Exploring Software Architecture Context

K. Eric Harper, Jiang Zheng
ABB Corporate Research
940 Main Campus Drive, Raleigh, NC USA 27606
{eric.e.harper, jiang.zheng}@us.abb.com

***Abstract*--- Architecture description can be modeled as a set of alternative choices and decisions, where the rationale and tradeoffs for each decision are documented and understood as needed to inform subsequent decisions. Each decision, based on ISO/IEC/IEEE 42010, pertains to one or more stakeholder concerns. These concerns combined with the system environment and scenarios provide architecture design context that clarifies the motivation for decisions. Subsequent authors have introduced the notion of an influencing decision force, using a many-to-many relationship with concern, to provide further context for decisions. For both concerns and forces it is left to the architect to identify the nature of this context. This paper proposes a systematic process for identifying and documenting design context in support of architectural decisions. For our work decision force is used as a central unifying aspect of the architecture framework metamodel. We extend the decision Forces Viewpoint to capture detailed design context descriptions, and add features for tagging the architecture description elements to facilitate identification of commonality, classification, and specialization. Initial feedback from industry stakeholders indicates this approach should be explored further.**

Keywords— software architecture; architecture descriptions; architecture viewpoints; architecture decisions

I. Introduction

Classical rhetoric and journalism students are taught that the context of a story can be captured by asking the 5W and 1H questions: Who, What, When, Where, Why, and How. These same questions can be posed to understand and document software architecture design context. The answers to the context questions are captured by modeling architecture description as a set of scenarios, alternatives, and decisions. Using this architecture framework it is possible to help stakeholders informally explore the design context of a proposed system and its environment, and in the process determine the priorities that drive the necessary decisions.

All architecture descriptions must at some point demarcate between the system components and their external environment. This system context boundary informs the audience which aspects of the design can be specified and influenced, and which external factors and events determine the interactions with the system. Software engineering accounts for this bifurcation with requirements analysis followed by separate software design. This approach can be taken when considering architecture description as a set of decisions. Identifying and documenting architecture design alternatives and decisions, with traceability to the forces, concerns, and affected stakeholders, is part of an architecture framework.

This work was made possible by ABB Corporate Research.

This paper proposes a systematic methodology for identifying and communicating architecture design context to bridge the gaps between external and internal stakeholders in support of architecture decisions. The decision force [1] concept, referred to simply as *force* below, is used as a central unifying aspect of the metamodel. We propose extending an existing framework [12] to enhance the Forces Viewpoint, and implement features for tagging the architecture description elements to provide taxonomies for specifying commonality, classification, and specialization. Our viewpoint extension fits into the framework to capture detailed architecture design context descriptions inspired by the Quality Attribute Workshop (QAW) [3][4] approach defined by the Software Engineering Institute, reserved previously for non-functional requirements (NFR).

II. Background and Related Work

Context in requirements engineering [5] and enterprise architecture [6] documents the Why and relates the design to organizational and business intentions, recognizing that the reasons behind the alternatives and choices may not be captured in the models. Software requirements [8] has evolved into producing a set of artifacts, starting with marketing requirements to create the vision, followed by system requirements to allocate responsibilities to hardware and software, and finally the technical requirements to capture the necessary hardware and software characteristics of the system.

Market requirements describe elicited needs or the problem to be solved as input from external stakeholders, such as customers, end users, external system teams, domain experts, auditors, senior management, support staff, operations staff, and gate owners, in their language. Sometimes the specifications can be written down as verbatim statements from those external stakeholders. These requirements consider the system from a broader perspective than specific stakeholders, including challenges related to requirements prioritization, traceability, product lines, release planning, and requirements tradeoffs.

Technical requirements describe one set of decisions (of many possible) for solution of the elicited needs, or the problem from the perspective of the technical team. Based on market requirements, the technical team translates, refines, and documents technical requirements in domain-specific language. These requirements must be traced back to the market requirements they address, and verified and validated for their intended use before release. It is important not to lose distinction in the descriptions between constraints that are outside control of the architecture, and internal characteristics of the system that can be negotiated through tradeoffs.

978-1-4799-1922-2/15 $31.00 © 2015 IEEE
DOI 10.1109/WICSA.2015.22

The idea for architecture description as a set of decisions, especially documenting the rationale, had its birth in the first related standard, IEEE 1471 [9]. Architecture decision as a first class entity in the description metamodel was promoted several year later [10]. The follow-on ISO/IEC/IEEE 42010 [2] recognized decisions as an integral aspect of architecture description, with an annex confirming use of decisions in practice as a metaphor for architecture. A dedicated group of researchers has carried the work forward with the addition of decision force [1] to the architecture description metamodel, and a proposed set of viewpoints for the decision documentation framework [12].

An alternative approach [13] for partitioning architecture documentation proposes the design space is described in three parts: concerns, problems and solutions. The authors consider architecture design context as a property of concern, defined formally as "conditions that influence design decisions but … are not specified explicitly as requirements [and] environmental [factors] that [influence] a decision."

Experience shows that NFRs have a significant influence on architecture decisions [4]. Quality Attribute Workshop (QAW) [3] is a facilitated, early intervention method used to generate, prioritize, and refine Quality Attribute Scenarios (QAS) to determine the necessary system characteristics before the software architecture is completed.

For each NFR, a detailed QAS is developed according to a template with six elements: (1) *Source of stimulus*: a human, a computer system, or any other entity that generated the stimulus; (2) *Stimulus*: condition that needs to be considered when it arrives at a system; (3) *Environment*: conditions in which the stimulus occurs; (4) *Artifact*: the entity that receives the stimulus; (5) *Response*: activity undertaken after the arrival of the stimulus; and (6) *Response measure*: the way a response is measured when it occurs. These elements may be mapped conceptually to the 5W and 1H questions.

III. Architecture Context

An architecture description documents design context by identifying the Who, What, Why, and How using instances of the elements in the architecture description metamodel. The context is revealed using the names of the instances, their relationships and detailed descriptions. Context is in part the relationship of the system to its environment, especially the interactions across the system context boundary. Context also includes the states of the interacting components that create constraints, and external factors and events.

Software development organizes ideas and relationships as elements of the software engineering process. Stakeholders provide input for the system as documented by requirements and use cases. Given this guidance the components and communication can be implemented by following the steps for analysis and design.

The metamodel [1] for the architecture decision framework has an ontology for representing design context. This metamodel is simpler than an earlier attempt [7] by the authors and is conformant to existing standards. The design context related to each element type in the metamodel is proposed in Table I with definitions and practical guidance for their use to ease the mapping of ideas to architecture documentation element types.

One key contribution from [13] is the associations between the design elements (Design Association Theory). A design problem has many of the same characteristics as a decision force and the associations could be explicitly documented with the design context instead of as a property of concern where there may be multiple possible contexts for a single concern.

With these mappings we answer four of the 5W and 1H questions. Using the architecture decision framework gets us closer to understanding and documenting the architecture design context. The full set of questions are covered by our extensions to the architecture decision framework described in the next section.

TABLE I. Architecture Description Element Definitions

Design Context	Definition		
	Element	**Formal**	**Guidance**
Who	Stakeholder	Having interest in system and concerns	Need to be satisfied
What	Concern	Influence on a system	Topic that must be addressed
Why	Force / Problem	Aspect to be considered	Motivation for concern, informs decision rationale
How	Decision	Conclusion or resolution of concern	Selection from alternatives

IV. Architecture Decision Framework

In ISO/IEC/IEEE 42010 architecture can be described as a set of decisions. Decisions are associated with stakeholder concerns as a many-to-many relationship. Subsequent authors have introduced the notion of an influencing decision force, using a many-to-many relationship with concern, to provide further architecture design context for the decision [1].

We extend the metamodel to show the influence of design context on decisions. Architecture decisions can have interdependencies where changing one decision affects others. Decision forces can contribute to each other in a hierarchy that suggests there are a number of solution alternatives. Concerns are framed from a stakeholder's perspective, but can be inter-related through common forces. Figure 1 shows the revised architecture decision metamodel and Section V provides an illustrative example.

The differences between our version of the metamodel and [1] are three-fold. First, the many-to-many relationship between force and concern is separated into two one-to-many associations, where each association is modeled by a Context class: one driven by external interest, another responding to the related forces. These associations document the design context explicitly for each force - concern pair. Second, a requirement element is added that documents a set of concerns, with one or more forces providing the motivation for the requirement. Lastly, stakeholders are shown as involved with decisions.

Splitting the force - concern relationship is significant because previously it was only an "is classified by" relationship between force and concern. The additional "is explained by" relationship supports our proposed architecture description process shown in Figure 2.

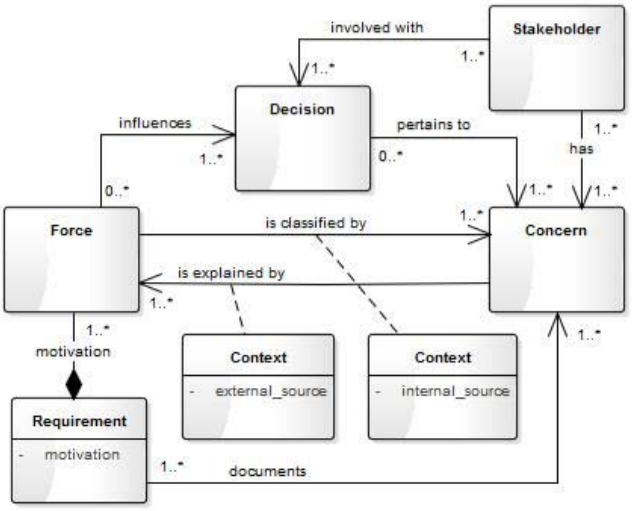

Fig. 1. Revised Architecture Decision Metamodel

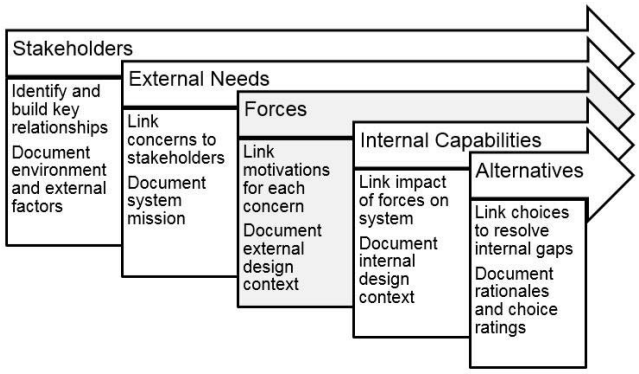

Fig. 2. Proposed Architecture Description Process

The description process highlights two steps that help us identify decision alternatives: 1) stakeholders and their external needs, and 2) the internal capabilities leading to alternatives that realize the solutions satisfying external needs. We propose that common forces motivate the concerns associated with the steps.

Our approach for explicitly documenting architecture design context is inspired by the template for QAS defined by the Software Engineering Institute. The proposed elements provide a concise representation of the design context as shown in Table II. Now we can document the complete 5Ws and 1H, where communication of the context fosters focused debate and hopefully rapid resolution of stakeholder differences.

Design context documentation is more likely for a project with tool support. Our architecture description framework is prototyped with extensions to the open source Decision Architect add-in [14] for Enterprise Architect [15]. As shown in Figure 3, the *Forces Viewpoint* is enhanced from [11] to add a design context documentation table complementing the decision matrix, where the rows in Table II correspond to columns in the Forces Viewpoint of the add-in when viewing the design context instead of the decision alternative ratings. A new *Context Viewpoint* is envisioned but not yet specified to manage individual force – concern pair details.

TABLE II. ARCHITECTURE DESIGN CONTEXT ELEMENTS

Element Type	Original QAS Type	Design Context	Description
Source	Source of Stimulus	Who	Component initiating external factors or events
Stimulus	Stimulus	What	Method and protocol for initiation
State	Environment	When	Enabling preconditions of the scenario
Artifact	Artifact	Where	Affected features or components
Response	Response	How	Manifestation of results, post-conditions
Benefit	Response Measure	Why	Expected quantified response range, effects

Our final contribution to the architecture decision framework, and embodied in the extension prototype, is a facility for tagging instances of elements in the architecture description metamodel with values using a common standard. Each stakeholder, force, concern and decision topic has placeholders for marking the element with category, context and type. The tags can be applied in any manner, but for our purposes category is used for finding commonality, context provides a means for classification, and type allows for specialization within a classification. Using tagged values provides description authors with a means for defining detailed taxonomies as needed without relying on custom profiles to extend the metamodel, and supports the ontology representation in our previous work [7].

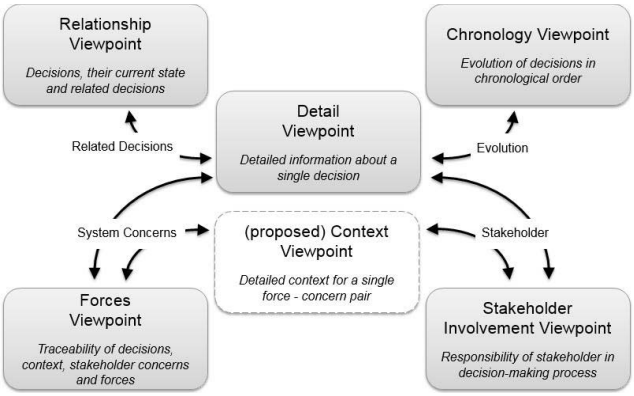

Fig. 3. Extended Architecture Decision Framework

V. ILLUSTRATIVE EXAMPLE

Practical use of design context in architecture description can be illustrated with a consumer product example. Figure 4 shows the system context diagram for an automobile power window control where the window is activated by commands initiated by the driver and passengers. The external need is to prevent a (child) passenger from operating the window and the identified necessary internal capability is access control for window operation. The common force that links these two concerns is child safety.

The left hand scenario in Table III summarizes the design context for the external need, and the right hand scenario for the internal capability. Use of the design context elements is as appropriate for the specific scenario and each element can be expanded with more detail. Agreeing on the design context,

stakeholders select alternative choices for the design, computing the lock setting based on: 1) the speed of the vehicle, 2) weight of the passenger, or 3) lock settings available to the driver. The motivations for each of the choices may differ in the two scenarios, but with a normalized scale the rating values can combined to determine a cumulative weight for each alternative.

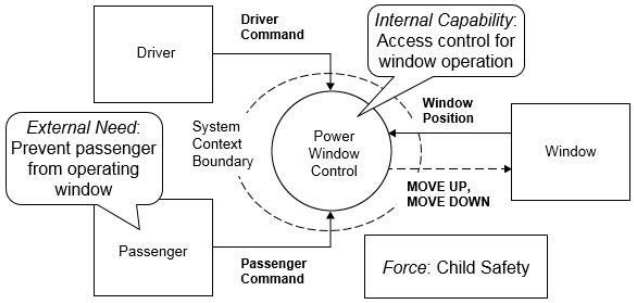

Fig. 4. Power Window Control – Illustrative Example [16]

TABLE III. POWER WINDOW CONTROL - SCENARIOS

Force: Child Safety	Scenarios	
	External: Prevent passenger from operating window	Internal: Access control for window operation
Source	Child passenger	Passenger window switch
Stimulus	Press window button	Up / down operation
State	Risk of accident	Window locked
Artifact	Power window control	Power window control
Response	Window movement	Control signal to window actuator
Benefit	No operation on window	No operation on window

VI. INITIAL INDUSTRY EXPERIENCE

Work on a prototype for the Decision Architect extensions to the Forces Viewpoint supporting architecture design context wrapped up late in 2014. Since then the authors' process has been executed for an industry project. A spreadsheet approach had been used to great benefit for quality attribute scenarios in previous activities, and the Enterprise Architect add-in alternative is much better suited to linking the design context to other aspects of the architecture model.

Initial ad-hoc feedback from stakeholders on the proposed metamodel has been mixed. None were using Decision Architect, the tool was not part of the solicitation for feedback. Some product architects are not open to proposals that create additional documentation tasks, especially if they do not fit with their established practices. On the other hand, other practitioners see the approach fitting with their efforts to define a product "solution surface" and to create a roadmap that drives the development, effectively a "knowledge roadmap". Product owners do not grasp the architecture theory but can see the value of the force linkage between concerns for application definition to provide a framework guiding people to a solution.

VII. CONCLUSIONS AND FUTURE WORK

Our architecture decision metamodel extends the standard for architecture description to incorporate explicit modeling and documentation of the scenarios associated with concerns, linked by common motivating forces that bridge the discussions between external and internal stakeholders. The well-known quality attribute scenario elements are leveraged to describe architecture design context by answering the 5Ws and 1H questions.

The proposed architecture framework is more likely to be adopted if it avoids additional documentation burdens for practitioners. The table entry format is an improvement because it does not require linking architecture elements by drawing lines. The Decision Architect Context Viewpoint is yet to be implemented and can assist further in searching and navigating complex software architectures.

REFERENCES

[1] U. van Heesch, P. Avgeriou, and R. Hilliard, "Forces on Architecture Decisions - A Viewpoint", Proceedings of the 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA-ECSA'12), Aug. 2012, pp. 101-110.

[2] ISO/IEC/IEEE 42010:2011, "Systems and software engineering - Architecture description", Dec. 2011.

[3] M. R. Barbacci, R. Ellison, A. J. Lattanze, J. A. Stafford, C. B. Weinstock, and W. G. Wood, "Quality Attribute Workshops, Third Edition", Technical Report, CMU/SEI - 2003 - TR - 016, August 2003. http://www.sei.cmu.edu/reports/03tr016.pdf

[4] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice. Second Edition, Addison-Wesley, 2007.

[5] D.T. Ross and K.E. Schoman, "Structured Analysis for Requirements Definition", IEEE Trans. Soft. Eng., Vol. SE-3, No. 1, January 1977.

[6] E. Yu, M. Strohmaier and X. Deng, "Exploring Intentional Modeling and Analysis for Enterprise Architecture", 10th IEEE Int. EDOCW, pp. 32-40, October 2006.

[7] K. E. Harper and A. Dagnino, "Agile Software Architecture in Advanced Data Analytics", Proceedings of the 11th Working IEEE / IFIP Conference on Software Architecture (WICSA'14), May 2014.

[8] K. E. Wiegers, "Software Requirements: Practical techniques for gathering and managing requirements throughout the product development cycle", Second Edition, Microsoft Press, 2003.

[9] IEEE 1471-2000, "IEEE Recommended Paractice for Architectural Description of Software-Intensive Systems", September 2000.

[10] P. Kruchten, "An Ontology of Architectural Design Decisions in Software-Intensive Systems", Proc. 2nd Groningen Workshop on Software Variability Management, December 2004.

[11] C. Manteufel, D. Tofan, H. Koziolek, T. Goldschmidt and P. Avgeriou, "Industrial Implementation of a Documentation Framework for Architectural Decisions", IEEE/IFIP Conf. on Software Architecture, 2014.

[12] U. van Heesch, P. Avgeriou and R. Hillard, "A documentation framework for architecture decisions", J. Syst.Software, vol. 85, issue 4, pp. 795-820, April 2012.

[13] A. Tang, M.F. Lau, "Software Architecture Review by Association,", J. Systems and Software, Vol. 88, pp. 87-101, February 2014.

[14] Decision Architect, https://decisions.codeplex.com/

[15] Enterprise Architect, Sparx Systems http://www.sparxsystems.com/

[16] The Mathworks, "Power Window – MATLAB & Simulink", http://www.mathworks.com/help/simulink/ug/power-window-example-case-study.html