architecture. For each topic, we thus wanted to understand its impact on the architecture of an entire automotive system. Architecture evaluation based on quality attribute scenarios was chosen as the evaluation instrument. We decided to use ATAM as the starting point for how to organize the evaluation. ATAM appeared to us to be the de facto standard architecture evaluation method, and, since we wanted to identify tradeoffs between different business goals, the outcomes of ATAM were a good fit. We were aware that ATAM would be heavy weight for use in a continuous setting, but we had the ambition to adapt it for more light-weight use. Also, connecting project outcomes to business drivers was valued highly, and there was hope that the strictness of ATAM would help shape and get the evaluation going. To allow this paper to be read on its own, without consulting the report defining ATAM (Kazman et al., 2000), Section 2 gives a summary of the method.

The evaluation efforts spanned a range of activities in the project. Table 3 gives an overview, describing for each activity the phases of ATAM it maps to, the activity itself, and the resulting data available to us for analysis. At the level of the entire project, WGA performed preparations, row WGA–WS in Table 3, corresponding to Phase 0 of ATAM. An initial training session on ATAM (row PROJ-Training) was given to participants from all the project workgroups. Business drivers, to be used during the evaluation, were also elicited (row PROJ-Drivers). Fig. 1 shows a timeline of the case, giving an overview of how the project activities relating to the evaluation map to the *Data Collection*, *Data Analysis* and *Refinement & Validation* stages of this study.

All the workgroups knew there would be a tradeoff evaluation towards the end of the project; however, the reports created were not made with an ATAM evaluation in mind. Using ATAM as a way to organize the evaluation was a later decision. Since the respective workgroups had the expertise on each topic, WGA could not alone prepare the output from the workgroups for an ATAM evaluation. The workgroups were therefore tasked with running a first evaluation of their work. This would include the creation of artifacts necessary for the evaluation, such as quality attribute scenarios. Each workgroup held evaluation workshops, rows WG–WS–B to WG–WS–E in Table 3. One of these workshops, WG–WS–BC, also served as training, where a consultant at P14 with ATAM experience participated, as mentioned above. Which ATAM phases these workshops could cover depended on how early during the project a workgroup started its evaluation, and how the evaluation task was perceived. The perception differed considerably across the workgroups and the individual participants. Some were enthusiastic, mainly in workgroups WGD, WGE, and WGF. Most were positive, albeit taking a wait-and-see stance, mainly reacting to efforts from the project lead through WGA. Some were negative to the efforts, actively challenging the evaluation approach. All groups carried out Phase 1, but not all continued with Phase 2. With the necessary materials available, WGA would then facilitate continuous evaluation at the project level, to find tradeoffs between the ongoing work of the workgroups. We participated in this work, the outcome forms part of the data. In Fig. 1, this is shown to the left in the *Data Collection* stage.

Partway through the evaluation work, P1 (the project lead partner) continued internal architecture evaluation experiments in parallel with the project work. This is shown in the middle of Fig. 1, as part of the *Data Collection* stage. These experiences were then fed back to the project, both to WGF, informing the demonstrator implementation of tool support for architecture evaluation, and as part of the project final report. In Table 3, this corresponds to rows Demonstrator, P1-Eval, and PROJ-Report. Our units of analysis are thus both the evaluation work done in the project and the work done internally at partner P1.

Throughout the project, we gradually adjusted the evaluation process. This was both in response to problems encountered, such as the challenge of securing simultaneous participation of stakeholders, the difficulty of concluding the evaluation sessions in the allotted time, or opposition to the whole approach. The work items from this process are part of the data used in this study. Table 3 covers these on rows Database, PROJ-Process, and NGEA-ATAM.

### 4.4. Data analysis

Throughout the project, all evaluation workshops conducted by the project workgroups were attended by at least one of the authors. We thus actively participated in the evaluation process. Additionally, our role was to bring in state-of-the-art research on architecture evaluation.

Each activity listed in Table 3 created data for this study. The data itself is diverse. It consists of the evaluation reports created during the project, but also of work items created during the

**Table 3**
Overview of activities related to the evaluation, both in NGEA and at P1.

| Label | Process Phase | Activity, data available for analysis listed in parenthesis |
|---|---|---|
| | | *Project level, doing ATAM* |
| PROJ-Prep | Phase 0 | Preparation in WGA for evaluation, including the choice of ATAM. Discussion of the need for architecture evaluation, decision to use ATAM (Slides, discussion notes, decision) |
| WGA–WS | Phase 0 | Workshops in WGA to plan evaluation, WGA formally asks other workgroups to prepare input to ATAM process (Slides, emails) |
| PROJ-Training | Phase 0 | Seminar on ATAM, given to the full project consortium. (Slides, discussions, documentation) |
| PROJ-Drivers | Phase 0 | Workshop in WGA to elicit business drivers. (Slides) |
| P1-Drivers | Phase 0 | Workshop with P1 giving input to the project on P1 business drivers. (Slides) |
| | | *WG level, doing ATAM* |
| WG–WS–B | Phase 0 and Phase 1 | WGB workshops (Slides, Scenario format spreadsheet) |
| WG–WS–BC | Phase 1 | Evaluation with ATAM expert from P14 on topologies and system-of-systems. (Evaluation report) |
| WG–WS–D | Phase 1 and Phase 2 | Two evaluation sessions in WGD on Adaptive AUTOSAR. (Workshop notes, evaluation report) |
| WG–WS–E | Phase 1 and Phase 2 | One evaluation session in WGE on Transparency. (Workshop notes, evaluation report) |
| Demonstrator | Support activity | WGF creating a tool implementation to support evaluation. (Proof-of-concept demonstrator) |
| | | *P1 internal work, doing ATAM* |
| P1-Eval | Phase 1 | Internal work with architecture evaluation at P1. We visited internal sessions as observers; the work items themselves are confidential, however. (Personal communication) |
| | | *Project level, adjusting ATAM to NGEA project needs* |
| Database | Support activity | Work towards a database of reusable assets. This covered business drivers and scenarios, and links between then. Served as starting input for the demonstrator. |
| PROJ-Process | Support activity | Continuous work in WGA to define and adapt the evaluation process to NGEA needs. (Internal document) |
| PROJ-Share | Support activity | Dissemination of adjustment ideas at trade-fairs (Slides, notes) |
| | | *Project level, validation, reflections, and dissemination to company partners* |
| FG | Reflections after ATAM efforts | Focus groups about the candidate principles. |
| PROJ-Report | Reflections after ATAM efforts | WGA report writing. Bringing up and discussing candidate principles. |

evaluation efforts, such as whiteboard sketches, notes, and slide decks. Additionally, materials (emails, reports, and slide decks) created when defining the evaluation, as well as training materials, are also included. We also visited internal evaluation sessions at P1-Eval as observers. Furthermore, one of the authors kept experience notes throughout the project. Fig. 2 shows four data examples: (a) process chart from the process definition activity PROJ-Process, (b) scenarios created by the workgroups, combined in the template we defined, (c) notes taken during WG–WS–D, and (d) whiteboard sketch from WGA–WS where we discussed how to organize the evaluation.

To derive our principles, we first did a post-hoc analysis of the data. Two of the authors separately went through the full data and wrote down their reflections. We then unified these reflections to a set of principles, constituting a first draft of the principles we present in Section 5. The *Data Analysis* stage of Fig. 1 shows this derivation of candidate principles, leading up to the focus group meetings.

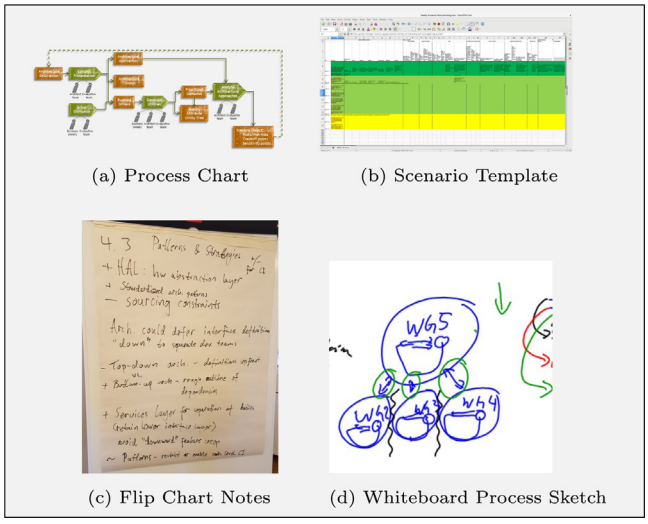


**Fig. 2.** Example Data.

### 4.5. Refinement & validation

As the project was closing, participants of WGA wrote a final report (PROJ-Report) on the evaluation efforts. Two of the focus groups (FG), where we reflected on the evaluation, were held during this writing period. Drafts of the principles that structure our study findings were also included in the report, and served as interview guide for the focus group meetings. Throughout the focus group series we refined the principles to their current form. In total, four sessions were held, by which time consensus had been reached about the set of principles; that the most important lessons learnt was covered in a small set of distinct principles. The focus group participants were senior architects from project partner P1. They brought their experiences from introducing architecture evaluation at P1, which included taking lessons learnt in the research project to the industrial context (P1-Eval). To the right in Fig. 1, we show the *Refinement & Validation* stage of the study.

The data from the evaluation efforts are similar to archive data, in that they were created for another purpose, (conducting the evaluation) rather than for this study (analysis of the evaluation). Although the diversity of the data creates a rich account of events to analyze, it does not remove this limitation. To handle this, we

use experience notes kept by one of the authors throughout the project, and we refine our initial analysis through focus groups. The focus groups serve as a different kind of data collection, one done for the specific purpose of this study. This allows triangulation between different kinds of data sources.

All authors participated actively in the evaluation efforts throughout the NGEA project. Furthermore, the industrial coauthors were, at the time of the evaluation work at P1 (P1–Eval), participating as employees. The interpretation of the data has also been extensively discussed among the authors, to achieve observer triangulation.

In line with our rationale for conducting a case study – to study a phenomenon in its natural context – our findings are inseparably tied to the studied case. By the study design, this realism of context comes at the expense of the possibility for generalization (Stol and Fitzgerald, 2018). Having two units of analysis allows comparing and contrasting the two. However, the possibility for generalization is inherently limited. Since we do not use statistical sampling, statistical generalization is not possible. Analytical generalization to similar contexts may be possible but would require separate validation and is thus a question for future studies.

## 5. Findings

We express our findings as principles for adapting architecture evaluation to continuous settings, summarized in Table 4. These principles are the reaction to difficulties we encountered both in the project and at company partners, and which are complex and not easily decomposed. In a continuous setting, the effort for evaluation must be significantly reduced, so that it can repeated. It is, however, unclear where effort can be reduced and how repeated evaluations can contribute to combined, overall, knowledge. In our experience, the problem has four major parts:

1. Without a clear and agreed upon demand for information, it is impossible to give a clear answer. A change in the demand for information will trigger a reevaluation in the continuous setting. A question that is too broad, too generic, or not made explicit will slow down evaluation to an extent that makes continuous evaluation unfeasible.
2. Without a way of decomposing the evaluation scope, it is impossible to evaluate continuously at the desired level of abstraction. However, limiting the scope does not guarantee that the evaluation will allow continuous learning. Completing the full scope could still be required.
3. Without a way to evaluate the impact of architectural decisions on the top-level product scope, the evaluation does not contribute to the body of product level architectural knowledge. In a continuous development setting, the body of architecture decisions will grow over time.
4. Without finding a constructive approach to architectural evaluation, technical stakeholders will feel that they are assessed personally. Otherwise, there is no clear value to technical stakeholders and therefore no buy-in. Without buy-in of technical stakeholders, continuous evaluation will not work.

In the presence of the previous problems, this makes it unlikely that architects throughout the product organization are keen on pro-actively help with providing knowledge about the value offered through architectural decisions.

In the following subsections, we describe each principle in a narrative style, and link back to particular project activities. For each principle, we first elaborate on the problem we faced; then, to connect to our data analysis, we describe observations from specific project activities informing our reflection; and, last, we describe how the principle helps mitigate the problem.

**Table 4**
Principles of continuous architecture evaluation.

| | |
|---|---|
| CAP I | Evaluate decisions on demand, in response to a clear stakeholder question |
| CAP II | Evaluate architectural decisions incrementally to manage evaluation scope |
| CAP III | Evaluate in the context of the full integrated product to support incremental architecture evaluation |
| CAP IV | Apply concepts of evaluation constructively to articulate the rationale for an architectural decision |

### 5.1. CAP I – evaluate decisions on demand, in response to a clear stakeholder question

#### 5.1.1. What is the problem?

In NGEA, the buy-in on the evaluation varied. Most participants were supportive of the general idea to perform evaluation, but the proposed approach was not universally accepted as beneficial. ATAM implicitly assumes that all stakeholders accept that there is value in the answer to the question of whether the architecture is fit for purpose. For any architectural evaluation, one has to decide for an appropriate level of detail, and whether continuous evaluation is needed or not. Since the project decided to aim for a rather low level of technical detail in a continuous, incremental way, the generic questions implied by ATAM were not suitable. In particular, they did not resonate well with the project, since no complete architecture was aimed for in each iteration of the evaluation. While this may be the result of the project setup, lack of buy-in of important stakeholders is a major risk for any architectural evaluation, and even at the company partners, we notice that the question of suitable abstraction level and continuous evaluation is important. In the absence of a complete evaluation on high-level in one sitting, the absence of an explicit, clear, and agreed upon demand (i.e. a clear question) creates a lack of involvement among stakeholders.

*Derivation of the principle from observations made during the study. (Table 3).* The activities PROJ-Prep and WGA–WS, informed by classical ATAM, asked for complete evaluation. WGB, in WG–WS–B, was first to try to fulfill their part of this demand, by preparing material for Phase 1 of ATAM from their material on architecture topologies. Not without struggle, ATAM steps 1 through 5 were attempted in the workgroup, going from business drivers via quality attributes to quality attribute scenarios. Without a clear demand for an answer to a stakeholder question, scenario creation proved challenging, particularly defining the response and response measure parts. The difficulties prompted an activity PROJ-Drivers, to elicit and refine business drivers on the project level, with the hope of clarifying the overall goal for the architecture to address. This did help clarify the project business drivers; for example, the business driver *the architecture should handle multiple brands and markets* impacts the quality attribute Flexibility. However, buy-in on the evaluation remained unaffected, and stakeholder questions also remained unclear and difficult to articulate, and thus hard to answer.

At P1 (the company internal setting), the evaluation efforts eventually received buy-in from most stakeholders. The way it was presented – wordings such as evaluation and assessment – needed work before senior architects bought in; we cover this further in CAP IV. One notable exception remained: middle-management responsible for human resource allocation; in other words, the decision of how much time that could be put on architecture evaluation. We thus see two reasons for the lack of buy-in, the lack of a clear stakeholder question, which the evaluation should answer, and the perception that the approach was too heavyweight. The latter concern we address further under CAP II.

#### 5.1.2. How does CAP I help?

For a continuous setting, this principle addresses *when* to evaluate *what*. An evaluation that is to be performed throughout a continuous way of working may well be ignored if there is no demand for the evaluation results. If it is unclear whether the proposed approach will address the task at hand, evaluation may be seen as waste. This can cause a lack of feedback and thus a lack of direction for the evolution of the product. Demand for evaluation could come directly from the stakeholders, wanting to know how an architecture supports their concerns. Then, it is also clear when the demand is fulfilled — only clear questions can receive a clear answer. When it is easy to see the value, it is easier to make a case for investing in architectural evaluation. Another possible source for demand is that a tradeoff between different concerns needs to be made at the implementation level. In the Demonstrator activity, we prototyped tool support for detecting evaluation needs from implementation changes (Fig. 3 shows screenshots). The prototype tool-chain had two parts: a database of scenarios, and architecture artifacts. The database allowed tracking how scenarios related to each other – for example if satisfying one scenario would constrain another scenario – and tracking which business driver(s) that motivated a particular scenario. The architecture artifacts then linked between scenarios and affected parts of the architecture. For the prototype, graph models were used as architecture artifacts. Accumulating a repository of evaluation artifacts gradually during development also facilitates reuse (see also CAP II).

Although the tooling was not deployed beyond prototyping, we see a need for tool support for continuous evaluation to be feasible. Evaluating architecture impact of implementation choices is a means of avoiding different balancings of stakeholder concerns throughout the product. CAP III covers this topic further.

In a continuous setting, large and one-time evaluation efforts are infeasible (Buchgeher and Weinreich, 2014). Rather than for complete evaluation, the demand will be for answering particular stakeholder questions. Specific demands occur as the implementation is underway, rather than in one big batch where all are initially known. The evaluation needs to answer the particular questions at hand, and match an overall iterative way of working. While quality attribute scenarios cover partial aspects in technical detail, in the NGEA project we were lacking a clear stakeholder question for the evaluation to address. In response to this, we propose evaluating decisions on demand as a way to divide the evaluation over iterations, and scope evaluation efforts.

### 5.2. CAP II – evaluate architectural decisions incrementally to manage evaluation scope

#### 5.2.1. What is the problem?

To provide valuable feedback through architectural evaluation in a continuous setting, a sufficient level of technical detail must be achieved. For evaluation at the level of technical detail of our case, an entire automotive architecture is too much to cover in one session. Our experiences are in line with claims from literature (Buchgeher and Weinreich, 2014; Galster and Avgeriou, 2014) that scenario-based approaches could be too heavyweight for a continuous setting. Particularly, it may be infeasible to run scenario-based methods often on all scenarios and decisions. If, in addition, the question is unclear, as prompted CAP I, it is difficult to provide a concise answer; thus, evaluation will take even more effort.

*Derivation of the principle from observations made during the study. (Table 3).* Finalizing the evaluation sessions proved difficult both in NGEA and at P1. A particular difficulty was covering the intended scope of the sessions in the allotted time; for example, managing to go through chosen scenarios for a chosen approach and identify any tradeoffs. At P1, in line with Step 7 of ATAM, the evaluation sessions aimed to cover a large number of scenarios, elicited from stakeholders throughout the company. Prioritizing the scenarios and concluding Step 8 of the evaluation process was not reached within the available time. Thus, neither in NGEA nor at P1, the resulting pruning of the utility tree was reached.

Throughout the NGEA project, in the PROJ-Process activity, we refined the evaluation process. Various techniques were tried, to keep the evaluation scope manageable: We restricted both the technical decisions and the scenarios for evaluation. For example, in both sessions of WG–WS–D, the evaluation was restricted to only one decision: the choice between AUTOSAR Adaptive or AUTOSAR Classic for one ECU. For the latter session, the evaluation was limited to only one scenario. Reaching a conclusive outcome proved difficult nevertheless.

#### 5.2.2. How does CAP II help?

For evaluation to be run frequently, the approach needs to be lightweight. If we remove the ambition to evaluate all scenarios and all decisions every time, we potentially gain an evaluation approach that is feasible to run continuously. This would allow for frequent runs, within available resource bounds. By running it more frequently, a positive side-effect with respect to CAP I might occur, since, in a continuous setting, concrete needs for evaluating specific questions arise throughout development, thus providing clear questions. The need for evaluation exists for the entire duration of product evolution, but not of the same decision, or all decisions, each time. Incremental, partial, evaluation can work in a continuous setting.

We see two complementary approaches: (a) decompose the overall evaluation by demand, as suggested in CAP I, or (b) decompose by architectural decisions and their areas, then relying on automation to allow evaluation of all decisions to scale. Tool support such as our prototype (Fig. 3) can allow us to collate an overview of decisions taken throughout implementation. As the screenshots show, scenarios are stored in a suitable tool and information model, allowing for updating them as well as relating them to new decisions and incremental evaluation results.

### 5.3. CAP III – evaluate in the context of the full integrated product to support incremental architecture evaluation

#### 5.3.1. What is the problem?

Contrary to what we had anticipated, the quality attribute scenarios did not ultimately help clarify the architecture impact of the studied technical topics. Using scenarios did however help reveal a deeper aspect of the problem: the absence of a surrounding context, as part of which the solutions from the workgroups would address something. Without a way to evaluate architectural decisions with respect to the on the top-level product scope, the evaluation does not contribute to the body of product level architectural knowledge. Internal stakeholders tend to evaluate local architectural concepts with regard to local concerns. This is not because of ill intent or disinterest, but we experienced that at scale, establishing the link to global, system-wide concepts was very difficult to do.

*Derivation of the principle from observations made during the study. (Table 3).* In addition to the difficulty of finishing evaluations in the allotted time, we also struggled to reach conclusive outcomes of the evaluation sessions. Phase 1 of ATAM, where artifacts need to be articulated, seemed promising for reaching descriptions of the architectural impact. In particular, we hoped that the quality attribute scenarios would allow concretizing what was addressed at the level of each workgroup topic. In the context of evaluation, workgroups WGB–WGF, who addressed different technical topics, were tasked with describing the architectural

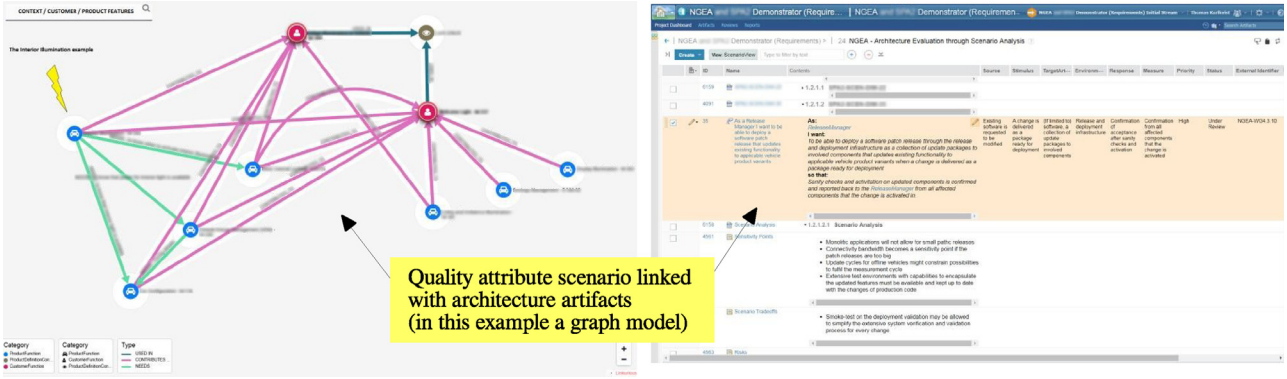


**Fig. 3.** Screenshots of the tool prototype from the Demonstrator activity: Quality attribute scenarios are stored in a database. Artifacts that describe the architecture are linked to affected scenarios, which can be leveraged for an overview during architecture evaluation. For confidentiality, text in the images has been blurred.

impact of their respective work. WGA would then evaluate trade-offs between them. The difficulty of finalizing evaluations at the workgroup level, WG–WS–B through WG–WS–E, meant that an overall evaluation at the project level never came to fruition. Relating the output of the workgroups to an overall architecture proved challenging.

#### 5.3.2. How does CAP III help?

For an architect to be able to provide feedback to developers on the architectural impact of a decision, it needs to be possible to evaluate the decision in the context of the full integrated product, to relate the impact in a part of the system to the entire product. Within the architecture evaluation of the NGEA project, relating the output of the separate workgroups to an overall architecture proved challenging. First, we tried to modify the scenario template, by drawing on the user story format familiar in continuous settings:

As `<stakeholder>` I want `<concern>` so that `<rationale>`

The stakeholder corresponds to the source in quality attribute scenarios; the concern corresponds to stimulus, artifact, and environment; and the rationale corresponds to response and measure. For example, *As sourcing manager, I want to change the supplier of a component, so that from the date the component is available, new cars can be produced with the new component within a week.*

The format change helped to facilitate communication and dialog. However, it did not help solve the deeper aspect of the problem; the lack of an artifact corresponding to a full integrated product, making the architectural impact of the studied topics elusive. Yet, improved communication certainly did not hurt.

Our focus groups confirm the need of relating implementation decisions back to the architecture level of the product. The impression is that decisions with architectural impact are taken day-to-day during implementation. It is, however, currently still difficult to identify the decisions that have this impact. This also makes it challenging for an architect to provide feedback to the developers.

ATAM emphasizes that the quality of the evaluation depends on the quality of the architecture description. The architecture description determines what can be evaluated; it needs to be possible to test the scenarios at hand against it. For continuous evaluation, this suggests continuously working on a purposeful architecture description. It is unclear if this is feasible for scenario-based approaches. One hypothetical possibility, raised from the P1–Eval and Demonstrator activities, is to reuse artifacts beyond the evaluation. For example, quality attribute scenarios can be used to express how a solution for a product is to be defined, linking how the stakeholders expect a solution to be constructed with why a particular construction is desirable. In a product verification context, quality attribute scenarios could be used to define the product quality assurance test cases. Thus, scenarios can add additional value beyond the architecture evaluation.

### 5.4. CAP IV – apply concepts of evaluation constructively to articulate the rationale for an architectural decision

#### 5.4.1. What is the problem?

In both settings of our case study – the NGEA project and the company setting at P1 – we encountered difficulties to get commitment of all important stakeholders. A reason for this is that stakeholders were reacting negatively to words such as architectural *evaluation* or *assessment method*. Project partners started to worry about whether this would affect how their work would be evaluated as part of the project. Architects felt that the sudden need for assessing their work conflicted with the trust that was put into them by installing them into this position.

While understandable, none of these fears matched the intention when it was decided to introduce architecture evaluation. In NGEA, the goal was merely to reason about when a certain architectural concept could be applied. In the company setting, we noticed a growing gap between developers and architects (Eliasson et al., 2015) and we hoped that a better line of argumentation of why a certain architectural decision was made would provide value.

With key stakeholders perceiving the idea of architecture evaluation so negatively, the original goals of architecture evaluation in both NGEA and at P1 were endangered. Stakeholders would stall, request and wait for additional input, not provide data, or stop to attend meetings. In this situation, a huge investment was already done to initiate Phase 0 and Phase 1 of our ATAM initiatives, yet, failure to achieve an evaluation appeared imminent.

*Derivation of the principle from observations made during the study. (Table 3).* In Phase 0, PROJ-Prep, WGA–WS, and PROJ-Training activities attempted to get everybody on board. In particular, the activities aimed to initiate an effective evaluation within the scope of the project agreement. Project partners sensed that the quality of their contributions, and thus indirectly themselves were evaluated and proceeded only carefully. Reports from WG–WS–B through WG–WS–E surfaced despite these challenges. WG–WS–D and WG–WS–E went furthest, to Phase 2 of the evaluation. Comparing these to the activities not proceeding that far, we found that workgroups WGD and WGE were more successful in using ATAM terminology to describe their knowledge constructively, as opposed to a more defensive stance, where groups challenged the evaluation approach, rather than reason about the architectural knowledge they provided.

*5.4.2. How does CAP IV help?*

With CAP IV, we aim to go back to our initial reasons for choosing ATAM, and for conducting architecture evaluation. In the NGEA setting, this was mainly to allow us to qualify any outcomes concerning new technologies and their architectural implications and to investigate how architectural decisions could affect the ability to do continuous integration or deployment.

At P1, in the separate company-internal efforts, the aim was to increase the impact of architecture on continuous development. Developers perceived the architecture as not helpful or even outdated. We therefore aimed to improve two aspects: Firstly, we aimed to provide better rationales and arguments on why certain architectural decisions were made. We hoped that this would address any misperceptions among developers. Secondly, we aimed to establish a feedback cycle towards continuous architecting efforts to ensure that architectural decisions indeed are of high quality and usefulness for coordinating development at scale. CAP IV thus aims to make use of the common language, terminology, and structure that ATAM provides, but not primarily for a group of assessors to investigate an architectural decision, but for the authors of this decision or concept, to allow them to make their argument and to present it in a useful way.

It is not our intention through CAP IV to replace architectural evaluation preformed by a wide range of stakeholders, but to complement it.

## 6. On the interplay of the principles

In this section, we answer our research questions and narrate the interplay and cross-cutting implications of our proposed principles, with examples of feedback on architecture quality in a continuous setting.

### *6.1. Timely feedback on whether a specific capability is supported by the current architecture (RQ1)*

ATAM does suggest to evaluate architectural decisions against quality scenarios, which in turn are derived from quality attributes and business drivers. In our context, we derive business drivers from business goals. A driver is then a condition, capability, or constraint that relates to the potential of reaching a business goal (in ATAM usually seen as system functionality, goals, constraints, and desired non-functional properties).

In a continuous setting, we must assume that business goals evolve and, therefore, also the view on business drivers. For example, the business goal to support shared mobility may raise privacy concerns. A business driver in this context could be the *ability to authenticate the vehicle user without a physical key* and the *ability to divide the usage information into separate access domains.*

If the need for such business drivers emerges, management might approach the architecture group with the demand of checking whether such capabilities are supported by the current architecture or, if not, how hard it would be to create them. In line with our CAP I, this is an opportunity to be embraced: here is a clear question that, with a well-established continuous architecting environment, should be possible to answer efficiently. With the right infrastructure in place, this answer can be given within hours rather than days. It demonstrates value to management (they can decide whether to commit), and provide important rationale to development. Further, this partial answer can provide input on similar answers in the future.

In this scenario, the ability to evaluate architectural decisions incrementally (CAP II) is crucial. Ideally, the architects take into account only those architectural decisions and scenarios, that are relevant for answering the question to management. If a new business driver is given, quality scenarios may have to be adjusted or newly created as well as aligned with the existing set of ATAM-related information items (see Fig. 3 for an example).

Of course it is important that the answer to management is evaluated in the context of the full integrated product, taking into account the latest, incremental information about the overall architecture, its decisions, and relationship to other business drivers (CAP III). We argue that at a realistic scale, this demands appropriate tool and database support to integrate partial, incremental information (CAP II).

Such tooling benefits in our experience from applying evaluation concepts constructively (CAP IV), which suggests a common language shared among all stakeholders and based on ATAM to describe the findings with respect to the business question. This also allows the use of a strong information model on how database items relate to each other. The prototype in Fig. 3 gives an example of first steps towards such an infrastructure. Additional automation promises to further quicken the feedback loops.

### *6.2. Clarity about which information to provide to support architectural evaluation in a continuous setting (RQ2)*

In the NGEA setting, the discussion of ATAM and preparations during Phase 0 quickly lead the project consortium to refine the quality criteria for project reports. Each project report was expected to answer how its content would affect electrical architecture or vice versa, how architecture could affect the content. For example, a report on different network topologies made an effort to discuss how each topology would affect related architectural decisions. A report on continuous integration proposed quality attribute scenarios that should be used when deciding about architectural tradeoffs.

We found the quality attribute scenarios to provide a way to constructively articulate the value an architectural decision provides. This allows us to create a solid argument about the value of an architectural decision and to "sell" it to the organization. We believe that these are strong indications on how constructive use of ATAM terminology and concepts helped to better describe project outcomes, thus providing an example of concrete practices to implement CAP IV.

By relying on ATAM terminology constructively (CAP IV), architectural knowledge is provided in a language that stakeholders find increasingly familiar. This benefit is facilitated by a strong information model and support to integrate this architectural knowledge on the system level (CAP III). As a consequence, even small increments in architectural evaluation can be described in a useful way, which enables CAP II. In addition, reusable knowledge can be stored in a way that allows us to answer questions on demand, thus facilitating CAP I.

### *6.3. Making architectural evaluation frameworks fit for continuous evaluation (RQ3)*

Over the course of our case, we worked on modifying the evaluation process (activity PROJ-Process), with the aim of better supporting our goal of feedback in a continuous setting. Specifically, our recommendations for architecture evaluation in continuous settings are:

- Before running an instance of architecture evaluation, bring everyone on board (one of the purposes of Phase 0 in ATAM). We however recommend to revisit this onboarding and at the same time to limit the scope of 'everyone'. CAP I asks for commitment and CAP II limits the scope.

- Before running an instance of architecture evaluation, make sure that all stakeholders understand and agree on the ratio of effort to the expected value of the evaluation. Our CAPs help to do this in a continuous setting: CAP I demands for a sponsor, CAP II limits the scope and effort needs, CAP IV suggests that even with a small effort value can be created since architectural knowledge can be packaged in a more useful way.
- To support continuous architectural evaluation, agree about the level of abstraction on the architecture to evaluate. CAP II asks for this kind of scoping, CAP III complements this, since still global value must be demonstrated.
- The people with the power to change the architecture should be part of the evaluation. CAP I asks for this kind of commitment.
- Be realistic about how much can be achieved in one meeting. CAP II suggests relevant scoping, CAP III helps with determining the value to be provided through the evaluation meeting, and CAP IV may allow preparing the meeting to significantly speed up the process.
- Have a strong facilitator, to keep meetings on track and to avoid lengthy discussions out of scope. CAP I and CAP II allow focusing on the value that evaluation should support.

## 7. Implications for research

Continuous approaches to software engineering rely on short feedback cycles. Small changes – deltas – are integrated frequently, which creates a closed feedback cycle on the level where integration happens. If that is the level of a whole system, the system-level impact of a change in one part or subsystem can be assessed. While there exist recommendations to do architecture evaluation early and often (Knodel and Naab, 2016), it is not clear which parts can be done frequently and continuously. Recent proposals to automate architectural evaluation focus on syntactical, structural aspects of the architecture, and the match between architecture and implementation (Buchgeher and Weinreich, 2014). When trying to semantically connect architectural decisions to business drivers and stakeholders, there is a trend towards scenario- and meeting-based evaluation. However, the continuous aspect of the evaluation is not addressed in such scenario-based methods and we experienced a lack of conceptual support. In particular, common process descriptions of ATAM focus on a single run. While we hoped to run evaluation multiple times with small adjustments, we discovered difficulties to do so. Our CAPs capture the core of adjustments needed in our experience. Yet, they leave questions for future research that must be investigated to make continuous architecting and architecture evaluation more focused and fast. In the following, we report the main open questions we identified.

*How to balance the tradeoff between too high-level of abstraction and too low-level of abstraction in architectural evaluation?* On the one extreme, one could opt only for infrequent evaluations on very high-level, removing the need for dedicated support for continuous evaluation. On the other extreme, one could choose very low-level evaluations, which might be easier to automate. Both approaches may provide value in a continuous setting, especially if they are combined based on an approach on a middle level, in which frequent meeting- and scenario-based evaluations are incrementally combined into a growing base of architectural knowledge. However, what we had in NGEA and wanted to understand the architectural impact of were specific technically detailed topics. As a system is evolving, decisions on the implementation level need to be related back to the architecture level, to avoid unwittingly impacting the architecture in uncontrolled ways. For evaluation approaches that in this way balance the level of abstraction, we believe our CAPs to provide guidance, yet further future research questions arise.

*How to balance scope and frequency of meetings in continuous scenario-based architecture evaluation?* We believe that the ambition in ATAM to invite all important stakeholders to all meetings precludes frequently repeated evaluations. Our CAPs suggest to evaluate on demand (setting the frequency), in evaluation meetings that are scoped based on particular scenarios or decisions (setting the scope), yet incrementally brought together on system level. Open questions remain, however, regarding the frequency versus effort associated with such meetings.

*How to improve feedback (speed and quality) through smart automation and tooling?* Current ways of, for example, continuous integration achieves feedback largely through automation. The integration, notably including testing, runs with little to no manual intervention. Our findings point to a need for similar short, frequent, feedback on architecture. With evolving systems, architecting the entirety in advance becomes infeasible. New needs will emerge, similar to how we wanted to understand architecture impact of specific topics. For architecture, however, it is unclear how to provide suitable support through automation and tooling, as, for example, unit testing frameworks do for software testing and integration. Future research could investigate whether for example (i) automatic evaluation of architectural debt, (ii) automated decision impact analysis techniques, (iii) visualization tools, or (iv) analysis tools of quality attributes (for example timing or security) are feasible and useful in this context.

*How to make the right information available to an agile team so they can take responsibility for the architectural impact of their work?* We believe that suitable architecture evaluation in continuous development can increase the quality of architectural decisions as well as the way they are communicated with important stakeholders. In this way, additional value can be provided to agile software teams, that can align on carefully evaluated system-wide tradeoffs instead of local over-optimization. Our CAPs target this usefulness to teams directly.

Yet, open questions remain on which concrete practices can be introduced to implement these principles of continuous architecture evaluation. How should architectural work be included in backlogs? How can architectural debt be made visible? How to balance long-term strategy and short-term concerns? How to facilitate feedback from different development teams or disciplines on architectural decisions? In an agile environment, that encourages autonomy of teams with broad competencies, ways for sharing architecture responsibility with the teams, while ensuring quality of the architecture from a systems perspective, are needed.

## 8. Concluding remarks

This paper aimed at investigating how to perform architecture evaluation in continuous development and, specifically, how architecture evaluation may provide valuable feedback during development of continuously evolving systems. The study is based on our experience of performing architecture evaluation in a research project in the automotive domain, and also at the company leading the project. We report our findings in terms of four principles for adapting architecture evaluation to continuous settings (Section 5), as well as their interplay and cross-cutting implications (Section 6). We also provide a discussion of the findings and highlight some open questions that frame interesting future research directions (Section 7).

## CRediT authorship contribution statement

**S. Magnus Ågren:** Conceptualization, Methodology, Investigation, Writing – original draft. **Eric Knauss:** Conceptualization, Methodology, Investigation, Writing – original draft. **Rogardt Heldal:** Conceptualization, Methodology, Investigation, Writing – original draft. **Patrizio Pelliccione:** Conceptualization, Methodology, Investigation, Writing – original draft. **Anders Alminger:** Conceptualization, Investigation, Validation. **Magnus Antonsson:** Conceptualization, Investigation, Validation. **Thomas Karlkvist:** Conceptualization, Investigation, Validation, Software. **Anders Lindeborg:** Conceptualization, Investigation, Validation.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This study was funded through the Vinnova project Next Generation Electrical Architecture (NGEA). We thank all participants in the architecture evaluation efforts in NGEA for great collaboration and project outcomes despite the difficulties we shared as described in this paper. The authors also acknowledge financial support from Centre of Excellence on Connected, Geo-Localized and Cybersecure Vehicle (EX-Emerge), funded by Italian Government under CIPE resolution n. 70/2017 (Aug. 7, 2017). Lastly, we thank Colin Venters for sharing his insights on carrying out an ATAM evaluation.

## References

Ågren, S Magnus, Knauss, Eric, Heldal, Rogardt, Pelliccione, Patrizio, Malmqvist, Gösta, Bodén, Jonas, 2019. The impact of requirements on systems development speed: a multiple-case study in automotive. Requir. Eng. 24 (3), 315–340.

Babar, Muhammad Ali, 2014. Making software architecture and agile approaches work together: Foundations and approaches. In: Agile Software Architecture. Elsevier, pp. 1–22.

Babar, Muhammad Ali, Capilla, Rafael, 2008. Capturing and using quality attributes knowledge in software architecture evaluation process. In: 2008 First International Workshop on Managing Requirements Knowledge. IEEE, pp. 53–62.

Barber, K. Suzanne, Graser, Tom, Holt, Jim, Baker, Geoff, 2003. Arcade: early dynamic property evaluation of requirements using partitioned software architecture models. Requir. Eng. 8 (4), 222–235.

Bashroush, Rabih, Spence, Ivor T.A., Kilpatrick, Peter, Brown, T. John, 2004. Towards an automated evaluation process for software architectures. In: Hamza, M.H. (Ed.), IASTED International Conference on Software Engineering, Part of the 22nd Multi-Conference on Applied Informatics. Innsbruck, Austria, February 17–19, 2004, IASTED/ACTA Press, pp. 54–58.

Buchgeher, Georg, Weinreich, Rainer, 2014. Continuous software architecture analysis. In: Agile Software Architecture. Elsevier, pp. 161–188.

Choi, Heeseok, Yeom, Keunhyuk, 2002. An approach to software architecture evaluation with the 4+1 view model of architecture. In: Ninth Asia-Pacific Software Engineering Conference, 2002. IEEE, pp. 286–293.

Cunningham, Ward, 1992. The WyCash portfolio management system. SIGPLAN OOPS Mess. 4 (2), 29–30.

Eliasson, Ulf, Heldal, Rogardt, Pelliccione, Patrizio, Lantz, Jonn, 2015. Architecting in the automotive domain: Descriptive vs prescriptive architecture. In: 2015 12th Working IEEE/IFIP Conference on Software Architecture. IEEE, pp. 115–118.

Eloranta, Veli-Pekka, Koskimies, Kai, 2010. Using domain knowledge to boost software architecture evaluation. In: European Conference on Software Architecture. Springer, pp. 319–326.

Erder, Murat, Pureur, Pierre, 2016. Validating the architecture. In: Erder, Murat, Pureur, Pierre (Eds.), Continuous Architecture. Morgan Kaufmann, Boston, pp. 131–159 (Chapter 6).

Galster, Matthias, Avgeriou, Paris, 2014. Supporting variability through agility to achieve adaptable architectures. In: Agile Software Architecture. Elsevier, pp. 139–159.

Hohl, Philipp, Münch, Jürgen, Schneider, Kurt, Stupperich, Michael, 2017. Real-life challenges on agile software product lines in automotive. In: Proc. of Int. Conf. on Product-Focused Software Process Improvement. PROFES, pp. 28–36.

ISO/IEC, 2011. ISO/IEC/IEEE 42010:2011 systems and software engineering – Architecture description.

Jeong, Gu-Beom, Kim, Guk-Boh, 2006. A study on software architecture evaluation. In: International Conference on Computational Science and Its Applications. Springer, pp. 1032–1041.

Kazman, Rick, Gagliardi, Michael, Wood, William, 2012. Scaling up software architecture analysis. J. Syst. Softw. 85 (7), 1511–1519.

Kazman, Rick, Klein, Mark, Clements, Paul, 2000. ATAM: Method For Architecture Evaluation. Technical Report, Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst.

Knodel, Jens, Naab, Matthias, 2016. Pragmatic Evaluation of Software Architectures. Springer.

Kruchten, P., Nord, R.L., Ozkaya, I., 2012. Technical debt: From metaphor to theory and practice. IEEE Softw. 29 (6), 18–21.

Li, Jinhua, Guo, Zhenbo, Zhao, Yun, Zhang, Zhenhua, Pang, Ruijuan, 2007. Towards quantitative evaluation of UML based software architecture. In: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. SNPD 2007, vol. 1, IEEE, pp. 663–669.

Nord, Robert L., Ozkaya, Ipek, Kruchten, Philippe, 2014. Agile in distress: Architecture to the rescue. In: International Conference on Agile Software Development. Springer, pp. 43–57.

Ovaska, Eila, Evesti, Antti, Henttonen, Katja, Palviainen, Marko, Aho, Pekka, 2010. Knowledge based quality-driven architecture design and evaluation. Inf. Softw. Technol. 52 (6), 577–601.

Pelliccione, Patrizio, Knauss, Eric, Heldal, Rogardt, Magnus Ågren, S., Mallozzi, Piergiuseppe, Alminger, Anders, Borgentun, Daniel, 2017. Automotive architecture framework: The experience of volvo cars. J. Syst. Archit. 77 (C), 83–100.

Runeson, Per, Höst, Martin, 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131.

Salger, Frank, 2009. Software architecture evaluation in global software development projects. In: OTM Confederated International Conferences "on the Move to Meaningful Internet Systems". Springer, pp. 391–400.

Scheerer, Max, Busch, Axel, Koziolek, Anne, 2017. Automatic evaluation of complex design decisions in component-based software architectures. In: Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design, pp. 67–76.

Stol, Klaas-Jan, Fitzgerald, Brian, 2018. The ABC of software engineering research. ACM Trans. Softw. Eng. Methodol. 27 (3), 11.

Stupperich, Michael, Schneider, Stefan, 2011. Process-focused lessons learned from a multi-site development project at daimler trucks. In: Proc. of 6th Int. Conf. on Global Software Engineering. ICGSE. Helsinki, Finland, pp. 141–145.

Van Der Valk, Rob, Pelliccione, Patrizio, Lago, Patricia, Heldal, Rogardt, Knauss, Eric, Juul, Jacob, 2018. Transparency and contracts: continuous integration and delivery in the automotive ecosystem. In: 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track. ICSE-SEIP, IEEE, pp. 23–32.

Wohlrab, R., Eliasson, U., Pelliccione, P., Heldal, R., 2019. Improving the consistency and usefulness of architecture descriptions: guidelines for architects. In: 2019 IEEE International Conference on Software Architecture. ICSA, pp. 151–160.

Woods, Eoin, 2019. Democratising software architecture. https://www.infoq.com/news/2019/04/ICSA-2019-Software-Architecture/.

Zalewski, Andrzej, Kijas, Szymon, 2013. Beyond ATAM: Early architecture evaluation method for large-scale distributed systems. J. Syst. Softw. 86 (3), 683–697.