A method for analyzing architectural drivers when engineering a system architecture

Applied in a case of developing an automotive drive system platform

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

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

Abstract— **A very important task in systems architecting is to understand the needs of the system and identify which ones have architectural ramifications, i.e., architectural drivers. The understanding of architectural drivers enables the later engineering tasks including evaluation of architectural alternatives. Systems engineering guidelines provide models and advice for what information entities to consider, but only limited proposals of how to proceed. In this paper, we device and present a method to perform analysis of architectural drivers and we apply it to an industrial case of developing a hybrid electric drive system for heavy automotive applications. We present data on what practitioners expect from such a method, we present the method and rationale, and preliminary results from applying the method to the case. We note that the process and information model are fairly general and could be considered useful for any developer of a complex system. We believe the proposed method closes some of the gap between the general models described in the system engineering guidelines and an industrially applicable method.**

Keywords—system architecture; architecture analysis; architectural requirements; automotive systems

I. INTRODUCTION

Engineering 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 qualities and overall project success. Inadequate definition of system architecture accounts for a large portion of the rework costs [1] and a flawed architecture is generally considered to be one of the reasons that a system fails to meet its goals [2]. The system architecting process include many of the key elements of the systems engineering process and span from identifying needs and crucial design considerations to performing decision-making among alternatives. A very important task in systems architecting is to understand the needs of the system and identify which ones have architectural ramifications, i.e., architectural drivers. The understanding of architectural drivers enables the later engineering tasks including decision-making. Therefor it becomes very important to have an effective method identify and structure the architectural drivers for a particular system under development.

Systems engineering guidelines [3][4][5] do provide models describing the involvement of artefacts, information entities and concepts, but provide limited guidance of how to proceed and perform the actual activities of architectural driver analysis.

The system engineering guides are limited both in preciseness of definition e.g., what defines an architectural requirement, and in defining the relations between the information entities. The guidance is also limited in process description, e.g., what order to proceed through the work tasks. Knowing what information to search for and how to proceed is central. Such questions need to be considered by any development team that faces an architectural driver analysis in an actual case.

In this paper, we present a method to perform analysis of architectural drivers and we apply it to an industrial case of developing a hybrid electric drive system for heavy automotive applications. The empirical findings of data on what practitioners expect from such a method are presented together with a detailed method description and rationale. Also, we present preliminary results from the case study.

The objective of this study is to define a method for eliciting and defining architectural drivers, that is practically useful in the studied case. We have earlier studied what information is critical in a pre-study phase [6] and the results from that phase are now used as a part of the input for architectural driver analysis. The study includes finding the expectations and criteria for method usefulness by the architecture team involved in the case. The goal of analyzing architectural drivers is to define architectural knowledge enough to drive the later phases of analysis and decision-making. We label all these activities as "analyzing architectural drivers" in this paper and in the title.

The paper is structured as follows. Section II introduces the central literature we have used to propose a method. Section III describe the case that we have studied; the problems faced, and the demands on a useful method as expressed by practitioners. Section IV describes the considerations in forming the method. Section V defines the proposed method and section VI outlines results from using the method and discuss future work. Section VII summarize and conclude the paper.

II. Related work

The method framework for engineering system architectures, MFESA [3], is a framework for 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 activities, as shown in Figure 1. The framework gives an indication of the order in which tasks are typically executed. Task number two, "Identify the architectural drivers", covers guidance for the activities related to eliciting and analyzing the system requirements that have architectural ramifications, i.e., architectural drivers. In this study, we instantiate task 2, and add elements of other theories to provide a workable method in the development project.

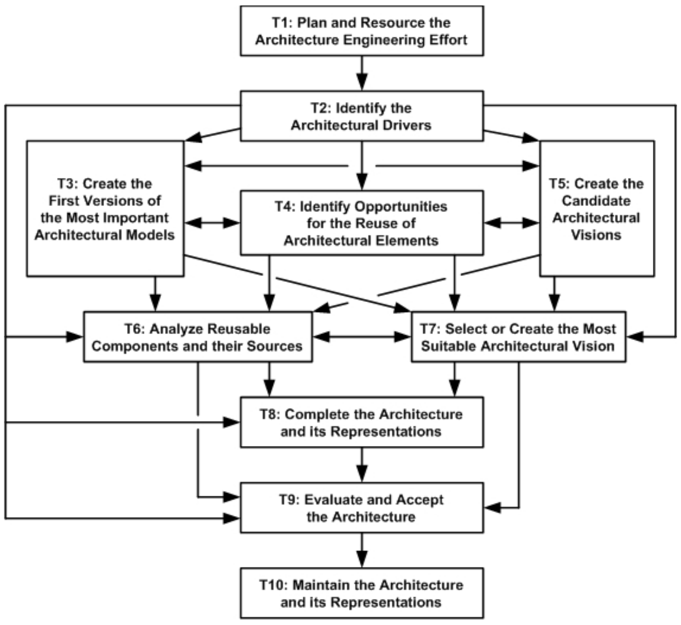
T1: Plan and Resource the Architecture Engineering Effort
T2: Identify the Architectural Drivers
T3: Create the First Versions of the Most Important Architectural Models
T4: Identify Opportunities for the Reuse of Architectural Elements
T5: Create the Candidate Architectural Visions
T6: Analyze Reusable Components and their Sources
T7: Select or Create the Most Suitable Architectural Vision
T8: Complete the Architecture and its Representations
T9: Evaluate and Accept the Architecture
T10: Maintain the Architecture and its Representations

Figure 1. MFESA Architecting Tasks (figure from the MFESA framework)

The CAFCR model by Muller [4] 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 [7], 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.

The Quality Attribute Workshop, QAW [8] provide a procedure to identify important quality attributes for software architectures. The procedure utilizes scenarios to express system usage, and provide a stepwise process for refining scenarios.

Riedemann and Freitag [9] describe an overview of how to utilize techniques for modeling system usage. Alexander and Maiden [10] describe a classification of scenarios used for system development and describe stories and use cases as two types of scenarios. Cockburn [11] describe a full guide on how and when to use a use case effectively.

III. Case description

The company we have studied, BAE Systems, has previously developed a customizable hybrid electric drive system intended for a few similar heavy automotive applications. The goal of the development effort is now to develop customizable hybrid electric drive systems to a large number of products with a wider scope of applications. A hybrid drive system should typically accommodate some of the following functionalities: re- generation of motion energy, optimized performance of motion actuators, optimized combustion engine control, productivity enhancement. The increased ability to control electric components compared to conventional automotive components provide the possibility to develop new functions that improve or optimize performance. The success and quality of such a drive system is heavily dependent on what fuel effectiveness and performance can be achieved for each product in particular applications.

The drive system is intended for use in many automotive applications and involves problems of adaptation and system boundary. The design needs variability and flexibility enough to accommodate vehicles that may have different architectures, system decomposition and design philosophy. Examples of design solutions that may well differ is paradigms for fault handling, diagnostics, and modes of operation. Design of an automotive subsystem will involve these types of complexity.

Developing a drive system platform may be performed at the same time and coordinated with development of several drive systems for specific applications. Platform development is inter-twined with each individual development project and provides and receives information and assets. As the development progress, more automotive applications will be considered.

In summary, the development effort that is studied in this paper can be categorized as a complex mechatronic platform for adaptation to a range of automotive applications.

A. *The problem of engineering a workable method*

When used, the MFESA produces a tailored system architecting method for the specific effort that is being undertaken. This tailored method does provide a model for how to engineer system architectures. It describes which information entities are useful and to some extent how they relate to each other. It says, for instance, that architecturally significant requirements can be classified into architectural concerns, and from there architectural risks can be derived and so on. But, it doesn't say how. In order to achieve an engineering procedure, we would need to decide on an ordered process, and criteria for what to classify into different information entities, and their relations. Looking into the task 2 of the MFESA, we note that definitions of the used information entities and their relationships are lacking. It is left to the user to tailor definitions of information entities, relations, and process. So, for instance, it is not clear to a practitioner what criteria to use to find the architectural risks or architectural concerns, and furthermore it is left to the user to define how these concepts relate to each other in the particular case. This means we can choose to have each architectural risk derived from architectural requirements, concerns or other things, with one to many relationships if we judge useful. These choices affect the procedure greatly.

In summary, the problem that the project group of our case is facing is to define how to perform analysis of architectural drivers in the case. A stepwise process is needed and definitions of work products that fulfill the needs of later stages of analysis and decision-making.

B. Practitioners expectations on method properties

The architecture team involved in the case expresses what properties the envisioned method must exhibit. By depth interviews in the pre-study phase and open discussions with the team members we have listed a number of method properties, MPs, that are desired in order to be considered useful in the case.

- MP1: The method must produce a result that can be used to compare alternative architectural solutions, and evaluate suitability according to known criteria. Later stages of architectural work must be provided with good enough artifacts and information to support decision-making.
- MP2: The project group of engineers wants a relatively lightweight solution. The footprint of the method must be in accordance with the project team size.
- MP3: The method must be simple in the sense that it is explainable and structurally understandable. A method that is difficult to explain would risk being distrusted.
- MP4: It is preferable to the greatest extent possible to achieve a divide and conquer method where stepwise results are achieved that can be individually addressed. The method should not use a model where many complex relations exist between the used artifacts.
- MP5: The method should select abstraction of artifacts so that there is a manageable amount of artifacts.
- MP6: The method should avoid incomplete results that need further elaboration in later stages, or at least a way to clearly define criteria for how good is good enough for the next stage. Preferably also a notion of how complete the result is before advancing to later stages.

IV. Steps and Considerations in Forming the Method

We propose a method for architectural driver analysis. This includes a structure and process for how to proceed with the engineering activities. The goal of the method is to identify and list the important system requirements with system architecture ramifications. The method should exhibit, if possible, the characteristics specified by the method properties, MPs. Here, we describe how we selected each element of the method and explain the choices we made when forming it.

A. Steps in defining the method

We have instantiated and tailored the MFESA guidelines for performing the architectural driver analysis. We did this for our context in the studied project. This work produced a list of steps, and work products that should be used in the process of architectural driver analysis. On top of this we have developed the method by some additions and clarifications.

1. We altered the use of architecturally significant requirements to the use of architecturally significant use-cases as Muller proposes [4].
2. We defined concepts proposed by MFESA to a larger extent and defined their relationships. We present a UML diagram depicting concepts and their relationships.
3. We proposed a stepwise process for carrying out the work. We present an overview of the process with steps and work products in a sequential procedure.

B. Adding a story and a use case

We want the architectural drivers developed as early as possible in the development process. We want a low footprint. We also want them to support later decision-making. The stakeholder stories should be expressed so as to achieve these goals. In order to define the architectural drivers, we can choose to express the stakeholder statements in different ways.

The ATAM produce scenarios that are prioritized. These act as statements against which the quality goals of the system will be judged. In the same way, we want a list of statements whose fulfillment can individually be estimated when considering an architectural solution. The MFESA propose that architecturally significant requirements are specified during the requirements analysis phase and then propagated to a later phase where they should be further elaborated and updated. Considering, the MP1 and MP6 statements of our practitioners, we decided to diverge from this suggestion of specifying needs as exact requirements and then proceed to identify the architecturally significant ones. Instead, we use the idea of stories proposed in section 4.3 in Muller [4]. This would enable judgments of fulfillment in much the same way as the ATAM approach.

The stakeholder statements on what the system need to be able to do needs to be analyzed, elaborated, and eventually expressed as requirements. For the early phase of the architecting process though, this will not suffice and complete set of requirements will be finished too late. We choose to view the statements as user stories. This enables interviewees to express freely their statements and us to quickly interview and document a large number of stories. A requirement is a more formal statement that either will or will not be fulfilled. The system shall follow a standard, shall exhibit a physical property, shall perform a function. Architectural considerations are not aimed at meeting the shall requirements. Architecting needs to weigh together information on system use and determine what architectural alternatives accommodate the system qualities that are most important to the system usage. The idea of architecture analysis is to aid in this difficult task. A use-case or scenario lends itself to expressing a need in a format where the stakeholder need can be described in a less normative way.

We propose a stakeholder requirements workshop and interviews where stakeholders from the complete lifecycle of the platform system get to state their needs of the system. All the statements are to be recorded in a list of stories. Subsequently, we can elaborate on the list and identify the architecturally significant stories. Having identified the architecturally significant ones, we can proceed and develop those into a set of corresponding, properly described, use cases. We choose to refine user stories into use-cases and progress by elaborating the architecturally significant ones by defining detailed scenarios.

The ATAM propose the use of utility trees. On the system architecture, as opposed to the software architecture level, we do not want to restrict the stakeholder elicitation to encompass only quality attributes and therefor we adopt the same fundamental listing of stakeholder statements where fulfillment can be estimated, but we do not adopt the structure of utility trees. Rather, we let the stakeholder tell their story without too much structure, and then elaborate to find the architecturally significant stories and develop them into use-cases that act as the need against which to judge fulfillment.

In later phases of architecting, when choosing between alternative architectural mechanisms, we would rather have a use-case description where an experienced engineer can judge each alternative as supportive of the use-case in a continuous scale. We choose to avoid using full requirements for the purpose of architecting because it would delay the process and they would provide non-optimal support for decision-making.

V. A METHOD FOR ANALYZING ARCHITECTURAL DRIVERS

We use three views to describe the proposed method of analyzing architectural drivers.

- A process description with steps and work products.
- Definitions of the involved concepts.
- A concept relation diagram for the method.

1) *Process description*

The process for analyzing architectural drivers that we propose includes the activities and work products as depicted in Figure 2. Previous work indicates what information is important in phases prior to the phase of analyzing architectural drivers [6]. We assume that the system boundary and the system stakeholders are identified. The MFESA model describe a general set of advice showing process steps together with input and output work products that should be considered when planning an engineering architecture effort.

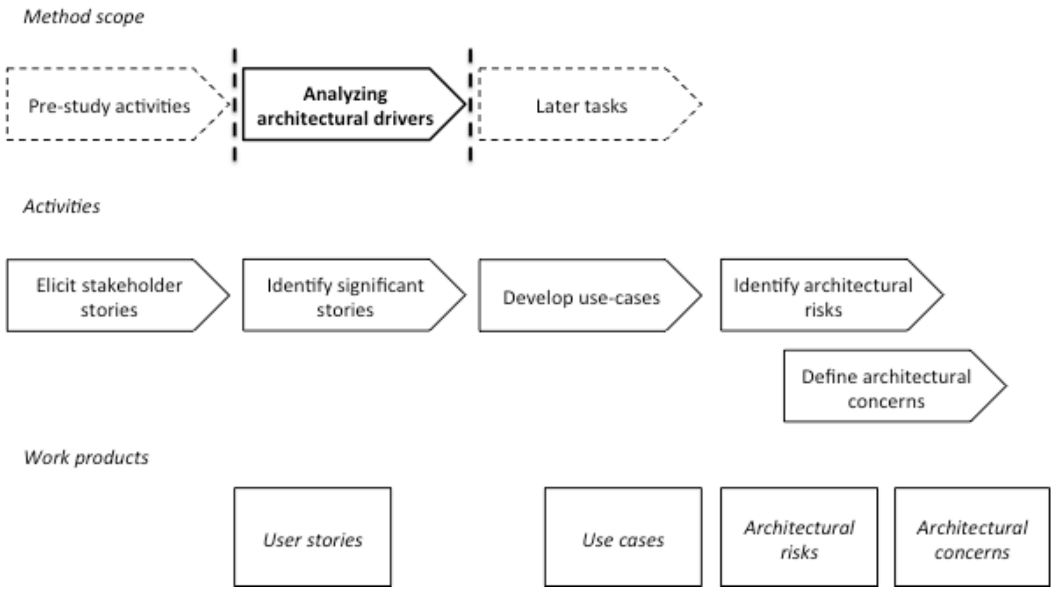

Figure 2. A process for architectural driver analysis.

The method for analysis of architectural drivers, that we propose, start with the activity of eliciting stakeholder user stories. Stakeholders tell user stories as a narrated description of a sequence of events. These user stories can be elicited by interviews or small workshops and should generate a list of stakeholder user stories. Some level of quality control is needed here and the list should be made free of stories that are 1, incomprehensible or 2, technical jargon is central or 3, no sequence of events identifiable, e.g., a simple shall requirement. The next step is to identify those stories that are architecturally significant. We distinguish which ones are architecturally significant based on the same principle as proposed by the MFESA. The architecture team must identify the architecturally significant product or process stories by experienced discussion. The engineers judge what will have architectural ramifications and affect the engineering of the system architecture. Now, there is a list of architecturally significant user stories, and the next step is to develop each into a use-case.

A use case is a description of how the system is used in its context including both functions and quantitative statements on performance and other qualities. Typically a use-case needs elaboration from the simpler style of a story. Each use case should be elaborated until it has a success scenario that includes potential variations, and the expected result. Any system qualities that are involved should hold quantitative statements. This step ends with having a use-case model of the system in its complete life cycle.

For each use-case, the architecture team brainstorms, and identifies architectural risks and opportunities, each with estimates of probability and severity. Those are used to remove too risky use-cases. Assigning architectural risk and architectural opportunity was based on team judgment. In effect, this step can reduce the scope and boundary of what the system is to do.

Based on the use-cases, the architectural concerns should be defined. An architectural concern is in our method, just like in the MFESA, a cohesive set of architectural drivers; in our case the drivers that are the architecturally significant use-cases. Use cases that seem related to the same area of design space are grouped together to form architectural concerns. The architecture team analyzes the use-case and identifies architectural concerns that are defined by a set of related use-cases. The architectural concerns will be used in later phases to identify architectural candidate solutions.

2) *Definitions of the involved concepts*

A method to analyze architectural drivers, require us to identify and structure information on architectural concepts. A model of analysis elements is chosen to represent the relevant information. In order to perform the analysis, we must have workable criteria for how observations from the case fall into the model structure so as to properly describe the case. Here, we describe the definitions and interpretation of the analysis elements that we use for our proposed method. The definitions are adopted from literature, and our interpretation is stated explicitly.

Alexander [10] proposes that a scenario is a common term that entails different specialized formats including, stories, use-cases, and more. Muller [4] propose that user needs can be expressed both as a story and as a use-case and that they can be developed and interchanged to be understandable by different stakeholders and used for different purposes. For the purposes of analyzing architectural drivers, we propose to use the concepts of user stories to capture the needs expressed by the stakeholders, and to develop some of them into more descriptive use cases that better detail the flow and success criteria.

The story and use case are concepts that we add on top of the MFESA proposed method. For the rest of the concepts that we use for analyzing architectural drivers definitions, we use the MFESA definitions.

User story: There are many texts stating different things on what a story is. We use the Wikipedia definition as we find it captures the essence of the most common definitions. Wikipedia defines a user story: “a user story is one or more sentences in the everyday or business language of the user of a system that captures what the user does or needs to do as part of his or her job function”. It seems relevant for describing needs of a system in a development effort.

In our method, we interpret the concept of a user story as a free form description of a stakeholder to explain a story of how their work will be done and results will be achieved. The story seems a natural choice as it enables free sentences on the use expressed in the business language of the stakeholder.

Use-Case: Cockburn [11] defines: “A use case is a description of the possible sequences of interactions between the system under discussion and its external actors, related to a particular goal”. We choose to interpret this in a broad sense as proposed by Muller [4]: “A use-case is a description of how the system is used in its context with a combination of functions and a quantitative description of performance and other qualities. The analysis results are used to explore the design options.” We select this definition because we, like Muller, want the use-case to specify a broad use of the system and to support later phases of exploring design options.

Architecturally significant: A scenario (a story or a use-case) is architecturally significant when a solution likely affect the architecture as judged by the architecture team. The architecture is considered the most important structure and principles of the system with high impact on the development effort outcome.

Architectural Concern: An architectural concern is an aggregate of related architecturally significant use cases. It identifies an area of the design space that all the use cases relate to.

The architecture team identifies architectural concerns by grouping use cases into cohesive sets and categorizes them according to the MFESA: Quality concerns, Constraint concerns, Functional concerns, Process concerns, Interface concerns, and Data concerns.

We let the architecture team identify architectural concern based on the use cases. The idea is to identify a set of concerns that are directly related to the architecturally significant use cases. The list of architectural concerns will later be used to drive candidate solution identification.

Architectural Risk: Any risk primarily related with the architecture. A risk has a harm severity and a probability associated.

With this rather vague definition, we let the architecture team identify, based on experience, any risks that are associated with each architecturally significant use case.

3) *An concept relation diagram for the method*

One thing that is missing from system engineering guidelines on the topic of system architecting is a precise information structure and a description of how method operators would operate on this structure. Many concepts and good advice are presented, but lacks a fundamental preciseness in their advice.

For the effort of analyzing architectural drivers, we propose an information model that entails enough elements to address the wanted characteristics of the practitioners as described in section III.B. The elements are defined in the previous section. In Figure 3. we show the relations between elements here we describe the rationale for selecting this relation structure.

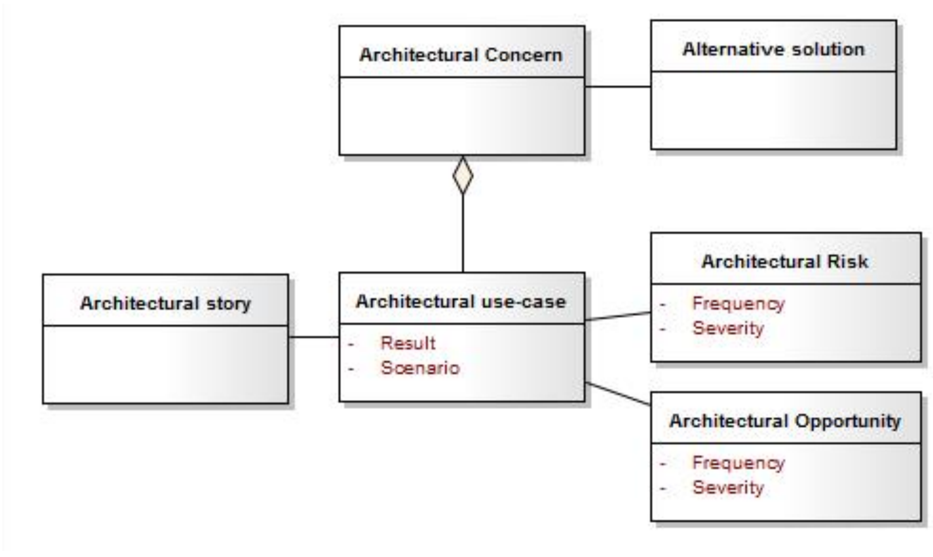

Figure 3. Information structure for analyzing architectural drivers.

Each stakeholder need is described in a *user story*. Each user story that is deemed significant is developed into a *use case*. The use case includes at least a main scenario and the description of a successful result, and possibly also variant scenarios, and quantifiable system qualities. An *architectural concern* is an aggregate of use cases with a common theme that identifies an area of concern in the design space. *Architectural risks* and *architectural opportunities* are derived from discussions around the use cases. A use case can involve several risks and opportunities, and those can in turn be related to other use cases. For each architectural concern there should be a number of alternative *architectural solutions* identified.

VI. Preliminary results from using the method

The stakeholder workshop was performed with 29 persons representing 11 roles involved in the life cycle of the drive system platform. Most roles were represented by one or two people, whereas developers where represented by 16 people. A total number of 1090 user stories were recorded.

All were categorized by using a number of type descriptions. There were 200 process related stories, 314 stories that relate to the platform system, 518 that were considered product specific. Each story was also categorized by the role and life cycle phase. We used the life-cycle model "High-tech commercial integrator" from the INCOSE systems engineering handbook. 21 stories were deemed invalid and 153 were considered as needing more clarification to be understood. Out of the total 1090 stories, less than 10% were considered to have architectural ramifications. We elaborated on the ones that were deemed architecturally important and developed a descriptive use case for each one of them. We grouped the architecturally significant use-cases together into cohesive sets that define the architectural concerns as proposed by the MFESA framework. In later stages, we plan to use the architectural concerns to drive a discussion on potential solutions, and generate different candidates for architectural mechanisms that could solve the needs. For each use case, we elaborate on what risks and opportunities could be foreseen. Sometimes these risks and opportunities showed to link to more than one use case.

A. Analysis of MP fulfillment.

To test the proposition that the method is usable, we would need more studies, but we discuss the indications from the case and analyze each property requirement of the method (MP).

MP1: By using the two types of scenarios; less formal stories and more descriptive use cases, the method should provide a result that can later be used as a base for evaluating architectural decisions.

MP2: The method was possible to carry out and there was no specific comments about it 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.

MP3: By using stories, we got stakeholders to talk rather freely and this part of the method was considered understandable. Also, we got a first list of statements quickly.

MP4: The stories, architectural use cases, and architectural concerns relate in a simple way and enable the engineers to do one thing at a time. The architectural risks and opportunities involve many-to-many relations to the use cases and that seems to hinder the practice of considering one thing at a time.

MP5: Using the given definitions and interpretations of the architectural driver elements in our method, there turned out to be less than 50 architectural use cases and architectural concerns identified on the system level.

MP6: The process turned out to iteratively refine stakeholder statements of use until the use case was considered understandable. No immediate need was seen to refine the use cases later, but more work and possibly more use cases will be required for coming break down into subsystems. We find no way of estimating completeness of the definition of architectural drivers. This is a general weakness of the method.

All MPs considered, we believe we see an indication of that two things turned out that were wanted by practitioners in the studied case. First, we get a list of the needs more quickly than if we would try to define requirements. Secondly, the method produces use cases that can effectively act as supporting the evaluation of architectural candidates.

B. Analysis of the method and future work

We note that the MPs are general in nature. Based on experience from the field, we would say that they are not specific to the studied case. If so, then a method that addresses them would be a general one.

An architecture evolves during iterations of system analysis, and the understanding of the architecture becomes clearer as the effort progresses. With this in mind, it is easy to see how an engineering task of finding the architectural drivers can yield a result whose completeness is unknown. If it is impossible to estimate completeness, a method is surely questionable. It seems very difficult to address this issue in system architecting, where issues vary from case to case.

We see that our method, like the systems engineering guidelines, does not provide a completely specified procedure. There is still some interpretation to be done by a practitioner looking to implement a procedure for architectural drivers analysis in his or her development effort. But we argue that the proposed method closes some of the gap between the general models described in the system engineering guidelines and an industrially applicable method.

The case shows some complexity in the sense that the system under development is a platform intended to be used as a base of a series of customized sub systems seems to affect the method needs to some extent. A platform is a set of reusable assets that should be optimized for a different life-cycle than that of a typical product. Previous activities of defining the system scope and identifying system stakeholders seem extra critical. The life-cycle of a platform can involve non-intuitive users and needs. We believe that the difficulty of specifying requirements is accentuated in such an effort. Future work includes studying the possibility of specialized analyses for platforms and product-line theory.

VII. Conclusion

In this paper we have presented a method for analyzing architectural drivers within an effort of engineering a system architecture. This includes; a process description with steps and work products, definitions of the involved concepts, and a concept relation diagram for the method.

We present a set of general demands on such a method as expressed by the architecture team in the studied case. Based on the practitioners demand, we have instantiated and tailored the MFESA guidelines for performing the architectural driver analysis. In addition, we developed the method by some additions and clarifications. We propose the use of architecturally significant use-cases as Muller proposes them in his book Systems Architecting: A Business Perspective. Definitions of the concepts proposed by MFESA are presented and their relationship defined. We present a UML diagram depicting concepts and their relationships and a stepwise process for carrying out the work.

We characterize the case of developing a hybrid electrical drive system and apply the method in the case. We analyze the use of the method and conclude that the process and information model are fairly general and could be considered and also useful for any developer of a complex system. The method does not fully define the procedure, but we argue that the proposed method closes some of the gap between the general models described in the system engineering guidelines and an industrially applicable method.

We find that there are no simple rules of thumb for what is to be considered in a system architecture analysis method. The architectural issues are context dependent and does not allow for the same approach for different systems. But the same difficulty does not apply when defining information concepts and their relationships. It may be possible to define a general information model with relationships between information entities and populate it differently for widely different systems.

We propose a method that can be used together with the MFESA framework, and we argue that it is not necessarily a specialization that disqualifies any particular system context from using it, rather any complex development effort could benefit from the added clarifications.

Acknowledgements

This work has been supported by Swedish Energy Agency.

References

[1] B. Boehm, R. Valerdi, E. Honour, “The ROI of Systems Engineering: Some Quantitative Results for Software-Intensive Systems,” Systems Engineering, 2008, 11, (3), pp. 221-234.

[2] P Wallin, S Larsson, J Fröberg, J Axelsson, “Problems and their mitigation in system and software architecting,” Information and Software Technology, 2012

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

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

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

[6] Fröberg, J., Cedergren, S., Larsson, S., “Eliciting Critical Information in a Pre-Study Phase of Developing a Drive System Platform for Automotive Applications,” Proceedings of the International Conference on Industrial Engineering, Systems Engineering and Engineering Management, September, 2011.

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

[8] 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.

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

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

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