Defining a Method for Identifying Architectural Candidates as Part of Engineering a System Architecture

Joakim Fröberg, Stig Larsson, Sara Dersten
School of Innovation, Design and Engineering
Mälardalen University, Sweden
[joakim.froberg,stig.larsson,sara.dersten]@mdh.se

Per-Åke Nordlander
BAE Systems AB
Örnsköldsvik, Sweden
per-ake.nordlander@baesystems.se

Engineering system architectures for complex systems involves the tasks of analyzing architectural drivers, identifying architectural concerns, identifying valid architecture candidates, and evaluation of alternatives. One problem to overcome when architecting a system is the identification of valid of architectural candidates. We have developed a step-wise method for performing system architecture analysis and tested it on a sub-system in a project developing a drive system for heavy automotive applications. In this paper we present the complete method of nine steps for engineering an architecture and we elaborate in detail on the procedure to identify architectural candidates based on previously identified architectural drivers. We present a diagram depicting the proposed information model, its concepts and their relationships. In addition, the expectations on such a method as expressed by practitioners have been elicited, and we elaborate on the validity by examining how well the method indicate fulfillment. Our conclusion is that the proposed method does not fail to deliver on any of the needs and this gives an indication of usefulness. When identifying architectural candidates it is important to use proper criteria in the process. Our conclusion is that the practitioners should focus on candidates that affect the system at hand (within system boundaries), and on the candidates that address the architecturally significant system use. This is reflected in our method where we prescribe evaluation of the design candidates by validating that they solve only the right problem and by ensuring that they address the system at hand.

Keywords—system architecture; architecture analysis; architecture evaluation; architectural candidate

I. INTRODUCTION

The architecture of a system involves some of the decisions that, more than others, affect the outcome of a development effort in terms of meeting system goals, achieving system effectiveness and overall project success. Engineering the system architecture of a complex system involves the tasks of analyzing architectural drivers, identifying architectural concerns, identifying valid architecture candidates, and evaluation of alternatives.

Systems engineering guidelines provide models and advice for what information entities to consider when engineering the architecture of a system, e.g., architectural concerns, but only limited guidance of how to do it. The guides are limited both in preciseness of definition of the information entities, e.g., what defines an architectural requirement, and also in process description, e.g., how do the information entities relate and what order to proceed through the work tasks. These questions need to be addressed by any development team that faces an architectural analysis in an actual case of engineering a complex system.

One problem to overcome when architecting a system is the formation of suitable architectural candidates. We assume that we have identified architectural drivers for the given system, but what architectural candidates do in fact address those? What architectural candidates do affect the quality of the system in the way that is defined by the architectural drivers? An efficient procedure must produce candidates that are valid and address the architectural needs of the system at hand.

II. LITTERATURE STUDY

"The Method Framework for Engineering System Architectures", MFESA [1], describes tailoring methods for engineering a system architecture for a specific development endeavor. Based on the context of the system, a set of steps, and work products can be instantiated to form a specialized method tailored for the particular case. The MFESA framework gives a complete view of the field of engineering a system architecture and divide the field into areas covering ten different types of tasks, T1-T10. T2 covers how to identify architectural drivers and both T3 and T5 cover how to come up with candidate architectural solutions. The MFESA task descriptions are general and needs to be interpreted and translated into a precise workable structure in order to be instantiated.

The CAFCR model by Muller [2] describe a wide range of aspects of the system architecting process including advice for understanding customer objectives and application. The model proposes the use of stories and use cases for utilization in a context of system architecting.

The Architecture tradeoff analysis method, ATAM [3], provides a usable method to elicit usage scenarios by using a technique of utility trees. You start with a stakeholder expression of a utility, e.g., maintainability, and then break it down into scenarios that are prioritized. These act as statements against which the quality goals of the system will be judged. In our proposed method, we model use-cases and divide them into life-cycle processes.

The Quality Attribute Workshop, QAW [4] provides a procedure to identify important quality attributes for software architectures. The procedure can be used for architectures of general systems as well and utilizes scenarios to express system usage, and provides a stepwise process for refining scenarios.

Analyzing architecture needs is based on understanding the system use and this can be captured and modeled in different ways. Riedemann and Freitag [6] describe an overview of how to utilize techniques for modeling system usage. Alexander and Maiden [7] describe a classification of scenarios used for system development and describe stories and use cases as two types of scenarios. Cockburn [8] describe a full guide on how and when to use a use case effectively. We have chosen to utilize use-cases to model system use.

III. DEFINING A METHOD FOR ENGINEERING AN ARCHITECTURE

This section presents the considerations of forming the method. We describe what is done in the study, what practitioners expect from the method, the information model that we chose, and how we interpreted MFESA procedures and how we defined central terms and concepts.

A. What is done in this study?

We have previously defined a method to elicit and define architectural drivers [9], and described the necessary steps to identify also the architectural risks and opportunities. In this paper we add the steps necessary to define architectural candidates based on the architectural drivers, and identify methods for the evaluation of the different options. This means that we can present a complete method of nine steps for engineering architectures. The development of this method was made through

- Definition of the criteria for what practitioners in our case company perceive as a practical method for analyzing system architecture. This was done using interviews with practitioners.
- Instantiation of a workable method by tailoring the MFESA tasks that are applicable to the case. This includes defining process steps, inputs, and outputs for tasks two, T2, three, T3, and five, T5 described in the MFESA framework.
- Interpretation of the meaning of the steps described in MFESA and inclusion of necessary additions and changes to produce a coherent method.
- Discussions regarding the method validity and usefulness with practitioners.

In this paper, the focus is on the describing the details of the method steps used for the identification of architectural candidates. In addition, of the preceding and later steps needed are explained to allow a complete description of the procedure.

B. Practitioners' expectations on an analysis method

The engineers that have participated in this study use a company specific systems engineering guide similar to the INCOSE systems engineering handbook [4]. The engineers express the need to support early decision-making among architectural alternatives. A useful method would need to plug in with the existing development process without making fundamental changes to the process. The method must clearly evaluate which candidate is best suited, preferably by comparable measures. The first results are needed early in the development process before all requirements specifications are completed. The method would need to be relatively lightweight in relation to the team-size. A method that is simple and understandable is preferred, and activities should as much as possible provide a divide-and-conquer approach where problems can be individually addressed. The method should not use an information model where many complex relations exist between the used artifacts.

In summary, the practitioners seek these basic method qualities:

- Support evaluation of architecture alternatives
- Impose low footprint
- Cover complete system usage and scope
- Based on quantifiable entities
- Provide analysis results in early phases
- Simplicity and separation of concerns

These method qualities seem fairly general and we have used them throughout the study to guide forming our proposed method.

C. Defining the information model

MFESA provides a generic model for engineering tasks in the realm of system architecting and also covers many areas. We define a method of nine steps that constitute a specialization that is concrete, but for only the tasks related to analyzing architectural drivers, identifying candidates, and evaluation. Our method aims to provide useful interpretations of generic advice. Things that must be addressed by a development team, and thus provide an exact but perhaps less generic engineering advice. In this paper, we describe in detail the steps of identifying architectural candidates.

Instantiating the MFESA tasks produces a list of artifacts and steps to perform, but no full interpretation of what constitutes the artifacts or description of how to perform the steps. We point out three additions that cannot be derived from the MFESA framework and could be useful in other cases. The significant additions and changes made by us were:

1. We interpret and define the concepts proposed by MFESA and define their relationships. We present a UML diagram depicting concepts and their relationships in an information model in Figure 1. A

precise notion of relationships helps to validate the candidate architectural alternatives.

2. We employ use-cases as a means to model and identify architecturally significant requirements. The Quality Attribute Workshop, QAW [5], the ATAM [3], and the CAFCR [2] model propose similar procedures to define and express system usage in scenarios. We choose to start with capturing use-cases and progress by elaborating the architecturally significant ones by defining detailed scenarios.

3. We propose a stepwise procedure for carrying out the work. We present an overview of the process with steps and work products in a sequential procedure.

The first thing that was needed to instantiate MFESA, was to choose how to interpret some of the central concepts. In MFESA, an architectural driver is defined as a significant requirement for the architecture, and an architectural concern is defined a cohesive set of architectural drivers. This is open to many interpretations and a more precise definition is needed to elicit the information from the stakeholders of the real case. We choose to group together architectural drivers that are related to the same area of design space. This often means a choice of concept for a system capability or functionality, e.g., the concept for telematic system communication.

Task 3, T3, aims to capture the most important architectural decisions and also make sure that all architectural concerns are adequately addressed. Task 5, T5, aims to verify that architectural candidates support the relevant architectural concerns. Sets of candidates are formed into competing architectural visions and those are evaluated and the most suitable is selected.

In order to perform these tasks, it needs to be identified what exact architectural concerns do affect the performance of system use and also how different architectural candidates cause different performance in system use. For instance, it must be established which candidates positively or negatively affect which use cases.

We have chosen to form the above information model to concretely define which architecture candidates affect which use-cases. This model is one out of many possible, but we aimed at fulfilling the demands for clarity and simplicity.

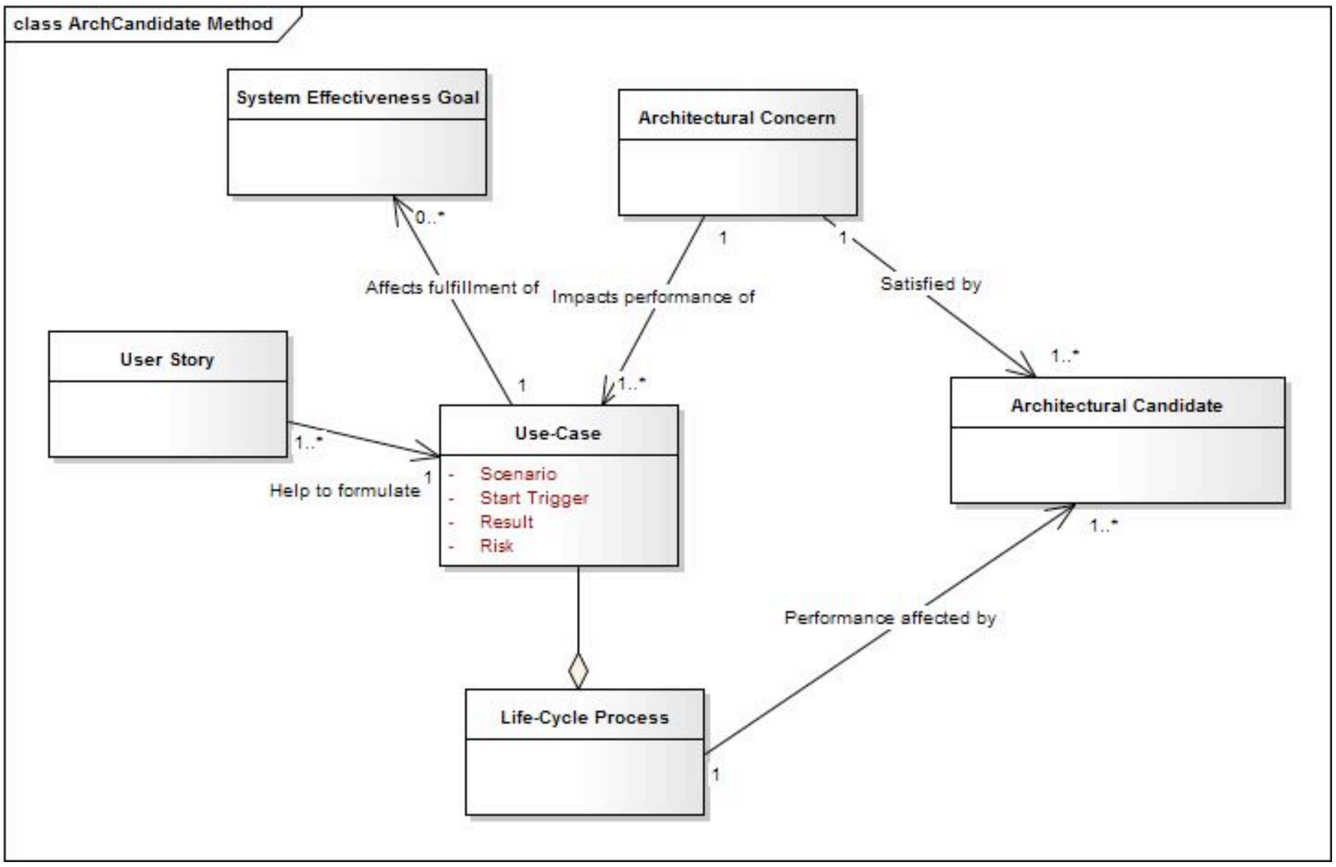

Figure 1. Information model for identification of architectural candidates.

The system use is documented using use-cases and a set of use cases make up a life cycle process. A number of user stories are analyzed and grouped into relevant use-cases. An architectural concern is a set of open design decisions whose choices for solution will affect a number of use-cases. The model stipulates to define relations for the ones that do. For each architectural concern there may be a number of architectural candidates that provides a solution but also gives different effects on some or all of the related use cases. The architectural candidate can be judged by its overall impact on a life-cycle process.

IV. A NINE STEP METHOD FOR ENGINEERING AN ARCHITECTURE

The resulting method to perform system architecture analysis is summarized by the following activities in a stepwise procedure:

1. Model use cases of the system in its complete life cycle. Stakeholder interviews and workshops are used to produce user stories that are used to formulate use-cases describing the system use through the life cycle. Each use-case has a main scenario, a starting point, a result, and possible risks and opportunities. Each life-cycle process is an aggregate of all its use-cases.

2. Remove too risky use-cases. Assign architectural risk and architectural opportunity based on team judgment to use-cases. Estimate severity and probability for each risk/opportunity and filter out too risky use-case. In effect, this step reduces the scope and boundary of what the system is to do.

3. Relate use-cases to architectural concerns. An architectural concern is an area of design space where the choices for solution has impact on the performance of use cases. For instance, a communication concept could be an architectural concern whose choice for a technical solution would affects how well the system performs in use. The architecturally significant use-cases are those that affect system effectiveness measures. The architecture team makes the judgments of which use-cases are associated with which architectural concerns.

4. Assign system level measures of effectiveness to use-cases. It is noted which use-cases do affect system level goals on effectiveness. The system effectiveness goals that are judged important for the system under consideration are associated with the affecting use-cases.

5. (Optional: Assign priority importance weight). The importance of each use-case and each effectiveness measure can be estimated. The team can choose to do this or omit it depending on what level of confidence are held of the estimates.

6. List candidate architectural alternatives for each concern. Alternatives are listed by brainstorming and discussion. Each alternative is validated against all use-cases that are related so that it is addressing the right problem.

7. Estimate value of each candidate. The architecture team estimates a value of the candidate compared to a baseline. The baseline could be a previous system or any agreed upon reference system. The estimate can be a monetary value, e.g., a cost saving (or increase) of a candidate represented by a particular proposal of a new hardware topology. The performance of each life-cycle process should be estimated.

8. Calculate overall indication of candidate suitability. The estimates can be used to calculate sums to give indication of the best-fitted architectural choice. The set of choices (an architectural vision) can be compared to another set.

9. Validate the set of architectural choices. Validate that the set of architecture choices can co-exist and does address all use-cases. Some of the concerns may not be possible to treat separately, but act as cross-concerns.

A. Example – Selecting a telematics architecture for an automotive system

We choose a theoretical example to illustrate the method with no details from the case. The example is chosen to present the method and it illustrates how architectural candidates are found in a simpler way without the level of detail of a real case. The number of use-cases and the level of technical detail of the candidates are greatly simplified.

We describe each step of the method, but focus on the tasks related to the identification of architectural candidates. We assume that architectural drivers have been elicited but describe all steps briefly in order to be coherent. We do not explicitly address the evaluation in this paper. The evaluation is important and will be performed after the identification of candidates.

1. *Model use cases of the system in its complete life cycle.* We assume that the design of the telematics system is judged as important for the performance of the process of maintenance for an automotive drive system. The life-cycle processes were modeled with stakeholders and use cases. There is OEM support personnel, Service personnel, developers, and sales people involved. One out of the many use cases involve that an service engineer gets the vehicle usage data (drive cycle profile, and diagnostic data) that has been logged when the system is used at a customer.

2. *Remove too risky use-cases*. The idea of a self-learning system could be considered. The use cases caused by having such a system could be assigned with risks and opportunities. Evaluation could show that risks for development time are too great and lead to a decision of not having that scope.

3. *Relate use-cases to architectural concerns*. The question of how to design the telematics system is identified as one architectural concern. We go through the use-cases and not which ones whose results would potentially be affected by different solutions. This

concern is decided to be related to the use-case of getting vehicle usage data, as different solutions would yield different functionality and also different response times.

4. *Assign system level measures of effectiveness to use-cases*. The user stories have left us with a number of statements of for all stakeholders on what is effectiveness in the different use-cases. The time needed to get the vehicle usage data is an important measure of effectiveness for the system in order to fulfill the preventive maintenance life-cycle process.

5. *(Optional: Assign priority importance weight)*. Some or all the use cases may be assigned an importance factor, e.g., the value of this particular use-case or life cycle. This may be measured by monetary value or by some other count. In this example we did not elaborate, as it is likely to affect mostly the evaluation.

6. *List candidate architectural alternatives for each concern.* The user stories may involve strong statements of what architectural candidates are preferable. At this point, by using the model, it is possible to go thorough each use-case to see if and by how much it is affected by each of the proposed candidates. This leads to a discussion of refinement of the proposals. It also uncovers candidates that address qualities that are out of scope of the actual system use. Lets say we end up with two architectural candidates. Either we could have usage data extracted by the service technician using a laptop-based tool, or we could have a fitted system that enables run-time queries by the OEM service engineer. (We choose extremes in the interest of space here)

7. *Estimate value of each candidate.* The two candidates affect the result of the "get usage data" use-case and it can be assigned a monetary value of that service. It is compared with the cost of the system and the estimated price of such a service. This could very probably become a large task for a large number of use-cases.

8. *Calculate overall indication of candidate suitability.* Summarizing the value of each candidate is necessary. This could involve a large amount of estimates, but for a given case there could be simpler estimations. We did not elaborate more in this study.

9. *Validate the set of architectural choices.* There may be dependencies between candidates and this would not allow us to evaluate each candidate separately. This can be addressed by grouping candidates into architectural visions like the MFESA propose. The total value of the candidate should dictate the choice.

V. Discussion on using the method

In addition to the theoretical example, we have participated in a development project and were given the task to find and analyze architecture drivers and alternative candidates and to evaluate suitability. This includes analyzing requirements, identify architectural alternatives, and deciding on architectural alternatives. We defined the base for our method by using the MFESA framework and added some method components from other theories. Still, the resulting method is not directly applicable. In order to perform the method, we had to clarify the interpretation of some of the work products and define the relationship between information entities. In addition, we had to specify a stepwise working procedure. Some of the additions could be considered as case-specific tailoring and some may be useful in general.

In order to validate the usefulness of our proposal, we analyze the usefulness criteria as expressed by the practitioners. We focus on the parts of the method that aid to identify architectural candidates.

A. Support evaluation of architecture alternatives

The method does provide the information model to define the relation of each architectural candidate, via the architectural concern, to the system use and to keep track if that is affecting system effectiveness goals. There is thereby a base of information to enable evaluation of the architectural candidates.

B. Impose low footprint

The method was possible to carry out for a sub-system to be evaluated and it was not perceived as being too resource heavy. Achieving a lightweight solution is a relative goal and we are not able to test the demand in any precise way.

C. Cover complete system usage and scope

The procedure of modeling the system use cases and then to group them into life-cycle processes does force the engineers to analyze and decide what is and what is not included in the system scope. Each use-case includes information on how it is run through and what is the valuable output, and also how it starts and ends. The method does provide a way to see what architecture candidates that does affect the system use and effectiveness goals and also allows to model all effects on system use within the system scope.

D. Based on quantifiable entities

The evaluation of architecture candidates is done by measuring and estimating real measures of cost and other measures of effectiveness. It is not based on estimated relative measures of quality fulfillment.

E. Provide analysis results in early phases

There is no hinder to model system usage early based on stakeholder stories without waiting for a complete requirement specification. As the development progress, there will be improved precision in specifications, but for this study we didn't consider how to iteratively improve architecture related methods.

F. Simplicity and separation of concerns

The information model seems simple, but there are no restrictions on how many use-cases connect to how many architectural concerns and system effectiveness measures. Complicated cross dependencies could arise, but for our sub-system the model does provide a simple way to find and evaluate candidates.

VI. Conclusion

In this paper we have presented a method for performing engineering of a system architecture and tested it on a sub-system in a case of developing a heavy automotive drive system. We tailored the method based on the MFESA framework and made necessary changes to suit the specific case. The tailoring includes precise definition of artifacts and selection of used tools such as use-case definitions. We present the method as a stepwise procedure, and provide preliminary analysis of the use.

We analyze the use of the method and conclude that the case context and practitioner needs seem fairly general and could be considered and also useful for any developer of a complex system. We propose a method that can be used as an extension to the MFESA architectural method framework, and we argue that our additions could be useful in other cases as it is not necessarily a specialization that disqualifies any particular system context from using it. We present an information model and a procedure to define our supportive method.

When identifying architectural candidates it is important to do two things - 1. focus the effort on only the system at hand (system boundary) 2. focus on the candidates that actually address the architecturally significant system use. This can be done by asking the domain specialists for candidates and iteratively increase the precision. Our method prescribe to help practitioners focus their design candidates by validating that they solve only the right problem and by focusing it to address the system at hand.

Acknowledgments

This work has been made possible by funding from the Swedish Energy Agency and support from Mälardalen University and the research institute SwedishICT/SICS.

References

[1] Firesmith, D.G., Capell, P., Hammons, C.B., Latimer, D.W., and Merendino, T.: “The Method Framework for Engineering System Architectures,” Taylor & Francis, 2008.

[2] Muller, G., “Systems Architecting: A Business Perspective,” CRC Press, 2011.

[3] Kazman, Rick; Klein, Mark; Clements, Paul. "ATAM: Method for Architecture Evaluation (Cmu/Sei-2000-Tr-004)." 2000.

[4] Barbacci, Mario; Ellison, Robert; Lattanze, Anthony; Stafford, Judith; Weinstock, Charles; Wood, William. "Quality Attribute Workshops (Qaws), Third Edition, Technical Report Cmu/Sei-2003-Tr-016." 2003.

[5] INCOSE, “Systems Engineering Handbook - A Guide for System Life Cycle Processes and Activities,” Version 3.2.2, 2011.

[6] C. Riedemann, R. Freitag, “Modelling Usage: Techniques and Tools,” IEEE Software 26(2). 20-24, 2009.

[7] Alexander, I., and N. Maiden. "Scenarios, Stories, and Use Cases: The Modern Basis for System Development." Computing & Control Engineering Journal 15 (2004).

[8] Cockburn, A., Writing Effective Use Cases. Vol. 1: Addison-Wesley Boston, 2001.

[9] Fröberg, Joakim; Larsson, Stig; and Nordlander, Per-Åke. "A Method for Analyzing Architectural Drivers When Engineering a System Architecture." In SysCon (2013 IEEE International Systems Conference). Orlando, Florida, 2013.

[10] P. Clements, R. Kazman, M. Klein, “Evaluating Software Architectures – Methods and Case Studies,” SEI Series in Software Engineering, Addison-Wesley, 2002.