Goals, Questions and Metrics for Architectural Decision Models

Marcin Nowak, Cesare Pautasso
Faculty of Informatics, University of Lugano (USI)
http://saw.inf.unisi.ch/
marcin.nowak@usi.ch, c.pautasso@ieee.org

ABSTRACT

Architectural decisions are the key element behind the design process leading to a software architecture. Making software architects aware of the implications of their decisions is only the beginning of what can be achieved by capturing the rationale and the constraints influencing the decision making process in a reusable body of architectural knowledge. In this paper we propose a metric-based approach to the analysis of architectural decision models. Using a hierarchically-structured approach we identify a number of useful goals and stakeholders involved in the architectural design process. Next, we sketch a set of metrics to provide data for the evaluation of the aforementioned goals. Our aim is to stimulate a discussion on how to find indicators relevant for software architects by measuring the intrinsic properties of architectural knowledge.

General Terms

Design, Documentation, Measurement, Verification, Complexity

Keywords

Software Architecture, Architectural Decision Modeling, Visualization

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; D.2.2 [**Software Engineering**]: Design

1 Introduction

Software metrics have been widely studied in the context of software reengineering [12] and software process improvement [21]. In general, the goal of applying metrics to software has been to assess the maturity of a software development organization [26] and quantitatively compare the quality of the code artifacts it produces [20]. Concerning software architecture, metrics have been applied to observe and compare architectural models (e.g., [17]) within different architectural analysis methods [10]. As opposed to measuring the observable properties of an actual software architecture [28], in this paper we focus on measuring properties of the knowledge captured as part of software architecture decision models. This knowledge is an important business asset for software design organizations [2, 22]. Our intent is to extract additional insight from the captured knowledge and the decisions by applying quantitative and qualitative metrics. Measuring the knowledge helps to evaluate its quality in order to improve it. Thus, it also helps to improve the quality of the outcome of the design process which makes use of the knowledge to guide decision-making. Observing the dynamics of the design process of a software architecture also helps to estimate the complexity involved with a given project, to keep track of changes in the design and to estimate how quickly it is progressing towards completeness.

The paper makes the following novel contributions. It proposes to apply the GQM (Goal-Question-Metric) methodology [23] to measure software architectural knowledge. It outlines a set of useful goals for the main stakeholders involved in the design of a software architecture. For each goal, we present a set of relevant questions which are answered by combining 26 different metrics. Metrics can be applied to the entire design space or to narrower scopes as well as to the decisions made within a specific project space.

The approach and the metrics that we have designed can be applied to any software architectural decision meta-model which complies with a minimal set of assumptions which are shared by most issue-based information systems (IBIS [9]). By showing how to define metrics starting from the intrinsic properties of a generic architectural meta-model, the approach goes beyond basic tradeoff analysis performed by aggregating cost/risk factors associated with different design solutions [15]. Since the metrics we propose do not rely exclusively on specific attributes being associated with the knowledge items, we can obtain useful insight without requiring additional input and extra effort to keep the model up to date.

In this paper we provide an initial theoretical investigation sketching a potentially useful set of design-space metrics. This investigation makes a strong assumption that the software architectural knowledge is modeled completely. Relaxing this assumption for measuring incomplete design spaces is outside the scope of this paper. Dealing with the analysis of partially unknown knowledge can be seen in the context of a broader set of research challenges [19]. We plan to evaluate the metrics as we complete their implementation in the Software Architecture Warehouse tool.

The rest of this paper is structured as follows: in Section 2 we discuss the benefits of introducing metrics to observe and analyze architectural decisions models. Section 3 contains the related research which has inspired our work. In section 4 we give some background on the knowledge meta-model assumptions and the GQM method used to design our metrics. Section 5 covers five goals covering a broad range of design scenarios. In section 6 we define 26 metrics grouped according to the aspects of the knowledge meta-model being measured. Section 7 concludes the paper.

2 Motivation

Once the knowledge embedded in a software architecture has been made explicit and it has been captured in a reusable decision model, the opportunity arises to measure it in order to address the following needs.

It is important to assess the quality of the captured knowledge so that, for example, the maturity of different knowledge repositories can be compared. Our expectation is that everything else being equal, an architecture designed using high quality knowledge it is likely to be of higher quality than one designed using a lower quality decision model. To this end, unless a set of well-defined metrics is introduced in the context of a given knowledge meta-model, it becomes difficult to evaluate the quality of the corresponding captured knowledge.

Another useful expected outcome of measuring architectural knowledge is the possibility of estimating the effort required to obtain a design. Architects approaching knowledge domains unknown to them can use metrics to assess the domain knowledge's intrinsic complexity by observing the entanglement of the relevant design issues and related architectural alternatives.

With the goal of speeding up the overall design process, metrics can also be applied to help with the pruning of irrelevant decisions so that architects can not only focus their efforts on the relevant ones, but also identify the critical decisions which should be initially attacked. Decisions can be prioritized according to different metrics which weigh their estimated impact against the corresponding constraints.

Additionally, metrics make it possible to keep track of the progress of decision-making. In general, it is particularly challenging to decide when a design is completed and the architect can stop making decisions. Measuring the dynamics of the decision-making process can help to gather the necessary evidence that a project has reached its fixed point and all the relevant decisions have been completed in a consistent way.

3 Related Work

Significant efforts have been invested in researching efficient methods for structuring, managing, and analyzing the body of knowledge accumulated in software engineering projects [2, 22]. Concerning the software architecture design process, architectural decision models [13] were proposed as the formal means to capture the assumptions, constraints, issues, and rationale for a certain solution domain in a coherent and reusable body of knowledge.

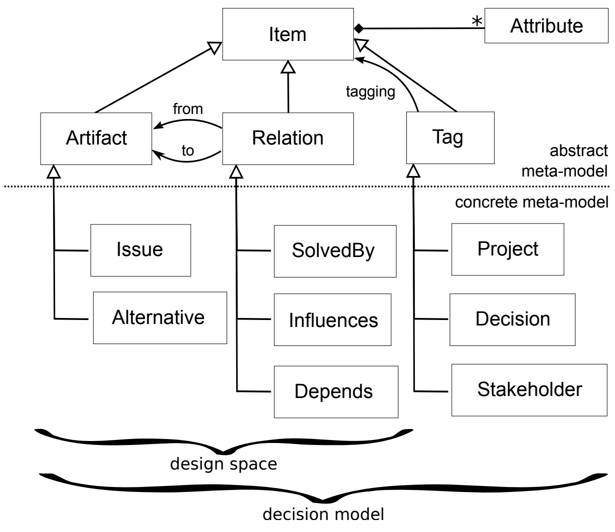


Figure 1: Basic knowledge meta-model.

In this context, various methods have been proposed to support software design [10], development [12], and evolution [5] processes with the use of qualitative and quantitative metrics. The analysis of high level software properties such as size, complexity, cohesion and coupling metrics was described in [12]. A systematic approach for aggregating low-level metrics to give meaningful answers to specific questions was proposed by [18]. Metrics have also been applied to quantitatively observe the properties of specific software architectures. First attempts to software architecture analysis were made with the qualitative SAAM method [14]. This was followed by quantitative methods for architecture evaluation based on cost-benefit analysis (CBAM [7]) and quality attributes (ATAM [15]). The idea of incorporating GQM into the software architecture analysis was proposed in SIMPLE [8] for the purpose of software product line evaluation.

In this paper we shift the focus onto measuring the software architecture knowledge captured in reusable decision models. We propose a minimal architectural knowledge model that is supposed to be augmented with additional attributes such as risk factors or effort estimates required to perform a different kind of 'what-if' analysis to compare different design solutions obtained over architectural knowledge. We do not assume or require any particular number of attributes, and instead we focus on observing the intrinsic properties that can be measured out of the basic knowledge found in most architectural decision meta-models.

4 Background

4.1 Knowledge meta-model

Following the observation that software architecture design is a wicked problem [27] we present a basic software architectural meta-model which generally follows from the IBIS (Issue Based Information System [9]) and the Question-Option-Criteria [16] concepts.

The meta-model which we present in this paper should be considered as a minimal subset of concepts appearing in [1, 6, 25, 29], that is still sufficient for the purpose of demonstrating our ideas. As shown in Figure 1, the meta-model defines elements of two abstraction levels. The generic part is composed of three kinds of knowledge *Item* elements: a graph of *Artifact* nodes linked by *Relation* edges and classified by various *Tag* types. Any instance of this type is uniquely identified by its ID and can carry a predefined set of attributes specific for a given type. For example in the case of a design issue these could be: name, description, and recommendation.

These generic elements are instantiated into a meta-model which adopts a subset[1] of the architectural knowledge concepts proposed by Zimmermann [29]. To be precise, we use knowledge artifacts such as *Issues* and *Alternatives*, together with the basic *SolvedBy* relation connecting an *Issue* with the corresponding *Alternative*. The relation of *Dependency* represents constraints between different *Alterative* artifacts. We also include the *Influence* relation, which can freely occur between any combination of *Issues* and *Alternatives*.

A given knowledge item can be tagged with multiple tags, thus providing support for arbitrary, multi-dimensional classification of the knowledge. Here we introduce three specialized tag types which are going to be used in this paper to define a number of useful metrics:

Decision this type of tag represents the possible outcomes of the decision-making process. Tags are attached to the *SolvedBy* relation between an *Issue* and an *Alternative*, since we allow the reuse of alternatives shared among multiple issues. In practice, the possible outcomes of a decision are represented by three tags of this type: *Positive* - denoted with $+$ (the alternative has been selected), *Negative* - denoted with $-$ (the alternative has been rejected) and finally *Neutral* - denoted with N (the alternative has been considered but no preference has been recorded). The lack of decisions is denoted with the symbol $\emptyset$ (the alternative has not yet been considered – see Figure 3). Multiple decision tags recorded over the same *SolvedBy* relation enable fine-grained modeling of consensus-reaching processes, for example through voting, where multiple designers can provide their own contribution to the final decision model.

Project tags are used to mark the relevance of a particular knowledge item in the context of a project and thus its membership in a given project space [19].

Stakeholder tag type represents individuals or roles that are involved in the decision making process concerning the artifacts to which they are attached to.

Tag-based classification of knowledge items is particularly useful for selecting the scope of items to be measured with the use of a given metric. Alternatively a metric's scope can be defined by following relations to and from knowledge items and thus bring indirect influences into consideration. The ultimate value of metrics calculated within a scope, containing multiple knowledge items, is the result of aggregating values measured for individual knowledge items. As we are going to show, the aggregation strategy depends on the specific metric.

[1]These basic concepts can turn out to be insufficient to cover more complex aspects of real-world scenarios, and thus should to be extended to fit with the particular needs of different design domains or processes.

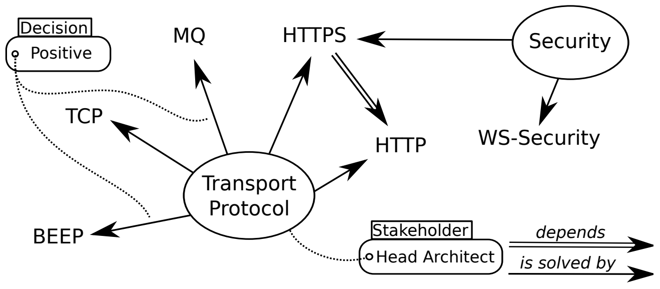


Figure 2: An example of design space with two Issues, six Alternatives and eight Relations.

4.2 GQM methodology

To define and organize our metrics for architectural decision models, we follow the guidance of the GQM (Goal Question Metric) framework [3, 4, 11]. We have selected from [23] a set of five goals that is intended to ensure a successful architecture design process and should be general enough to be used in conjunction with any knowledge-based design methodology. Later, we decompose each goal into a set of questions considering the relevant properties of the design space and the decision model. For each question we have defined a number of metrics which can be combined to serve as a basis for an answer.

Following [24], every metric is associated with a scale: either qualitative (nominal or ordinal) or quantitative (interval or ratio). Most of the metrics we define use a ratio scale over different ranges (i.e., N, non-negative integers, `{true,false}` booleans, `[0,1]` rational numbers, and `time` timestamps).

4.3 Stakeholders

Every collaborative design effort involves multiple stakeholders. First, we identify the main software architecture design roles (after [11, 30]) so that we can relate them with specific goals in the next section.

Project Manager: responsible for providing the equipment and the human resources required to perform the design tasks. He/she also keeps track of their progress in order to deliver the design (and the implementation) meeting all milestones and deadlines agreed with the customers.

Head Architect: performs the main technical role in the design process. She/he is responsible for obtaining a consistent and feasible design fulfilling all application requirements. As a head of the design team his/her responsibility is to oversee the design and to communicate with technical and business-level project stakeholders.

Business Analyst: translates customer needs into formalized requirements and communicates them to the technical members of the team.

Scribe is a professional writer in charge of documenting which issues were considered during design meetings and capturing the rationale behind the chosen alternatives so that the architect's knowledge and intention can be accumulated in the decision model.

5 Goals and Questions

5.1 Prioritize Decisions (G_1)

Stakeholder: Head Architect.

Bootstrapping the design process for a complex project can be a challenging task. In order to support this task we suggest the use of metrics which provide architects with guidance that will dynamically prioritize decisions according to their properties and relations.

Questions:

1. What is the impact of this decision on other decisions? This can be answered by counting the number of outgoing influence relations (*influence out-degree* (M_4)) and the number of decisions which depend on it (*dependency out-degree* (M_2)). In case of top-down design, highly influential decisions are likely to be taken first.
2. How strongly is this decision influenced or constrained by other decisions? The metrics *influence in-degree* (M_3) and *dependency in-degree* (M_1) can be combined to answer this question. Early identification of the design issues which require to make decisions which depend on many other issues can ease the preparation of preliminary design sketches while keeping the decision model consistent. Leaving these issues for late arbitration can result in extensive changes towards the end of design process to ensure consistency.
3. How complex is the decision? This can be learned from the *complexity* (M_5) metric. Depending on the chosen design strategy, it can be beneficial to begin the design by considering complex issues, or – on the contrary – to leave them until the end by first tackling the simple issues.
4. How many people are involved in this decision? By observing metrics counting the number of *individuals involved* (M_{15}) and the number of *roles involved* (M_{16}) the architect can estimate the communication effort and amount of compromise required to take a given decision. This becomes especially important in projects having a complicated stakeholder structure.

5.2 Prioritize Alternatives (G_2)

Stakeholder: Head Architect

As the architect ponders each design issue, metrics can simplify the task of comparing the value of various alternatives being studied. Clearly selecting a final alternative in the design requires the input of the architect, but the process of pruning unwanted alternatives and the corresponding iterative refinement of the design can be supported by the following questions.

Questions:

1. When was an alternative created? Long-living knowledge artifacts are likely to remain stable in the future and therefore might be preferred by conservative architects taking the long-term perspective. Long-lived alternatives can be detected with the *age* (M_{23}) metric.
2. How many times was this alternative given a positive decision? The fact that a given alternative was often selected in a given past timeframe or in a set of interesting projects could help to make a strong argument for selecting it again. Comparing the *positive decisions* (M_9) discounted by the *negative decisions* (M_{10}) that have been attached to the alternatives in the context of a particular issue can provide a valuable insight in case multiple architects are involved in a collaborative design scenario.
3. How often was an alternative reused? The fact that some alternative was considered relevant for multiple projects is already a good recommendation for it. This can be measured by counting the number of *project references* (M_{17}).
4. How many times was this alternative changed and when did the last change happen? On the one hand, a very high number of *changes* (M_{20}) can be a sign of suspect volatility. On the other hand no changes at all in combination with a low number of *positive decisions* (M_9) and a long time since the *last change* (M_{24}) may indicate that the particular alternative has been abandoned.
5. When was a new *Solved by* relation created to this alternative? Knowing the last time when there was a new reference created between a given alternative and an issue is an interesting measure of reuse. Finding out that there are recent new references might indicate that a particular concept proves to be reusable in many situations. The time since the alternative was referenced can be obtained with the *last referenced* (M_{25}) metric.
6. How many times was this alternative viewed and by how many people? Alternatives with high counts of *visits* (M_{19}) and *visitors* (M_{21}) are likely to be candidates worthy of consideration. This metric cannot be used in isolation, but should be considered in combination with the those mentioned before.

5.3 Ensure design consistency (G_3)

Stakeholder: Head Architect.

Consistency is a crucial property of the architectural design because the implementation of inconsistent designs is not feasible. A decision model can be called fully consistent if all possible decisions were made and there are no conflicting decisions, that is decisions made about all alternatives (within the scope of a project) are of the same type. A decision model which contains conflicting decisions is inconsistent. A model that contains no conflicting decisions, but is incomplete (not all decisions are taken) is only partially consistent, given by the proportion of the number of taken decisions.

Questions:

1. Does the decision model contain conflicting decisions? That is, have all alternatives been tagged with decision tags of the same kind (either Positive or Negative)? The presence of conflicts is indicated by looking at the number of alternatives for which there are conflicting decisions (*collision* (M_{11}) metric).
2. How complete is the decision model within modeled project space? What is the percentage of decisions which have been taken in relation to all decisions in given project space? Under the assumption that the project space is modeled completely, this can be measured as the

$$\textit{completeness} = 1 - \frac{\textit{no decisions}(M_{10})}{\textit{total number of alternatives}(M_{13})}$$

5.4 Monitor design progress (G_4)

Stakeholders: Head Architect, Project Manager

Tracking the design progress is essential for project management. An approximate estimate of the design progress can be drawn by observing the dynamics of the decision model and answering the following
Questions:

1. Is the design progressing? The progress can be seen by measuring the rate of change of the previously defined *completeness* of the decision model as well as the time since the *last decision* (M_{26}) was made.
2. Is the design changing? Counting the number of changed decisions within a given timeframe gives an idea of the rate of change of the design. For example, decisions may be changed from *negative* to *positive* and vice versa. The metric of *total number of decisions* (M_9) can also be sampled over time and should be interpreted in relation to the *total number of alternatives* (M_{13}).

5.5 Assess knowledge quality (G_5)

Stakeholder: Analyst, Scribe, Head Architect.

We expect that the quality of the domain knowledge contained in a decision model has a direct influence on the final quality of the design. In case of large-scale projects it is often difficult or practically impossible to manually inspect the background information based on which all design decisions have been made. Having the means to estimate the quality of the domain knowledge gives architects and analysts the opportunity to increase their awareness of the quality of the knowledge they are relying upon during their decision-making. It also helps to direct the efforts of the scribes towards the knowledge items which need rework in order to improve their quality. We do not claim that it is possible to specify an absolute measure for assessing the quality of the knowledge contained in a decision model. Still, an automated measurement mechanism should help to identify outliers so that the design team can focus on them.
Questions:

1. How exhaustive is the description of the knowledge? By localizing which spots in the design space have a low *descriptiveness* (M_{18}), scribes and other design team members can decide where to direct additional documentation and knowledge-gathering effort.
2. How complex are relations within a given scope of the knowledge? Measuring *complexity* (M_5) and observing its distribution within a given scope, designers can identify outliers and irregularities. Having done that, they can decide to refactor the knowledge by decomposing design issues that are too complex into smaller ones. Likewise, they may refine underdeveloped and disconnected artifacts by adding missing relationships to them.
3. Is a given knowledge item well categorized? In order to be accessible and discoverable, the knowledge needs to be properly classified so that designers can locate relevant information by browsing through various cross-cutting categories and tags associated with them. Rich and meaningful *classification* (M_{14}) eases the identification of the issues representing important design concerns and can help to refine text-based search results.
4. How many designers authored a given item? A number of *authors* (M_{22}) that were involved in editing a given knowledge item can indicate the amount of accumulated experience of its multiple authors, who – like in most collaborative wiki-like knowledge management systems – had a chance of correcting, reviewing and building upon each other's content. Like in the previous goal (G_2 - Prioritizing Alternatives), this metric also cannot be used in isolation but should be combined with others. For example, items with a high number of *changes* (M_{20}) can be volatile or can have reached a good level of refinement depending on the item's *age* (M_{23}) and *time since last change* (M_{24}).

6 Metrics

6.1 Structural metrics

Structural metrics are defined over the topology of the graph established by the relationships among the knowledge items.

6.1.1 Dependency (M_1, M_2) and influence (M_3, M_4)

Domain: Issues, Alternatives, *Scale:* Ratio, *Range*: N

The *depends* relationship between artifacts is used to represent the fact that the choice of one alternative implies the choice of the other alternative. The *influence* relation on the other hand can appear between any combination of issues and alternatives. The following four metrics are used to gauge the cohesion between knowledge items established by the dependency and influence graph. The in-degree (number of incoming relations) shows how strongly a given artifact depends on or is influenced by others. Measuring the out-degree (number of outgoing relations) shows how many other artifacts are affected by it.

$$M_1(a) := |\ \textit{depends}\text{ relations coming into } a\ |$$
$$M_2(a) := |\ \textit{depends}\text{ relations outgoing from } a\ |$$
$$M_3(i) := |\ \textit{influences}\text{ relations coming into } i\ |$$
$$M_4(i) := |\ \textit{influences}\text{ relations outgoing from } i\ |$$

When measuring the dependency or influence degree of multiple items within a scope, only the relations to (and from) items outside of the scope are counted.

6.1.2 Complexity (M_5)

Domain: Issues, Alternatives, *Scale:* Ratio, *Range*: N

We define *complexity* to measure the entanglement of a particular knowledge item following its direct relationships. In practice, it can be interpreted as an estimator for the amount of effort required to analyze and decide upon a selected knowledge item due to the requirement to understand its relationships. In the context of an issue i related by *SolvedBy* with set of alternatives A_i, we define *complexity* as:

$$M_5(i) := |A_i| + \sum_{a \in A_i}(M_1(a) + M_2(a) + M_3(a) + M_4(a))$$

Likewise, for a design alternative a, being a solution for a set of design issues I_a, we define complexity as:

$$M_5(a) := |I_a| + \sum_{i \in I_a}(M_3(i) + M_4(i))$$

In case of a more elaborate model, this definition would need to be extended for additional relations. For multiple items within a scope, complexity is additive. An example of the complexity of the simple decision model of Figure 3 is given in Table 1.

	a_1	a_2	a_3	a_4	a_5	a_6	i_1	i_2
M_5	1	1	1	2	1	2	5	3

Table 1: Complexity metric values for the decision model of Fig. 3

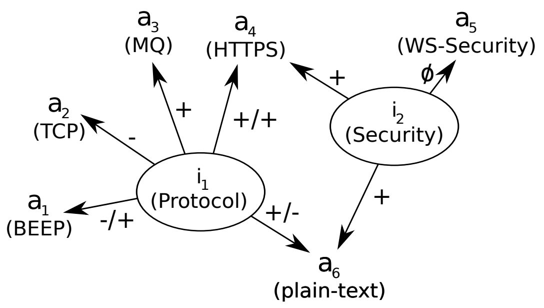


Figure 3: An example of an incomplete and inconsistent decision model over two design issues.

	i_1 Solved by						i_2 Solved by				
	a_1	a_2	a_3	a_4	a_6	i_1	a_4	a_5	a_6	i_2	
M_6	1	0	1	2	1	5	1	0	1	2	M_6
M_7	1	1	0	0	1	3	0	0	0	0	M_7
M_8	0	0	0	0	0	0	0	0	0	0	M_8
M_9	2	1	1	2	2	8	1	0	1	2	M_9
M_{10}	collision	decided	decided	decided	collision	0	decided	no decision	decided	1	M_{10}
M_{11}						2				0	M_{11}
M_{12}						3				2	M_{12}
M_{13}						5				3	M_{13}

Table 2: Fine grained decision metrics for the decision model of Fig. 3

6.2 Decision metrics

The purpose of decision metrics is to determine the consistency and the completeness of a decision model. The main assumption on the structure of the knowledge meta-model (see section 4.1) on which they are defined is that multiple decision tags can be applied to the same alternative (in the context of a given issue). For example, the decision model captures a fine-grained representation of individual decisions made by different stakeholders, which may vote to accept or reject a certain alternative. Thus, we distinguish two categories of metrics: fine-grained metrics, which take into consideration individual decision tags as they are attached to alternatives, and coarse-grained metrics, which aggregate all decisions that have been cast on each alternative solving a given design issue. We also introduce global outcome-oriented metrics, which look at the overall decision state of the various design issues in a given scope/decision model.

6.2.1 *Fine-grained metrics*

Domain: Solved by relations, *Scale:* Ratio, *Range*: N

If we assume that multiple decision tags can be attached to an alternative a in the context of each design issue (which is solved by the alternative in question), it becomes possible to count how many tags of each kind (*positive*, *negative* and *neutral*) there are.

$$\begin{gathered} M_6(a) := |\ \textit{positive}\ (+)\ \text{decisions attached to}\ a\,| \\ M_7(a) := |\ \textit{negative}\ (\text{-})\ \text{decisions attached to}\ a\,| \\ M_8(a) := |\ \textit{neutral}\ (\text{N})\ \text{decisions attached to}\ a\,| \\ M_9(a) := M_6(a) + M_7(a) + M_8(a) \\ \text{(total number of decisions)} \end{gathered}$$

These metrics are additive if they are taken in the scope of multiple alternatives. Examples are given in Table 2.

6.2.2 *Coarse-grained metrics*

Domain: Issues, *Scale:* Ratio, *Range*: N

In the context of a particular design issue i with set of related alternatives A_i, we distinguish between the following decision states over an alternative $a \in A_i$ (see Figure 3). The set of decisions contributing to the state of alternative a is defined as $D(a)$.

no decisions when no decision tags are attached (for example $I_2 \rightarrow A_5$),

$$\text{no decision}(a) \Leftrightarrow D(a) = \emptyset$$

collision when there are multiple conflicting (mixed *positive* and *negative*) decision tags (for example $I_1 \rightarrow A_1$ and $I_1 \rightarrow A_6$).

$$\text{collision}(a) \Leftrightarrow \exists d_j, d_k \in D(a) : d_j \neq d_k$$

decided when there is at least one decision tag, and all are of the same kind (either *positive* or *negative*; *neutral* tags are ignored). For example: $I_1 \rightarrow A_2$, $I_1 \rightarrow A_3$ and $I_1 \rightarrow A_4$.

$$\text{decided}(a) \Leftrightarrow \forall d_j, d_k \in D(a), j \neq k : d_i = d_j$$

These lead to the definition of the following metrics:

$$\begin{gathered} M_{10}(i) := |a \in A_i : \textit{no decision}(a)\,| \\ M_{11}(i) := |a \in A_i : \textit{collision}(a)\,| \\ M_{12}(i) := |a \in A_i : \textit{decided}(a)\,| \end{gathered}$$

Given that alternatives may not be found in any other state, it is possible to count the *total number of alternatives* (M_{13}) for a given issue i as:

$$M_{13}(i) := M_{10}(i) + M_{11}(i) + M_{12}(i)$$

6.3 Tag-based metrics

Design space exploration in search for relevant issues and alternatives can be very challenging. Assuming that tags can be used to classify knowledge items and thus ease the identification of relevant knowledge items, it becomes possible to measure how well a given item is classified, for which stakeholder it is relevant and within how many project decision models it has been reused.

6.3.1 *Classification* (M_{14})

Domain: Issues, Alternatives, *Scale*: Ratio, *Range*: N

The *classification* metric evaluates the richness of the classification associated with particular knowledge items i.

$$M_{14}(i) := |\,\textit{unique tags}\ \text{on}\ i\,|$$

In case of multiple knowledge items being measured, the metric counts the number of unique tags for all the items in the scope under consideration.

6.3.2 *Collaboration metrics* (M_{15},M_{16})

Domain: Issues, *Scale:* Ratio, *Range*: N

Learning how many individuals and how many roles are involved in the decision-making over a design issue i is important to estimate the implied collaboration and communication overhead. For example, decisions that require an agreement to be reached between multiple roles may involve formal design meetings and thus require more effort.

$$\begin{gathered} M_{15}(i) := |\ \textit{individuals}\ \text{responsible for}\ i\ | \\ M_{16}(i) := |\ \text{unique}\ \textit{role}\ \text{tags applied to}\ i\ | \end{gathered}$$

For example, an issue which needs to be decided by two developers A and B together with analyst C will result in 2 involved *roles* and 3 responsible *individuals.*

6.3.3 Reuse metric (M_{17})

Domain: Issues, Alternatives, *Scale*: Ratio, *Range*: N

The *project references* (M_{17}) metric is defined as the number of projects in which a given knowledge item i has been used.

$$M_{17}(i) := | \textit{ project} \text{ tags applied to } i \ |$$

When used to measure set of knowledge items, it only counts the number of unique projects referenced by all of the items in the set.

6.4 Content-based metrics

6.4.1 Descriptiveness (M_{18})

Domain: Issues, Alternatives, Decisions, *Scale*: Ratio, *Range*: $[0, 1]$

We assume that each knowledge item i is described by its attributes to convey relevant information about the concept it represents. For the purpose of learning how complete this description is, we propose a metric to measure the *descriptiveness* of knowledge items:

$$M_{18}(i) := \frac{|\textit{non-empty attributes of } i|}{|\textit{total attributes of } i|}$$

For example, an artifact that has no information in its attributes has a descriptiveness level of 0. On the other hand, artifacts where all attributes are non-empty has a *descriptiveness* of 1. The descriptiveness measured for a decision can be seen as the degree of *justification* measuring whether an explicit rationale for the decision is given (1) or is missing (0).

The aggregated value of *descriptiveness* over a scope is calculated as the average of the *descriptiveness* for each individual item in the scope.

6.5 Access Metrics

Domain: Issues, Alternatives *Scale*: Ratio, *Range*: N

Assuming that it is possible to log all accesses (i.e., read and write operations) to a decision model's knowledge items, measuring the access frequency and the change intensity can provide additional insight. To do so, for any knowledge item i, we introduce the *visits* (M_{19}), *changes* (M_{20}), *visitors* (M_{21}) and *authors* (M_{22}) metrics:

$$M_{19}(i) := |\text{read-only accesses to } i\,|$$
$$M_{20}(i) := |\text{write accesses to } i\,|$$
$$M_{21}(i) := |\text{users reading } i\,|$$
$$M_{22}(i) := |\text{users editing } i\,|$$

When measured over a scope of knowledge items, the first two metrics (*visits* and *changes*) sum the access counters for individual items, while the *visitors* and *authors* metrics count the unique users.

6.6 Dynamics Metrics

Domain: Issues, Alternatives *Scale*: Ratio, *Range*: *time*

In order to observe the dynamics of a decision model, we observe the access and modification time-stamps of particular knowledge items. For that purpose we propose four dynamics metrics: the first measures the *age* (M_{23}) of a knowledge item, while the other three measure the time since the last *change* (M_{24}), *reference* (M_{25}) and *usage* (M_{26}).

$$M_{23}(i) := t_{now} - time(\text{creation of } i)$$
$$M_{24}(i) := t_{now} - time(\text{last attribute change of } i)$$
$$M_{25}(i) := t_{now} - time(\text{last reference to } i)$$
$$M_{26}(i) := t_{now} - time(\text{last decision made over } i)$$

Goal	Metric
G$_1$. Prioritize Decisions	M_1 *dependency in-degree*
	M_2. *dependency out-degree*
	M_3. *influence in-degree*
	M_4. *influence out-degree*
	M_5. *complexity*
	M_{15}. *individuals involved*
	M_{16}. *roles involved*
G$_2$. Prioritize Alternatives	M_9. *positive decisions*
	M_{10}. *negative decisions*
	M_{17}. *project references*
	M_{19}. *visits*
	M_{20}. *changes*
	M_{21}. *visitors*
	M_{23}. *age*
	M_{24}. *time since last change*
	M_{25}. *last referenced*
G_3. Ensure Design Consistency	M_{10}. *no decisions*
	M_{11}. *collision*
	M_{13}. *total number of alternatives*
G_4. Monitor Design Progress	M_9. *total number of decisions*
	M_{10}. *no decisions*
	M_{13}. *total number of alternatives*
	M_{26}. *last decision*
G_5. Assess Knowledge Quality	M_5. *complexity*
	M_{14}. *classification*
	M_{18}. *descriptiveness*
	M_{20}. *changes*
	M_{22}. *authors*

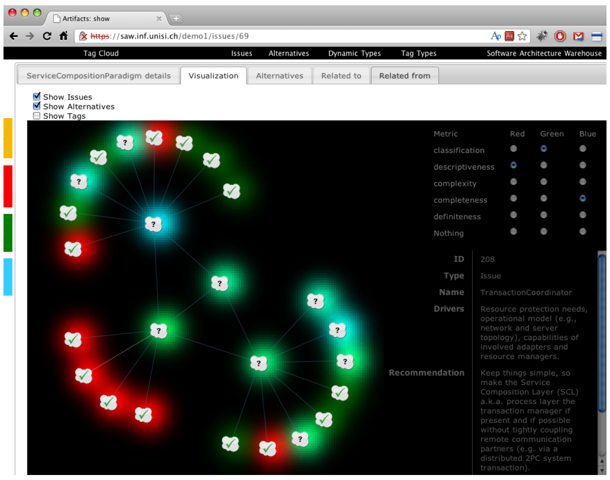
Table 3: Summary overview of the Goals with the corresponding Metrics

Aggregated metrics compute the age of the most recent time stamp over the items in the given scope.

7 Conclusion

In this paper we have presented an initial sketch of a novel set of metrics (summarized in Table 3) for observing and analyzing architectural decision models. The metrics support the goals of the main stakeholders of a software design project and enable them to quantitatively and qualitatively measure the properties of the knowledge that has been captured in the design space and reused across multiple decision models. Metrics enable architects not only to assess the complexity, the quality and the descriptiveness of the knowledge, but also to prioritize their decision-making according to various metrics. We also propose to use metrics for closely following the dynamics of the software design process, estimating and tracking the degree of completeness of a design, as well as detecting inconsistencies as they are introduced.

Due to space limitations we included a theoretical definition of the metrics and did not present here a complete evaluation of their practical usefulness. We have been implementing them as part of the Software Architecture Warehouse (SAW) project, which is a Web-based interactive design support tool for tracking decisions made by architects and managing architectural knowledge. The tool supports knowledge capture, analysis, and sharing among distributed design teams. It has been recently extended with an interactive 2D visualization of decision models (Figure 4) based



Metric	Red	Green	Blue
classification	●	◉	●
descriptiveness	◉	●	●
complexity	●	●	●
completeness	●	●	◉
definiteness	●	●	●
Nothing	●	●	●

ID	208
Type	Issue
Name	TransactionCoordinator
Drivers	Resource protection needs, operational model (e.g., network and server topology), capabilities of involved adapters and resource managers.
Recommendation	Keep things simple, so make the Service Composition Layer (SCL) a.k.a. process layer the transaction manager if present and if possible without tightly coupling remote communication partners (e.g. via a distributed 2PC system transaction).

Figure 4: Screenshot of the Software Architecture Warehouse tool in metrics visualization mode

on the metrics defined in this paper.

We will further continue this research with an analytic specification of the metrics and pursue an empirical evaluation of their usefulness in practice. Once the basic set of metrics has stabilized we will combine them to build detection strategies helping to provide feedback and guidance to architects as they make progress with their design. We plan to use the metrics to compare different existing architectural knowledge repositories in order to extract the differences between various knowledge domains.

Acknowledgements

This work is partially supported by the Swiss National Science Foundation with the CLAVOS project (Grant Nr. 125337).

8 References

[1] T. Al-Naeem, I. Gorton, M. A. Babar, F. Rabhi, and B. Benatallah. A quality-driven systematic approach for architecting distributed software applications. In *Proc. of the 27th International Conference on Software Engineering*, pages 244–253, 2005.

[2] A. Aurum, J. Ross, W. Claes, and M. Handzic. *Managing Software Engineering Knowledge*. Springer, 2003.

[3] V. Basili, G. Caldiera, and D. H. Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.

[4] V. R. Basili. Software modeling and measurement: the goal/question/metric paradigm. Technical Report TR-92-96, University of Maryland at College Park, College Park, MD, USA, 1992.

[5] R. Capilla, F. Nava, and J. C. Duenas. Modeling and documenting the evolution of architectural design decisions. In *Proc. of the 2nd SHARK Workshop*, 2007.

[6] R. Capilla, F. Nava, S. Pérez, and n. Juan C. Due˙A web-based tool for managing architectural design decisions. *SIGSOFT Softw. Eng. Notes*, 31(5):4, 2006.

[7] P. Clements, R. Kazman, and M. Klein. *Evaluating Software Architectures: Methods and Case Studies.* Addison-Wesley, 2001.

[8] P. C. Clements, J. D. McGregor, and S. G. Cohen. The structured intuitive model for product line economics (SIMPLE). Technical Report CMU/SEI-2005-TR-003, Software Engineering Institute, CMU, 2005.

[9] J. Conklin. *Dialogue Mapping*. Wiley, 2006.

[10] L. Dobrica and E. Niemelä. A survey on software architecture analysis methods. *IEEE Trans. Software Eng.*, 28(7):638–653, 2002.

[11] P. Eeles and P. Cripps. *The Process of Software Architecting*. Pearson, 2009.

[12] R. B. Grady. Successfully applying software metrics. *IEEE Computer*, 27:18–25, 1994.

[13] A. Jansen and J. Bosch. Software architecture as a set of architectural design decisions. In *Proc. of the 5th Working IEEE/IFIP Conference on Software Architecture WICSA '05*, 2005.

[14] R. Kazman, L. Bass, M. Webb, and G. Abowd. SAAM: a method for analyzing the properties of software architectures. In *ICSE '94: Proc. of the 16th international conference on Software engineering*, pages 81–90, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[15] R. Kazman, M. Klein, and P. Clements. ATAM: Method for architecture evaluation. Technical Report CMU/SEI-2000-TR-004, August 2000.

[16] A. MacLean, R. M. Young, V. M. E. Bellotti, and T. P. Moran. Design rationale. chapter Questions, options, and criteria: elements of design space analysis. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1991.

[17] M. E. Manso, M. Genero, and M. Piattini. No-redundant metrics for uml class diagram structural complexity. In *Proc. of 15th International Conference on Advanced Information Systems Engineering*, 2003.

[18] S. Morasca. Refining the axiomatic definition of internal software attributes. In *Proc. of the Second International Symposium on Empirical Software Engineering and Measurement, ESEM 2008*, pages 188–197, 2008.

[19] M. Nowak, C. Pautasso, and O. Zimmerman. Architectural decision modeling with reuse: Challenges and opportunities. In *Proc. of the 5th Workshop on Sharing and reusing architectural knowledge SHARK '10*, 2010.

[20] R. E. Park, W. B. Goethert, and W. A. Flora. Goal-driven software measurement – a guidebook. Technical report, Software Engineering Institute – Carnegie Mellon University, 1996.

[21] D. F. Rico and R. S. Pressman. *ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers*. J. Ross Publishing, Inc., 2004.

[22] K. Schneider. *Experience and Knowledge Management in Software Engineering*. Springer, 2009.

[23] R. V. Solingen and E. Berghout. *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development.* McGraw-Hill Inc., 1999.

[24] S. S. Stevens. On the theory of scales of measurement. *Science*, 103:677–680, 1946.

[25] A. Tang, Y. Jin, and J. Han. A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software*, 80(6):918–934, 2007.

[26] C. P. Team. CMMI for development, version 1.2. Technical report, Software Engineering Institute, 2006.

[27] H. van Vliet. Software architecture knowledge management. In *19th Australian Software Engineering Conference (ASWEC 2008)*, pages 24–31, 2008.

[28] R. Wojcik, F. Bachmann, L. Bass, P. Clements, P. Merson, R. Nord, and B. Wood. Attribute-driven design (ADD), version 2.0. Technical report, Software Engineering Institute – Carnegie Mellon University, 2006.

[29] O. Zimmermann. *An Architectural Decision Modeling Framework for Service-Oriented Architecture Design*. PhD thesis, Universität Stuttgart, 2009.

[30] O. Zimmermann and F. Mueller. Web services project roles. `http://www.ibm.com/developerworks/library/ws-roles/`.