Team Situational Awareness and Architectural Decision Making with the Software Architecture Warehouse

Marcin Nowak and Cesare Pautasso

Faculty of Informatics, University of Lugano, Switzerland
marcin.nowak@usi.ch,c.pautasso@ieee.org
http://saw.inf.usi.ch

Abstract. The core of the design of software architecture is all about architectural decision making. A high-quality design outcome sets high requirements, not only on the skills and knowledge of the design team members, but also on the management of the decision making process. We claim that in order to deliver high quality decisions, the design team needs to obtain a high level of situational awareness. To address this, we present an analysis of the problem of team situational awareness in design workshops and propose a model on how stakeholder positions help to build consensus within the argumentation viewpoint of architectural decisions. We show how the Software Architecture Warehouse tool has been extended to support the argumentation viewpoint within its live design document metaphor to provide support for co-located and distributed design workshops.

1 Introduction

As a result of the trend of globalization in the software industry, remote collaboration and decision making within distributed teams is growing in importance. Due to the complex nature of software systems, the design of software architecture holds many qualities specific to so-called wicked problems [6, 17] and often cannot be addressed with simple goal-driven optimization methods [7]. To address these problems, the discipline of Software Architecture Knowledge Management (SAKM [3]) was born and a number of systems specialized in architectural decisions management have been proposed (PAKME [4], ADDSS [5], ArchDesigner [2], (e)AREL [20], AD$_{kwik}$ [23]). Whereas a subset of these tools explicitly targets the collaboration needs of distributed or co-located design teams, only limited support is offered for raising the level of situational awareness in the context of design workshops.

This paper makes the following contributions: it proposes a novel argumentation viewpoint for capturing architectural knowledge, in which the positions of multiple stakeholders and design team members can be captured. The positions follow a well defined life cycle and their state can be aggregated to 1) determine the level of consensus around each design issue 2) track the progress of the design workshop; and 3) facilitate capturing the rationale of each decision made by the team. The argumentation view has been implemented as part of the Software Architecture Warehouse, a collaborative design tool based on the live design document metaphor. Thanks to its real-time synchronization of design spaces, it provides an additional, complementary communication channel that in our experience can enhance the efficiency of both the open, divergent and the closed, convergent phases of design discussions [10].

This paper is structured as follows: in next Section we present an a brief relation over background and related work about collaborative architectural decision management; then we delimit the scope of the problem addressed by our research. In Section three we introduce the concept of argumentation viewpoint. Section four positions situational awareness in the context of architectural decision making. The Software Architecture Warehouse - the collaborative design tool implementing the argumentation viewpoint – is presented in Section five. In Section six we present our preliminary evaluation results and wrap up with conclusions in Section seven.

2 Related Work and Background

The decision making process [13], and in particular the software architectural decision making process have been a subject of many studies [9]. The topic of collaborative design has been less studied and it is only partially supported in the ISO 42010 decision meta-model [1]. Out of the seven architectural decision modeling tools reviewed in [19] only three provide support for collaboration, but none of them is suitable for a low-latency, design workshop environment. Our work is complementary to existing frameworks and meta-models, since it targets dynamic decision making activities within a team.

2.1 The problem of collaborative design

The factors that limit the efficiency of the decision making process within a design team are manifold [10], e.g., the partial overlap of the participants' expertise, the complexity of the domain and the wicked nature of the software architectural design problem [18].

In our experience running design workshops, we have observed that the decision making process can be very chaotic, difficult to control and to organize without a proper reference framework and tool support. A solid framework for organizing the decision making process was proposed in [23, Chapter 7].

Another problem is related to the volatility of the decisions. Systematic recording and documentation of the discussion flow is needed to mitigate decision evaporation. An open challenge for the architectural decision management tools is to capture useful content as much as possible during the workshop without hindering the brainstorming or the decision making activities. The goal is to reduce the cognitive load required to record alternatives and decisions, without the need to resort to dedicated minute takers or scribes.

We also see a big potential in groupware support for creating an environment in which awareness of the design is shared between team members. Due to the inherently limited and partially overlapping expertise of each design team member, in order to achieve high decision quality, the efficient reuse of previous decision experience is essential. In other words, before making design decisions, it is essential to elicit and decide what is to be decided out of the available design space. The elicitation of design issues can be done offline as part of the workshop preparation, but the selection of relevant architecture alternatives sometime can only happen during the live brainstorming.

Another major difficulty in efficiently running architectural design workshops is to keep the focus of the entire design team on the same design issue. In a co-located design workshop, thanks to high bandwidth of face to face communication, depending on the size of the team, this requires some good moderation by the lead architect, but still may be time consuming. Due to the more limited communication bandwidth, in distributed workshops it becomes more challenging to keep the collective attention of all remote participants in focus. This is critical when pruning possible alternatives: as the decision making time grows near, all team members need to be aware of which decision is about to be made.

Another fundamental problem concerns the nature of the architectural design solutions. There are many ways how solution can be unsuitable for the stakeholders. The most critical cases are when solution is either internally inconsistent (decisions contradict each other), or unacceptable (due to violation of constraints). These two cases can be relatively easily eliminated when using a systematic decision making process that includes solution validation activities (see [23]). It is often the case that there are multiple valid, acceptable solutions. In such situation the best solution candidate should be chosen by evaluating its value for the stakeholders. Given that only some of the qualities of the solution are easy to assess quantitatively, this process can be automated (see [7]) only to a certain extent. When the alternative solutions lie on the Pareto frontier, it becomes necessary to trade-off different quality attributes against one another. It is particularly challenging to do so without a high level of situational awareness among the design team.

All in all, out these challenges, in this paper we target the need for 1) preparing the discussion by re-using existing architectural knowledge; 2) focusing the attention of the team; 3) recording the individual position of each stakeholder; 4) making the consensus building process transparent.

2.2 From situational awareness to good decisions

One fundamental assumption that we are making in this paper is that good design decisions are well-informed decisions made by a team with a high level of situational awareness.

Situational awareness is the term used to describe the perception and comprehension of a particular environment, which can vary – as proposed by [8] – across three levels:

- **Perception** (SA_1) – the status, properties, features of relevant elements of the environment are recognized and monitored,
- **Comprehension** (SA_2) – making sense of, recognizing relations, and interpreting the values of the attributes perceived on the previous level (SA_1),
- **Projection** (SA_3) – predictions over the future state of the environment are made based on the knowledge about its current condition (SA_1) and expected dynamics (SA_2).

The original application of the concept of situational awareness was in the applications involving efficient decision making within fast-changing, dynamic environments such as emergency services or battlefield operations. Under such conditions, for the sake of efficiency, decision making is often centralized and authoritative and must happen within strict time limits. Such strategy is often not suitable for the situations when the expertise required to make decisions is distributed among multiple stakeholders.

Although conditions requiring situational awareness of the battlefield are significantly different from the ones within an architecture design workshop, we find a certain number of similarities that lead us to propose to apply the concept of team situational awareness to enhance the efficiency and quality of the collaborative architecture design process. In particular the situational awareness shared among the whole team can help to efficiently argument and build consensus about each decision. A design team sharing a high level of situational awareness can gather relevant information, interpret it from different viewpoints of the involved stakeholders, exchange (well grounded and justified) positions based on assumptions, expectations and predictions over the quality of the resulting architecture, and eventually converge to a single consensus decision.

3 Decision Model and Argumentation Viewpoint

The starting point of our considerations is the decision meta-model proposed in the standard ISO 42010 [1] (see Figure 1). We propose to use the Architectural Decision entity (see Figure 2) to establish a relation between a design issue (representing the problem domain) and multiple design alternatives (from the solution domain). This is similar to what Kruchten proposes in [14] with a relation type named **is an alternative to** which is meant to relate decisions "addressing the same issue". Similarly to [11], we propose to introduce the Design Issue as a first-class entity. An advantage of representing design issues and design alternatives explicitly, is that identification and reuse of design decisions is promoted [15].

The argumentation viewpoint we propose consists of:

Design Issue – A reusable aspect of the system design (from the problem domain) that can can be addressed with one or more design alternative to produce an architectural decision model.

Design Alternative – An action, method, or pattern that can be used to address particular design issues. In some cases, each design alternative can be reused within the context of multiple design decisions.

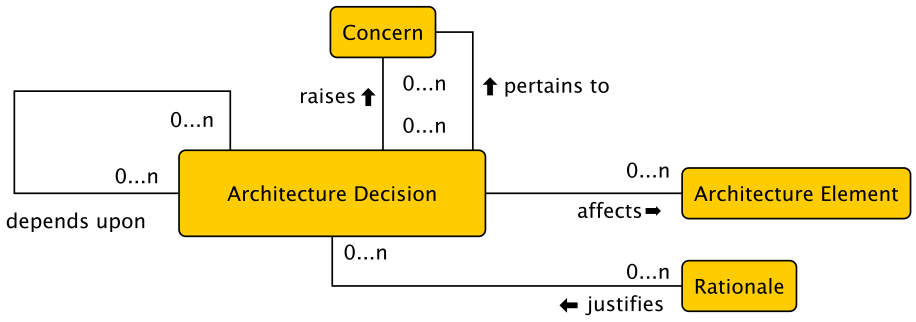


Fig. 1. Elementary architectural decision meta-model after ISO 42010 [1]

Position – A subjective take of a design team member on a design alternative applied in the context of particular design issue. For example, the position can be positive, negative, or neutral. The rationale for the position can also be captured with a natural language description. This can be complemented by a weight associated to the uncertainty or confidence level of the position.

In the simplest case, undecided or open architectural decision would be represented just by the design issue with no alternatives or positions associated to it. Normally, the agenda of a design workshop includes a set of open design issues to be discussed. During the workshop, the design team elicits, generates or captures one or more design alternatives that are related to the design decision under discussion. At this point, positions are used to state the subjective evaluation of each stakeholder or each design workshop participant. Additionally, positions can be justified by relating them to the *decision force* or an *action* (see [22]) that a particular stakeholder recommends to be taken. This provides added value by helping to refine and express the rationale justifying the position. The uncertainty of the position can be explicitly expressed by the stakeholder so that its weight can be taken into account while bringing the decision process to the end. The result is a closed architectural decision which binds the design issue with the chosen alternative.

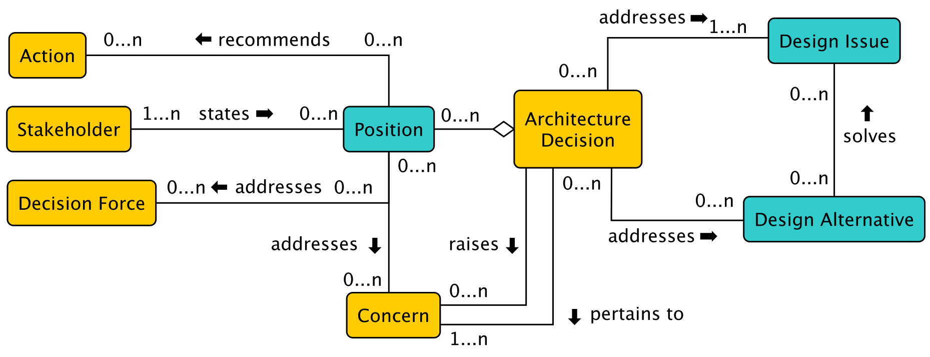


Fig. 2. The argumentation viewpoint meta-model of the architectural decision with Position related to other decision model elements: Action, Stakeholder taken from [21], Decision Force from [22].

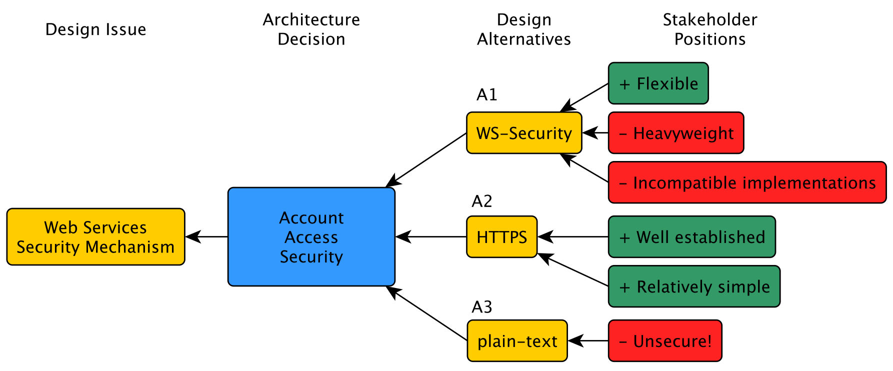


Fig. 3. An example design decision from the service oriented architecture design space together with a number of positions

In Figure 3 we present an example of a design decision from the design space of service oriented architectures. Three design alternatives have been proposed to address the design issue of selecting a Web services security mechanism. Six stakeholders positions have been recorded. Colors and symbols reflect the actions (see [21, Fig. C.3]): green (+) for *validate* and red (-) red for *reject* respectively.

3.1 The life cycle of positions within alternatives

At the beginning of the workshop, each architectural decision starts with no stakeholders' positions recorded (Figure 4). The *aligned* state is reached when all the positions associated to one alternative refer to the same *action*. For example, in Figure 3 all positions related to HTTPS are positive. This can be interpreted as representing the state of consensus among all stakeholders. The *colliding* set of positions exists when positions refer to more than one different *action*. In the example, both positive and negative positions are associated with WS-Security.

In this situation, when stakeholders cannot agree on the action to be taken, the architect leading the workshop can solve the conflict by overriding the conflicting positions expressed by the team members. Thus, after manually naming

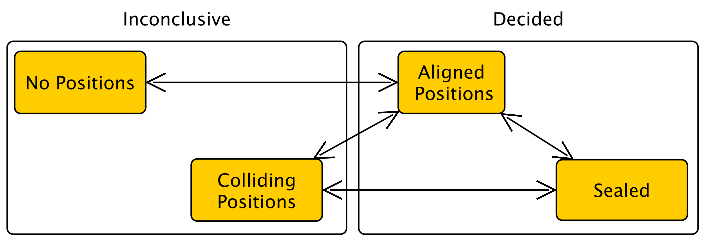


Fig. 4. The state diagram of the life cycle of architectural alternatives. The state of the alternative is aggregated from the actions of its positions.

one action as being the outcome of the discussion, it will proceed to seal the alternative, marking the end of the discussion. The state in which there are either *no positions* recorded, or positions are *colliding* will be referred as *inconclusive*; conversely *aligned* or *sealed* positions will be referred to as *decided*.

3.2 The life cycle of a design decision

The aggregated state of architectural decisions made over the design alternatives within the context of a particular design issue can be conveniently used to monitor the progress of the decision making process (see Figure 5).

Design decisions about a given design issue start their life-cycle with *no alternatives* recorded. As the design progresses, stakeholders elicit (or reuse) one or more relevant design alternative, leaving the design decision in the state with *no decisions*, since no single alternative has yet been selected. Later, stakeholders record their positions and make decisions. In the situation when at least one alternative is in an *inconclusive* state, one can speak about *incomplete* choice. The case when all design alternatives are *decided*, we recognize three types of *complete choice*. To distinguish them we need to check not only whether there is an agreement on the positions about the alternative, but also on whether the agreement is about a positive (i.e., acceptance, validation, approval – see [21]), or negative (i.e., rejection) decision. Rejected alternatives are discarded and based on how many alternatives remain we distinguish: 1) the *conclusive choice* happens when there is exactly one remaining alternative; 2) the *inconclusive choice* happens where there are multiple acceptable alternatives; 3) the *warring choice* represents the case where no alternative is left on the table.

In the example shown in Figure 6, we see a design decision with four alternatives. The first two (BEEP, TCP) have been rejected, while the last two (MQ, HTTPS) have been validated. Therefore the state of decision is inconclusive, since there is more than one alternative left. Assuming that only one alternative is required to settle the issue, another iteration of the discussion will be required to refine the choice among the two remaining alternatives, for example, based on additional constraints given by other design issues, concerns or decision forces.

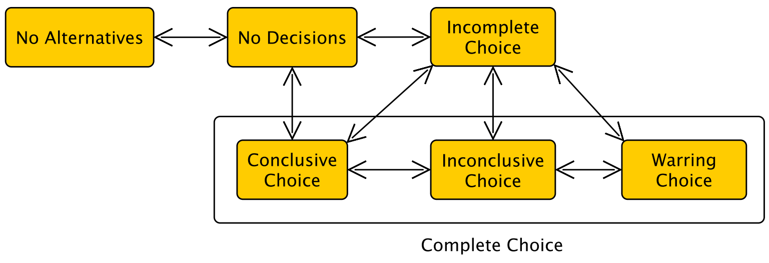


Fig. 5. The state diagram of the life cycle of a design decision. The state of the decision is aggregated from the state of its alternatives.

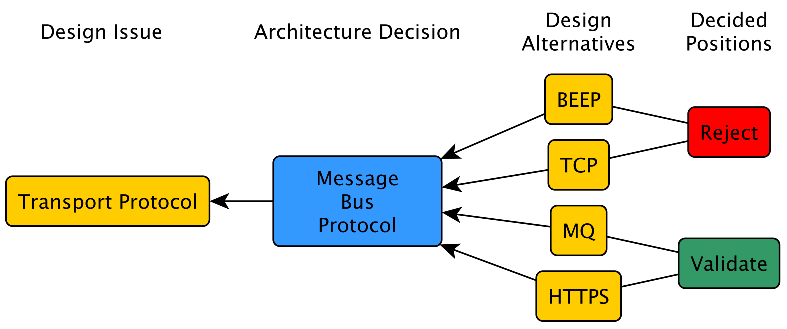


Fig. 6. An example design issue of a transport protocol selection with four design alternatives (protocols) and a complete, inconclusive choice between the alternatives

4 Shared situational awareness of the design space

In the previous sections we have introduced the concept of situational awareness, we scoped the problem of collaborative decision making and finally, we proposed the argumentation viewpoint for modeling architectural decisions. In this section we combine those elements into a mechanism for supporting design teams in the efficient design of software architecture.

First we shall explain that by shared situational awareness, in the context of the architectural decisions, we understand 1) providing decision stakeholders with a customized view over the design space that delivers exactly the information needed for making high quality decisions, 2) providing the means to synchronize the focus of attention of the team, and 3) recording, sharing and analyzing the positions of the design workshop participants. The aim of the first two techniques is to bring the situational awareness of the team to the level of perception (SA_1), whereas the third enables comprehension (SA_2).

Architecture design is a process leading the design team towards the creation of software architecture that has qualities desired by the stakeholders. In general terms, decision making within the design can be principally divided into two modes [12]. The first mode, so-called *open*, happens when the design discussion tends to be divergent and exploratory both in the problem and the solution domains. In this mode, new design alternatives are discovered and new design issues are identified. The *closed* mode, instead, is used to evaluate features and properties of elicited design issues and alternatives. At first, during the fast triage, unfeasible design alternatives are quickly excluded from the scope of the design space. Next, based on the stakeholder's evaluation choices are made within the "closed" project design space. In fact, there is no strict temporal separation between these two modes of operation. Often, transitions between open and closed are needed due to the refinement of the available domain expertise that implies need for readjustment of the choices previously made. In a way, the synergy between open and closed decision making modes is similar to the twin peaks model relating software architecture and requirements engineering [16].

Within these modes we are going to make a distinction between collocated and distributed team configurations. In both setups we assume that all team members have personal computers and network connectivity. In the collocated configuration the team is sharing a meeting room with common facilities such as whiteboard and beamer. In the distributed configuration, additionally we assume that team sites are connected through audio (and video) conferencing systems.

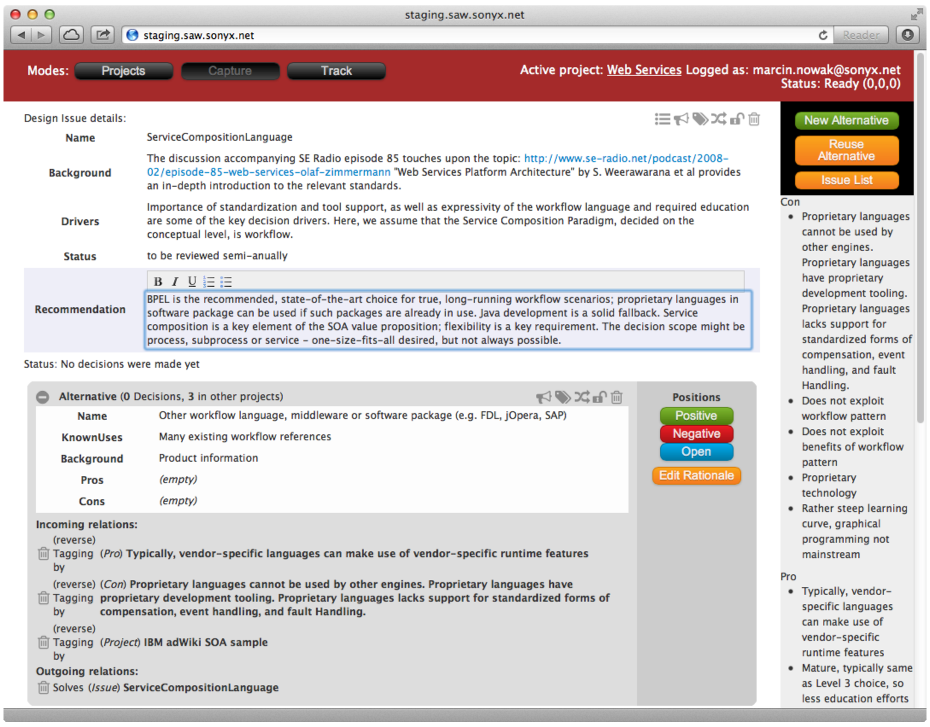
4.1 Open mode, divergent discussion

In the open mode, the design team brainstorms freely over an open design space, creates, edits and destroys design issues, alternatives and decisions. The main needs of the team operating in this mode are related to the efficient capturing of decision model items without getting in the way of the decision making process. The captured information needs to be delivered to the all design team members in a form that stimulates further brainstorming. Not getting in the way is particularly important in the co-located scenario, when the bandwidth of the face to face communication is very high and – for some kind of interactions – a collaboration tool may become an obstacle. The clear benefit of tool-supported collaboration in this phase lies in the efficiency of generating new design alternatives in parallel, since each participant can propose his ideas through the tool interface. The moderator can drive the discussion towards the new contributions in due time, but in any case, the proposed alternatives do not evaporate. Furthermore, thanks to the shared view over the design space and the low-latency with which additions are propagated, everyone on the team is aware about everyone else's contributions (and thus redundant contributions can be culled).

4.2 Closed mode, convergent discussion

In the closed mode, the design team focuses mainly on the evaluation of the elicited design elements in order to find and agreement over a possibly optimal solution. To this end we find it particularly important to build a shared awareness of stakeholders' positions about the design alternatives in question. This can be achieved by sharing each position in real-time within the context of particular design issues and alternatives. For small, co-located teams such awareness can be intuitively built by the design team leader summarizing the current state of the discussion, but in large and/or distributed teams, management of explicit stakeholders' positions is essential.

The efficient capture and reuse of the design discussion comes with the risk of easy derailment in case when it is not moderated appropriately. Discussion moderation of the co-located team can be done by the lead architect by bringing the attention of the group to the particular topic, which can be for example visualized on the beamer. The non-verbal communication bandwidth in the distributed configuration is very limited, so we see a big potential in asynchronous sharing of design space pointers and view references in a manner similar to instant messaging systems. These pointers identify a design issue, alternative or a decision and are particularly useful to bring the attention of the design team to



Active project: Web Services Logged as: marcin.nowak@sonyx.net
Status: Ready (0,0,0)

Design Issue details:

Name	ServiceCompositionLanguage
Background	The discussion accompanying SE Radio episode 85 touches upon the topic: http://www.se-radio.net/podcast/2008-02/episode-85-web-services-olaf-zimmermann "Web Services Platform Architecture" by S. Weerawarana et al provides an in-depth introduction to the relevant standards.
Drivers	Importance of standardization and tool support, as well as expressivity of the workflow language and required education are some of the key decision drivers. Here, we assume that the Service Composition Paradigm, decided on the conceptual level, is workflow.
Status	to be reviewed semi-anually
Recommendation	BPEL is the recommended, state-of-the-art choice for true, long-running workflow scenarios; proprietary languages in software package can be used if such packages are already in use. Java development is a solid fallback. Service composition is a key element of the SOA value proposition; flexibility is a key requirement. The decision scope might be process, subprocess or service - one-size-fits-all desired, but not always possible.

Status: No decisions were made yet

Alternative (0 Decisions, 3 in other projects)

Name	Other workflow language, middleware or software package (e.g. FDL, jOpera, SAP)
KnownUses	Many existing workflow references
Background	Product information
Pros	*(empty)*
Cons	*(empty)*

Positions

Incoming relations:

(reverse) Tagging by (*Pro*) **Typically, vendor-specific languages can make use of vendor-specific runtime features**

(reverse) Tagging by (*Con*) **Proprietary languages cannot be used by other engines. Proprietary languages have proprietary development tooling. Proprietary languages lacks support for standardized forms of compensation, event handling, and fault Handling.**

(reverse) Tagging by (*Project*) **IBM adWiki SOA sample**

Outgoing relations:

Solves (*Issue*) **ServiceCompositionLanguage**

Con
- Proprietary languages cannot be used by other engines. Proprietary languages have proprietary development tooling. Proprietary languages lacks support for standardized forms of compensation, event handling, and fault Handling.
- Does not exploit workflow pattern
- Does not exploit benefits of workflow pattern
- Proprietary technology
- Rather steep learning curve, graphical programming not mainstream

Pro
- Typically, vendor-specific languages can make use of vendor-specific runtime features
- Mature, typically same as Level 3 choice, so less education efforts

Fig. 7. A detailed view of the design issue during the collaborative editing of one of its attributes

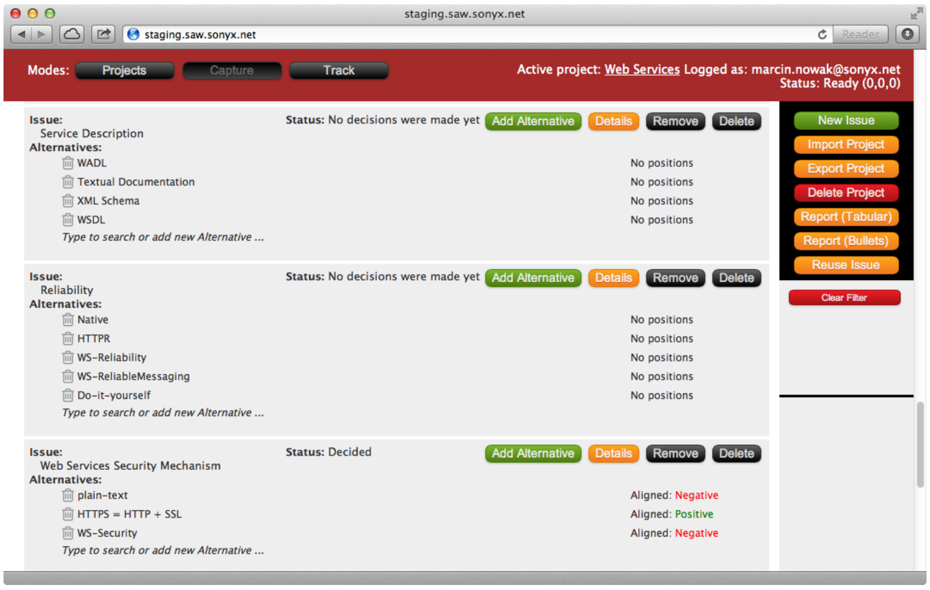
a particular attribute or feature. Being able to efficiently share a precise reference to a view over the design space is very useful to synchronize the focus of attention quickly and then proceed with the decision making to converge.

5 The Software Architecture Warehouse

In this section we present details of the prototypical functionality that we implemented in the Software Architecture Warehouse (SAW) to address the needs and requirements introduced in the previous sections.

5.1 Shared design space awareness

SAW is implemented as a tool to help the entire software architecture design team achieve a high level of situational awareness about architectural decisions and the corresponding design space being traversed during a design workshop. In order to provide designers with an elementary (perception) level of the shared awareness (SA_1) we have introduced the live design document metaphor. Any



Active project: Web Services Logged as: marcin.nowak@sonyx.net
Status: Ready (0,0,0)

Issue: Service Description
Status: No decisions were made yet
Alternatives:
WADL — No positions
Textual Documentation — No positions
XML Schema — No positions
WSDL — No positions
Type to search or add new Alternative ...

Issue: Reliability
Status: No decisions were made yet
Alternatives:
Native — No positions
HTTPR — No positions
WS-Reliability — No positions
WS-ReliableMessaging — No positions
Do-it-yourself — No positions
Type to search or add new Alternative ...

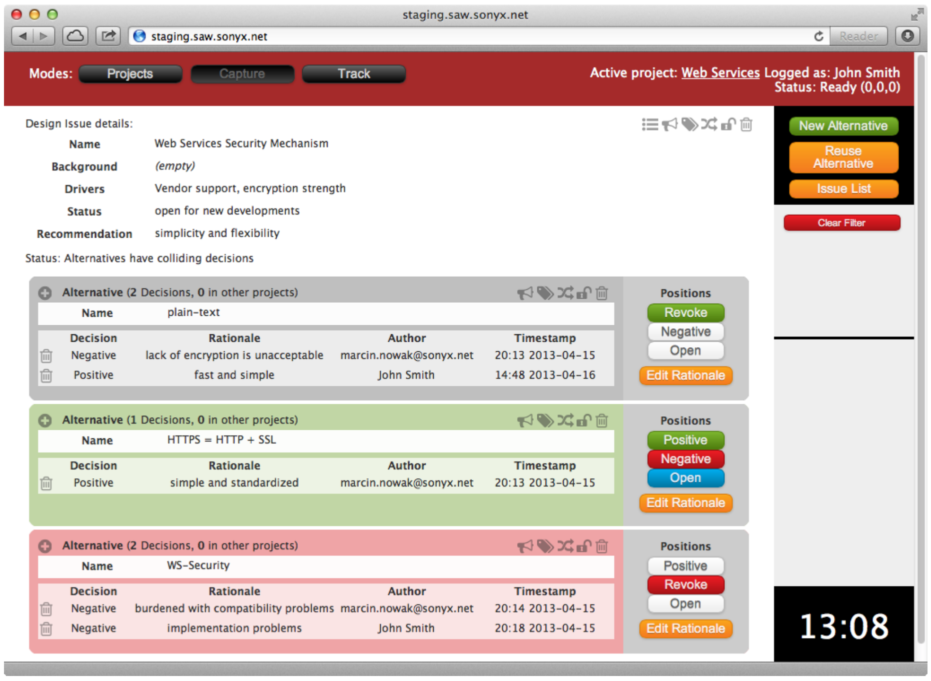
Issue: Web Services Security Mechanism
Status: Decided
Alternatives:
plain-text — Aligned: Negative
HTTPS = HTTP + SSL — Aligned: Positive
WS-Security — Aligned: Negative
Type to search or add new Alternative ...

Fig. 8. A project details view listing referenced design issues together with design alternatives

change to the design elements and relations between them are immediately propagated (with low-latency) to all the design team members participating in the workshop (see Figure 7). Due to the connected nature of the architectural decision representation, the live-document paradigm extends beyond content updates within particular views. To this end, SAW propagates design space alterations to all views. For example alterations made to a design alternative are instantly reflected in the project details and project summary views (see Figure 8). To deal with conflicting edits, SAW follows an optimistic strategy whereby users can see which parts of the document are being edited by others and thus can refrain from entering concurrent modifications. The same mechanism also helps users to see where the attention of other users is being directed.

Additionally, in order to ease interpretation of the decision state, and thus bring users to a higher level of situational awareness (comprehension - SA_2) we have implemented visual aids indicating the state of particular design space elements. For example the decision status of the design issues and alternatives is color-coded so that stakeholders can get an overview at first glance about the level of consensus (see Figure 9). Also in the case of positions, new contributions can be entered in parallel and updates are immediately propagated to all participants.

Targeting the projection level of situational awareness (SA_3), participants may base their positions on the knowledge associated with each design issue alternative (e.g., decision drivers, concerns). Likewise, they may navigate through the design space following arbitrary kinds of relationships (influence) between



Active project: Web Services Logged as: John Smith
Status: Ready (0,0,0)

Design Issue details:

Name	Web Services Security Mechanism
Background	(empty)
Drivers	Vendor support, encryption strength
Status	open for new developments
Recommendation	simplicity and flexibility

Status: Alternatives have colliding decisions

Alternative (2 Decisions, 0 in other projects)

Name plain-text

Decision	Rationale	Author	Timestamp
Negative	lack of encryption is unacceptable	marcin.nowak@sonyx.net	20:13 2013-04-15
Positive	fast and simple	John Smith	14:48 2013-04-16

Positions: Revoke, Negative, Open, Edit Rationale

Alternative (1 Decisions, 0 in other projects)

Name HTTPS = HTTP + SSL

Decision	Rationale	Author	Timestamp
Positive	simple and standardized	marcin.nowak@sonyx.net	20:13 2013-04-15

Positions: Positive, Negative, Open, Edit Rationale

Alternative (2 Decisions, 0 in other projects)

Name WS-Security

Decision	Rationale	Author	Timestamp
Negative	burdened with compatibility problems	marcin.nowak@sonyx.net	20:14 2013-04-15
Negative	implementation problems	John Smith	20:18 2013-04-15

Positions: Positive, Revoke, Open, Edit Rationale

Fig. 9. A view presenting a design issue with three alternatives, two of which have aligned positions (second and third), the other (first) has colliding positions.

issues and/or alternatives. This way, the impact of decisions can be analyzed from a global perspective.

5.2 System architecture

Client-Server split – Traditional Web applications rely on the thin-client paradigm. Over time, many server-side Web-frameworks were conceived to cope with the growing complexity of Web applications (RoR, Django, etc.). The traditional Web applications leave all MVC layers to be handled by the server side, leaving only view rendering for the Web browser. This approach has the advantage of containing all application code within a single location, however it is not suitable to support the live document metaphor. Since every user interaction with the system triggered a call to a rather heavy server-side stack, the result was rather limited scalability. In the process of architecting and implementing SAW we have soon realized that the level of interactivity required to realize the desired liveness of the user experience could not be implemented with the use of traditional server-side frameworks. To this end we have implemented server-side SAW as a thin layer wrapping a NoSQL database. The interactive user-interface is implemented following the MVVC pattern in JavaScript (with Backbone `http://backbonejs.org/` and Marionette `http://marionettejs.com/`).

Node graph observer, notification system – SAW uses the graph paradigm to persist decision models and design spaces. In order to deliver high user awareness over the shared design space, a suitable data replication mechanism is needed. We have implemented a light-weight notification mechanism, which distributes identifiers of the altered graph nodes, so that clients can reload node data if needed. In case when the graph structure changes, by creating or removing edges between nodes, a notification of this event is propagated to the nodes influenced by the change (see Figure 10). The notification system is very general and has also been used to implement the view pointer broadcast feature.

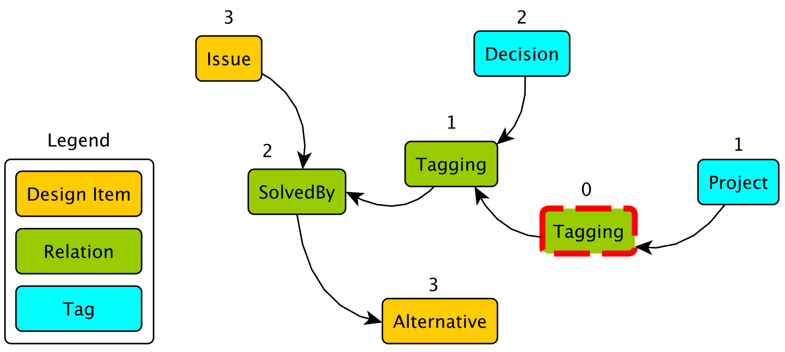


Fig. 10. Event propagation over the shared graph model. Views can subscribe to observe changes within a certain distance of their model elements

6 Formative Evaluation

The concept of team situational awareness and its support within the Software Architecture Warehouse has been validated through three formative evaluation cycles over the past 2 years. Different releases of SAW has been used in more than 50 co-located design workshops, with groups of 5-10 students attending each session. In some cases, the same participants have repeatedly used the tool, and provided us with feedback about its progress, performance and usability. The participants played the role of software architects (including the lead architect), software developers as well as other stakeholders, such as customers or end-users of the systems being designed. SAW has also used in distributed design workshops over conference calls and in hybrid workshops with some co-located participants and others connecting remotely. For space reasons we focus in analyzing the experience we have gathered in the co-located design workshops. The feedback received has helped to refine the concept of team situational awareness and improve the tool usability and scalability.

We have observed that the usage patterns and load may greatly vary in intensity over a design workshop session (in average 2h), making the real-time performance requirement very challenging to achieve without sufficient resources on the server-side and over an unreliable network. We have tested the performance of the system, and there is no noticeable delay of event propagation with up to 20 participants who are collaboratively editing a design space made of up to 100 issues (with 5 alternatives each). The tool is also ready for a Cloud-based deployment and each tier can be separately distributed for additional performance.

Concerning the impact on the cognitive load of the lead architect, we have found out that only users that have accumulated some experience with the tool's user interface can be effective in capturing the discussion while leading it. In other cases we had to resort to recruiting minute takers (or scribes) that would act as a proxy between the lead architect at the whiteboard and the design decisions tracked by the tool and displayed with the beamer. In general, since all participants have the possibility to contribute their input into the shared knowledge repository, over multiple sessions, we observed that it was no longer necessary to employ a single dedicated scribe as this role was spontaneously shared among all participants, after they realized about the presence of the additional communication channel.

The feature to broadcast pointers over the design space was suggested by one user in order to make it efficient to navigate to a specific design view. The user would copy and paste the URI of the page displaying the relevant information and share it with the rest of the participants with an instant messenger. After observing this behavior we decided to implement explicit support for this feature by taking advantage of the existing notification infrastructure. This way we can guarantee that it is very efficient to ensure that all participants are seeing the same view at the same time.

Concerning the tracking of positions within the argumentation view, we experimented with two levels of detail. The initial lightweight solution was a simple positive or negative vote over each alternative. Then we added the ability to retract positions and recast votes, since people needed to be able to change their mind as the consensus building process was taking place. At a more fine-granular level, users can also enter the rationale and confidence level of their position. This required additional time and effort and has been met with some resistance. In particular, not all users can immediately and independently provide a rationale for their position and prefer to wait for others to express their viewpoints and piggy back their position on the previous ones.

Another feature added based on explicit user feedback, was the ability to seal the state of decisions to explicitly mark the conclusion of the discussion over certain issues. This has also been used to track the progress of the workshop. This way the tool can provide a separate list of open issues, which need to be decided upon - this list keeps shrinking during the closed phase of the discussion, providing all participants with a sense of accomplishment, while the list of sealed and decided issues grows.

We have also observed that SAW added an additional communication channel to the discussion, in a way that workshop participants could contribute to the design space without interrupting the ongoing discussion. Similarly, some participants which were intimidated by the lead architect, felt empowered to make their contributions through SAW, silently and in the background. Once discovered by the rest of the team, these contributions have often proven to be highly relevant for the quality of the final design.

In the feedback surveys we conducted, the majority of workshop participants reported that thanks to the possibility to access shared positions of other designers, they felt more confident about the quality of the decisions being made during the design workshops actively supported with SAW.

7 Conclusion

In this paper we have performed an analysis of the problem of collaborative decision making in the context of software architecture design workshops. Based on the idea of enhancing the situational awareness of the whole design team, we have proposed a novel argumentation viewpoint of the standard software architectural decision model and discussed the life cycle of design decisions within the open and closed phases of a lightweight collaborative design process. The concepts presented in this paper are fully implemented by the Software Architecture Warehouse, a prototype architectural decision management tool targeting real-time support for co-located and distributed design teams. Selected aspects of the tool architecture have been discussed together with the promising results of our preliminary evaluation.

Future work aims on developing metrics and detection strategies to raise further the team situational awareness to the projection level (SA_3). In the near future we plan an extensive evaluation with our industry partners for closely studying the dynamics of collaborative design processes within distributed design teams.

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

References

1. ISO/IEC 42010 – Systems and software engineering – architecture description, 2011.
2. 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*, 2005.
3. M. A. Babar, T. Dingsøyr, P. Lago, and H. van Vliet. *Software Architecture Knowledge Management - Theory and Practice.* Springer, 2009.
4. M. A. Babar and I. Gorton. A tool for managing software architecture knowledge. In *Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent, SHARK-ADI '07*, 2007.

5. 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.
6. J. Conklin. *Dialogue Mapping*. Wiley, 2006.
7. T. de Gooijer, A. Jansen, H. Koziolek, and A. Koziolek. An industrial case study of performance and cost design space exploration. In *International Conference on Performance Engineering*, 2012.
8. M. R. Endsley. Theoretical underpinnings of situation awareness: a critical review. In M. R. Endsley and D. J. Garland, editors, *Situation Awareness Analysis and Measurement*, Mahwah, NJ, USA, 2000. Lawrence Erlbaum Associates.
9. D. Falessi, G. Cantone, R. Kazman, and P. Kruchten. Decision-making techniques for software architecture design: a comparative survey. *ACM Computing Surveys*, 43, 2011.
10. R. Y. Hirokawa and M. S. Poole, editors. *Communication and Group Decision Making*. SAGE Publications, Inc, 2nd edition, 1996.
11. A. Jansen and J. Bosch. Software architecture as a set of architectural design decisions. In *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture WICSA '05*, 2005.
12. D. S. Kerr and U. S. Murthy. Divergent and convergent idea generation in teams: A comparison of computer-mediated and face-to-face communication. *Group Decision and Negotiation*, 13:381–399, 2004.
13. G. Klein. *Sources of Power*. MIT Press, 1999.
14. P. Kruchten, P. Lago, and H. van Vliet. Building up and reasoning about architectural knowledge. *Quality of Software Architectures*, pages 43–58, 2006.
15. M. Nowak, C. Pautasso, and O. Zimmerman. Architectural decision modeling with reuse: Challenges and opportunities. In *Proceedings of the 5th Workshop on Sharing and reusing architectural knowledge SHARK '10*, 2010.
16. B. Nuseibeh. Weaving together requirements and architectures. *IEEE Computer*, pages 115–119, 2001.
17. C. Potts and G. Bruns. Recording the reasons for design decisions. In *ICSE*, pages 418–427, 1988.
18. C. Potts and G. Bruns. Recording the reasons for design decisions. In *Software Engineering, 1988., Proceedings of the 10th International Conference on*, pages 418 –427, apr 1988.
19. M. Shahin, P. Liang, and M.-R. Khayyambashi. Architectural design decision: Existing models and tools. In *Joint Working IEEE/IFIP Conference on Software Architecture 2009 and European Conference on Software Architecture 2009, WICSA/ECSA 2009*, pages 293–296, 2009.
20. 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.
21. U. van Heesch, P. Avgeriou, and R. Hilliard. A documentation framework for architecture decisions. *Journal of Systems and Software*, 85(4):795–820, 2012.
22. U. van Heesch, P. Avgeriou, and R. Hilliard. Forces on architecture decisions - a viewpoint. In *Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, WICSA/ECSA, Helsinki, Finland, August 20-24*. IEEE, 2012.
23. O. Zimmermann, J. Koehler, F. Leymann, R. Polley, and N. Schuster. Managing architectural decision models with dependency relations, integrity constraints, and production rules. *Journal of Systems and Software*, 82(8):1249 – 1267, 2009.