A Classification of Value for Software Architecture Decisions

Ulrik Eklund and Thomas Arts

Department of Applied IT
Chalmers Univ. of Technology/Göteborg University, Sweden
ulrik.eklund@ituniv.se

Abstract. This paper introduces a classification for decisions originating from work performed by architects. With the creation of a new architecture, all observed decisions were documented using an existing taxonomy extended with the introduced classification. In the first four months, 80 decisions were documented. Not all decisions have the same value for the architecture and one needed a classification to reason about importance of decisions. After realization of the first increment of the architecture a sanity check was performed: The architects showed how the six most important design artefacts and the fifteen most important architectural constraints and prerequisites were related. The relationship was via decisions and the classification helps to reduce the work to make and maintain this connection over time. The classification is dynamic and over time decisions can be classified differently. This enables architectural learning by pointing out which decisions were taken too early or had little impact.

1 Introduction

The classification introduced in this paper originates from a practical problem a group of architects was confronted with. In their preparation of a software architecture for a new product, they have to take a large number of decisions. Design artefacts such as specifications, models and code remain, but the 'why' is lost over time. In the lifetime of a product, but in particular when an architecture for a new product is created, an answer to the 'why' question is of utmost importance; "Did we base this decision on technology that now is replaced?"; "Did we take this decision because the company decided for a specific business unit to implement it?". If the reasons for a decision has been invalidated, then it would be wise to revisit that decision, but one can only do so, if the reasoning around the decision is documented.

The architects were in particular interested in the relationship between design artefacts and the prerequisites for the architecture, which includes business and technical requirements, and design constraints. During the work we noticed a demand among the architects to discuss and understand more in detail how prerequisites and artefacts were related, especially as a rationale for the architecture as a whole. This lead to a new classification of decisions supporting reasoning about the value or usefulness of a decision, also over time.

M. Ali Babar and I. Gorton (Eds.): ECSA 2010, LNCS 6285, pp. 368–375, 2010.

We aim to aid in answering questions like "do the architects spend their time on the right/best decisions?", "Are there some decision the architects should *not* make?" and "which decisions could be reused?". Our contribution is a classification of decisions and show that this helps architects to:

- detect possible decisions that need to be elaborated on,
- detect decisions that need discussion with stakeholders,
- detect over time whether the architects spend their time on taking the right decisions, those that create true value for the organisation. These are the decisions that would be impossible, less efficient or more costly if they would be made by an other stakeholder, for example an acquirer or programmer.

The focus on the vital decisions become even more necessary if architects are a limited resource or one has a lean perspective on software development, i.e. eliminate spending time on issues not creating value.

Related Work. Kruchten et al. [1] have noticed the need of explicitly documenting design decisions and recognize that this is often omitted in practice. They present an ontology to help documenting and analysing design decisions. In our case the architects build upon this existing ontology. Tang et al. [2] focus much on the relation between prerequisites, decision and artefacts and have tool support for documenting this by means of a UML model profile [3]. This approach would support a change impact analysis of the architecture, e.g. [4].

A difference between the two mentioned approaches is that Tang et. al. only describes relationship between prerequisites and design outcomes and not between decisions themselves. If such relations need to be expressed, then a design outcome from one decision must be modelled as a prerequisite for another. Kruchten et al. on the other hand give no extra status to prerequisites; these are decisions at the beginning of the chain of relations and one may use a decision attribute to document their special rationale.

2 The Case of Documenting Decisions

How can one help the architect to make a limited set of decisions, and still do a proper job? We expect that each "architectural requirement" or prerequisite relates via a number of decisions to at least one design artefact, most likely a few. Similarly we would expect each design artefact to be traceable to at least one prerequisite. If we consider the software architect to be the link between the requirement owners (stakeholders) and the software design, then part of the job of the software architect is to take decisions such that the set of architectural artefacts is a smallest set covering the prerequisites.

In our case a team of software architects is appointed to deliver a new software architecture, but one of them has the extra task as an industrial PhD student to document all decisions taken. This is communicated to the team and everyone agrees on the usefulness of that extra task being carried out. The decisions were

documented in a systematic way with attributes[1] similar to the ones by Kruchten et al. [1]. Additional notes were taken describing how the decision was taken, e.g. was it made by the lead architect, by consensus after discussion, or if alternatives were not even discussed?

In the first four months, 80 decisions were documented relating to both the process of defining the architecture and the resulting artefacts. After first increment of the software architecture a sanity check on the work was to be performed. In order to do so, the architects showed how the six most important design artefacts and the fifteen most important architectural prerequisites and constraints were related by various decisions.

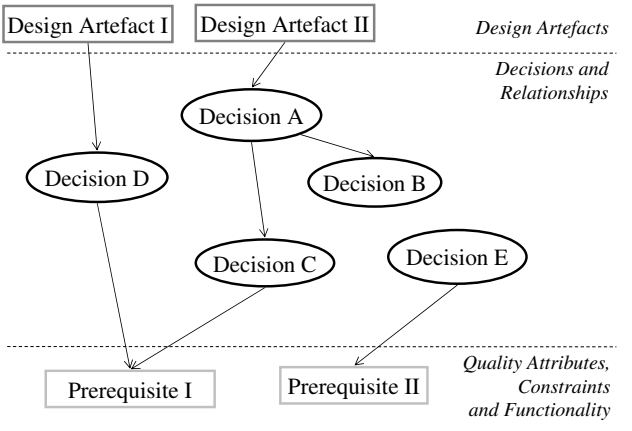

Fig. 1. The relationship between decisions and other elements in a simple graphical notation. The relationships to other architectural artefacts are inspired by [2].

When documenting decisions care was taken to relate them. Whenever a new decisions was added, it was related to already existing decisions if possible. Kruchten et. al. have a rather elaborate categorisation of different relationships [5], [1], but in practice this richness of relations is a bit overwhelming; the simplest thing to determine is whether a decision depends upon another. We propose a very simple relationship of "is influenced by"[2]. If decision B is influenced by decision A then decision B must be re-evaluated if decision A is removed or changed. This simple relationship would make it possible to evaluate how far the influence of a single decision reaches. The link between artefacts and decisions and between decisions and prerequisites were made as part of the documentation process (Fig. 1), sometimes within the team of architects, sometimes by the PhD student alone. These relations are also characterized as "influenced by".

Future work involves evaluation of this classification together with an analysis of the benefits at Volvo Cars.

[1] Epitome (or decision itself), Rationale, Scope, Authors, State, and Category.
[2] Note that the relation 'influenced by' is the inverse of Kruchten's 'depends on'.

3 Classes of Architectural Design Decisions

We now want to talk about the decisions by characterizing the decisions based upon how valuable the decisions are for the organization. Our point of view is that each design artefact should be based upon a decision taken and that decisions are taken to meet some prerequisite.

If we consider for simplicity the prerequisites and design artefacts as decisions as well, then in this way one obtains a directed graph of decisions. When the Ph.D. student studied this graph, it was observed that certain relations were missing, since it was believed that two nodes were related, but no path existed between them. In those cases it turned out that an implicit decisions was taken they were added to the graph (similar to "Implicit and undocumented decisions" mentioned in [1][3]. Obviously, there may still be implicit decisions not recognized this stage and therefore not made explicit, hence undocumented. The implicit decisions are the first we want to define a name for in order to talk about them.

Oblivious Decisions are the decisions that the architects are not aware that they are making and at best are documented in hindsight. Examples include earlier experience, implicit company policies to use certain approaches, standards, and the like. These were the most difficult decisions to observe since the observer was native to the setting he observed, i.e., he was as accustomed to the 'of course' knowledge as the other architects. The existence of them is based on a theoretical reasoning rather than empirical observation.

Example: Typical examples are decisions where there is only one alternative. This can be due to technical limitations, but also that the consensus is so strong or the decisions was taken so long ago that no-one is aware of any alternatives.

3.1 Classification by Relations in the Decision Graph

We base our terminology on the directed graph, of which Fig. 1 is an example, obtained by relating all decisions and including prerequisites and design artefacts as nodes in the graph. We first divide the decisions in four main classes, corresponding to the following relation with the decision graph:

Exterior Decision. A node in the graph that has a path to a prerequisite.
Interior Decisions. A node in the graph that has no path to a prerequisite.
Effectual Decisions. A node in the graph for which there exists a path from a design artefact to this node.
Ineffectual Decisions. A node in the graph for which there exists no path from a design artefact to this node.

Exterior decisions have a clear stakeholder that drives the decision and the decisions give value to the organization by bringing the requirements of a stakeholder closer to the design artefacts. Decision A, C and D in Fig. 1 are typical Exterior decisions.

[3] "The architect is unaware of the decision, or it concerns 'of course' knowledge."

Table 1. Example of a decision directly affecting the design outcomes driven by a number of business concerns

Name:	#35 AUTOSAR Basic Software
Epitome:	The basic software of the electronic control units (ECU) in the electrical system shall follow the AUTOSAR standard.
Rationale:	
Scope:	All software in the electrical system
History:	Director nn, 200x-xx-xx, 1st version
Category:	Exterior Effectual decision
Note:	This is an assumption that the architects have worked on since the project start in 2008 and was observed as a decided fact rather than when the actual decision was made by management.

Example: A typical example of an exterior decision is the use of the AUTOSAR standard [6], which supports a number of business decisions and defines a number of standardized software components that are part of the design outcomes (cf. Table 1 for the documented observed decision).

Interior decisions are decision necessary for the architecture to progress. Decision B in Fig. 1 is a typical Interior decision. Within this class of decisions, we discriminate two kinds of decisions: *Imposed Decisions* and *Supporting Decisions*.

Imposed decisions. Decisions that are imposed on the architects and need to be resolved for the design of the architecture to progress. There is no stakeholder that drives the decision, but a choice needs to be made in order to progress. Normally the choice made limits certain future business cases. An experienced architect will need a solid knowledge about the system and what needs to be resolved in order to have a finished product.
Example: For a connected car [7] it is important to know if the car manufacturer will offer all services or if 3rd parties also shall have a possibility to offer services (in some sort of open innovation scenario). This is really a business decision which acts as an architectural prerequisite but if it is not known the architects need to make an assumption to progress the work with the technical solution in the car anyway.

Supporting Decisions. A supporting decision is a decision necessary for the architects to progress, but not discernible for other stake holders than the architects themselves.
Example: An architecture team is tasked with developing both a product line architecture and the architecture for the first instance. They can then decide between first developing the product line architecture and use that as a basis for the product architecture. Or they can first define a product architecture and then generalise that to a product line. Either way it is not discernible for any stakeholders which decision they made if they are both delivered at the same time.

Table 2. Example of a decision directly affecting the design outcomes

Name:	#25 Choice of deployment views
Epitome:	The logical architecture components will have three deployments: 1. Logical architecture components onto hardware (ECUs) 2. Logical architecture components onto systems 3. Logical architecture components onto organisation The deployment will be modelled separately from the logical package structure in the UML model.
Rationale:	
Scope:	The entire logical architecture, the entire life-span
History:	Architect nn1, 2009-06-08, 2nd version
Category:	Exterior Effectual decision
Note:	Consensus decision after several discussions. Original decision observed at working meetings of the logical architecture team.

Effectual decisions result in a visible Design Outcome. In a design document driven organization, these are typically the decisions that the software architects are expected to make and document the outcomes in various views. An example of an effectual decision is seen in Table 2 stating how the design outcome will be presented. However, the software architect should obey to the principle of "an architect should make as few decisions as possible, deferring the rest until later in the lifecycle" [8].

Ineffectual Decisions are those decisions that address a prerequisite but are never visible in a design artefact.

We have not found any examples of Ineffectual decisions in our study but an analysis of design decisions and their relationship to Concerns and Outcomes shows that these types of decisions can also exist if the classification should be considered complete. Decision E in Fig. 1 is a typical Ineffectual decision.

4 Value of Decisions

By defining classes of decisions as above, one can determine by the position in the graph what kind of decision is taken (Exterior or Interior, Effectual or Ineffectual) and for those that are Interior, one can determine by studying the decisions whether it is an Imposed decision or a Supporting decision. This can then help to determine the value of a decision or to detect decisions that need elaboration or more discussion with stakeholders. The value depends on the organizational context. In the context of Volvo Cars, some guidelines can be formulated for decisions that probably need more attention than other decisions. For example, interior imposed decisions are most important to document, since they indicate that a technical decisions is made before the business stakeholders have made up their mind. In other words, further discussion with stakeholders is required; either immediately for the software release under development or at a later stage when the business propositions are clearer.

Ineffectual decisions may in the case of Volvo Cars indicate that the architects are not yet ready with their work, since the rôle of the software architect is seen to produce the initial top-level design and Volvo Cars is a design artefact-driven organization. In other organizations it may be the other way around, where many effectual decisions may indicate overambitious software architects that deal too much with details.

Exterior and effectual decisions are, of course, also important to document, since the "why" will be forgotten when the Design Outcome is fixed, but these decisions are relatively easy to trace in the organization and can potentially be reconstructed.

5 Classifying Decisions over Time

The classes of decisions we described before are statically determined. However it was observed that decisions change over time for various reasons and this needs to be addressed when evaluating the value or usefulness of them. Two classes emerged after the involved architects analysed the 80 observed decisions in our case. In order to asses the value of the decisions over time the analysis needs to be iterated.

Unstable decisions are those decisions that change over time due to added or changed prerequisites, given that these prerequisites were hard to foresee. At the moment of this analysis, we have not yet been able to identify such decisions, but since our study only lasted four months and the products based on the architecture are manufactured in more than seven years this is not unexpected.

Premature decisions are decisions that show to be erroneous over time. They had to be changed without new prerequisites emerging, because they were based upon incorrect interpretation of prerequisites or forgotten prerequisites.

Example: A decision on what to include in the architecture description was changed from: Items/headings that are known to be included in the reference architecture description with a comment that future information will be included in future versions of the document, to: There will not be any empty headings with TBD (to be defined) in the architecture description. This decision was changed by the lead architect after three months. Because of too little contextual information available at the moment that the decision was taken, this precise decision had to be adjusted quickly after.

Expedient decisions are those decisions that do not change over time (thus it depends upon when in time one determines their status whether they are expedient or not). Expedient decisions are unchanged when prerequisites are added or changed.

6 Conclusion

With the proposed classification of decisions it should be possible to reason about the value of decisions, as seen from the architects perspective, both when

the decisions are are made and later in retrospective. The classification should support post-mortem analysis if the architects spend their time on the most useful issues, especially in the view of the architects being a limited resource.

A software architect should observe and take care of Interior Imposed decisions, since they form a potential risk for the architecture. If one is to re-use decisions in a next project, then the Interior Imposed and Interior supported decisions need to be evaluated thoroughly. Exterior decisions also need to be re-evaluated, but the situation of having a stakeholder for them eases that task.

When gaining experience from working with software architecture it is important to observe which decisions become unstable, premature or stay expedient throughout the product lifetime. In particular premature decisions indicate a learning opportunity for software architects.

In practice it seems impossible to document all architectural decisions in the lifetime of a car, in particular to maintain the documentation of these decisions. A learning organization starting to document decisions will become better in choosing which decisions to maintain.

If an organisation is interested in re-using architectural knowledge from previous projects and systems, it should also be interested in what subset of this knowledge that is *useful* for the architects to re-use. We believe this paper presents a classification and an associated in-depth terminology to use in such analyses.

Acknowledgements. This work has been financially supported by the Swedish Agency for Innovation Systems (VINNOVA) as part of the FFI program. We are grateful for all the time fellow architects have contributed in discussions.

References

1. Kruchten, P., Lago, P., van Vliet, H.: Building up and reasoning about architectural knowledge. In: Hofmeister, C., Crnković, I., Reussner, R. (eds.) QoSA 2006. LNCS, vol. 4214, pp. 43–58. Springer, Heidelberg (2006)
2. Tang, A., Han, J., Vasa, R.: Software architecture design reasoning: A case for improved methodology support. IEEE Software 26(2), 43–49 (2009)
3. Tang, A., Jin, Y., Han, J.: A rationale-based architecture model for design traceability and reasoning. Journal of Systems and Software 80(6), 918–934 (2007)
4. Jansen, A., Avgeriou, P., van der Ven, J.S.: Enriching software architecture documentation. Journal of Systems and Software 82(8), 1232–1248 (2009)
5. Kruchten, P.: An ontology of architectural design decisions in software intensive systems. In: 2nd Groningen Workshop on Software Variability, pp. 54–61 (2004)
6. AUTOSAR: AUTomotive open system ARchitecture, AUTOSAR (2009)
7. Automotive technology: The connected car. The Economist (June 2009)
8. Tyree, J., Akerman, A.: Architecture decisions: demystifying architecture. IEEE Software 22(2), 19–27 (2005)