Collaborative Software Architecture Decisions: Structure and Dynamics

Doctoral Dissertation submitted to the
Faculty of Informatics of the University of Lugano
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Marcin Aleksander Nowak

under the supervision of
Prof. Dr. Cesare Pautasso

September 2014

Dissertation Committee

Prof. Dr. Mehdi Jazayeri	University of Lugano, Switzerland
Prof. Dr. Michele Lanza	University of Lugano, Switzerland
Prof. Dr. Patricia Lago	VU University Amsterdam, the Netherlands
Prof. Dr. Olaf Zimmermann	University of Applied Sciences of Eastern Switzerland, Rapperswil, Switzerland

Dissertation accepted on 12 September 2014

Research Advisor	PhD Program Director
Prof. Dr. Cesare Pautasso	***Prof. Dr. Igor Pivkin*** **and** ***Prof. Dr. Stefan Wolf***

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Marcin Aleksander Nowak
Lugano, 12 September 2014

Abstract

The complexity of modern computer systems is often comparable to that of biological systems. As much as this complexity can be effectively hidden from the end-user, it is inherently absorbed in the design of the system. Software Architecture is an effective design abstraction that allows designers to divide and conquer the complexity. A modern way of looking at the Software Architecture is to see it as a set of principal design decisions. The design of Software Architecture for large and complex systems often requires expertise exceeding what can be delivered by the individual software architect; therefore, successful design relies on effective collaborative decision making within the design team of diverse domain experts.

We tackle the problem of collaborative decision making in the software architecture design teams by proposing the decision argumentation viewpoint extension to the architecture description standard. Its main purpose is to support fine-grained decision argumentation modeling. Within the viewpoint, we devise the architecture decision consensus lifecycle and a design issue choice state machine that enable precise characterization of the decision state.

Based on the argumentation viewpoint we define an analytical framework designed to estimate the structural and temporal characteristics of decision models. The framework comprises fifteen metrics and offers a comprehensive look into the state and dynamics of the decision making process.

Building upon this foundation, we designed and implemented the Software Architecture Warehouse (SAW) - a tool to assist software architects in collaborative decision making during architecture design workshops. SAW features low-latency, structured architecture decision capturing and decision consensus management.

Furthermore, the Software Architecture Warehouse is accompanied by the implementation of the decision argumentation metrics framework. Finally, we evaluate the framework by applying it on the decision spaces recorded during the master's course on Software Architecture and Design. We conclude with a discussion over the interpretation of differences observed between the workshops assisted by the use of the Software Architecture Warehouse and those supported by EtherPad, an alternative unstructured collaborative editor. In the last words, we address threats to validity, limitations, and we hint at future work.

Kurzfassung

Die Komplexität moderner Computersysteme wird oft mit jener biologischer Systeme verglichen. Auch wenn diese Komplexität erfolgreich vor dem Endbenutzer verborgen werden kann, ist sie inhärent im Systemdesign enthalten. Die Software-Architektur ist eine effiziente Software-Abstraktion, die den Entwicklern ermöglicht, diese Komplexität aufzuspalten und somit zu überwinden. Eine moderne Perspektive auf die Software-Architektur besteht darin, in ihr eine Gruppe von wichtigen Designentscheidungen zu sehen. Die Entwicklung einer Software-Architektur für große und komplexe Systeme erfordert häufig Fachwissen, welches das eines einzelnen Software-Architekten übersteigt, daher beruht ein erfolgreiches Design auf effizienter kollaborativer Entscheidungsfindung innerhalb eines Entwicklungsteams aus Experten für verschiedene Bereiche.

Die vorliegende Arbeit versucht, das Problem der kollaborativen Entscheidungsfindung in einem Software-Entwicklungsteam zu lösen, indem die Entscheidungs-Argumentationsperspektive als Erweiterung zum Architekturbeschreibungsstandard vorgeschlagen wird. Ihr Hauptzweck ist, eine kleinteilige Entscheidungs-Argumentationsmodellierung zu unterstützen. In dieser Perspektive soll der Architekturentscheidungs-Konsens-Zyklus entworfen werden und einen Automaten zum Auswahlstatus von Designfragen, der eine präzise Charakterisierung des Entwicklungsstatus erlaubt.

Auf der Entscheidungs-Argumentationsperspektive basierend wurde ein analytischer Rahmen definiert, um die strukturellen und zeitlichen Charakteristika von Entscheidungsmodellen zu ermitteln. Der Rahmen umfasst 15 Metriken und bietet einen umfassenden Überblick über den Status und die Dynamiken des Entscheidungsprozesses.

Auf dieser Grundlage aufbauend, soll das Software Architecture Warehouse (SAW) entwickelt und implementiert werden – ein Instrument, das Software-Architekten während des Architektur-Workshop in der kollaborativen Entscheidungsfindung helfen soll. Das SAW enthält eine strukturierte Architekturentscheidungsaufnahme und ein strukturiertes Entscheidungskonsens-Management, beide mit einer geringen Latenz.

Desweiteren wird das Software Architecture Warehouse ergänzt um die Implementierung eines Frameworks von Entscheidungs-Argumentations-Metriken.

Schließlich evaluiert die vorliegende Arbeit das Framework durch Anwendung auf die Entscheidungsräume, die während des Master-Kurses über Software-Architektur und Software-Design erfasst wurden. Die Arbeit schließt mit einer Diskussion der beobachteten Unterschiede zwischen der Benutzung des Software Architecture Warehouse und EtherPad, einem alternativen unstrukturierten kollaborativen Editor. Weiter werden im Schlusswort noch Limitierungen des gewählten Ansatzes angesprochen und mögliche zukünftige Forschungsansätze skizziert.

Sommario

La complessità dei moderni sistemi informatici è spesso comparabile con quella dei sistemi biologici. Per quanto questa complessità possa essere efficacemente nascosta all'utente finale, essa è intrinseca nel design del sistema stesso. L'Architettura del Software è un'efficace astrazione a livello di design che permette ai progettisti di suddividere e gestire la complessità.

Oggigiorno, l'Architettura del Software può essere vista come un insieme di decisioni di design. Il design dell'Architettura del Software per sistemi grandi e complessi spesso richiede l'esperienza di più di un singolo architetto del software, infatti, la realizzazione di un design di successo si basa sulla collaborazione nel processo decisionale di un team di designers formato da esperti di diversi settori.

Noi affrontiamo il problema della collaborazione del processo decisionale in un team di designers dell'Architettura del Software proponendo un'estensione al modello che funge da punto di vista argomentativo nel processo decisionale agli standard descrittivi dell'architettura.

Lo scopo principale è quello di supportare la modellazione dettagliata di argomentazioni nelle decisioni. All'interno dell'estensione è elaborato il ciclo di vita delle decisioni a livello architetturale, come anche i problemi legati al design, per caratterizzare in modo preciso lo stato della decisione.

Lavorando sul punto di vista delle argomentazioni, abbiamo definito un framework analitico studiato per stimare le caratteristiche strutturali e temporali dei modelli decisionali. Il framework comprende quindici metriche e offre una comprensiva introspezione nello stato e nelle dinamiche del processo decisionale.

Basandoci su ciò, abbiamo studiato e sviluppato il Software Architecture Warehouse (SAW), uno strumento in grado di assistere gli architetti del software nel processo decisionale collaborativo durante i workshops architetturali. SAW offre un modello per registrare le decisioni architetturali e gestire il processo decisionale in modo strutturato e a bassa latenza.

Software Architecture Warehouse comprende inoltre l'implementazione del framework per le metriche argomentative delle decisioni precedentemente menzionato. Valutiamo infine il framework applicandolo sullo spazio delle decisioni raccolte durante il corso di master di Software Architecture e Design. Concludiamo con un'interpretazione delle differenze osservate fra i workshops che utilizzavano

Software Architecture Warehouse e quelli che invece facevano uso di EtherPad, un sistema non strutturato di collaborazione alternativo. Nell'ultimo paragrafo menzioniamo i punti deboli, limitazioni e diamo suggerimenti su sviluppi futuri.

Streszczenie

Złożoność współczesnych systemów komputerowych jest porównywalna ze złożonością systemów biologicznych. Jest ona często ukryta przed końcowym użytkownikiem poprzez wchłonięcie jej przez architekturę systemu. Architektura oprogramowania jest wydajną abstrakcją projektu informatycznego, która umożliwia projektantom okiełznanie ich złożoności. Współcześnie, architektura oprogramowania jest postrzegana jako zbiór fundamentalnych decyzji projektowych. W związku z tym, że projektowanie złożonych systemów komputerowych często przekracza kompetencje pojedynczego architekta, aby uzyskać wysokiej jakości projekt, niezbędna jest skuteczna współpraca pomiędzy ekspertami z różnych dziedzin przy podejmowaniu decyzji projektowych.

W tej pracy podejmujemy temat grupowych decyzji projektowych w zespołach pracujących nad architekturą oprogramowania. Proponujemy rozszerzenie standardu dokumentacji architektury oprogramowania (ISO 42010) o punkt widzenia skupiający się na szczegółowej argumentacji decyzji projektowych. W ramach tego punktu widzenia definiujemy cykl życia konsensusu dla decyzji i maszynę stanów kwestii projektowych.

W oparciu o ten punkt widzenia zdefiniowaliśmy model analityczny służący do badania struktury i dynamiki modeli decyzji projektowych. Model ten składa się z piętnastu metryk i umożliwia dogłębny wgląd w stan faktyczny i proces podejmowania decyzji.

Na tej podstawie zaprojektowaliśmy i zaimplementowaliśmy Software Architecture Warehouse (SAW) – narzędzie które wspomaga architektów w procesie grupowego podejmowania decyzji projektowych. SAW wspiera szybkie i uporządkowane dokumentowanie decyzji oraz zarządzanie konsensusem.

Dodatkowo SAW jest wyposażony w implementacje wyżej wymienionego modelu analitycznego. Przydatność tegoż modelu demonstrujemy na przykładzie analizy danych zebranych w ramach magisterskiego kursu architektury oprogramowania oferowanego na uniwersytecie w Lugano. W oparciu o zebrane dane i model, przedstawiamy interpretację różnic pomiędzy grupami projektowymi; jedną wspomaganą przy użyciu SAW, a drugą grupą wykorzystującą generyczny grupowy edytor tekstu (EtherPad). Na zakończenie omawiamy ograniczenia i stosowalność naszych wyników, oraz przedstawiamy perspektywy na przyszłość.

Acknowledgments

This work would not be possible without support and encouragement of many people. First, I would like to thank my thesis advisor, Prof. Dr. Cesare Pautasso. Without his support and supervision, this work would not be possible at all. Thanks to his open mind and encouragement, I was able to stay on track through the dire straits of the research process. His meritorious critique and advice assured a high-quality outcome.

I should strongly acknowledge the effort that, my colleague, Vasileios Triglianos has put into proofreading the final draft of the thesis. His fresh eyes have had a significant impact on the editorial quality of this dissertation. I would like to thank the rest of my research group: Dr. Saeed Aghaee, Masiar Babazadeh, Dr. Daniele Bonetta, Vincenzo Ferme, and Dr. Achille Peternier. I would like to thank my thesis committee, Prof. Dr. Olaf Zimmermann, Prof. Dr. Michele Lanza, Prof. Dr. Patricia Lago, and Prof. Dr. Mehdi Jazayeri for their direction and invaluable advice along this project.

I feel obliged to mention Jan Lalek, Andrzej Rybczynski and Adam Matusiak of Lafot. Work that we have done together has been a forming experience that have inspired me to pursue this particular research field. I am also beholden to a number of fabulous managers that I had the pleasure to work with: Dirk Weidemann, Reinhard Koehler, for involving me in super interesting projects and for all the personal development opportunities given to me.

A significant amount of development work on the Software Architecture Warehouse has been contributed by the master students of the Faculty of the Informatics. Your work is greatly appreciated. I am confident that during this work we all have learned an important lesson about web application development. Thank you: Ievgenii Riabokon, Masiar Babazadeh, Adnan Alhariri, Mark Pruneri, Omar Elabed, and Alessandro Andreani.

I owe a big thank you to Dr. Cherilyn Keall for her invaluable advice on style and language, which gave a finishing touch to this dissertation.

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

Danksagung

Diese Dissertation wäre ohne die Unterstützung und den Zuspruch vieler Menschen nicht möglich gewesen. Darum möchte ich die Gelegenheit ergreifen, mich an dieser Stelle bei ihnen zu bedanken. An erster Stelle danke ich meinem Betreuer Cesare Pautasso. Ohne seine Unterstützung und seinen Rat wäre diese Arbeit sicherlich nicht möglich gewesen. Seine Aufgeschlossenheit und Ermutigungen haben mir geholfen mich nicht im Forschungsprozess zu verlieren. Seine konstruktive Kritik und seine produktiven Ratschläge haben für die hohe Qualität des endgültigen Resultats gesorgt.

Mein besonderer Dank gilt darüber hinaus meinem Kollegen Vasileios Triglianos, der viel Zeit und Mühe in das Korrekturlesen der endgültigen Version investiert hat. Seiner Aufmerksamkeit ist nichts entgangen und seine Ratschläge haben die Lesequalität dieser Dissertation deutlich verbessert. Ich möchte ebenfalls den übrigen Mitgliedern meiner Forschungsgruppe danken: Dr. Saed Aghaee, Masiar Babazadeh, Dr. Daniele Bonetta, Vincenzo Ferme und Dr. Achille Peternier. Desweiteren möchte ich den Prüfern, Prof. Dr. Olaf Zimmermann, Prof. Dr. Michele Lanza, Prof. Dr. Patricia Lago, und Prof. Dr. Mehdi Jazayeri für ihre Anleitung und wertvollen Ratschläge danken.

Ebenso zu Dank verpflichtet bin ich Jan Lalek, Andrzej Rybczynski und Adam Matusiak. Unsere Zusammenarbeit war für mich eine wichtige Erfahrung, die mich dazu inspiriert hat, weiterhin in diesem Feld zu forschen.

Mein Dank geht auch an eine Reihe von ausserordentlichen Managern, mit denen ich das Vergnügen hatte zusammenzuarbeiten, vor allem an Dirk Weidemann und an Reinhard Koehler dafür, dass sie mich in sehr interessante Projekte einbezogen haben und mir viele Möglichkeiten zur persönlichen Entwicklung geboten haben.

Einen bedeutenden Beitrag zur Entwicklungsarbeit des Software Architecture Warehouse haben die Masterstudenten der Fakultät für Informatik geleistet. Ich bin davon überzeugt, dass wir alle während dieser Arbeit viel über die Entwicklung von Web-Anwendungen gelernt haben. Vielen Dank an: Jewgeni Riabokon, Masiar Babazadeh, Adnan Alhariri, Mark Pruneri, Omar Elabed und Alessandro Andreani.

Schliesslich danke ich noch Dr. Cherilyn Keall für ihre wertvollen Ratschläge in Bezug auf Sprache und Stil, die dieser Dissertation den letzten Schliff gegeben haben.

Diese Arbeit wurde teilweise vom Schweizerischen Nationalfonds (SNF) im Rahmen des CLAVOS-Projekts gefördert (Stipendium Nr. 125337).

Ringraziamenti

Questo lavoro non sarebbe stato possibile senza il supporto e l'incoraggiamento di molte persone.

In primis, vorrei ringraziare il mio supervisore di tesi Prof. Dr. Cesare Pautasso. Senza il suo supporto e la sua supervisione questo lavoro non sarebbe stato possibile. Grazie alla sua apertura mentale ed al suo incoraggiamento, sono stato in grado di rimanere in pista affrontando le difficoltà del processo di ricerca. Le sue critiche ed i suoi consigli hanno assicurato l'alta qualità del risultato.

Vorrei ringraziare in modo particolare Vasileios Triglianos per il suo sforzo nella lettura della bozza finale della tesi. Il suo aiuto ha avuto un impatto rilevante sulla qualità editoriale del manoscritto. Vorrei anche ringraziare il resto del mio gruppo di ricerca: dr. Saeed Aghaee, Masiar Babazadeh, dr. Daniele Bonetta, Vincenzo Ferme, dr. Achille Peternier. Vorrei ringraziare il mio comitato tesi, Prof. Dr. Olaf Zimmermann, Prof. Dr. Michele Lanza, Prof. Dr. Patricia Lago, e il Prof. Dr. Mehdi Jazayeri per la loro direzione e preziosi consigli per questo progetto.

Sento il dovere di menzionare Jan Lalek, Andrzej Rybczynski e Adam Matusiak della Lafot. Il lavoro svolto insieme è stato un'esperienza formativa che mi ha ispirato a proseguire in questo particolare campo di ricerca.

Sono anche grato a una serie di favolosi manager con cui ho avuto il piacere di lavorare: Dirk Weidemann e Reinhard Koehler, per avermi coinvolto in progetti molto interessanti e per tutte le opportunità di sviluppo personale che mi hanno offerto.

Una notevole quantità di lavoro di sviluppo sul Software Architecture Warehouse è stato svolto dagli studenti di master della facoltà di Scienze Informatiche. Il vostro lavoro è stato molto apprezzato.

Sono convinto che durante questo lavoro, abbiamo tutti imparato delle lezioni importanti sullo sviluppo di applicativi Web. Grazie a: Ievgenii Riabokon, Masiar Babazadeh, Adnan Alhariri, Mark Pruneri, Omar Elabed e Alessandro Andreani.

Devo un grande ringraziamento a Dr. Cherilyn Keall per i suoi preziosi consigli sullo stile ed il linguaggio che hanno dato un tocco finale a questa dissertazione.

Questo lavoro è stato parzialmente supportato dallo Swiss National Science Foundation con il progetto CLAVOS (Grant Nr. 125337).

Podziękowania

Ta praca nie była by możliwa do zrealizowania bez wsparcia i zachęty wielu osób.

W pierwszej kolejności chciałbym podziękować mojemu promotorowi Profesorowi dr. Cesare Patuasso. Bez jego pomocy i nadzoru ta praca w ogóle nie doszłaby do skutku. Dzięki jego otwartości umysłu i wsparciu w trudnych momentach, byłem w stanie pozostać na właściwym kursie. Konstruktywna krytyka i merytoryczna pomoc była gwarancją wysokiej jakości ostatecznego wyniku.

Chciałbym bardzo podziękować mojemu koledze – Vassileos Triglianos za wysiłek, który włożył w korektę ostatecznej wersji tej pracy. Jego świeży punkt widzenia znacznie podniósł jakość edytorską tekstu. Chciałbym również podziękować pozostałym członkom mojej grupy badawczej, to jest: dr. Saeed Aghaee, Masiar Babazadeh, dr. Daniele Bonetta, Vincenzo Ferme, dr. Achille Peternier. Specjalne podziękowania należą się moim recenzentom: prof. dr Olaf Zimmermann, prof. dr Olaf Zimmermann, prof. dr Michele Lanza, prof. dr Patricia Lago, oraz prof. dr Mehdi Jazayeri, za ich nieoceniony wkład w przygotowanie tej pracy.

Czuję się zobowiązany podziękować Janowi Lalkowi, Andrzejowi Rybczyńskiemu i Adamowi Matusiakowi z firmy Lafot. Czas, który spędziliśmy pracując razem był doświadczeniem które zainspirowało mnie do naukowego zgłębiania architektury oprogramowania.

Chciałbym również podziękować dwóm świetnym managerom z którymi miałem przyjemność pracować: Dirk Weidemann i Reinhard Koehler, za zaangażowanie mnie w interesujących projektach i umożliwienie rozwoju.

Znacząca część pracy przy implementacji Software Architecture Warehouse została wykonana przez studentów studiów magisterskich uniwersytetu w Lugano. Jestem przekonany, że w czasie naszej współpracy wszyscy wiele nauczyliśmy się na temat aplikacji internetowych. Dziękuję: Ievgenii Riabokon, Masiar Babazadeh, Adnan Alhariri, Mark Pruneri, Omar Elabed i Alessandro Andreani.

Gorące podziękowania należą się dr Cherilyn Keall za jej nieocenioną pomoc i poradę językową, która nadała ostateczny kształt tej pracy.

Ta praca była częściowo finansowana przez Szwajcarską Fundację Naukową (SNF) z projektu CLAVOS (Grant nr 125337).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this dissertation we explore the collaborative design of software architecture using a contemporary approach to software architecture, viewing it as a set of principal design decisions [TMD09]. Efficient software architecture design calls for new ways of making and managing decisions as a team [TGA13]. We propose an extension to the architecture description standard (ISO 42010) that focuses on decision argumentation, and we observe it under experimental conditions.

Modern computer systems often reach levels of complexity comparable to biological systems. Software Architecture is a design abstraction often used to master such complexity so that it can be managed by the designers. In order to deliver high-quality architecture design for complex software system, the design team often needs to cover vast areas of expertise that effectively exceed the capacities of a single software architect [TGA13]. At the same time, physically bringing together domain experts to participate in an architecture design workshop creates a significant logistical overhead, in particular if the meetings need to take place regularly over an extended period of time. With help of teleconferencing technologies it is possible to bring together otherwise physically distributed teams. Unfortunately, telecommuting inherently reduces communication bandwidth between the team members. Constrained communication is particularly a problem for medium and large design teams.

A significant difficulty in running architecture design workshops is in the codification of the decisions in a way that is not overly intrusive for the process, but leaves outcome that is informative for the designers and developers, and can be reused in future designs. Model-driven engineering is a widespread technique commonly used to address the complexity of software systems. Unfortunately, on the other hand, the tools and methods used to model effectively software architecture fall short when used to model design decisions during dynamically progressing design workshops. On the other hand, general purpose collaboration tools often miss the expressive power required for detailed modeling. Specifically for software architecture, the ISO 42010 standard proposes a model of architecture design decisions [ISO11]. The standard and its extensions effectively model decision status, rationale, and many other aspects, but not the argumentation that leads to the decision itself.

1.1 Problem Statement

The problem with architecture decisions is how to make good decisions. The best way to evaluate architecture decisions would be to observe how successful the design that results from the decision making is. In practice there is a multitude of factors that influence the success (or failure) of a particular design, that combined with the long time and significant effort required to implement the system, make it very difficult to track the impact of individual decisions. Instead of this holistic approach, we have decided to focus first on individual decisions. In this setting, we have found that good decision needs to be 1) **smart** and 2) **well informed** [Han05]. Our research does not aim at supporting decision makers with artificial intelligence or data mining technologies, so we assume that the design team is smart enough to understand the problem domain and based on this information make smart (rational) decisions. In terms of decisions being well informed, we make the assumptions that, thanks to the requirements engineering, the designers are provided with enough information about the problem domain and that they have enough expertise as a group, to successfully challenge individual design issues.

An ability to perceive, understand and predict the dynamics of an environment, constitute so-called Situational Awareness (SA) [End00]. In literature, three levels of SA are recognized. The first level concerns solely the perception of the environment. The second level is attributed both to perception and comprehension. Finally, the third degree of SA is related to the capacity to predict future dynamics of the environment based on its perceived past and present states. We follow the heuristic according to which a high degree of SA is essential for delivery of high-quality architecture decisions.

Due to the fact that during an architecture design workshop the architecture decisions are made collaboratively, another essential property of a good architecture decision is its **consensuality**. Related to it is the so-called Team Situational Awareness (TSA) [End95], which extends the concept of Situational Awareness. A high level of Team Situational Awareness requires not only that all design team members have a high SA individually, but also that this awareness be shared among themselves. In terms of collaborative architecture decisions, we assume that a high degree of TSA is reflected by the decision consensus.

Therefore we state the thesis of our work to be:

> *Support for low-latency, structured architecture decision argumentation improves the quality of the collaborative decision making process.*

In order to address this thesis we have pursued active research in three directions: architecture decision argumentation modeling, supporting collaborative architecture decision making, and estimation of collaborative architecture decisions quality. We introduce them in the following sections.

1.2 Architecture Decision Argumentation Modeling

Effective management of architecture decision models requires a well tailored meta-model. Surveying the state of the art we have realized that there exists a number of competing decision meta-models. In order to retain maximum flexibility in regard to the decision meta-model, we have decided to step-up the abstraction level and operate on the elementary structure of a knowledge graph [Zha02]. This way our methods can be easily applied regardless of the particular meta-model choice.

We have adopted ISO 42010 standard architecture description as the basis of the decision meta-model. Due to the fact that the standard does not offer explicit argumentation modeling, we proposed a specific decision argumentation viewpoint. Within the viewpoint, we have devised a notion of position that is used to represent fine-grained decision argumentation of individual decision makers. An analysis of the aggregated state of multiple positions within the context of design decisions let us define a decision consensus lifecycle. Similarly, an aggregated state of multiple decisions, that are related to the design issue, lets us define the design issue choice state.

1.3 Collaborative Architecture Decision Making

In order to investigate the state of the art in supporting architecture decision management, we have inspected a broad range of generic and specialized tools. The outcomes of our survey are organized into a design space centered around two core design issues: the collaboration paradigm, and the supported liveness level. Within this design space, we have identified a combination of decisions that would serve for the purpose of supporting collaborative decision making. This provided us with the basis for the design and implementation of the Software Architecture Warehouse – a prototypical tool supporting software architects during the architecture design workshops with high liveness and a structured argumentation model.

1.4 Estimation of Collaborative Architecture Decisions Quality

The prerequisite for managing quality is the ability to estimate it [LM06]. In order to measure qualities of the collaborative architecture design decisions, we have devised an analytical framework of decision metrics. The metrics are designed to analyze structural and dynamic characteristics of decision models, with a particular focus on the collaborative decision argumentation. The framework consists of fifteen metrics that are geared towards three questions: 1) how aligned are the decisions, 2) how volatile is the consensus over the decisions, and 3) how democratic are the decisions.

The Software Architecture Warehouse is accompanied by the analytical framework implementing the aforementioned decision metrics suite.

1.5 Contributions

This dissertation advances the state of the art in collaborative architecture decision making with the following three contributions:

Architecture Decision Argumentation Viewpoint – an extension of the contemporary software architecture description standard (ISO 42010), which allows comprehensive modeling of the collaborative argumentation behind the architecture decisions,

Software Architecture Warehouse (SAW) – a collaborative, web-based architecture decision management tool designed to support collocated and remote architecture design workshops by providing features supporting high degree of team situational awareness within the architecture design team. SAW implements the Architecture Decision Argumentation Viewpoint in practice,

Collaborative Architecture Decision Metrics – an analytical framework designed to estimate structural and dynamic characteristics of a collaborative architecture decision space.

1.6 Structure

The rest of the dissertation is structured as follows:

In Chapter 2 we provide broad background covering three major fields that our research stems from. We introduce the discipline of Software Architecture Design, specifically its quality concerns. As the discipline is very knowledge intensive, the considerations over Architectural Knowledge lead us into the second major field, Knowledge Management (KM). Within KM we investigate specifics of the knowledge-based decision making process with a peculiar focus on the so-called wicked problems and strategies for addressing them. Finally, we wander into the field of Computer Supported Collaborative Work to build a foundation for the collaborative design of software architecture.

In Chapter 3 we dive into Software Architecture modeling and explore the design space covering the collaborative Software Architecture design paradigms and the tool liveness. Within the design space of seven design issues we investigate and recover decisions implied by 17 tool designs.

Chapter 4 introduces typical scenarios of software architecture design; architecture synthesis, architecture analysis, and a hybrid of both. Analyzing these scenarios we state two research problems:

1. Collaborative architecture design decision consensus,
2. Quality of the collaborative architecture decisions.

Following that, we have elicited respective research questions:

1. How to support collaborative software architectural decision making?
2. How to identify and quantify properties of a good, collaborative design decision making process?

Finally, we arrive at the research thesis for our work.

In Chapter 5 we propose a viewpoint extension of the ISO 42010 standard decision model aimed at modeling decision argumentation. With it, we thoroughly examine choice and consensus state spaces. The Decision Argumentation Viewpoint opens completely new opportunities for the analysis of an architecture design workshop dynamics. In order to address them, in this chapter, we propose a set of Goals, Questions and Metrics, together with matching interpretation model.

In Chapter 6 we document the Software Architecture Warehouse – our applied contribution to the field of collaborative architecture decision modeling. The description covers a range of features that make it stand out from the competition.

Chapter 7.2.3 covers the Analyzer – an implementation of argumentation viewpoint analytics.

Next, in Chapter 8 we provide a report on the formative evaluation of the SAW and a demonstration of the analytics framework, that we have performed within the Software Architecture and Design courses at the University of Lugano.

Finally, Chapter 9 wraps up the thesis with a summary, conclusions and perspectives on future research.

Chapter 2

Background

Software architecture is the discipline practiced by software architects. It emerged out of the experience with the design and implementation of complex software systems gathered by software engineers. The foundations for defining Software Architecture were laid by Perry and Wolf [PW92]. The first case-study based approach to the field was presented in 1994 by Garlan and Shaw [GS94]. Despite numerous attempts of the community [SEI10], no universally recognized definition of Software Architecture exists. Bass et al. proposed in [BCK03]:

> *"The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships between them. "*

The IEEE 1471-2000 standard [oEE00] states:

> *"Architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution."*

A number of modeling languages were proposed in order to provide a common communication platform among software architects. These range from the generic Unified Modeling Language (UML [FS99]) to specialized *Architectural Description Languages* (ADLs). A comprehensive review and comparison of ADLs is provided by Medvidovic and Taylor in [MT00]. Given the sheer size and long lifespans of today's software systems the description provided by Kruchten et al. [KOS06]

> *"Software architecture ... is the key to achieving intellectual control over [their] enormous complexity. "*

exposes the most crucial function of software architecture - an abstraction used to tame complexity.

The amount of detail embraced by the architecture can be enormous. In order not to overwhelm the recipient, the right balance between abstraction and minuteness is needed. One of the ways to strike this balance is to view architecture from the perspective (viewpoint) emphasizing a particular aspect of the system. Architecture frameworks (Kruchten 4+1 views [Kru95], Zachman [Zac87], TOGAF [Jos09], Siemens 4 views [SNH95]) define a broad range of architectural viewpoints tailored either to the particular stakeholder, characteristic, or feature of the design.

The difference between architecture, design and implementation (or code) has been well exposed by Eden et al. [EK03]. These concepts belong to different abstraction levels and can be classified according to their **intensionality** and **locality**, which are two orthogonal dimensions of abstraction:

> *"A specification is **intensional** iff there are infinitely-many possible instances thereof. Conversely, all other expressions are **extensional**. "*

For example, architecture and design are intensional, because there are infinitely many implementations that can fulfill them. Whereas, implementation is is a specific software program.

> *"A specification φ is **local** iff the following condition holds: If φ is satisfied in some design model ϖ then it is satisfied by every design model that subsumes ϖ."*

For example, design is local, because it comprehensively describes structures within its scope. Architecture is non-local because particular pattern or style can apply to specific abstraction level. Making it possible to apply another pattern or style for lower (or higher) abstraction.

The abstraction levels of architecture, design and implementation are summarized in Table 2.1.

	Intensional	**Extensional**
Non-Local	Architecture	-
Local	Design	Implementation

Table 2.1. Architecture vs. Design vs. Implementation

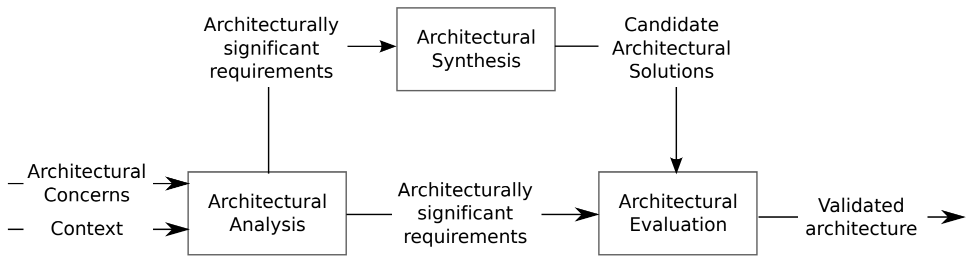


Figure 2.1. Generalized architecture design process model after Hofmeister et al. [HKN+07]

There exists no one, universally recognized division between architecture design and implementation. Grady Booch wrote [Boo06] that all architecture is design, but not all design is architecture, which is right within the scope of prescriptive design. A contradicting claim can be made about descriptive artifacts: every system has an architecture, but some systems have no design.

2.1 Architecture Design Process

Architecting is typically seen as the activity linking the requirements [vL08] of the problem domain [Jac95] to the design artifacts of the solution domain. The relation between requirements and architecture is represented well by the Twin Peaks model [Nus01] (see Figure 2.2). The Twin Peaks model adopts the spiral model of software project management for iterating between architecture and requirements. It allows the design process to cope with the "I'll know when I see it" (IKIWISI [Boe00]) effect. The consecutive iterations are more refined and contain larger amount of detail.

As shown in Figure 2.1, the architectural design process first analyzes the input architectural concerns and context to elicit the architecturally significant requirements [Dur11]. These are then used to synthesize many candidate solutions, which will then be evaluated and compared to allow the selection of the best architecture.

In [CLvV07] Clerc, Lago and van Vliet present the results of a survey made among architects to determine their actual profile in terms of *abstraction levels* and *roles*. They find no clear boundaries between what they call architecture

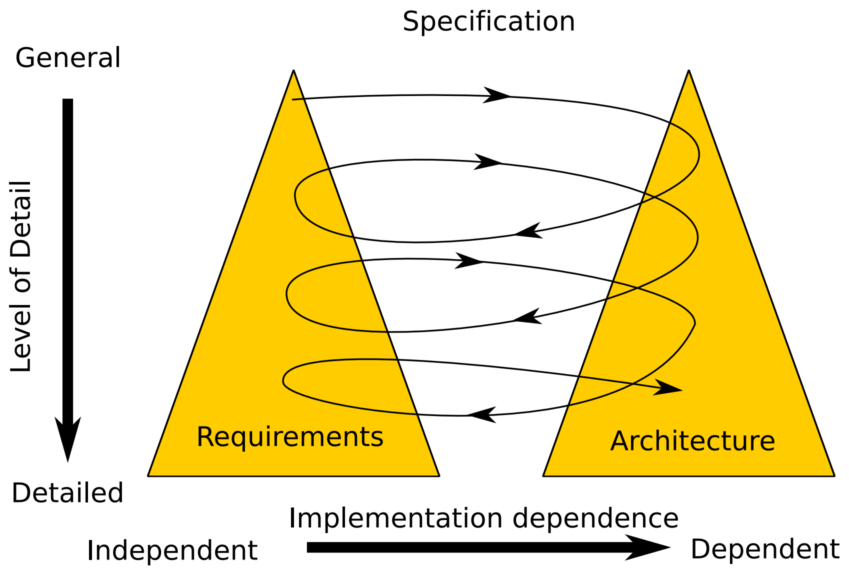


Figure 2.2. Twin Peaks model of concurrent and progressive development of requirements and architecture [Nus01]

levels (e.g., enterprise vs. systems vs. software vs. information architecture) and stakeholders (e.g., project manager vs. technical developer). Instead, they identify clusters of the stakeholders' characteristics with diverse demands on the architectural knowledge access (see Table 2.2).

Cluster Label	**Architectural Roles**
Communicator	architectural educator, project leader
Low-Level	designer, developer, reviewer of the code
Specialist	consultant, technical specialist
High-Level	architect, reviewer of the architecture
Other	end-user, lead architect, security consultant

Table 2.2. Clusters of architectural roles characteristics [CLvV07]

2.1.1 Quality of Software Architecture

The quality of Software Architecture has a very multi-dimensional nature. A selection from the basic collection of qualities of a software product as defined in ISO 9126 [ISO91] can be found in Table 2.3. Qualities of the software end-

FUNCTIONALITY	The architecture shall satisfy the stated or implied needs.
EFFICIENCY	The architecture shall solve stated or implied needs efficiently with respect to hardware usage (e.g. memory usage and CPU load)
RELIABILITY	The architecture shall perform its required functions correctly.
REUSABILITY	It shall be possible to reuse the architecture in other applications in the same environment.
MAINTAINABILITY	The effort required to make changes to the architecture shall be low.

Table 2.3. Software product qualities after ISO 9126 [ISO91]

product are influenced by the quality of the whole chain of design artifacts and processes leading to it [EC09]. Taking the generic software architecture design process (Figure 2.1), a rough approximation of the chain of quality implications between the process activities can be pictured as in Figure 2.3. In Figure 2.4 we show an equivalent diagram of implications between the design artifacts.

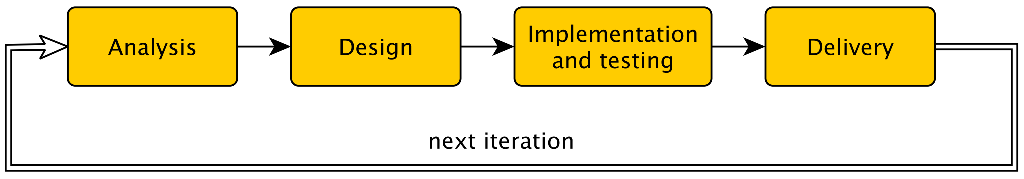


Figure 2.3. Chain of quality implications between design process activities

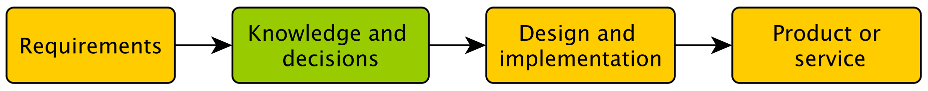


Figure 2.4. Chain of quality implications between design artifacts

Our claim is that the quality of all these artifacts is linked and therefore in order to get predictable quality of the final product, one needs to be able to estimate and control the quality of the artifacts preceding it in the chain.

2.1.2 Architectural Knowledge

Over time, some designs have proven to be particularly successful and out of the chaos of spontaneous creativity, patterns emerged [HWBYZ13]. Patterns can be found at all abstraction levels [Ede05, EHK06] ranging from implementation, through tactics, up to strategy. A number of design patterns were first documented in the cornerstone book by Gamma et al. [GHJV94]. Since then a significant effort has been invested in the distillation and documentation of architectural knowledge in order to make it reusable [Zim11]. To some extent this knowledge remains proprietary as a highly valuable asset, but at the same time there is a broad selection of published books of patterns [BMR+96, SSRB00, KJ04, BHS07a, BHS07b, Mah06, HW03, Fow02]. To this day, the number of documented architectural patterns is estimated to be over 7500 and is growing [HWBYZ13].

At this point it, makes sense for us to focus on the definition of architectural knowledge. After Avgeriou et al. [AKL+07] we use the following:

> *"Architectural Knowledge (AK) is defined as the integrated representation of the software architecture of a software-intensive system (or a family of systems), the architectural design decisions, and the external context/environment."*

More specifically, de Boer and others, in [dBFL+07], propose a meta-model of Architectural Knowledge that neatly embraces elements and actions related to Architectural Knowledge (see Figure 2.5).

Following the accumulation of architectural knowledge, it became clear that reusing existing successful designs increases quality and time efficiency and reduces the cost [GHJV94] of the design process. Most importantly, by reusing successful designs, architects can focus on the design aspects within their expertise and deliver valuable innovations [Tra95]. As a result, the research focus in *software architecture* shifted towards the analysis of the decision making process [Bro10].

Initial approaches to the formal representation of decisions come from research in the field of artificial intelligence [Lee89] (late '80s). The importance of decision capture for the design of mission-critical systems was spotted in [ROJ90]. The capturing and processing of design decisions alone appeared to be clearly

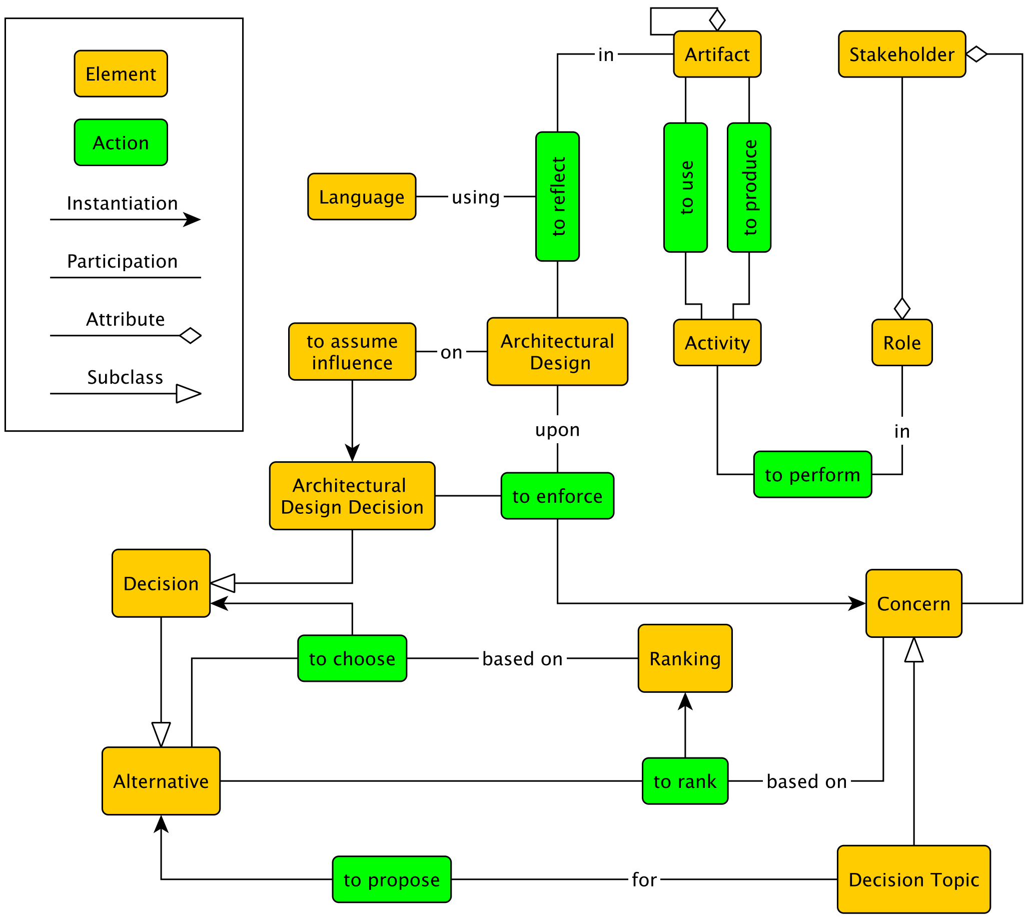


Figure 2.5. CORE Architectural Knowledge model [dBFL+07]

insufficient without knowing the rationale behind these decisions [Lee97]. Development in this direction resulted in the *Rationale-Based software engineering* method [Bur05, BCMM08] which rapidly acquired recognition in the field of Software Architecture [CSM08, Cap09, JAvdV09]. Furthermore in [Bos04] Bosch argued that decisions should be considered as *first-class* entities in software architecture. More recently Jansen and Bosch in [JB05] proposed the following decision-centric [TMD09] definition of *software architecture*:

> *"A software system's architecture is the set of principal design decisions made about the system."*

Apart from the engineering perspective on the *architectural design decisions*, an important consideration of economical benefit was stated by Clements in [Cle07]. A broader, more general, broader study on the evaluation of project decisions in software engineering can be found in Hoover et al. [HRLT09].

2.2 Knowledge Management

In a comprehensive monograph on knowledge management, Dalkir [Dal11] introduces a number of goals characterizing the knowledge management initiative:

$\mathbf{G}_1$ *Generating new knowledge*
$\mathbf{G}_2$ *Accessing knowledge*
$\mathbf{G}_3$ *Applying knowledge for the purpose of decision making*
$\mathbf{G}_4$ *Embedding knowledge in processes, products and/or services*
$\mathbf{G}_5$ *Representing knowledge in documents, databases and software*
$\mathbf{G}_6$ *Facilitating knowledge growth through culture and incentives*
$\mathbf{G}_7$ *Transferring knowledge*
$\mathbf{G}_8$ *Evaluating the value of knowledge and its impact*

These goals fit very well in the two-dimensional knowledge management activities space (knowledge spiral) proposed by Noanaka and Takeuchi [NT95] (Figure 2.6).

Within the knowledge management cycle, newly created ($\mathbf{G}_1$) tacit knowledge is codified into explicit knowledge in some kind of formal representation ($\mathbf{G}_5$ and $\mathbf{G}_7$). The codified knowledge can be used in the product design process ($\mathbf{G}_3$ and $\mathbf{G}_4$) as well as for the improvement of the design practice itself ($\mathbf{G}_6$). Next, individual consumption of the explicitly codified knowledge ($\mathbf{G}_2$) together with the assessment of the utility of particular knowledge in the application context ($\mathbf{G}_8$) leads to a new start of the knowledge cycle. Since the aforementioned goals are very adequate for software architectural knowledge management, we adopted them to serve as a foundation of the functionality provided by the Software Architectural Warehouse introduced later in Chapter 6.

For the purpose of further scoping the field of software architectural decisions management activities, we found it useful to demonstrate how the knowledge management model devised by Choo ([Cho06], Figure 2.7) clearly positions *decision making* in the cycle of knowledge management. The cycle starts with the *sense making* stage consuming *experience* coming from the external environment and resulting in the set of *shared meanings*. *Shared meanings* stand for the codification of knowledge in the process of *knowledge creating* from explicit *external knowledge and information*. The same *shared meanings*, created earlier during *sense making*, are used in combination with *new knowledge and capabilities* as a ground for *decision making*. As decisions influence their design environment, the *experience* on which they are based on needs to be re-evaluated, thus starting the next *knowledge cycle*.

2.2.1 Decision Making Process

Giving a formal model of the decision making process was a subject of intensive investigation for the management and organizational researchers. Choo in

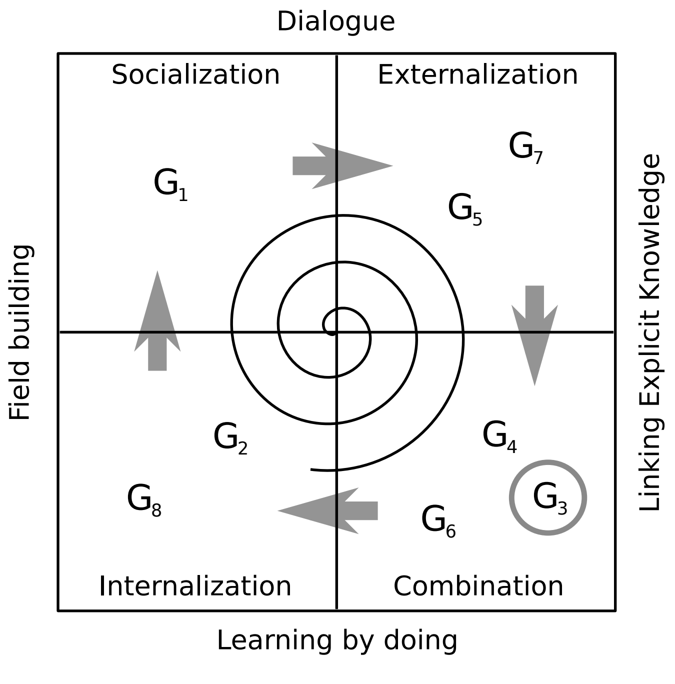


Figure 2.6. Nonaka and Takeuchi spiral model of knowledge conversion. [NT95]

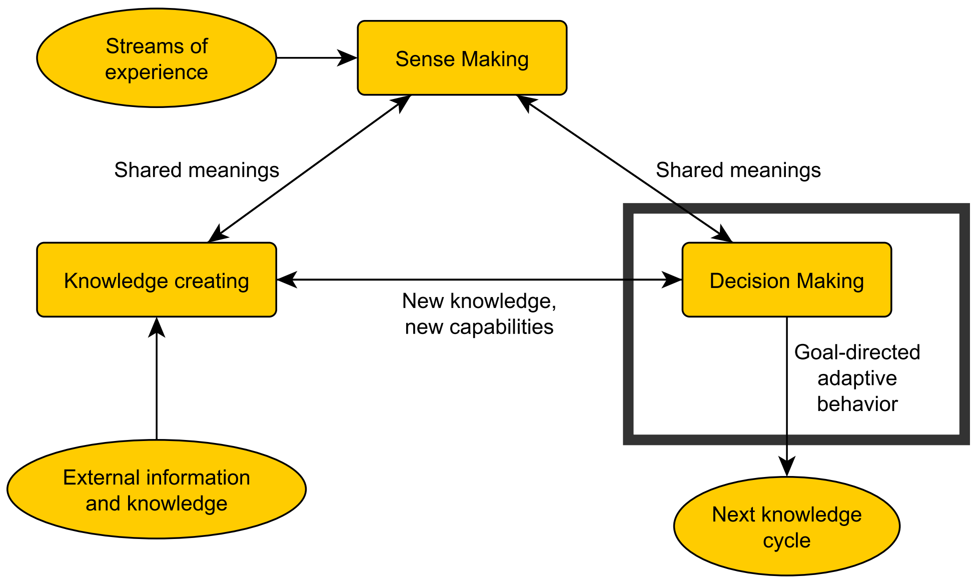


Figure 2.7. Overview of Choo's knowledge model [Cho06]

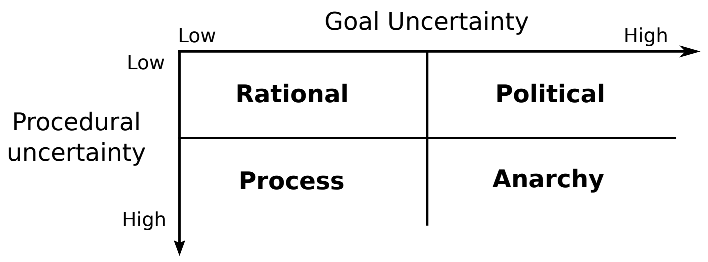


Figure 2.8. Four basic decision making modes identified by Choo in [Cho06].

[Cho06] recognized four modes of decision making and classified them along the dimensions of goal and procedural uncertainty (Figure 2.8). The *anarchic* way is the least favorable mode for making decisions, where goals are ambiguous and the processes to reach those goals are unclear. The *political* mode is characterized by the certainty about the process, but is burdened by conflicting goals and stake-holder interests. Contrary to that, the *process* mode of decision making is goal directed, but due to fuzzy procedures it faces difficulties with choosing between multiple variants and alternative solutions. The sweet-spot and the most desired decision making mode is the *rational* mode, which is goal directed and guided by well-defined rules and routines combined with an explicit performance analysis.

The first step in the direction of an improved decision making process is a reduction of *procedural uncertainty*, which can be obtained by implementing an elementary, four-stage decision process as pictured in Figure 2.9 [Kle99]. The first

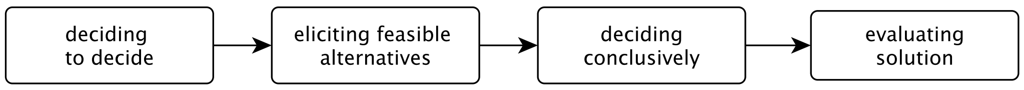


Figure 2.9. An elementary, four-step decision making process

stage of *deciding to decide* consists of the identification of the issues which require decisions to be made for. This can be done by inspecting goals, concerns and requirements relevant to the given stage of the design. The second stage of the decision making process involves the *elicitation of the feasible alternatives* and the elimination of those which are unfeasible in the context of a particular issue. At this stage of the process, the decision status is not yet conclusive. The *conclusive decision* state is reached in the process of evaluating the elicited alternatives and irrevocably choosing the best one. Finally, an alignment of the decision within the context needs to be checked by *evaluating solution* characteristics. Decision making processes are the subject of in-depth studies within disciplines ranging from administration [Sim97, Chapter 5] to software design [Bro10]. In software architecture, the decision making (micro)process was proposed by Zimmermann in [Zim09] as a part of the SOA Decision Modeling framework (SOAD).

The second step for improvement of the decision making process is the reduction of goal uncertainty through appropriate goal elicitation and management. Goal setting and management are subjects of investigation for the field of strategic management [NHC07], and thus are beyond the scope of this dissertation.

2.2.2 Wicked Problems

The decision making process presented in Section 2.2.1 makes sense in the context of an individual decision. In practice software architectural decisions form a dense network of interdependencies and mutual influences [TGA13]. One way to tackle the complexity of a decision making process is to formalize both artifacts and process. This approach (the so-called “first generation approach”) appears to be successful at solving the well structured, so-called “*tame* problems” [Rit72] which can be exhaustively formulated and which have a specified, binary function that can be used for verifying the validity of the solution. In a way, *tame* problems can be seen as those which have algorithmic methods of arriving at the solution with clearly specified stopping conditions [KS08]. Following that, similarly to computational problems, normally there is a range of well-defined operations that can be executed to transform (reduce) a *tame* problem into another tame problem, which eventually will lead to the solution. Rittel came up with an 8-step generic procedure to approach *tame* problems [Rit72]:

1. understand and define the problem, define the measure of *effectiveness*,
2. understand the context of the problem,
3. analyze the information - identify constraints and inputs,
4. generate solutions - define the *solution space*,
5. assess (evaluate) solutions - estimate their *effectiveness*,
6. implement the chosen solution,
7. test if the solution has expected *effectiveness*,
8. modify, correct the solution.

The 5th step of the aforementioned procedure can be efficiently addressed using a general morphological analysis [Rit06] that has proven to be very successful when applied to well defined, multidimensional problems of choice within finite design spaces. Unfortunately, this approach is unfeasible in dynamic and uncertain environments.

In [Rit72, RW73] Rittel defined the notion of *wicked* problems, which in contrast to *tame* problems can be characterized by the impossibility of providing an exhaustive formulation. A complete formulation of a *wicked* problem can be obtained only after the problem is solved. Attempts to obtain it upfront are futile. Contrary to *tame* problems, alternative solutions to the wicked problem can be evaluated only in relative terms. Thus, it is impossible to give a guarantee that a given solution candidate is the best possible one; hence, *wicked* problems have no stopping rule. Another characteristic of the *wicked* problem is the difficulty concerning the evaluation of the solution alternatives, which arises from the fact that evaluation criteria are multidimensional and subjective in the context of particular stakeholders. Finally, *wicked* problems are mostly one-of-a-kind, prototypical instances. Designers, therefore, lack a complete image of the dynamics of the system under design. Alternatively, if the system itself (partially) reuses an existing design, it might be the case that the reproduction of the original environments is either very hard or impossible, thus significantly reducing the utility of the prior experience.

Characteristics of *wicked* problems match many problems which (construction) architects face in landscape planning and urban design. In fact, Liskov in [LG86] and later Cripps in his blog entry[1] have pointed out that these characteristics match problems of modern software architecture design of big computer systems. An identification of *wicked* problems led to the development of the social techniques of reaching design consensus [Coy05] and tools such as IBIS (Issue Based Information-System [Con05, CB88]) aimed at supporting the deliberation

[1]`http://softwarearchitecturezen.blogspot.com/2010/02/wicked-problem-is-one-that-for-each.html`

process. The experience gathered in the collaborative solving of *wicked* problems led to the identification of three elementary solution strategies [Rob00]:

Authoritative - focused on the centralized decision power of a single person. Due to the fact that it doesn't require any consensus making effort, it can be potentially time-effective and thus efficient in heavily time-constrained conditions. The centralization of authoritative decision making has the downside that the person making decisions, acting solely on the basis of his own expertise, is prone to overlook essential decision context changes. For the same reason, the decision maker is likely to misinterpret the requirements and misunderstand the needs of the stakeholders influenced by the decision.

Competitive - a strategy promoting diversity of alternative solutions which are put into competition for the purpose of choosing one that is going to be implemented. Typically this strategy is used in architectural competitions to choose the best design. Thanks to the comparative evaluation of multiple solutions, the competitive strategy is much more likely to fulfill the requirements and the needs of stakeholders. Unfortunately, it is also burdened with disadvantages. The biggest one is that it focuses on the *zero-sum game*, which means that choosing one solution alternative means a complete rejection of all others, even if they have some desirable features. Another one is that it is often successful only in a limited context and results in a (potentially violent) run for local power. If applied in a broad context, due to the fact that the winning solution provider *takes all*, eventually this strategy can very easily turn into the **authoritative** strategy, thereby canceling out the benefits of the competition.

Collaborative - follows a principle of *win-win* problem solving and avoids solutions in which the *winner takes it all*. Rather than playing a *zero-sum game*, it focuses on distributing *pie shares* based on winners and losers under an assumption of a variable sum game that seeks to enlarge the pie for the benefit of all stakeholders. Implementation of the collaborative strategy sets quite high requirements on the stakeholders' organization culture. Taking into account that knowledge required for the decision making is distributed among multiple stakeholders, it is critical for the process to obtain clearly communicated individual judgements together with their grounding (rationale). Offering many advantages over the other two strategies, the collaborative strategy is not free of disadvantages. The biggest one is that every additional stakeholder taking part in the process increases the *transaction cost* of the decision making process.

2.3 Computer Supported Collaborative Work

The decision making process [Kle99] and, in particular, the software architectural decision making process have been the subjects of many studies [FCKK11]. The topic of collaborative design has been less studied, and it is only partially supported in the ISO 42010 decision meta-model [ISO11]. Out of the seven architectural decision modeling tools reviewed in [SLK09a], 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.

2.3.1 The Problem of Collaborative Design

The factors that limit the efficiency of the decision making process within a design team are manifold [HP96], e.g., the partial overlap of the participants' expertise, the complexity of the domain and the wicked nature of the software architectural design problem [PB88].

In our experience running design workshops, we have observed that the decision making process can be very chaotic, and 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 [ZKL$^+$09, 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 needing to resort to dedicated minute takers or scribes.

We 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 sometimes 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 collocated design workshop, thanks to the 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 a solution can be unsuitable for stakeholders. The most critical cases are when a 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 [ZKL^{+}09]). It is often the case that there are multiple valid, acceptable solutions. In such situations, 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 [dGJKK12]) 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.

2.3.2 Positioning in Environment

Baecker et al. in [BGBG95] propose a very interesting classification of computer supported collaboration work in two dimensions – the time and the location of participants. Collaborative decision making activity fits within these dimensions as follows:

C_1 – Collocated, synchronized – The design workshop involves the complete design team within a single meeting room. Typically the workshop would be moderated by the head architect, otherwise communication among team members can be direct and unconstrained.

C_2 – Dislocated, synchronized – High-grade experts are often too valuable to be asked to travel for assignments of short duration, hence holding design workshops is often a necessity. Broadband networking would be typically used to share audio and video, as well as other media.

C_3 – Collocated, time-shifted – Due to the gradual (and progressive) nature of large-scale architecture design, decision making needs to take place in multiple iterations. Continuity and consistency of decisions are essential to achieving high quality of the final design.

C_4 – Dislocated, time-shifted – Similar to the previous case (C_3), but distributed.

2.3.3 From Situational Awareness to Good Decisions

High-quality, good decision needs two elementary ingredients – a sufficient amount of information, and a smart decision maker. The apprehension of the environment for making decisions is scientifically known as Situational Awareness (SA). Although supporting the smartness of a decision maker lays beyond the scope of our research, we believe that there is a vast space for improvement in the field of Situational Awareness. In particular, complex undertakings require teams to deliver high-quality decisions that require a high degree of Team Situational Awareness (TSA). Situational awareness can vary – as proposed by [End00] – 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 among, 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 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 situations in which the expertise required to make decisions is distributed among multiple stakeholders.

Although the conditions requiring situational awareness on 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 applying the concept of team situational awareness to enhance the efficiency and quality of the collaborative architecture design process. In particular, situational awareness shared among the whole team can help it to efficiently argue and build consensus about each decision. A design team sharing a high level of situational awareness can gather relevant information, interpret it from the 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 on a single consensus decision.

2.4 Collaborative Design of Software Architecture

Since a software architecture is an abstraction used to manage the complexity of large systems, collaborative design is a method of dividing the design effort into chunks that can be addressed by individual designers within their areas of expertise. Computer Supported Collaborative Engineering (CSCE) was mainly developed in the field of mechanical engineering [SHL08]. Even if there are significant similarities between established engineering design disciplines and software architecture design, the latter is still young and fast developing. Specifics of software architectural design, such as high volatility and intangibility of the design outcome, set challenging requirements for knowledge sharing. The topic of supporting architects in architectural knowledge sharing was extensively studied by Farenhorst et al. [FdB09, FJFH09] and Clerc et al. [CdVL10, Cle11], resulting in a monograph on Software Architectural Knowledge Management [BDLvV09].

The decision-making process introduced in Section 2.2.1 makes perfect sense in the context of an individual decision. This approach (the so-called "first generation" approach) appears to successfully solve the well-structured, so-called *tame* problems [Rit72] that can be exhaustively formulated and have a specified, binary function which can be used for verifying the validity of the solution. Contrary to that, a complete formulation of a *wicked* problem can be obtained only after the problem is solved.

2.5 Summary

In this chapter we scoped the domain of collaborative software architecture design. We also sketched the elementary elements of the software architecture design process. Next, we discussed the quality of software architecture, in particular, the propagation of quality through the artifacts used by the software design process. We introduced general aspects of knowledge management relevant for software architecture design, thus positioning architectural knowledge management in its context. Finally, we introduced wicked problems and strategies of addressing them. We wrapped up this background chapter with the discussion of Computer Supported Collaborative Work (CSCW) and its ties to the collaborative design of software architecture.

Chapter 3

Related Work and the State of the Art

In this Chapter we review the state of the art in architectural decision modeling and making, we introduce the cornerstone decision meta-models, and we systematically review the design space covering a range of tools supporting decision and knowledge management. Readers interested in the emerging discipline of software architecture metrics should refer to [SCB14].

3.1 Architecture Views and Decision Modeling

As we introduced in the previous chapter, the complexity of modern computer systems can be very high. The abstraction offered by software architecture is a good tool for addressing it. Nevertheless, the diversity and number of software architecture elements can easily overwhelm both stakeholders and designers. Architectural viewpoints, which provide a particular perspective on software architecture elements, are particularly good at exposing specific aspects of the system under design. For example, Kruchten [Kru95] proposed four architectural views (Logical, Process, Development, and Physical) that are interconnected with a set of usage scenarios (see Figure 3.1). Even if these viewpoints have proven to be very effective in depicting architecture elements, in order to include the notion of decisions as first-class entities (see Section 2.1.2), the framework of 4+1 views needed to be extended with an extra decision dimension [KCD09]. A particular set of specific attributes of architectural decisions was proposed by Tyree and Akerman [TA05] in the form of a decision template. In [SLK09a] Shahin et al. survey nine architecture decision meta-models and study a mapping between Tyree's decision template attributes and particular decision meta-model elements. The conclusion that can be drawn from of this survey is that there is no single, complete mapping between decision meta-models. This fact can also

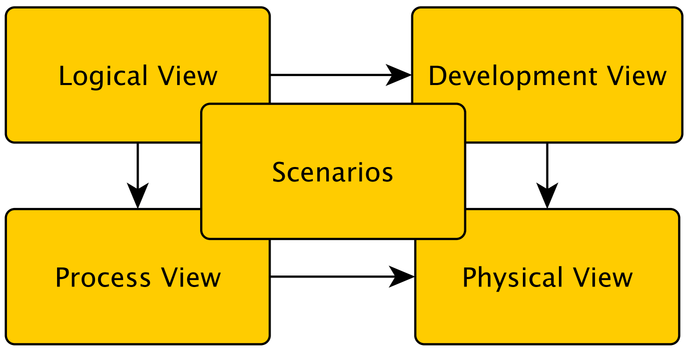


Figure 3.1. Kruchten's 4+1 architectural view model [KCD09]

be interpreted as a need for (some degree of) flexibility of a decision meta-model to the needs of the particular project case.

A decision meta-model worth mentioning because of the significant sample of a high-quality reusable architectural decision model (RADM), is the SOAD meta-model (see Figure 3.2) proposed by Zimmermann et al. [ZKL+09].

Here we provide more details on major elements of the ISO 42010 model:

Architecture Decision – is a part of architecture description and is considered the key to the architecture of the system under design [ISO11]. It states the architecture's direction. The notion of architecture decision is consistent among the inspected meta-models,

Architecture Rationale – documents arguments, Pros and Cons, that were used to arrive at the particular design decision,

Concern – ISO 42010 defines concern as interest in a system relevant to one or more of its stakeholders,

Group – is a cross-cutting designation or label attached to decisions within a particular design domain or design abstraction level,

depends loop – represents specific, unilateral dependency relation between the architecture decisions,

AD Element – a work product used to express an architecture,

Stakeholder – an active or passive participant in the design process who has interest in and/or influence on the design.

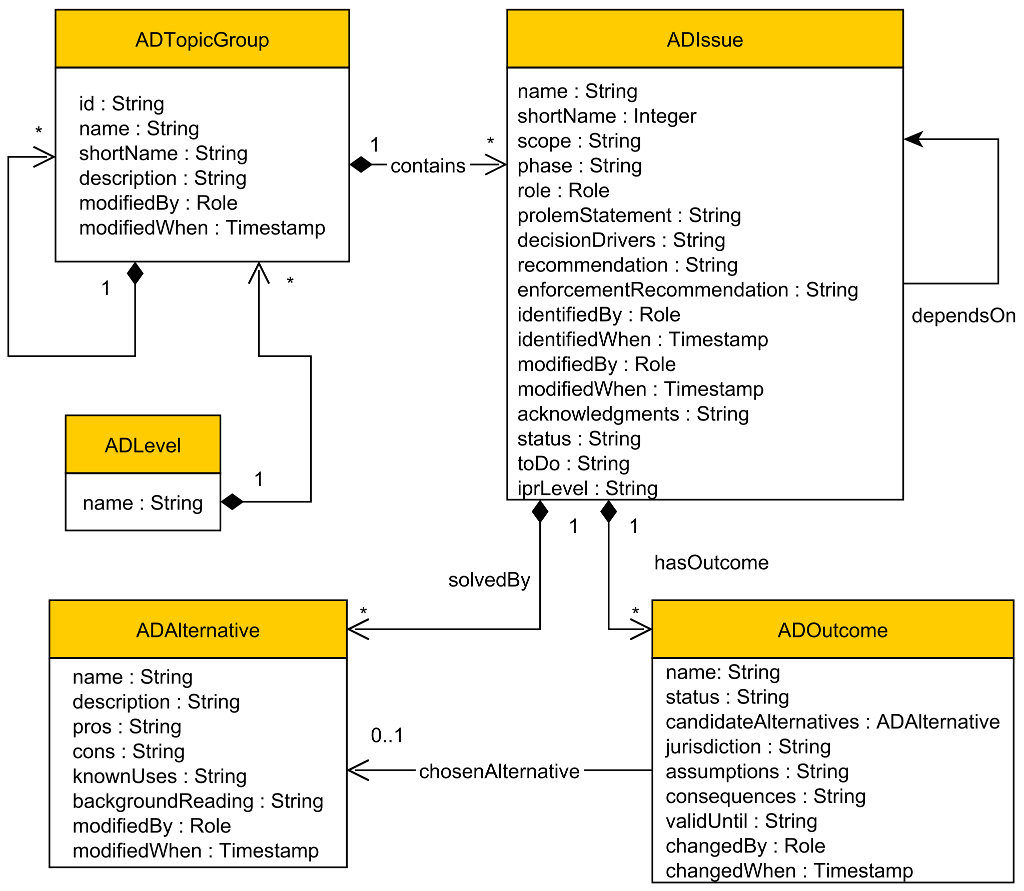


Figure 3.2. SOAD UML Meta-Model [ZKL+09]

ISO 42010 [Iso11]	**SOAD [ZKL+09]**	**CORE [dBFL+07]**	**Tyree Akerman [TA05]**
Architecture Decision	Architectural Decision	Architectural Design Decision	Decision
-	Decision Drivers	Concern	Assumption, Constraints
-	AD Alternative	Alternative	Positions
Architecture Rationale	Justification	decision loop	Argument
Concern	Problem Statement	Decision topic	Issue
Group	AD Topic	-	Group
-	Status	-	Status
depends loop	dependsOn	decision loop	Related Decision
-	-	to reflect	Related Artifact
AD Element	-	Artifact	Artifact
-	AD Outcome	-	Resulting Context, Consequence
Stakeholder	Role	Stakeholder	Stakeholder
Iteration	Phase	-	-

Table 3.1. Mapping between elements of the Tyree AD template, CORE, SOAD, and ISO 42010 Architecture Decision Meta-Models

In Table 3.1 we have summarized the incomplete mapping between the various architecture decision meta-models (ISO 42010, SOAD, CORE, and Tyree-Akerman template). The interpretation of the mapping between meta-models needs to be done with a particular care, due to the fact that it is not isomorphic. This means that there is no one-to-one mapping between the meta-model elements. For example, concern from the ISO 42010 standard only partially maps to problem statement (SOAD), decision topic (CORE), or an Issue (Tyree Ackerman), but can be used as a generalization of stakeholder requirements (functional and non-functional).

It can be noticed that the meta-model proposed by the ISO 42010 standard (see Figure 3.3) does not offer representation to some of the elements existing in the other meta-models, such as decision status or decision alternatives (see example in Figure 3.4). In order to circumvent this limitation, the standard is open for extensibility with custom viewpoints. Van Heesch et al. [vHAH12a] proposed a documentation framework that defines three additional viewpoints – chronological, relationship, and stakeholder involvement viewpoints.

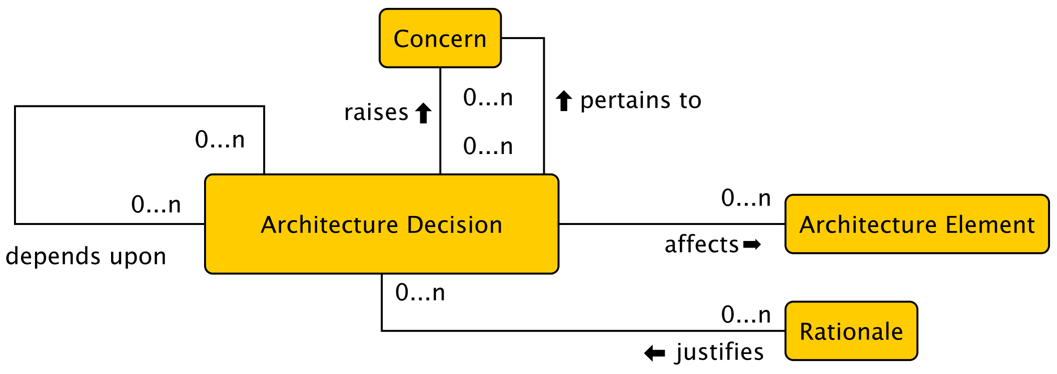


Figure 3.3. Elementary architectural decision meta-model after ISO 42010 [ISO11]

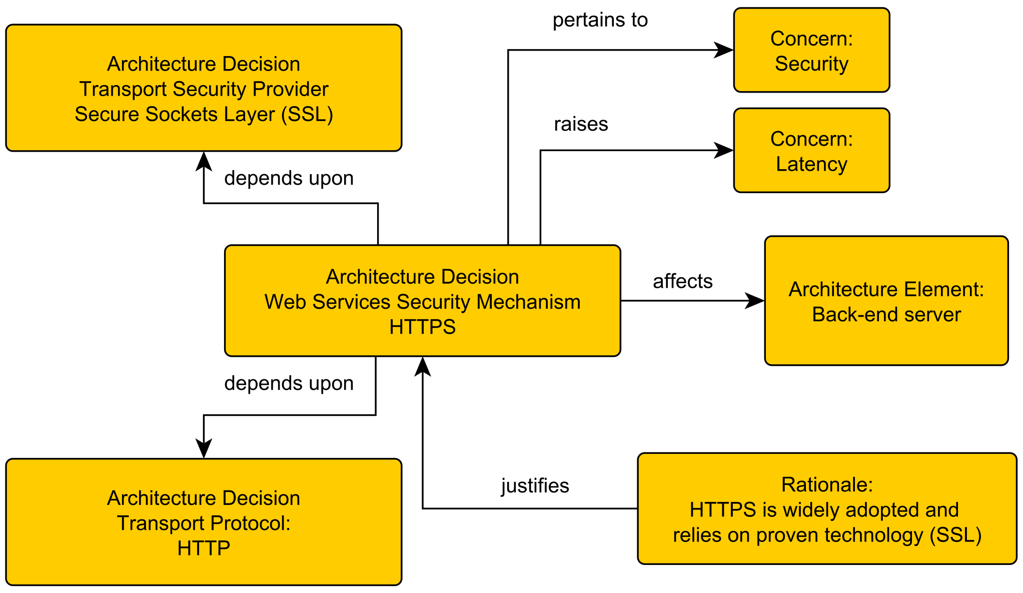


Figure 3.4. An example of architectural decision model after ISO 42010 [ISO11]

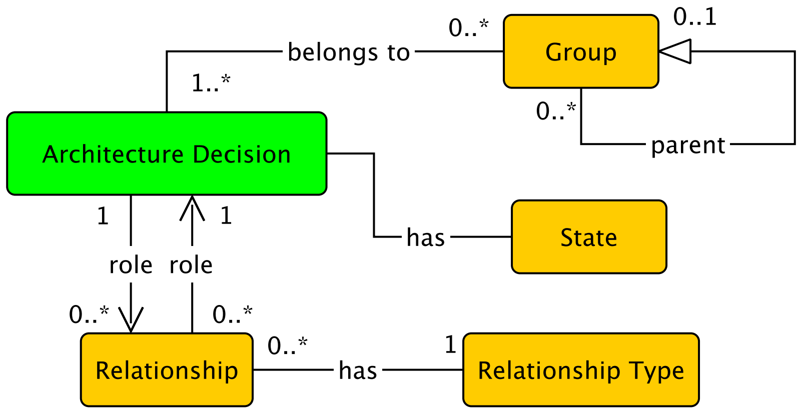


Figure 3.5. Meta-model of relationship viewpoint for architecture decision [vHAH12a]

The custom relationship viewpoint (see Figure 3.5) extends the generic *depends loop* with custom-typed relations and it is intended to make decision (change) impact analysis possible. The particular feature of this view is that it is very specific about the Relation Type. Additionally this viewpoint enables classification of the decisions within the hierarchical system of Groups. The relationship view is a snapshot of the architecture (as a whole) at a particular stage of development and contains no information about its evolution.

Contrary to the relationship viewpoint, the chronological viewpoint (see Figure 3.6) focuses particularly on the architecture's evolution. It adds the notion of Iteration, which is meant to be used to organize decisions within the design process iterations. The Iteration Endpoint has specific date and type (like Milestone, Release or Snapshot) and can aggregate multiple Iterations.

The stakeholder involvement viewpoint (see Figure 3.7) is intended to document the contribution of particular stakeholders to the decision model. Van Heesch et al. define a set of actions (such as formulate, propose, discard, validate, and confirm) that effectively change the State of the decision. According to this viewpoint, each architecture decision is related to at least one action (formulate).

The aforementioned viewpoints are not completely orthogonal; that is, some element types recur in more than one viewpoint. This redundancy is intentional and meant to improve the readability of the views.

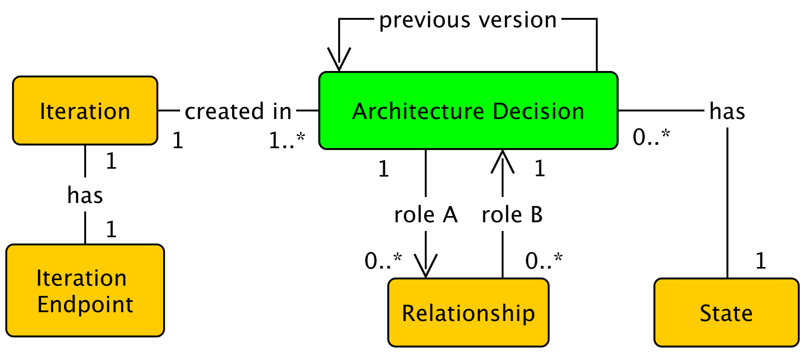


Figure 3.6. Meta-model of chronological viewpoint for architecture decision [vHAH12a]

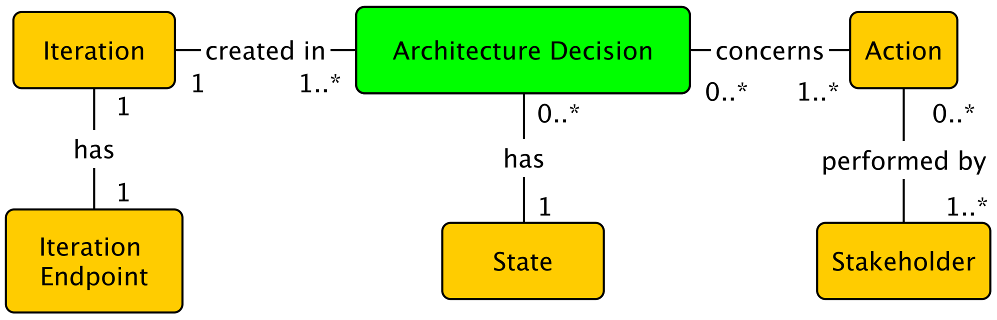


Figure 3.7. Meta-model of stakeholder involvement viewpoint for architecture decision [vHAH12a]

3.2 Architectural Knowledge Management and Decision Support Tools

In this section, we review selected architecture decision support and generic collaboration tools from the perspective of collaborative work. The design space we present is structured with design issues and design alternatives that are addressed by the architecture decisions as depicted in Figure 3.8. The description of each design alternative is accompanied by a summary of benefits and challenges together an illustrative example (see [ANP12]).

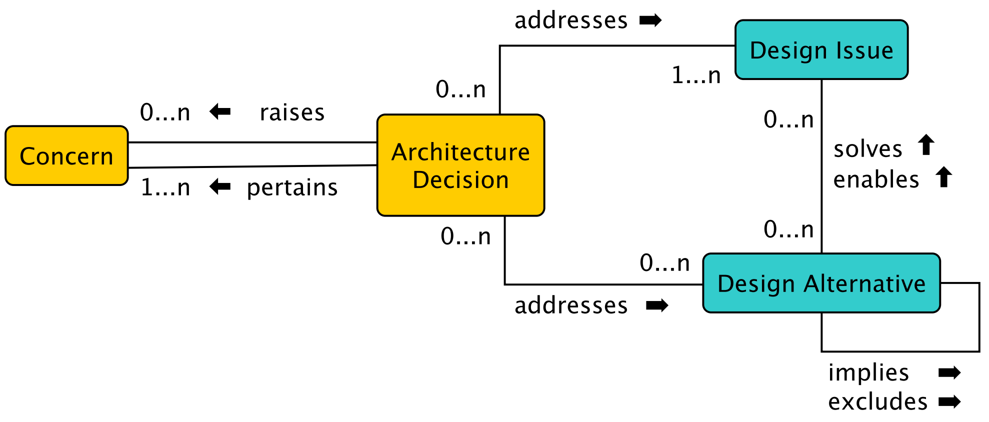


Figure 3.8. The Architecture decision viewpoint exposing relations between design space elements: design issues and alternatives

We divide the analysis into three parts, with the first focusing on the collaboration features, the second on the attributes of the adopted decision meta-model, and finally, the third inspecting the technical aspects.

3.3 Collaborative Design Paradigms

We have identified different collaborative design paradigms that are defined by the corresponding architectural decisions on the model location (centralized vs. distributed), by the update propagation mechanism (manual vs. automatic) and by the tool support for an explicit meta-model for capturing architectural

Paradigm	**Model Location**	**Synchro-nization**	**Meta-Model type**	**Examples**
Wiki	Centralized	Manual	Implicit	SEI-ADwiki
Shared Repository	Centralized	Manual	Explicit	ADDSS, ADkWIK, ADvISE, DA, Compendium, EAGLE, KA, SDA, PAKME, SEURAT
Collaborative Editor	Centralized	Automatic	Implicit	EP, GD
Blackboard	Centralized	Automatic	Explicit	ODR, SAW
Shared File	Distributed		Explicit	Archium, AREL, EA

Table 3.2. Summary of decisions leading to each Collaborative Design Paradigm level

decisions. Before we compare the various paradigms in Section 3.3.4, we present each of these issues and alternatives individually.

3.3.1 Design Issue: Model Location

The decision model is a digital representation of the deliberations happening during a design workshop. The collaborative editing of the model requires simultaneous access to it by multiple participants. This can be supported by two deployment alternatives for the model: centralized and distributed.

Design Alternative: Centralized

In the centralized topology, the consistent data model is kept on one node (e.g., a database server or a version control system repository), and multiple clients present a more or less synchronized view over the content.

Benefits: The single point of synchronization makes it possible to provide a single stream of updates by serializing all changes applied by each client. It is also clear where to find an authoritative copy of the model and from where to retrieve the latest version.

Challenges: Centralized deployments do not scale well with a large number of clients due to all traffic being processed by one server. The centralized server can also prove to be a single point of failure that can effectively break the system (not only because all information may be lost but because a failed server would hinder the actual collaboration).

Design Alternative: Distributed

The distribution of the decision model means that every peer keeps its own replica of the decision model and there is no single, master node that holds most up-to-date revision. Peers can freely synchronize the model between themselves.

Benefits: The distributed topology is robust against the failure of any single node. Design work can continue even if the central server is not available. Since design decisions are stored locally, the privacy of the designers is respected.

Challenges: The lack of a central server makes it likely that conflicts will be introduced into the model, as changes are first applied locally and independently by each peer, which will merge them only later with collaborators.

3.3.2 Design Issue: Synchronization

Collaborative work on decisions requires the model to be synchronized between all the views that are presented to each stakeholder, who may be accessing the same model over multiple devices or channels. The main challenge of achieving model synchronization involves propagating and integrating updates that can come from multiple sources, and thus solving possible inconsistencies due to conflicting updates.

Design Alternative: Manual

The manual alternative means that users need to explicitly trigger the tool to perform the synchronization. Users may save local changes and propagate them elsewhere, or decide to fetch remote changes and integrate them with their local information. In case conflicts are detected, the tool delegates to the users the responsibility for resolving them.

Benefits: Manually triggered updates give the user full control over the content that is shared with others. This can be particularly important if connectivity is limited. Manual propagation also helps to retain the semantic atomicity of changes [HL08].

Challenges: One problem with manual updates is that the synchronization feature can become very intrusive and distract from the main task at hand (i.e., making architectural decisions). The other difficulty comes from the observation that manually triggered updates tend to be coarse-grained and thus more likely to collide with the changesets from other clients. Conflict resolution for big changesets is often a very tiresome task that requires manual intervention.

Design Alternative: Automatic

Automatic synchronization implies that the tool does not require users to intervene in changeset propagation; instead it propagates to other stakeholders local changes and integrates changes of other stakeholders automatically. Typically this behavior generates a large number of fine-grained change notifications that need to be aggregated and applied to bring the model up-to-date.

Benefits: Automatic change propagation can be very helpful in dynamic, rapidly changing situations such as brainstorming sessions because of the fact that the design team can focus on the task instead of being occupied with keeping their peers up to date.

Challenges: Constrained connectivity with high latency can make frequent update propagation very difficult because it may be necessary to buffer updates in larger changesets that can be transferred more efficiently. Specific techniques (e.g., operational transformation) can be applied for automatic conflict resolution. In any case, conflicts tend to happen less frequently due to the small granularity of changes and their real-time propagation.

3.3.3 Design Issue: Meta-Model Type

The architecture decisions elicited during the design workshop are not always naturally formatted according to a specific decision meta-model [SLK09b]. The tool supporting architecture decision making can either be decision meta-model agnostic or operate within the structure of a particular meta-model.

Design Alternative: Implicit Meta-Model

The tools operating with an implicit meta-model (e.g., plain text editors, word processors, visual diagramming tools) do not enforce any particular structure on the recorded content, leaving it to the editors to ensure that the captured content fits with the required structure. These tools can be useful in informal brainstorming discussions, when few decisions that need to be taken and decisions are not related.

Benefits: The unconstrained form of the decision log can be beneficial for recording side-notes and commentary that typically would not fit into a formal decision meta-model. Such side-notes often record ideas and information that can be later refined to fit into the elements of the decision model (e.g., capturing the rationale of individual stakeholder positions).

Challenges: Since no structural constraints are enforced, it becomes difficult to ensure the structural integrity of large decision models. In particular, consistency verification and inference about related decisions would need to be computed manually and thus would require a lot of effort for large and complex dependency graphs.

Design Alternative: Explicit Meta-Model

Capturing and operating decisions within a specific meta-model constrains the form of the captured decisions.

Benefits: Giving an explicit structure to the decision model enables a tool to automatically process its content, e.g., for automated verification, quality control, and what-if analysis. Additionally, metrics about a large number of decisions can be automatically computed, aggregated and compared.

Challenges: The main problem with the strict enforcement of the decision meta-model is that it would significantly slow down brainstorming, in a particular when the design team is not very experienced at using a particular decision meta-model, or when the complexity of the tool's user interface hinders the rapid capturing of content. Another problem is that a specific decision meta-model can be unsuitable for capturing important, domain-specific aspects of the design process.

3.3.4 Design Issue: Collaborative Design Paradigm

Table 3.2 summarizes how the design issues we have illustrated so far concur in defining the collaborative design paradigm supported by a tool.

Design Alternative: Shared File

Collaboration based on the exchange of decision models persisted in files is one of the easiest alternatives to adopt, since it completely separates the concern of modeling decisions (which is done using an editing tool that operates on local files) from the concern of sharing information about the decisions (which can be done with different file sharing approaches). Persistence in files is the simplest way to store decision models. The data encoding can vary between plain text and binary encoding, but the main characteristic of file-based collaboration is the choice of the file sharing mechanism, which can vary from a centralized file server (e.g., NFS or SMB) to a cloud-based folder synchronization service (e.g., Dropbox[1] or BitTorrent Sync[2]). These impose the constraint that there is only one active editor of file at a time. More refined mechanisms for conflict detection and merging are offered by version control systems such as Subversion[3] or Git[4]. These, however, may not always be able to merge conflicting updates for decision models stored in binary files.

Benefits: An advantage of the fact that each stakeholder can own his own copy of a file is that there is no need for continuous connectivity for read-only operations.

Challenges: The major problem with file-based decision model persistence is the exchange of updates among stakeholders. In a situation in which two or more clients modified the decision model simultaneously, the file-sharing mechanism would not be able to help with merging resulting conflicts. Version control systems can help to detect and manage conflicts. However, merge operator applied to the plain-text file representation is unaware of the model semantics.

Design Alternative: Wiki

Wiki systems [LC01] are typically used to manage generic knowledge repositories, which can be contributed to and shared by many stakeholders. The main characteristic of the wiki-based collaboration is that it remains model-agnostic, since it typically provides a simple formatting language to describe the syntax of Web pages and automatically infer the presence of hyperlinks.

[1]`https://www.dropbox.com/`
[2]`https://www.getsync.com/`
[3]`http://subversion.apache.org/`
[4]`http://git-scm.com/`

Benefits: The main advantage of using a wiki-based system to support a collaborative decision making tool lies in the bulk of the functionality that can be reused. In particular, we refer to content micro-formatting, revisioning history, access control, and persistence.

Challenges: Wikis can mitigate problems that are due to the lack of support for an explicit meta-model by means of templates, whereby users may choose among different structures for new pages. However, no guarantee can be made that users will not modify the predefined structure given by the template. Wikis also offer various forms of locking at the level of individual pages to avoid introducing conflicts in the content; that is, while a user is editing a page, other users may be prevented from doing so on the same page.

Design Alternative: Shared Repository

The shared repository paradigm builds upon the **wiki** alternative, with the difference that the repository is aware of a meta-model that has been tailored to capture architectural decisions. In this category ,we include content management systems, semantic Wikis, and form-based user interfaces that are customized to a specific decision meta-model. Shared repositories also feature a centralized deployment of the model and manual synchronization.

Benefits: Collaboration over shared repositories offers functionality that is specifically designed to support the whole lifecycle of architectural decisions, it can simplify the interpretation of the decision model and provide guidance inferred from the content of the decision model itself.

Challenges: The main limitation of shared repositories stems from the manual synchronization approach. In dynamic situations, where multiple stakeholders are changing the decision model rapidly and concurrently, a shared repository gets in the way. In such environments, the user interface views of the decision model often become out of date, and users need to dedicate their attention to refreshing them. To deal with this issue, some tools offer an automatic refresh feature, which – when implemented using periodic polling – may limit the scalability of the system without necessarily providing a robust solution to deal with conflicts.

Design Alternative: Collaborative Editor

Collaborative editors eliminate the limitations of wikis and shared repositories by supporting real-time synchronization of the content. Basic collaborative editors do not make any assumptions about the content meta-model and leave it up to the user to follow some conventions for capturing the knowledge, which is instantly shared with all other stakeholders.

Benefits: Time plays a very significant role in collaborative decision making. Thanks to the proactive delivery of the model updates with low-latency, it is possible to keep all stakeholders up to date. This feature can potentially improve the efficiency of the deliberation process.

Challenges: In situations when there is high latency between the server and the clients, it is likely that conflicts will still occur. Some approaches have been proposed to resolve such conflicts automatically. By introducing some techniques to indicate the presences of and the focus of the attention of users, it is possible to prevent conflicts, since users become aware of what the others are doing with the content [HGS$^+$13].

Design Alternative: Blackboard

The blackboard-based collaboration paradigm combines the benefits of the real-time collaboration of **collaborative editors** and the advantages of operating with a rich representation of the data, following an explicit meta-model. This means that stakeholders can rapidly go through a design discussion and hopefully converge on a consensus, while the tool takes care of recording the outcome according to the explicit structure of the decision model.

Benefits: Rapid feedback, team situational awareness, and structural integrity are the main benefits delivered by the blackboard collaboration paradigm.

Challenges: The concern for blackboard systems is that the rapid exchange of the changesets implies that large model changes are split into a large amount of small updates. Concurrent model editing can make it impossible to execute a large model change fully due to emerging interrelations such as dependencies or exclusions, thus leaves the user puzzled by a model entanglement that is difficult to fix.

Liveness	Synchronization	Conflict Prevention	Conflict Resolution	Example
None	Manual			Archium, ADkWik, ADDSS, EA, EAGLE, AREL, DA, SDA, SEI-ADWiki, SEURAT
Low	Automatic	Locks		Compendium
Medium	Automatic	No	Manual	ODR
High	Automatic	No	Automatic	EP, GD, SAW

Table 3.3. Summary of decisions leading to each Liveness level

3.4 Liveness

We propose to adopt the concept of liveness introduced by Tanimoto in [Tan90] to describe the level of responsiveness of the decision model to changes introduced by multiple stakeholders. In addition to the previously discussed Synchronization issue (Section 3.3.2), for which a manual alternative implies a non-existent liveness, we further refine the level of liveness based on how the tool supports Conflict Prevention and Conflict Resolution, as detailed in the following Sections.

3.4.1 Design Issue: Conflict Prevention

Concurrent read-write access to a decision model is an inherent necessity for enabling collaborative decision making. As long as concurrent read-only access does not present any difficulty, the concurrent reading and writing can cause conflicts. Locking mechanisms help to prevent more than one user from modifying the model (or parts of it) at the same time. The locking mechanism can be activated either explicitly by the user, or implicitly by the tool.

Design Alternative: None

Without a locking mechanism, concurrent write access to the decision model typically relies on the merging and conflict resolution mechanisms to handle cases in which more than a single user modifies the model at the same time.

Benefits: The core benefits of not using a locking mechanism is that it eliminates the complexity of protocols [Tau06] that need to deal with corner cases, such as stale locks, or client starvation. Removing locks also simplifies the user interface, since users do not have to be aware of the lock status of different model items, nor need to be granted access to the locks.

Challenges: Not using an explicit locking mechanism can be troublesome if changes introduced to the model are very interdependent. In such cases, the result of the changes might be either inconsistent or incomplete, thus inducing large rollback overhead.

Design Alternative: Locks

An explicit locking mechanism acts in two ways: first, it prevents concurrent access to particular elements of the decision model, and second it grants a user the exclusive access to the model and informs others about this fact.

Benefits: Having the ability to lock the decision model may help to force a diverging design discussion to converge, since only one user is empowered to perform updates to the model, i.e., to settle controversial decisions. Thus, being certain of having exclusive access to the model helps when the atomicity and consistency of the change need to be enforced.

Challenges: Locks, if applied inappropriately, can significantly hamper the throughput of concurrent work on the decision model. In particular, when large subsets of design issues, or even the whole model, are locked by a single user, others cannot actively progress with their contributions, but can only watch the progress of the other user, which may hamper an active brainstorming session during a design workshop. In such case, another communication channel might be needed to provide the input.

3.4.2 Design Issue: Conflict Resolution

In situations in which multiple users edit an architecture decision model concurrently, the changesets need to be merged in a way that preserves model consistency. We have observed two alternative approaches to solving this problem:

Design Alternative: Manual

With manual conflict resolution, the user is notified about the occurrence of the conflict and is explicitly asked to decide how to resolve it.

Benefits: The main benefit of this approach is that the user remains in full control of the outcome, so there is no chance of implicit model corruption that would escape the users' attention.

Challenges: Frequent updates leading to a large number of merge alternatives can overwhelm users. Also, the conflict notifications can be a very significant distraction, and manually solving conflicts introduces a delay in the information flow.

Design Alternative: Automatic

The user is not involved in the conflict resolution process. Typically a rule-based arbitrage algorithm is used to deterministically merge the changesets [EG89]. The more fine-grained the changesets are, the easier it is to apply the arbitrage algorithm consistently over the colliding changesets.

Benefits: The main advantage of automatic conflict resolution is its seamless operation when used with simple, linear data models, such as plain text.

Challenges: In situations in which changesets are large, or they are applied to non-linear data models (such as graphs), the arbitration results may no longer fit with the users' expectations. In turn, since conflicts are merged quickly without user intervention, there is a chance that for the model to become corrupted without the users being aware of it.

3.4.3 Design Issue: Liveness

In the context of the three design issues that we introduced in the previous sections, we propose to revisit the concept of liveness introduced by Tanimoto in [Tan90] and adopt it within collaborative architecture decision making as follows. Based on the combination of decisions (choices) across the alternatives related to the aforementioned issues, here we devise four distinctive levels of liveness.

Design Alternative: None

If a tool does not deliver any real-time collaborative decision making support, then one cannot speak about its liveness level. In other words, the absence of liveness is characterized by the need to **manually synchronize** the decision model.

Benefits: Manual synchronization significantly reduces the complexity of the tool design and thus the corresponding implementation effort as well, which may reuse existing information sharing technologies.

Challenges: If a design is to be made in a collaborative way, and the tool does not provide any automated mechanism for automatically sharing architectural knowledge and decisions, the stakeholders need to resort to alternative communication channels and generic information sharing tools, such as configuration management and version control systems (Subversion, GIT), file-sharing (Dropbox, BTSync) and video-conferencing, remote presence tools (e.g., Skype).

Design Alternative: Low

The low level of liveness characterizes tools that provide **Automatic** model synchronization combined with a pessimistic conflict prevention mechanism based on **locks**.

Benefits: Locking parts or even the whole model for exclusive access is a very simple and effective method of avoiding changeset collisions. The low liveness support is definitely suitable for stepwise refinement within a controlled and slow-paced workshop environment.

Challenges: Incautious use of locking can very easily lead to stale locks and thus significantly hamper the decision making process. In order to prevent that from happening, the decision making process needs to be carried out in a systematic way.

Design Alternative: Medium

The medium level of liveness differs from the previous one by not relying on the locking mechanism for conflict prevention. Instead, a conflict detection and mitigation mechanism is implemented. In particular, the conflict merge mechanism differentiates between the changesets that can be safely merged, and those that require users' intervention.

Benefits: The advantage of the medium level of liveness over the low level shows up in the situations in which multiple users edit the model simultaneously. The ultimate consistency of the model is assured by the user curating the model in cases where a collision is detected.

Challenges: Again, manual collision resolution can prove to be very inefficient in cases where multiple users intensively work on the same part of the model.

Design Alternative: High

The high level of liveness is delivered by tools that perform an automatic model synchronization and handle changeset merging in a manner that does not involve users, even in case of conflicts.

Benefits: The biggest benefit of a high degree of liveness is that it provides a very streamlined experience for users. This is particularly important in tightly time-constrained discussions and very dynamic situations.

Challenges: Carefree conflict resolution, without human oversight, might result in a blowback when the changesets get merged in a way that is consistent according to the merging rules but not anticipated by the team. Without explicit warnings built into the tool user interface, the conflict can remain unnoticed and thus corrupt the model.

3.5 Tools

In this section, we provide a very brief description of the tools that we surveyed in the process of building the design space that we have introduced earlier. Analyzing the application domain we have distinguished four categories of tools, which we arrange with increasing levels of specificity:

Knowledge – generic tools that can be applied in broad range of applications, typically relying on an implicit meta-model,

Decisions – tools focused on the decision support domain that are not customized to the domain of software architecture, and that operate on a minimal, but expandable explicit decision model,

Architectural Knowledge – knowledge management tools, tailored to the domain of software architecture, intended to manage a broad range of knowledge artifacts,

Architecture Decisions – a subset of **Architectural Knowledge** management tools that focus specifically on Architecture Decisions.

EtherPad (Domain: Knowledge)

EtherPad (EP) [GIZ08] is a collaborative, real-time editor (RTCE) enabling multiple authors to work on a single, unstructured text document. It is based on the Operational Transformations (OT) engine and the EasySync communication library. As an editor, it offers elementary text formatting features, comparable to those found in the Rich Text Format (RTF) specification. The EtherPad does not offer any argumentation, or process integration support features. The content contributed by the individual users is color-coded to ease identification. When applied as a tool to support a decision making process, there exists neither an automated way to enforce the application of some particular decision model, nor a mechanism to ensure its consistency.

Google Docs (Domain: Knowledge)

Google Docs (GD) [Goo13] is a cloud-based tool suite providing a set of generic office applications (Document, Spreadsheet, Presentation, and Drawing). It offers near-real-time collaboration, with rich-structure documents. The integration of the individual user contributions is automatic, and there is no explicit traceability of authorship for edited content, although it can be done implicitly by identifying changeset authors. A document meta-information can be shared between users using document annotations. Similarly to the EtherPad, Google Docs can be used to record architecture decisions in free form, but the tool does not provide any explicit decision making and modeling features. Differently from EtherPad, Google Docs does offer some limited degree of offline operation support.

Compendium (Domain: Decisions)

The Compendium [SSS+01] is a social science tool developed at the Open University. It implements the Issue-Based Information System concept introduced by Conklin and Bergman in [CB88]. It does not provide any software architecture related facilities. Decision model persistence is done either in the organization-centralized database or locally on the computer running the Compendium. There exists an experimental, live-collaboration extension that uses the jabber protocol to provide asynchronous decision model element sharing between users who are working concurrently, but there is no automated support either for validation or for conflict resolution within the model.

SEI-ADWiki (Domain: Architectural Knowledge)

Bachmann et al. [BM05] documented their experience of replacing documentation edited with a conventional rich-text editor (Microsoft Word) with a collaborative wiki (MediaWiki). The SEI-ADWiki initiative does not adhere to any specific architectural knowledge (or decision) meta-model, even though it extends the basic wiki editor with more structured, form-based editing views.

SEURAT (Domain: Architectural Knowledge)

Software Engineering Using RATionale (SEURAT) [Bur05] is an annotation-based rationale management system aimed at supporting software maintenance. It is built as an Eclipse plug-in extension. Structurally, it consists of the *argument editor and analyzer*, *rationale repository*, *inference engine*, and *argument ontology storage*. Within the RATspeak AK model it offers management of entities such as Requirements, Goals, Alternatives, Claims, and (design) Artifacts.

EAGLE (Domain: Architectural Knowledge)

EAGLE [FLvV07b, FLvV07a] is a Microsoft-SharePoint-based architectural-knowledge-sharing portal built within the GRIFFIN project at the VU University Amsterdam and the University of Groningen. EAGLE operates within a domain model (CORE) built on the basis of an extensive investigation of architecture design processes and interviews with practicing architects.

PAKME (Domain: Architectural Knowledge)

Process-based Architecture Knowledge Management Environment (PAKME) is a situational application by Babar et al. [BGJ05]. It is based on the open-source groupware platform (HyperGate), which was customized to fit the requirements for architecture design. The underlying platform provides it with typical groupware facilities such as user management and access control, but it does not provide concurrent access protection or model consistency validation. The decision rationale is captured individually for each architecture decision, and there exists no support for capturing multi-user argumentation.

AREL (Domain: Architectural Knowledge)

Architecture Rationale and Elements Linkage (AREL) [TJH07, Tan07] is a UML-based model to capture and document design rationale. It builds upon the generic modeling tool Enterprise Architect and Bayesian network analysis tool (Netica[5]). It provides decision support for qualitative (QLR) and quantitative (QAR) decision making, as well as decision support for architectural element traceability. It provides neither explicit collaboration nor argumentation features, as it relies on file-based architecture model persistence. It does support decision conflict resolution and model validation.

Knowledge Architect (Domain: Architectural Knowledge)

Knowledge Architect (KA) [LJA09, LJA10] is a tool suite designed to provide holistic support for architectural knowledge management. It comprises the centralized knowledge hub that is accessed by the components responsible for the

[5]http://www.norsys.com/netica.html

traceability management, automated checking (validation and conflict mitigation), and knowledge translation. A range of plug-in extensions was implemented to integrate the core knowledge repository with generic office suite applications (MS Word, Excel). It offers model flexibility thanks to the ontology-based decision model specification.

ADDSS (Domain: Architectural Knowledge)

The architecture design decision support system (ADDSS [CNPD06, CDN10]) is a web-based tool aiming at architectural knowledge management. It supports capturing and delivering architectural decisions, as well as functional and non-functional requirements and architecture documentation. It operates on a custom, yet flexible architectural decision knowledge representation and offers partial support for an iterative decision making process. It offers multiple viewpoints and perspectives on decision models.

Archium (Domain: Architecture Decisions)

Archium [JvdVAH07] is a research tool prototype developed at the University of Groningen to demonstrate practical applications of the theory of Software architecture. It served its purpose within the Grid for information about architectural knowledge (GRIFFIN). The core part of the Archium tool-chain is an architecture description language (ADL) compiler that can generate ArchJava and Java code. The Archium decision models are automatically validated and checked for decision consistency.

Architectural Decision Knowledge Wiki (Domain: Architecture Decisions)

AD_{kwik} is a situational wiki application presented by Schuster et al. in [SZP07]. It is built on top of the IBM proprietary QEDWiki system. It uses the SOAD framework-specific decision meta-model and offers support for the complete decision management life-cycle. The decision rationale (justification) is recorded as an attribute of the ADoutcome entity. There is no explicit support for the decision argumentation, but there exists a QEDWiki generic message board functionality, specific for every wiki entity. The QEDWiki locking mechanism prevents from concurrent write-access to the decision model elements, but there is no mechanism to validate the consistency of the decision model. The tool does not provide any push mechanism to notify involved stakeholders about concurrent changes in the decision model.

Open Decision Repository (Domain: Architecture Decisions)

Open Decision Repository [AAE$^+$14] is a tool initiated by the Software Engineering and Architecture Group at the University of Groningen, which was further developed within the Google Summer of Code project. It is implemented as a rich web application with the back-end provided by the J2EE technology stack. It supports architectural decision modeling within the ISO42010 meta-model with four additional viewpoints proposed by van Heesch in [vHAH12a]. ODR provides neither a locking mechanism, nor automatic changeset conflict resolution mechanism.

Solution Decision Advisor (Domain: Architecture Decisions)

Solution Decision Advisor (SDA) [MZ11] is an Eclipse RCP application that implements a (logically) layered architecture system built to manage architectural knowledge entities based on the custom decision-points meta-model. Users operate the system by using the Decision Point Knowledge Editor and the Decision Making Client implemented within the Eclipse Rich Client Platform, and the Eclipse Graphical Modeling Framework. SDA is designed as a multi-layered application with extensibility points planned within each layer. It offers both decision meta-model and decision making process flexibility. It also offers facilities for model consistency verification and conflict resolution.

ADvISE (Domain: Architecture Decisions)

The Architectural Design Decision Support Framework (ADvISE) [LTZ13] is an Eclipse-based Tool suite developed by the Software Architecture Group at the University of Vienna. Thanks to the adoption of OSGI-based extensibility, it offers a number of modular features such an Architectural Knowledge Transformations Toolkit, Architectural Decisions Modeling, and Fuzzy Decision Support. The main focus of the toolkit is knowledge and decision traceability and inference. It does not offer any explicit support for collaborative design and decision making.

Decision Architect (Domain: Architecture Decisions)

Enterprise Architect (EA)[6] is a commercial suite developed by SparxSystems. It is a versatile and expandable architecture modeling tool, broadly adopted in the industry. Manteuffel et al. [MTK$^+$14] presented an extension that implements documentation framework for architecture decisions proposed by van Heesch [vHAH12a]. The Decision Architect (DA) [KG14] has been developed at the University of Groningen in cooperation with ABB. It supports architecture decision documentation using five specialized documentation views proposed in [vHAH12b]. In terms of collaboration it relies on the infrastructure provided by the Enterprise Architect framework, that is either file-based or database backed blackboard model exchange.

[6]http://www.sparxsystems.com/products/ea/11/index.html

3.6 Potential Gap

After investigating the design space covering Collaboration Paradigm and Liveness, we realized that none of the surveyed tools supports a combination of blackboard-style collaboration and a high level of liveness. The need to address this gap in the state of the art laid a foundation for the design of the Software Architecture Warehouse (SAW).

Software Architecture Warehouse (Domain: Architecture Decisions)
Software Architecture Warehouse (SAW) [NP13] is a web-based tool supporting collaborative architecture decision making. It aims at promoting a high level of team situational awareness by delivering multiple, customized views of the decision model. SAW offers live updates, decision model flexibility, and modular extensibility.

3.7 Summary

In this chapter, we have surveyed the design spaces of the architecture decision meta-models and the collaborative aspects of the tools supporting architecture decisions modeling and management. Within the set of seven design issues, we have elicited two core issues addressing Collaborative Design Paradigm and Liveness. In total, we have inspected 19 alternatives and documented their advantages (pros) and disadvantages (cons). The presented decision space covers the state of the art and lead to identification of the goal to be filled in by the original contributions of this thesis.

In next chapter, we scope the field of our research, we analyze typical software architecture design scenarios and we identify core concerns for the stakeholders. Finally, we state research problems and define research questions that lead to our research thesis.

Chapter 4

Collaborative Software Architecture Decisions

In this chapter, building on the foundation of the background research and the state of the art presented earlier, we are going to guide the reader through the major practical concerns of some practitioners from the field. Later in this chapter, we define the Research Problems with corresponding Research Questions, and finally we state our research thesis.

4.1 Typical Scenarios

A software architecture design process typically can be framed into one of three scenarios, namely, Architecture Synthesis, Architecture Evaluation, and a mixture of the two. Here we provide brief descriptions of some typical project cases comprising such scenarios.

4.1.1 Architecture Synthesis

An in-house software architect working within the research and development department of a high-tech company is invited to work on an innovative, green-field project of a new product. The product requirements list is underspecified, and stakeholders' opinions are volatile. Nevertheless the team, through construction of many prototypes, discovers numerous patterns, methods and techniques for addressing stakeholders' concerns. The architect is concerned about the fact that the experience gathered during the construction and evaluation of the prototypes might either remain limited to a narrow group of prototyping experts or evaporate completely. In order to avoid that happening during the process of the product design, he or she carefully captures design issues and alternatives recognized during the design process thereby filling up the design space with content. As prototypical designs vary, each of them constitutes a (partially complete) alternative solution representing the set of design decisions. Together with the maturing of the product, the design space is filled with design alternatives offering diverse qualities for the final design. Design issues that require more research to explore new alternatives are related to the *unknown* alternative.

Due to the fact that there are multiple (non-technical) stakeholders involved in the design process, reaching a consensus in decision making within the context of particular design issues is a non-trivial task.

In cooperation with analysts, stakeholders are expressing business requirements by using qualitative and quantitative attributes attached to specific design alternatives, decisions, and decision points. Because of geographical distribution and low availability of stakeholders, many decisions need to be taken offline in a step-wise manner.

4.1.2 Architecture Evaluation

A medium-sized company has exhausted the capacities of its computerized order management system and thus has decided to invest in a new system suitable to its needs and scalable to adapt to future growth. A group of analysts has prepared concise documentation of the requirements and handed it over to an external team of architects. Based on the requirements specification, the architects have produced an intermediate architecture design candidate together with decision and rationale documentation. An independent external auditor is hired to assess the qualities and the progress of the architectural design. The alignment of the design with the requirements is analyzed with ATAM [KKC00] (or similar architecture evaluation methods). In order to estimate the design progress, the project manager wants to find out how many design issues are fully decided, how many were partially decided and finally, how many have not yet been not addressed. The architect is aware of the fact that design issues are very diverse in terms of complexity and influence on the design. In order to verify that the design is on the right track, he or she wants to be sure that issues with high impact and/or high complexity are addressed first and foremost.

4.1.3 Synthesis and Evaluation

An in-house architect of a highly specialized manufacturing company is asked to join a team brought together for the purpose of designing an innovative product. The design situation is a typical brown-field project case. The company has many years of experience in the field, but the architect has been with the company only for the past three years. Earlier designs were made by a team of two tenured architects, one of whom retired, the other of whom passed away. Over the years, the company culture encouraged explicit documentation of the design decisions taken. The product that is to be delivered consists of multiple sub-systems that were previously offered as separate products. The architect wants to reuse the experience gathered by his predecessors so that he will not repeat the mistakes and benefit from the experience of his predecessors. The first step is a preliminary identification of the design issues important for his design and then a pruning of the irrelevant ones. Having done this, he inspects past decision records in order to identify decisions that recurred in successful designs.

4.2 Scoping the Research Problem

The experience gathered through our past practice in software architecture design and collected from the community of active practitioners allowed us to recognize the following concerns about the design process:

Co$_1$ FRAGMENTATION OF EXPERTISE AND/OR DECISION POWER

Due to the complexity and heterogeneity of the topics addressed by architectural decisions, the expertise and decision power required to make good architecture decisions is often distributed among multiple experts. The architecture design workshop would typically gather the experts physically in a single room to facilitate the most efficient - direct communication. In cases where physical presence was not an option, the experts would telecommute with audio/video conferencing equipment. The constrained bandwidth of communication can make the problem of expertise fragmentation even more pronounced.

Co$_2$ INEFFICIENT CONCILIATION

By its nature, the brainstorming process can have a very non-linear pace. Sharp focusing and agility definitely help in the process. Unfortunately, organizational inertia and latency of communication can significantly hamper the process of reaching agreement about a decision (conciliation). In particular, in the brainstorming sessions including remote peers, keeping discussion focused and attaining constructive argumentation can be a very difficult task to accomplish.

Co$_3$ CHAOTIC BRAINSTORMING PROCESS

The project design space is a set of diverse architecture decision artifacts, such as design issues, design alternatives, positions, relations, and others. A well-modeled decision space is a great aid for the traceability and transparency of the decision process. In practice, building a well-structured decision model requires effort. This effort can be significantly reduced by providing efficient modeling guidance.

4.2.1 Team Situational Awareness and Architecture Design

As we introduced in Section 2.3.3, high-quality, collaborative architecture decisions require three conditions to be fulfilled. First, the decision makers need to be well informed. Second, they need to be smart in analyzing available information. Finally a good collaborative decision needs to be consensual.

Providing stakeholders with the right information about the context of the software architecture design is covered by disciplines such as requirements analysis [vL08] and architecture modeling and, therefore, lays beyond the scope of our research. Similarly, there exist plenty of computational methods aimed at assisting humans in decision making [PIPWJ08]. In our work, we make the assumption that the decision makers are sufficiently informed to make the decision in question and have enough skills to take the decision. Based on this assumption, we have focused our efforts on investigating the collaborative aspect of the architecture decision making process, and have distilled the concerns listed in the previous section into the following research problems:

4.2.2 Collaborative Architecture Design Decision Consensus (RP$_1$)

Collaboration [Gru94] has been recognized as a crucial element for effective engineering design [FWC93, SHL08] and planning [Rit72] for a long time. In [Par09] Parnas positions design documentation as a communication vehicle and emphasizes the importance of appropriate organization in order to provide knowledge accessibility. In the context of architecture decision making, collaboration among design team members plays an essential role in delivering high-quality solutions matching stakeholders' requirements and fitting within project constraints. To this end, we devise the following research question:

RQ$_1$ How to support collaborative software architecture decision making?

By supporting collaboration through improved communication among distributed members of the design team, we want to improve the quality of the decisions and thus the quality of the design process and design outcome. Architects are naturally at the center of collaborative decision making due to their central position among stakeholders, developers, and project management [Zim09, Chapters 2 and 3]. A positive correlation between system structure (integrity) and the social organization of the design team has been pointed out by Conway in [Con68]; hence, effective collaboration in decision making is essential for design integrity. Support for collaborative decision making aims at helping to transform a design “stream of consciousness” into a high-quality, reusable design body of architectural knowledge [Lag09]. We acknowledge the fact that the factors impeding design collaboration are multi-dimensional. Spatial distribution sets different demands for supporting collocated and distributed teams. Similarly, intensive, low-latency, highly interactive brainstorming has very different characteristics from a steady collaboration over a long period of time. The challenge that we see is that replacing face-to-face communication with technology needs to strike a very delicate balance between the overhead of the tool and the simplicity of natural, direct communication.

4.2.3 Quality of the Collaborative Architecture Decisions (RP_2)

The necessity of having explicitly documented architectural knowledge is broadly recognized by software designers [Zdu09]. Solely the fact of having explicit architectural knowledge does not automatically imply its utility for the design process. Following the thought of Tom DeMarco - "*You can’t control what you can’t measure*" [DeM86] - we claim that it is impossible to make good collaborative decisions [JB05] without understanding the characteristics that comprise the decision quality.

Taking into account the trend that puts architecture decisions in the center of the software architectural design [TMD09], a question about their quality naturally comes to mind. By learning about design decisions quality, one gets very important insight into the process that led to the creation of the design (see discussion on the quality of software architecture in Section 2.1.1).

As we have pointed out in Section 2.2.2, the software architecture design, comprising decision making, has properties of a wicked problem. This means that there is no analytical process that would lead to the ultimately correct solution. Therefore, we focus on the qualities of the decision making process that contribute to quality software architecture decisions (see Section 2.3.3).

RQ$_2$ How to identify, and quantify properties of a good, collaborative design

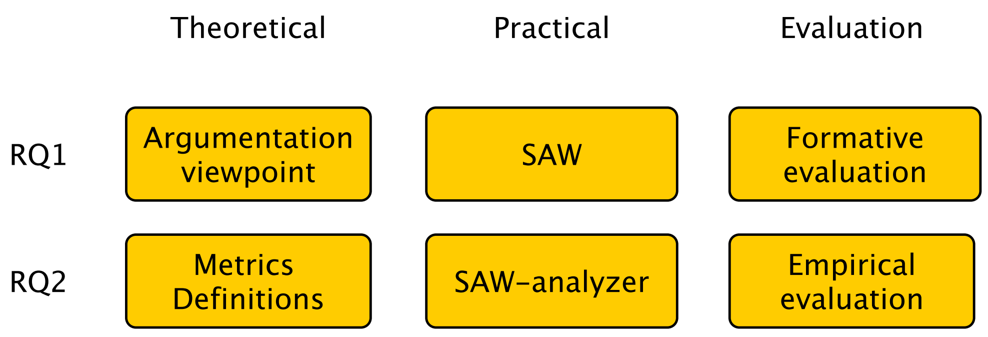


Figure 4.1. The meta-model of this dissertation

decision making process?

The estimation of the quality of the decision making process affects both architectural synthesis and evaluation. During the architectural synthesis [HKN$^+$07], the design team motivated by the desire to deliver the best design quality is going to strive for the best quality of the input (i.e., the reusable architectural knowledge) on which to base their design. In architectural evaluation [KKC00], post-mortem analysis of the qualities of the knowledge used in the design process opens a new perspective for the evaluation of the quality of the architecture itself. Measuring the qualities of the architecture decisions can also be employed for tracking the progress and dynamics of the design process, as well as used as a detailed source of information for the evaluation of the design output qualities.

4.3 Research Thesis

Based on the research problems and research questions stated in the previous sections, we state the following thesis of our research:

> *Support for low-latency, structured architecture decision argumentation improves the quality of the collaborative decision making process.*

In order to address this thesis we propose a research meta-framework (see Figure 4.1) that comprises three following aspects:

Theoretical – where we devise models, definitions and formal theory addressing a Research Question,

Practical – in which we provide tools and methods implementing the aforementioned theory,

Evaluation – where we report on the results of applying theory and practice in the experimental environment.

In order to address Research Question 1 (**RQ**$_1$), we devised a decision argumentation viewpoint extension to the ISO 42010 standard (see Section 5.1). In order to employ the argumentation viewpoint in practice, we have implemented the Software Architecture Warehouse, which we describe in Chapter 6 and discuss its formative evaluation in Section 8.1.

In the context of Research Question 2 (**RQ**$_2$), on a theoretical level, we devised a range of metrics to describe the structure and dynamics of the collaborative design decisions within the framework of the argumentation viewpoint introduced in Section 7.1. The practical aspect is an implementation of the metrics provided by the SAW-analyzer described in Chapter 7.2.3. Finally, Section 8.3 reports on the results of measuring design decision argumentation during the architecture design sessions.

4.4 Summary

In this chapter, we briefly introduced three typical scenarios concerning software architecture design process - Architecture Synthesis, Architecture Analysis, and a mixture of the two. Based on this scenarios, we identified three practical concerns and devised two research problems; one related to the support of the collaborative decision making and another to the estimation of qualitative and quantitative properties of collaborative decision making process. Finally, we stated the research thesis and defined an approach for addressing research questions, based on three pillars - Theoretical, Practical, and Evaluation.

In the following Chapter, we develop theoretical foundation for the collaborative architecture decision modeling – the decision argumentation viewpoint followed by its practical implementation – the Software Architecture Warehouse.

Chapter 5

Architecture Decision Argumentation Viewpoint

In this chapter, we provide the theoretical framework that lays the foundation of our applied contribution described in Chapter 6. In particular, we provide an argumentation viewpoint extension for the architecture decision model. Next, building upon this viewpoint, we deliberate over the decision argumentation lifecycle. Finally, we devise a set of software architecture metrics focused on estimating the dynamics of the architecture decision making process.

5.1 Decision Model and Argumentation Viewpoint

The starting point of our considerations is the decision meta-model proposed in the standard ISO 42010 [ISO11] (see Figure 5.1). We propose to use the Architectural Decision entity (see Figure 5.2) 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 [KLvV06] with a relation type named "is an alternative to" which is meant to relate decisions "addressing the same issue". Similarly to [JB05], we propose to promote the Design Issue as a first-class entity. An advantage of representing design issues and design alternatives explicitly is that the identification and reuse of design decisions is promoted [NPZ10].

Conforming to the recommendation about viewpoint definition from the ISO 42010, specification Annex B, the Decision Argumentation Viewpoint that we propose addresses the following concerns:

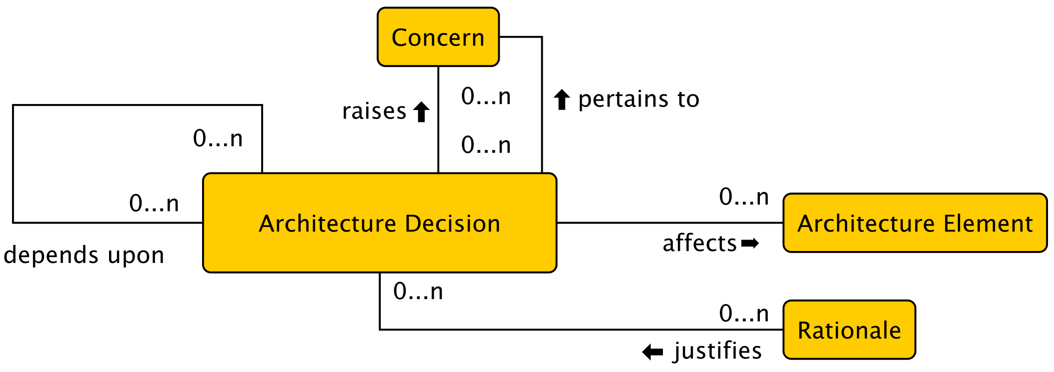


Figure 5.1. Elementary architectural decision meta-model after ISO 42010 [ISO11]

Choice Completeness – on a coarser level, the quality of the design depends not only on individual decisions, but on the completeness of the decision model. A high degree of completeness is desirable in mature design models.

Decision Integrity – as introduced earlier, most architecture decisions are taken collaboratively. The integrity of decisions is essential for high-quality of outcomes. An explicit representation of the individual positions of the decision makers is essential to ensure integrity.

The non-concerns for the decision argumentation viewpoint include aspects covered by the four viewpoints (Decision Detail, Decision Relationship, Stakeholder Involvement, Decision Chronological) proposed in [vHAH12a] and Decision Forces as proposed in [vHAH12b].

The core constructs of the Decision Argumentation Viewpoint are:

Design Issue – A reusable aspect of the system design (from the problem domain) that 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 alternatives can be reused within the context of multiple design decisions.

An explicit representation of design issues and alternatives offers a unique way to map reusable knowledge from the design spaces onto the project specific decision spaces. The practical difference between Decision Argumentation Viewpoint and other decision meta-models that do not distinguish between design issues and design decisions is that with this distinction it is possible, within a project,

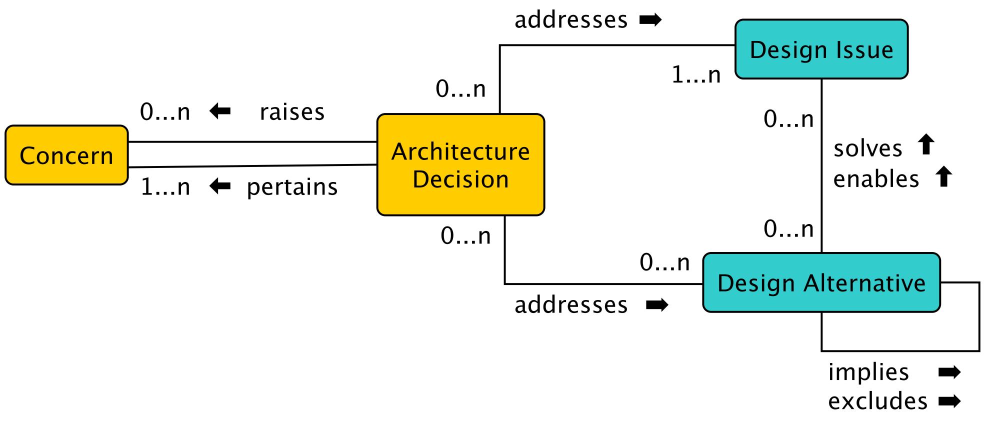


Figure 5.2. An architecture decision viewpoint exposing linkage with design space elements: design issues and alternatives

to address a single (reusable) design issue with multiple decisions that affect different architecture elements and relate to different concerns (see Figure 5.4).

All issues and alternatives should be identified with a unique name. Additional attributes shall be adjusted to a particular design domain. A practice-hardened set of attributes for design issues (Background, Status, Drivers, Recommendation) and alternatives (Known uses, Background, Pros, Cons) was proposed by Zimmermann in [ZKL$^+$09].

Building upon this extended decision viewpoint we devise a new decision model entity (see Figure 5.3), which we define as:

Position – A subjective take of a design team member on a design alternative applied in the context of a particular design issue. For example, the position can be positive, negative, or neutral. The rationale for the position can be captured with a natural language description. This can be complemented by a weight associated with the uncertainty or confidence level of the position. Additionally the position can be marked with a *revoke* flag that, in combination with the timestamp, makes it possible to keep a full record of past argumentation dynamics.

There are a number of relations that can exist among the argumentation viewpoint core elements (see Figure 5.3). The design alternatives relate to design issues with the **solves** relation. The fact that a given alternative **solves** a design issue means that it constitutes a valid outcome for the decision making process in the context of the particular design issue. Another important relation **enables** a design issue if a particular design alternative was chosen. Similarly, the **implies** relation between two alternatives indicates that the designated alternative needs to be chosen if the first one is. The design alternatives are, by default (implicitly) mutually non-exclusive. Therefore, we devise the **excludes** relation to model a situation in which two alternatives cannot be chosen at the same time. The user's position on the decision can relate to the elements of the other architecture decision viewpoints with the following relations:

recommends – an optional relation to the **action** from the stakeholder involvement viewpoint [vHAH12a],

states – a relation to one or more **stakeholders** from the ISO 42010 standard [ISO11],

addresses – an optional relation to the **decision force** from the decision forces viewpoint [vHAH12b],

addresses – an optional relation to the **concern** from the ISO 42010 standard.

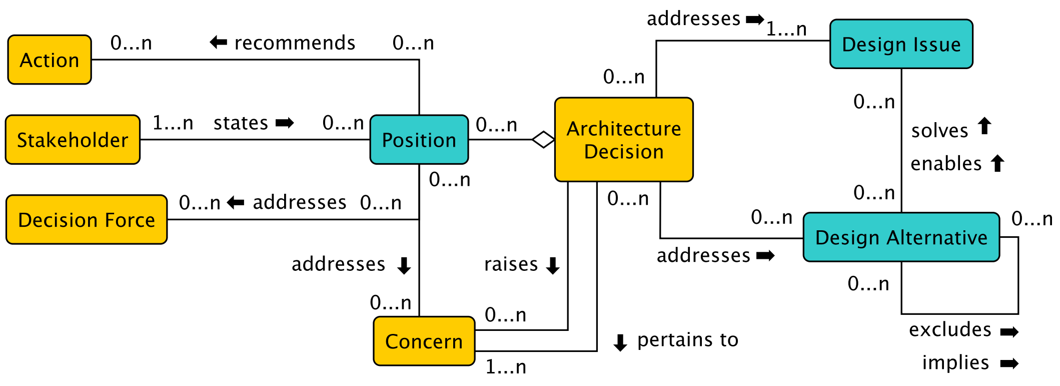


Figure 5.3. The argumentation viewpoint meta-model of the architectural decision with Position related to other decision model elements; Action, Stakeholder taken from [vHAH12a], Decision Force from [vHAH12b]

There are no constraints on the number of positions related to a particular design decision. A single user/stakeholder can contribute multiple positions as long as all except one are labeled with the **revoked** flag. In Figure 5.4 we provide a template for the view based on the argumentation viewpoint.

In the simplest case, an undecided or open architectural decision would be represented just by design issue with no alternatives or positions associated with it.

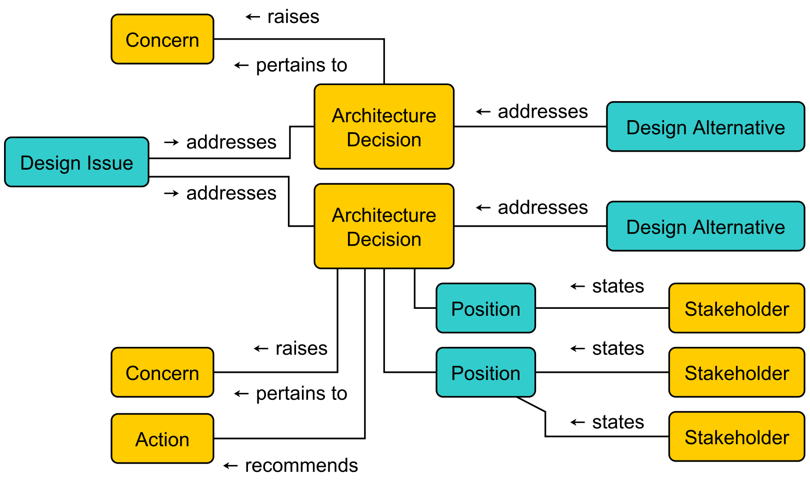


Figure 5.4. A template for the view created with the argumentation viewpoint

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 to an *action* (see [vHAH12b]) 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 a position can be explicitly expressed by the stakeholder so that its weight can be taken into account while bringing the decision process to an end. The result is a closed architectural decision which binds the design issue to the chosen alternative.

In Figure 5.5 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 position types: green/(+) for *positive* and red/(-) for *negative* respectively.

5.1.1 The Lifecycle of Positions within Alternatives

At the beginning, each architectural decision starts with no recorded stakeholders' positions (Figure 5.6). The *aligned* state is reached when all the positions associated with one alternative refer to the same position type. For example, in Figure 5.5 all positions related to HTTPS are positive. This can be interpreted as

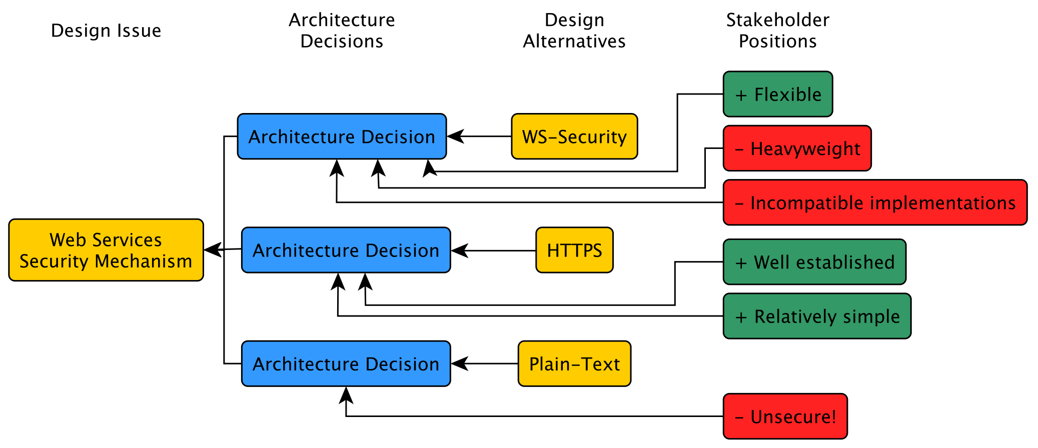


Figure 5.5. An example design decision from the service-oriented architecture design space together with a number of positions [PZL08]

representing the state of consensus among all stakeholders. The *colliding* set of positions exists when positions refer to more than one different position type. 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 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 either there are *no positions* recorded or positions are *colliding* will be referred to as *inconclusive*; conversely, *aligned* or *sealed* positions will be referred to as *decided*.

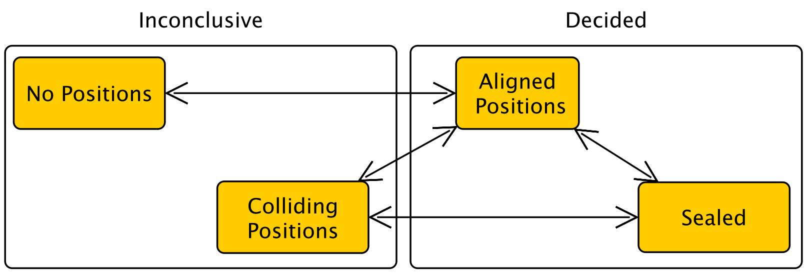


Figure 5.6. The state diagram of the lifecycle of a design alternative. The state of the alternative is aggregated from the actions of its positions

5.1.2 The Lifecycle 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 5.7).

Design decisions about a given design issue start their lifecycle with *no alternatives* recorded. As the design progresses, stakeholders elicit (or reuse) one or more relevant design alternatives, 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 situations in which at least one alternative is in an *inconclusive* state, one can speak about *incomplete* choice. In cases where 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 about the positions on the alternatives, but also about whether the agreement is about a positive (i.e., acceptance, validation, or approval – see [vHAH12a]) or negative (i.e., rejection) decision. Rejected alternatives are discarded, and based on how many alternatives remain, we distinguish: 1) a *conclusive choice* happens when there is exactly one remaining alternative; 2) an *inconclusive choice* happens where there are multiple acceptable alternatives; and 3) a *warring choice* represents a case where no alternative is left on the table.

In the example shown in Figure 5.8, 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 the 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.

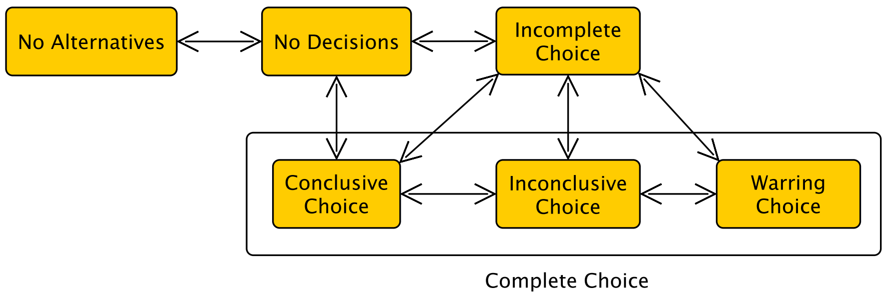


Figure 5.7. The state diagram of the lifecycle of a design decision. The state of the decision is aggregated from the state of its alternatives

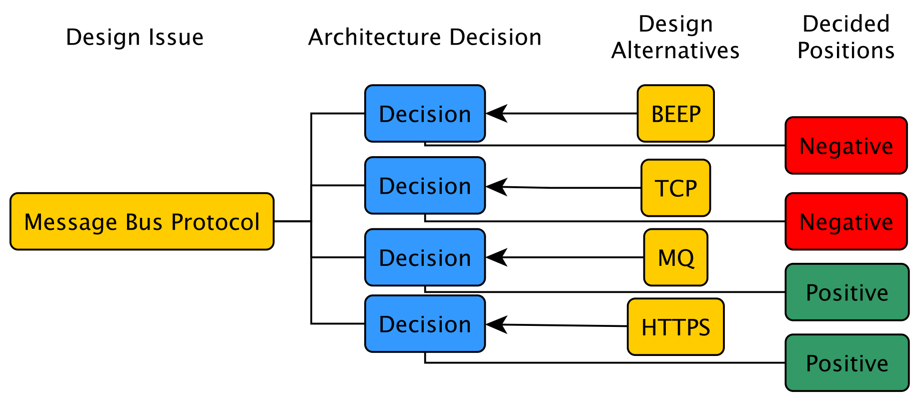


Figure 5.8. An example design issue about transport protocol selection with four design alternatives (protocols) and a complete, inconclusive choice between the alternatives

5.2 Summary

In this chapter, we identified two main concerns regarding collaborative architecture decision making and, further, we have defined a decision argumentation viewpoint to address them. Next, we elicited reusable decision assets such as design issues and design alternatives. Within the framework of the decision argumentation viewpoint, we defined state machines for decision consensus and choice state. Finally, we supported our considerations with illustrative examples. In the next chapter, we introduce the Software Architecture Warehouse – a web based tool designed to support collaborative architecture decision making. The SAW builds upon the theoretical foundation laid in this chapter.

Chapter 6

Software Architecture Warehouse

The Software Architecture Warehouse (SAW) is a tool prototype that we have designed with the warehouse concept in mind. We have extrapolated the warehouse concept into the discipline of architectural knowledge management. In a way similar to that of a data warehouse, the goal of the SAW is to accumulate architectural knowledge from existing software design projects in order to provide benefit for future projects (see Figure 6.1). In the larger picture, the Software Architecture Warehouse provides a platform to persist, analyze and provide reports on live characteristics of architecture decision spaces (see Figure 6.2).

In following sections of this chapter, we introduce the concept of shared design space awareness and explain how it influenced design of the Software Architecture Warehouse. We elaborate about the usage context and typical usage scenario with specific decision-making activities. Next we cover the most important SAW features and proceed with architecture decisions followed by the documentation of selected aspects of the design.

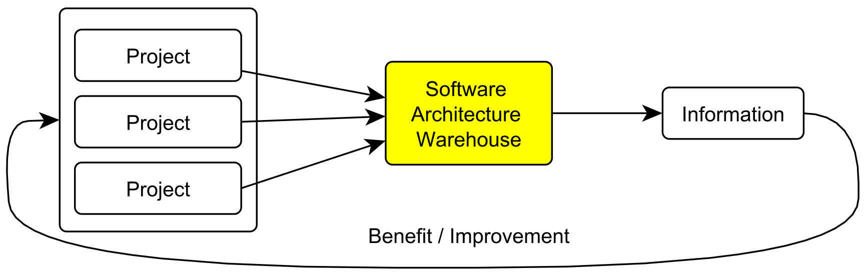


Figure 6.1. Logical information flow through the Software Architecture Warehouse

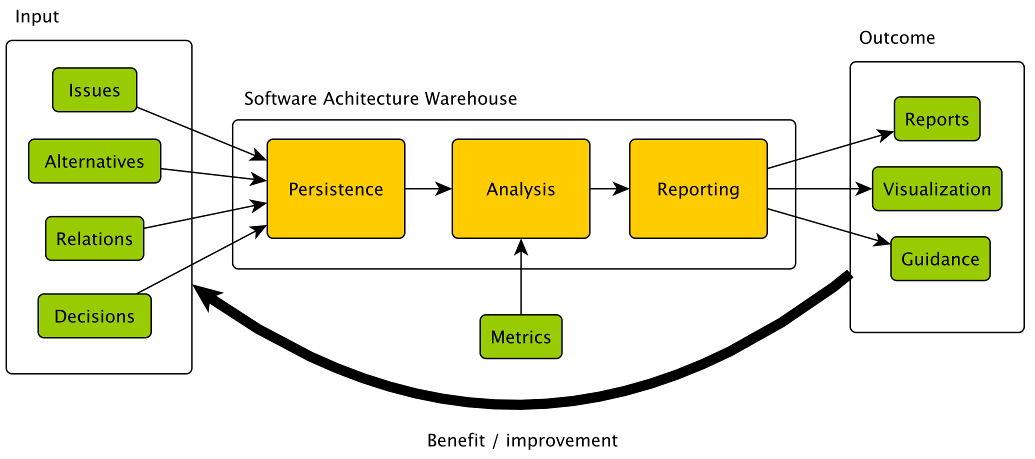


Figure 6.2. The software Architecture Warehouse in the context of its inputs and outputs

6.1 Shared Design Space Awareness

The 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 shared awareness (SA_1), we have adopted the live design document metaphor. Any change to the design elements and relations among them are immediately propagated (with low-latency) to all the design team members participating in the workshop. 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 changesets to all views. For example changes made to a design alternative are instantly reflected in the project details and project summary views (see Figure 6.5).

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, at a first glance can get an overview of the level of consensus (see Figure 6.8). 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 issues and/or alternatives. This way, the impact of decisions can be analyzed from a global perspective.

6.2 Usage Context

In the setting of workshop-based design processes, the Software Architecture Warehouse targets software architecture design teams and supports different user profiles and social configurations:

By **architects** preparing a design workshop. Individually, the architects browse the design space and elicit relevant design issues. In so doing, the architect can discover interesting issues organized in a multidimensional, tag-based classification, perform free-text searches, and follow relations to navigate among closely related issues. After the design workshop is over, the architect completes capturing the outcome of the workshop in SAW, which helps him/her to validate the decision models checking its argumentation consistency and completeness.

During the design workshop a **collocated team** brainstorms about the architecture under discussion using multiple communication channels, such as personal discussion, and whiteboard sketches. The goal of the SAW is to support the discussion without getting in the way. In particular, the SAW helps with the moderation of a free-form discussion by focusing the shared attention and understanding of the participants on discussion of specific design issues. The SAW also explicitly records decisions and helps to build consensus when opinions diverge.

A **distributed team** with telecommuting members has naturally limited communication bandwidth; hence, it is important to use an efficient and precise collaboration tool that is tailored to support collaborative design. Following the philosophy of WYSIWIS (What You See Is What I See [Wol93]), the SAW efficiently supports decision-context sharing among team members and complements other communication tools (such as instant messaging or videoconferencing).

6.3 Application Scenario

A typical application of the Software Architecture Warehouse can be demonstrated in the following case. An in-house architect is given the responsibility of designing a new system that should combine the experiences of the past projects carried out within his or her organization. Initially, he or she sketches a preliminary

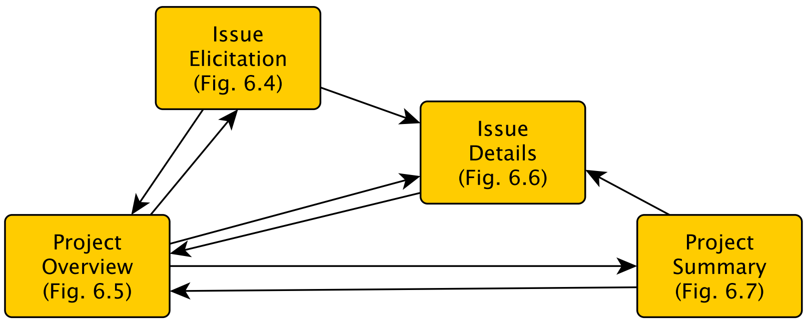


Figure 6.3. A navigation path through the UI views of the SAW, that architects can follow during the design workshop

design that fulfills the constraints imposed by the requirements. In the process, he or she identifies a number of design issues that can be addressed by employing well-known design patterns as well as design issues that have been previously encountered. Together with the design, he or she builds up a collection of open issues to be discussed with the development team at the next design workshop. Documented architectural knowledge (design and decisions [KLvV06]) is to be used as a starting point for the design workshop discussion. During the design workshop, the architect presents the initial design proposal, while the design team challenges her design and brings forth alternative solutions.

The workshop discussion is centered on design issues, lifecycle of which comprises the following activities:

Capture – issues are captured by giving them a name that appears as appropriate at the particular moment of the discussion. Captured issues do not have to be decided immediately, but they can already be associated with a small number of possible architecture alternatives.

Elicit – issues are reused by selecting them out of the warehouse and bringing them into the context of the current project.

Brainstorm – the team contributes additional details about the issue (such as background, decision drivers, and recommendations) and gains a shared understanding of what the corresponding design problem is about. In this phase, the set of possible alternatives grows as the discussion diverges.

Decide – each team member individually registers his or her position on whether alternatives should be rejected. Team members are immediately notified about the others' positions so that the team discussion can converge.

Feature	Data Warehouse (DW)	SAW
Collection	OnLine Transaction Processing System (OLTP)	Collaborative, rich-, web-client
Staging	Extraction Translation and Loading (ETL)	RoR web-server
Analysis	Data Marts	Meta-model in the web-client
Reporting	External System	Rich-, web-client

Table 6.1. Mapping between features of a data warehouse and the SAW

Rationalize – each position needs to be backed by some rationale.

Curate – during the live discussion of the design workshop, the context of each design issue is implicit. The explicit classification and specification of the context of each design issue is essential for preventing the evaporation of valuable knowledge. This activity is carried out soon after the workshop is over.

In practice, during the design workshop, each activity overlaps in time and we do not assume a linear flow (see Figure 6.3). It is up to the architect to prioritize and steer the discussion towards the most critical issues. After the workshop, the architect revisits the outcome of the discussion by separating the issues that have been closed (which may need to be curated and rationalized) from the ones that need further discussion (which may need to be further researched, e.g., to generate additional alternatives or to investigate and compare the alternatives among which the team could not reach a consensus).

6.4 SAW Features

A data warehouse is a specialized database system designed for the purpose of data analysis and reporting [Inm02]. Apart for data persistence (staging), the main purpose of the data warehouse is an analysis (mining) of the information that can be discovered only by leveraging the very large scale of the raw data received as input from online transaction processing systems.

Our approach covers the three aspects of warehouse functionality (staging, analysis, and reporting). In Table 6.1 we provide a mapping between typical OLAP data warehouse stack and specific modules of the Software Architecture Warehouse (see also Figure 6.10).

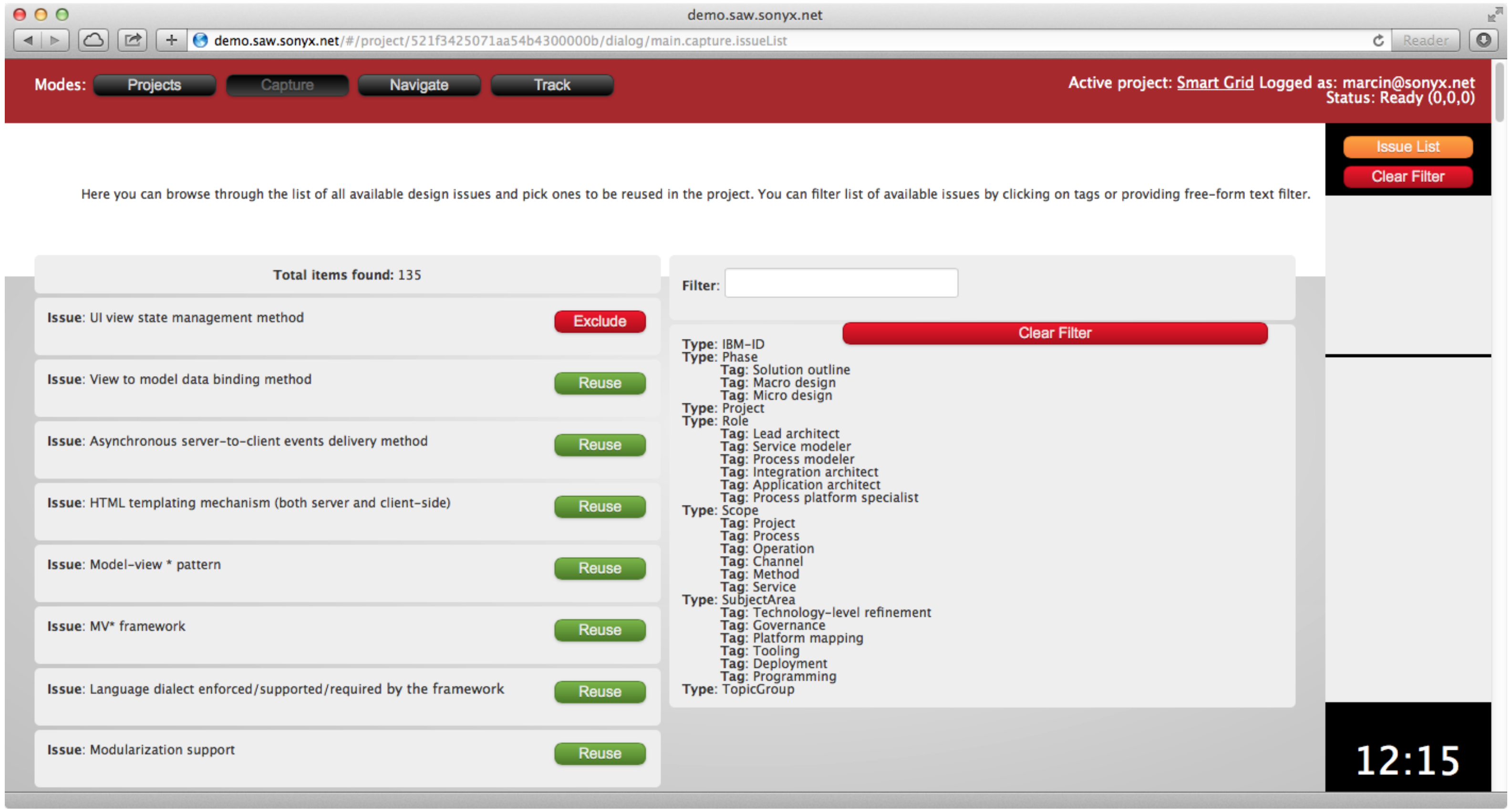
In the following sections, we introduce features of the Software Architecture Warehouse that make it stand out from the other collaborative architecture decision management tools we have listed in Section 3.5.

6.4.1 Project-Based Design and Reusable Design Issues

The meta-model supported by the SAW distinguishes between the problem and solution domains by representing design issues (in the problem domain) solved by one or more possible design alternatives (in the solution domain). The meta-model also separates reusable knowledge (the design space, which is composed of reusable design issues and alternatives) from context-specific information (the project space). The tool supports multiple projects, which can include references to relevant issues and alternatives augmented by the project-specific positions of individual designers.

6.4.2 Decision Elicitation

An important step during the preparation for a design workshop consists of *deciding what is to be decided*. Browsing through the decision repository gives the architect an opportunity to elicit design issues to be reused (see Figure 6.4). The SAW supports the issue elicitation process with a free-text search, filtering based on multi-dimensional tagging and tag-cloud browsing. Every elicited design issue is made immediately available to all designers participating in the project. Elicitation can also happen implicitly as architects are about to enter new design issues since the SAW will attempt to locate existing issues based on the tentative name of the issue about to be captured.


demo.saw.sonyx.net
demo.saw.sonyx.net/#/project/521f3425071aa54b4300000b/dialog/main.capture.issueList
Reader
Modes: Projects Capture Navigate Track
Issue List
Clear Filter
Exclude
Reuse
Clear Filter
12:15

Active project: Smart Grid **Logged as:** marcin@sonyx.net
Status: Ready (0,0,0)

Here you can browse through the list of all available design issues and pick ones to be reused in the project. You can filter list of available issues by clicking on tags or providing free-form text filter.

Total items found: 135

Issue: UI view state management method

Issue: View to model data binding method

Issue: Asynchronous server-to-client events delivery method

Issue: HTML templating mechanism (both server and client-side)

Issue: Model-view * pattern

Issue: MV* framework

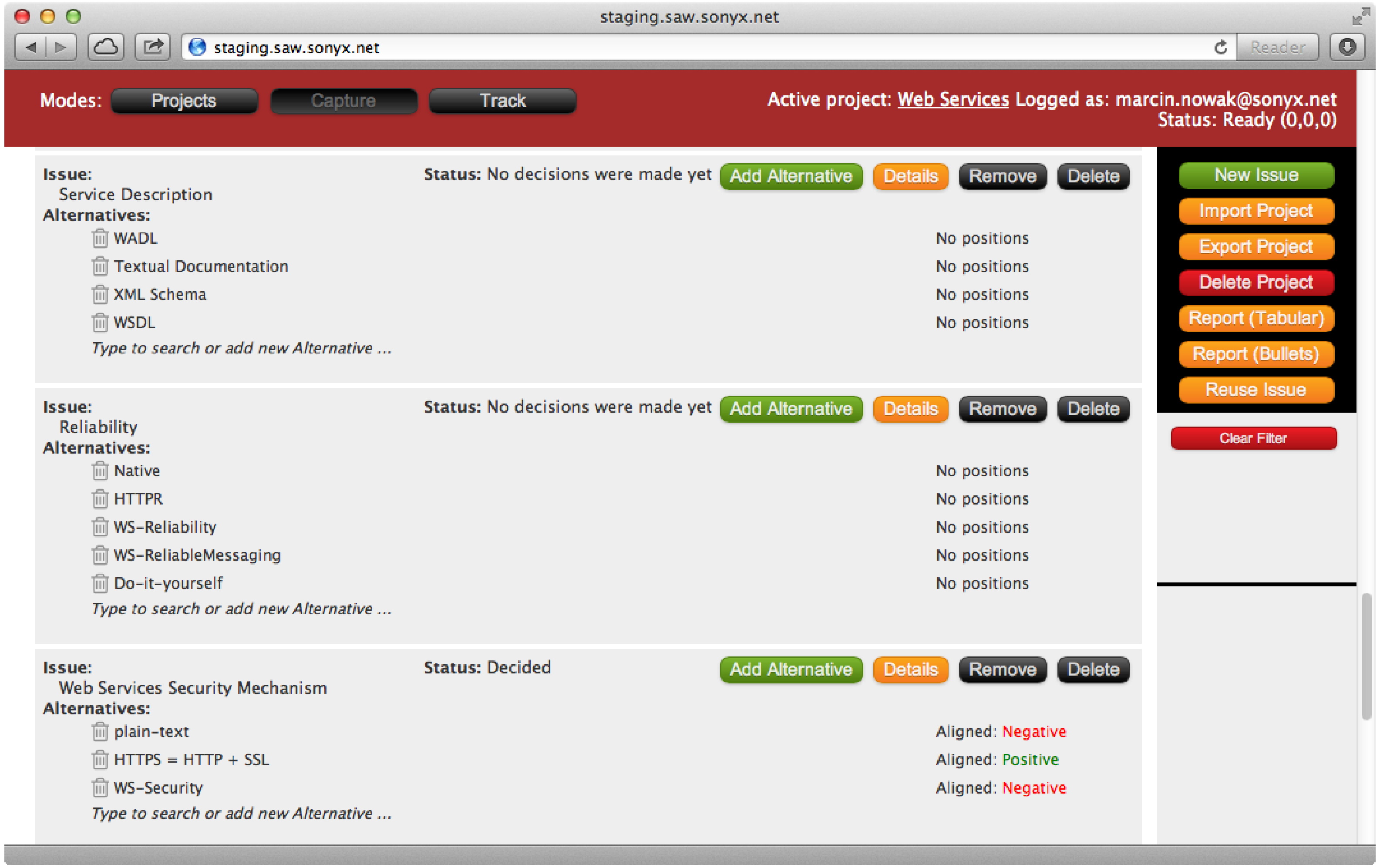
Issue: Language dialect enforced/supported/required by the framework

Issue: Modularization support

Filter:

Type: IBM-ID
Type: Phase
  Tag: Solution outline
  Tag: Macro design
  Tag: Micro design
Type: Project
Type: Role
  Tag: Lead architect
  Tag: Service modeler
  Tag: Process modeler
  Tag: Integration architect
  Tag: Application architect
  Tag: Process platform specialist
Type: Scope
  Tag: Project
  Tag: Process
  Tag: Operation
  Tag: Channel
  Tag: Method
  Tag: Service
Type: SubjectArea
  Tag: Technology-level refinement
  Tag: Governance
  Tag: Platform mapping
  Tag: Tooling
  Tag: Deployment
  Tag: Programming
Type: TopicGroup

Figure 6.4. Decision Elicitation: Reusable design issues (on the left) can be included in the current project by clicking on the Reuse button. The set of issues can be filtered by a free text search and tag-based classification (right)



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 ...

Figure 6.5. A comprehensive project overview presenting a number of design issues in the context of the project. Selected design issues are expanded with a list of design alternatives that are highlighted with a color-coded decision summary

6.4.3 Collaborative Brainstorming

Often the initial set of elicited design decisions is not sufficient to completely cover the architectural design. During the design workshop brainstorming sessions, participants capture and refine new design issues and alternatives. The SAW supports a low-latency delivery of updates so that designers can synchronize the discussion context and their contributions. The benefits of rapidly capturing design issues and alternatives are twofold:

1. Designers can easily make a contribution to the discussion by entering new alternatives in the tool without forcing an interruption of the discussion stream;

2. The evaporation of captured items is prevented so that architects can later come back and consider important issues that were likely to be overlooked if raised while the discussion was focused on a different topic.

6.4.4 Decision Making

The SAW records the positions of each team member on a particular design alternative in the context of the corresponding design issue. Every position can and should be supported by a brief statement describing the corresponding rationale. The position can be one of pre-defined types:

Positive – when the designer expresses that he or she considers a particular alternative to be viable in the context of given design issue,

Negative – when the alternative is not acceptable,

Open – to label alternatives which were considered during the main discussion, but their decision status is still neutral, as neither **positive** or **negative** can be applied.

Each position can be revoked with an **anti-position**, which reverts its status but also records the change of position in the discussion log (see Figure 8.3). Concerning the position types, the SAW meta-model is flexible and additional decision types can be easily introduced by tailoring the tool configuration to the needs of specific projects or design teams. For example if design alternatives have a quantifiable attribute that is relevant for the decision making, the decision meta-model can be tailored to accommodate it.

Choice state of the design issue can be monitored in the project overview (see Figure 6.5), which aggregates the individual consensus states of the alternatives related to it. From the same view, it is possible to export different reports containing a complete snapshot of the design space.

6.4.5 Position Conflict Management

During workshops involving multiple designers, conflicting positions are difficult to avoid in practice. The SAW not only provides designers with immediate updates on the status of each decision but also embraces the reality of position conflicts within its meta-model. This helps to support the early identification of conflicts and enables their resolution through consensus building so that the rework overhead can be minimized.

An aggregated overview of all team members' positions is provided in the project overview (Figure 8.2), which shows the number of **positive**, **negative** and **open** positions. A gray background indicates alternatives for which conflicts, i.e., the presence of positions of different types, have been detected. The positions and the corresponding rationale provided by each team member can also be inspected and edited in the Issue Detail View (Figure 6.9).

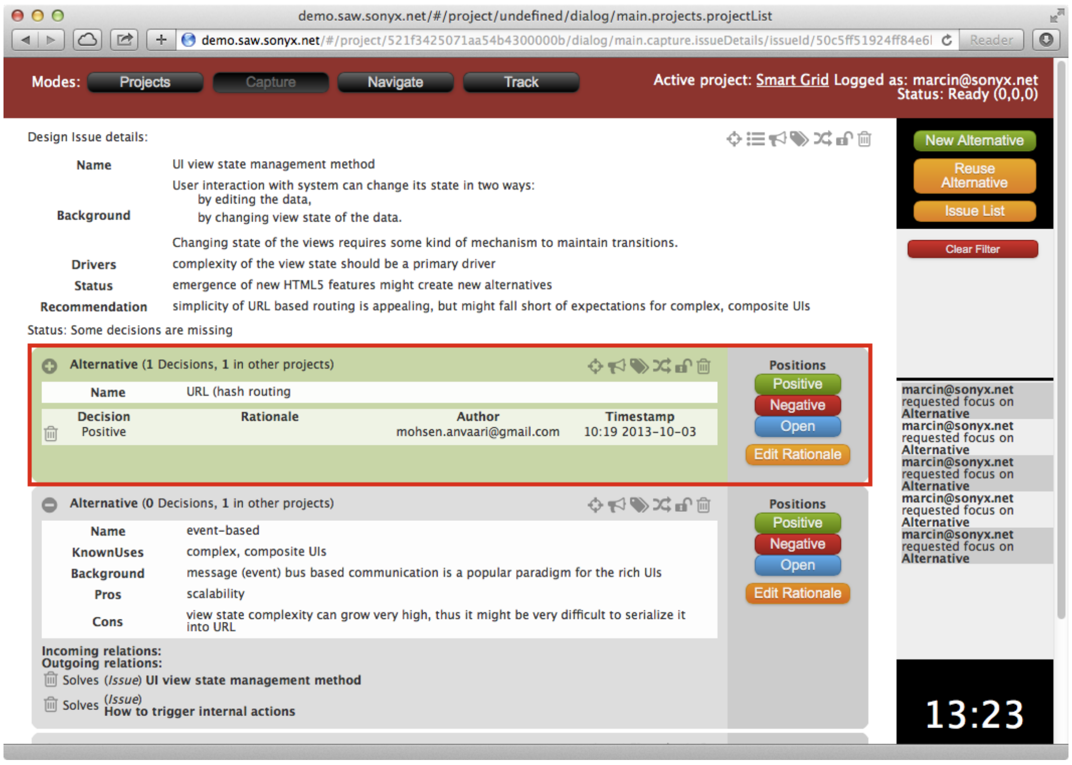
6.4.6 Focus Tracking and Convergence

Conducting remote workshops on a design space containing many issues presents the need to quickly focus the discussion on specific design issues. The SAW helps the team to gain a shared understanding of the focus of the discussion with a non-intrusive, mouse-tracking mechanism. Since all participants get an indication of whether other team members are pointing at the same item, this mechanism also prevents conflicting text edits on the same knowledge item.

Additionally we have equipped SAW with a mechanism for explicitly drawing designers' focus to a particular item. After a focus call the item is broadcaster, the item in question gets highlighted in the views that display it (see Figure 6.6). If the current view does not display the called item, the user can easily navigate to the right context by clicking on the focus call event in the notification pane.

6.4.7 Progress Monitoring

Given the non-linear nature of design discussions, it is important not to lose track of the progress of a design workshop. The SAW provides the Summary View (Figure 6.7) which allows for classification and sorting of issues based on their decision status and quick separation of issues that have been decided (a



demo.saw.sonyx.net/#/project/undefined/dialog/main.projects.projectList

demo.saw.sonyx.net/#/project/521f3425071aa54b4300000b/dialog/main.capture.issueDetails/issueId/50c5ff51924ff84e6

Active project: Smart Grid Logged as: marcin@sonyx.net
Status: Ready (0,0,0)

Design Issue details:

Name	UI view state management method
Background	User interaction with system can change its state in two ways: by editing the data, by changing view state of the data. Changing state of the views requires some kind of mechanism to maintain transitions.
Drivers	complexity of the view state should be a primary driver
Status	emergence of new HTML5 features might create new alternatives
Recommendation	simplicity of URL based routing is appealing, but might fall short of expectations for complex, composite UIs

Status: Some decisions are missing

Alternative (1 Decisions, 1 in other projects)

Name	URL (hash routing

Decision	Rationale	Author	Timestamp
Positive		mohsen.anvaari@gmail.com	10:19 2013-10-03

Alternative (0 Decisions, 1 in other projects)

Name	event-based
KnownUses	complex, composite UIs
Background	message (event) bus based communication is a popular paradigm for the rich UIs
Pros	scalability
Cons	view state complexity can grow very high, thus it might be very difficult to serialize it into URL

Incoming relations:
Outgoing relations:
Solves (*Issue*) UI view state management method
Solves (*Issue*) How to trigger internal actions

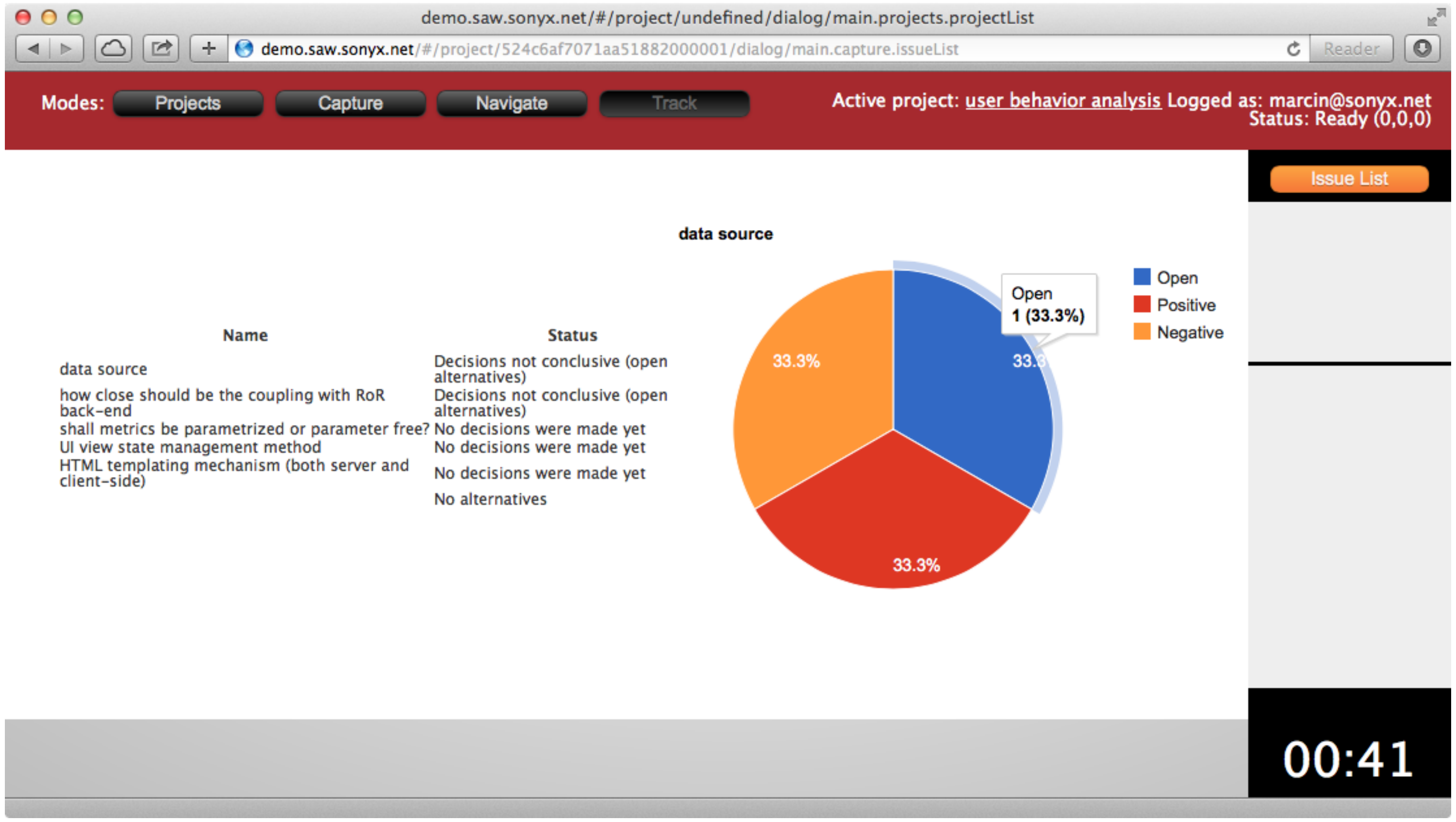
marcin@sonyx.net requested focus on Alternative
marcin@sonyx.net requested focus on Alternative
marcin@sonyx.net requested focus on Alternative
marcin@sonyx.net requested focus on Alternative
marcin@sonyx.net requested focus on Alternative

13:23

Figure 6.6. An Issue Detail view with one alternative highlighted by the focus call and number of focus calls visible in the notification pane on the right

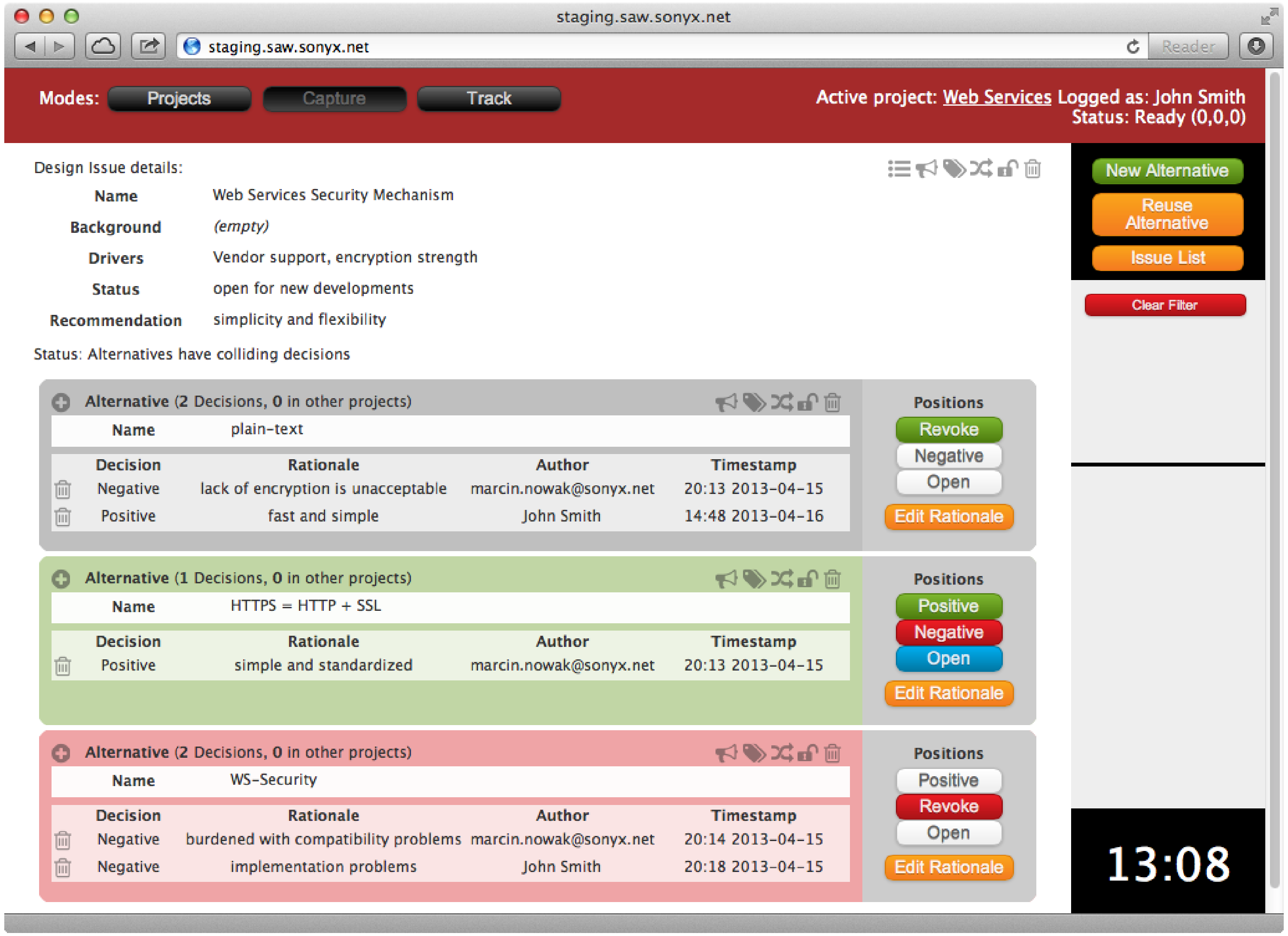
single alternative has been chosen), from issues that are still open (for various reasons). In particular, the SAW employs selected decision metrics (as introduced in Section 7.1.2) to detect **incomplete** issues (where some alternatives have not been accepted or rejected), **conflicting** issues (which contain at least one alternative that contains both **negative** and **positive** positions), and **inconclusive** issues (which have multiple alternatives with aligned positive positions and need to be further refined).

Typically architecture design workshops are time constrained. We have decided to equip SAW user interface with a large clock widget visible in the lower right corner of the screen, to help users to stay within the agreed time limit.


demo.saw.sonyx.net/#/project/undefined/dialog/main.projects.projectList
demo.saw.sonyx.net/#/project/524c6af7071aa51882000001/dialog/main.capture.issueList
Reader
Modes: Projects Capture Navigate Track
Active project: user behavior analysis Logged as: marcin@sonyx.net
Status: Ready (0,0,0)
Issue List
data source
Open
1 (33.3%)
33.3%
33.3%
33.3%
Open
Positive
Negative
00:41

Name	Status
data source	Decisions not conclusive (open alternatives)
how close should be the coupling with RoR back-end	Decisions not conclusive (open alternatives)
shall metrics be parametrized or parameter free?	No decisions were made yet
UI view state management method	No decisions were made yet
HTML templating mechanism (both server and client-side)	No decisions were made yet
	No alternatives

Figure 6.7. The Project Summary View with an overview of the design issues within the context of a project. The decision status is summarized (and color-coded) for each alternative and aggregated at the issue level.



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

Alternative (1 Decisions, **0** in other projects)

Name HTTPS = HTTP + SSL

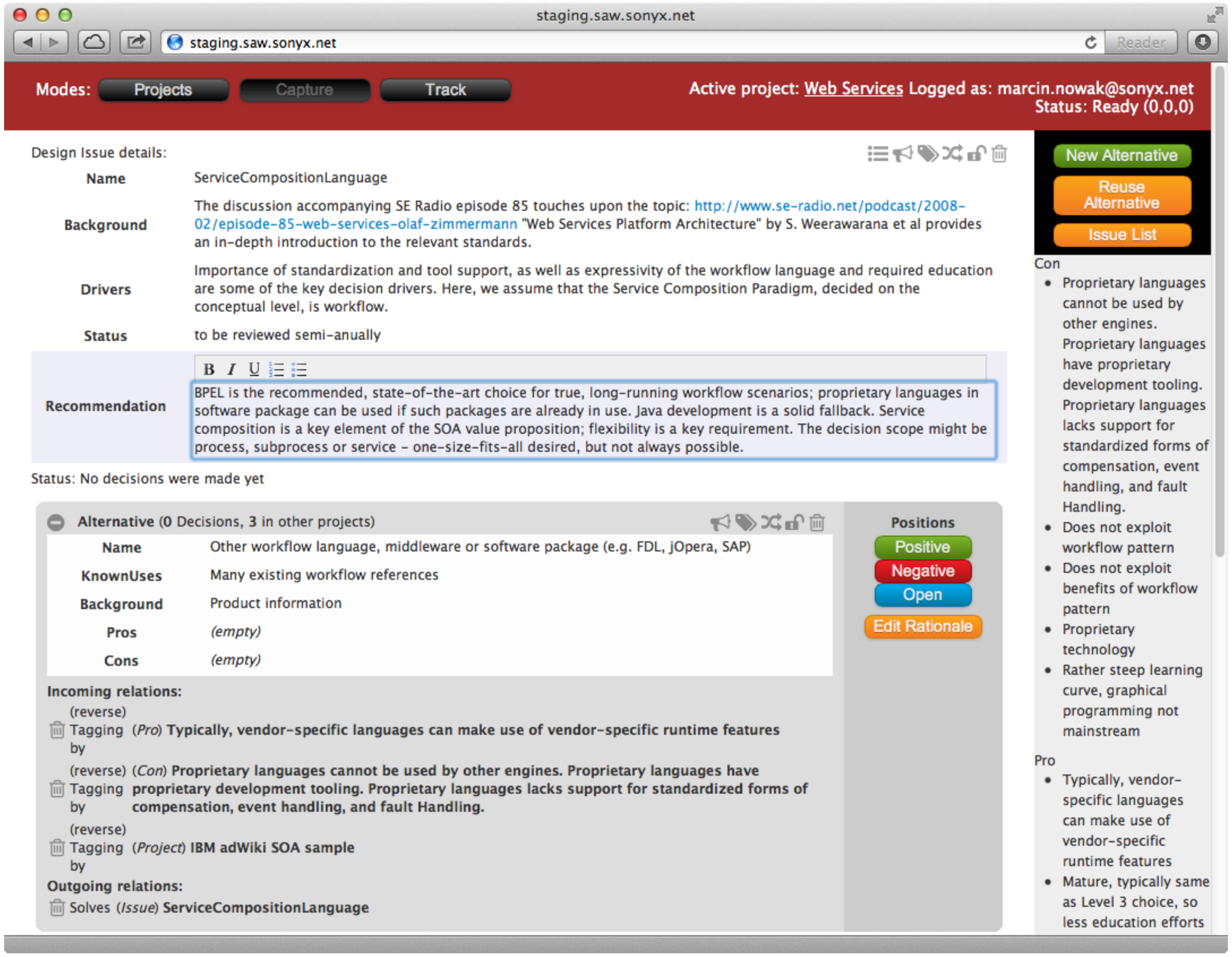
Decision	Rationale	Author	Timestamp
Positive	simple and standardized	marcin.nowak@sonyx.net	20:13 2013-04-15

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

Figure 6.8. A view presenting a design issue with three alternatives, two of which have aligned positions (second and third), the other (first) of which has colliding positions. Positions on the alternatives are presented together with a user-specified rationale and are color-highlighted based on the decision type and status.



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)*

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

Figure 6.9. The Issue Detail View presenting full information about design alternatives in the context of the particular design issue. Each alternative holds information about its known uses, background, pros and cons.

6.5 Architecture Design Decisions

In this section, we document the principal design decisions that influenced our design of the Software Architecture Warehouse. In order to approach this task systematically, we reuse the design issues that we introduced in Chapter 3 (Sections 3.3 and 3.4).

6.5.1 Design Issue: Model Location

The choice of the decision model location is driven by multiple factors. In the context of the SAW, we have elicited the following criteria:

Latency – low latency and thus high interactivity is essential for the effective collaboration of stakeholders. High latency, and thus low responsiveness, can very easily negate the utility of the collaborative decision support tool,

Ease of entry – as distributed design teams often work in non-trivial network conditions, it is essential to minimize the effort required to get started,

Reliability – the time that stakeholders invest in the process of decision making is very valuable; hence, reliability is essential for gaining the benefit of using a decision support system,

Feasibility of implementation – realistically, resources that were available to us for the implementation of the Software Architecture Warehouse were limited. Hence, it was important to reuse as much as possible from the solutions available within the state of practice and focus on the novel and essential aspects of the system.

Design Alternative: **Distributed** Decision: **Open**

The latency of access to the distributed decision model is lower for read-only access, as each of the clients stores its own copy of the model. In terms of set-up complexity, the distributed alternative is disadvantaged, because it requires either some kind of peer discovery mechanism or else tedious manual configuration of peers. The complexity of the mechanism required to provide reliable synchronization of the distributed model is also substantial.

Design Alternative: **Centralized** Decision: **Positive**

In terms of latency, the centralized model location is advantageous in a read-write scenario, because model changesets are distributed through the single node; hence, latency does not accumulate over consecutive hops. Setting the application up is also by far simpler, as the clients just need to be provided with the address of the central server. Reliability can be assured just by providing communication with the central node, regardless of topology of the clients. A big advantage of centralized deployment is that there exist many application frameworks that can be efficiently reused.

6.5.2 Design Issue: Synchronization

The synchronization of changesets among the clients is essential to keep all stakeholders up-to-date. When making decisions about the synchronization model we find the two following criteria important:

Latency – In the dynamic environment of the architecture design workshop, it is important that changesets are distributed as rapidly as possible so that the stakeholders have an opportunity to provide rapid feedback.

Consistency – The change of a decision model often proceeds gradually leaving, it in a tentatively inconsistent state. Tentative inconsistency can easily be misinterpreted and lead to confusion.

Design Alternative: **Manual** Decision: **Negative**

Manual triggering gives the user full control over the synchronization of the changeset exchange. This is can be advantageous in situations in which temporal consistency of the decision models needs to be maintained at all times. In practice though, manual synchronization is rather intrusive and easily gets in the way of the deliberation process.

Design Alternative: **Automatic** Decision: **Positive**

An automatic synchronization of decision model changesets means that users can focus on deliberation instead of being distracted by the tool's operations. Giving up user control over when the decision model is synchronized can lead to tentative decision model inconsistencies and thus to some confusion, but assuming low communication latency and a fast rate of change, it should not threaten the eventual consistency of the decision model.

6.5.3 Design Issue: Meta-Model Type

While considering type of the decision meta-model to be adopted, two important criteria come into consideration:

Ease of expression – Brainstorming, by definition, does not have a well-structured form. In practice, the least constrained the exchange of thoughts is, the better it is for the deliberation.

Consistency of outcome – Volatile deliberation can very easily result in chaotic and inconsistent results. Enforcing a lightweight meta-model provides a framework that can help to improve the consistency of the resulting decision model.

Design Alternative: **Implicit** Decision: **Negative**

An implicit meta-model relies on the users to formulate their input accordingly. It can potentially boost productivity in the short term, but in the long term has potential to result in inconsistent outcomes that are a very tedious to interpret and fix. In particular, such output is difficult to process automatically.

Design Alternative: **Explicit** Decision: **Positive**

An explicitly used meta-model in the tool, to some degree, constrains freedom of expression. Keeping in mind that the goal of deliberation is a consistent decision model, the constrained expression is easily offset by the fact that there is no effort required later for the sanitization of the model. Additionally, an explicit meta-model is important particularly for inexperienced decision makers, who benefit from guidance in the formulation of their contribution to the model.

As a result of choices that we made in the cases of the last three decisions, we have identified the **blackboard** style as the collaboration paradigm supported by the Software Architecture Warehouse.

6.5.4 Design Issue: Conflict Prevention

There are two criteria that we take into consideration when deciding upon the conflict prevention mechanism:

Penalty for corruption – when considering whether to prevent conflicts, one needs to ask about the cost of changeset conflict for deliberation. In particular, when conflict is not prevented, model corruption can occur.

Cost prevention – prevention of concurrent access requires additional actions to be taken by the stakeholders. The overhead caused by conflict prevention can be disruptive to the deliberation process.

Design Alternative: **Locks** Decision: **Negative**

An explicit locking mechanism is a heavyweight method of preventing two users from editing single decision model item at the same time. As the editing of a single particular design item can take quite some time, locking can disrupt the fast dynamics of decision making, particularly in the fast-paced and volatile conditions of the architecture design workshop. As time is a particularly valuable resource for the stakeholders in the decision process, the relative cost of preventing editing conflicts is quite high.

Design Alternative: **None** Decision: **Positive**

No locking implies that users can concurrently modify a decision model. This potentially leads to some degree of model corruption, but with an assumption low-latency and a high rate of change, we believe that this disadvantage is easily offset by the overall efficiency boost.

6.5.5 Design Issue: Conflict Resolution

In the case of unconstrained, simultaneous write access to the decision model, the occurrence of conflict is inevitable. When making choices among strategies for handling changeset conflict resolution, we have considered the following criteria:

Cost of intervention – The conflict resolution mechanism is in place to improve the efficiency of deliberation. The less intrusive the mechanism is, the better it is for the process dynamics.

Quality of the solution – The resolution of changeset conflict can be ambiguous. The quality of the result is important, because if it is not good enough, then intrusive intervention is required.

Design Alternative: **Manual** Decision: **Negative**

The upside of manual conflict resolution is that, the stakeholders involved in it can assess and approve the quality of the outcome. At the same time, manual conflict resolution is very costly in terms of the involvement of the stakeholders.

Design Alternative: **Automatic** Decision: **Positive**

The fact that automatic conflict resolution does not involve stakeholders and thus is not intrusive for deliberation, is very advantageous. The quality of automated conflict resolution might be questionable, but in environments with low-latency of communication, the stakeholders have an opportunity to fix badly resolved conflict assuring eventual consistency.

As a result of choices made for **Synchronization**, **Conflict Prevention**, and **Conflict Resolution**, the design of the Software Architecture Warehouse has a high level of **Liveness** (see Section 3.4).

6.6 Selected Design Aspects

In this section we have gathered design aspects that reach beyond the design space of Collaborative Architecture Decision Making Tools (see Section 3.2). The content of this section is not exhaustively covering design of the Software Architecture Warehouse, but selectively focus on the aspects that we found interesting or innovative. The reason for this is that, despite numerous iterations in design and implementation, SAW is still a prototypical tool. Therefore, its design is burdened with a significant amount of technical debt, which documented in detail would obfuscate the view and confuse the reader.

6.6.1 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.). 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 for supporting the live document metaphor. Since every user interaction with the system triggers a call to a rather heavy server-side stack, the result is rather a limited scalability and bad user-perceived responsiveness. In the process of architecting and implementing the 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 (MongoDB[1]). The interactive

[1]MongoDB NoSQL, document oriented database – `http://www.mongodb.org/`

user-interface is implemented following the MVVM pattern in JavaScript with Backbone[2] and Marionette[3] (see Figure 6.10).

Rationale: The traditional client-server split for web applications was not able to deliver user experience required to provide live collaboration support.

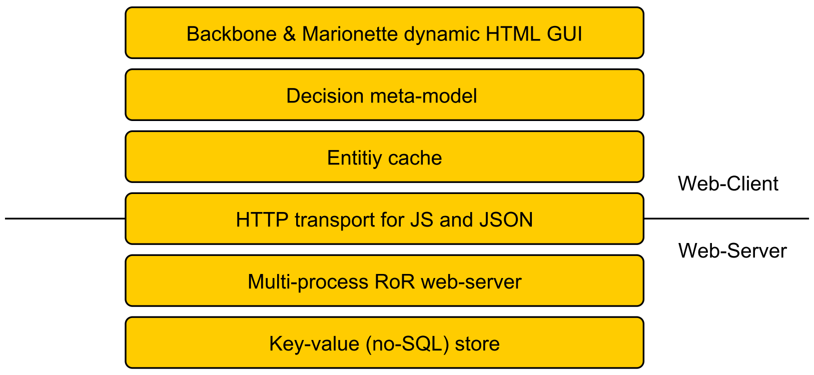

Backbone & Marionette dynamic HTML GUI
Decision meta-model
Entitiy cache
HTTP transport for JS and JSON
Multi-process RoR web-server
Key-value (no-SQL) store
Web-Client
Web-Server

Figure 6.10. Layering structure that exposes the client-server split of the Software Architecture Warehouse

6.6.2 Rich Web-Application

In order to support high levels of liveness that imply a high responsiveness of the user interface, the user interface of the Software Architecture Warehouse has been implemented as a highly modular, expandable JavaScript based web-application. At the moment of writing, the following modules have been implemented:

Main – covers the core functionality of the user interface and serves as a toolbox of general purpose widgets for other modules; we elaborate on it in more detail in the next section,

Capture – focuses on design capture and reuse of design issues and alternatives with support for tag-based classification and filtering,

Decide – provides features for users to cast their individual positions on alternatives and covers consensus and choice-state logic,

Projects – implements design and decision space management features, such as import, export, and reporting.

[2]Backbone javascript application framework – `http://backbonejs.org/`
[3]Marionette extension to Backbone application framework – `http://marionettejs.com/`

The **Main** module (see Figure 6.11) comprises the following units:

Data – covers decision meta-meta-model level entities such as **Item**, **Relation**, and **Collection** classes. It is also reosponsible for the entity cache,

Router – is responsible for parsing the hash-based navigation part of the URI and passing it over to the respective modules,

Context – implements a collection of objects made available to all widgets within the module. The main context is shared among the modules,

Meta-model – maintains the mapping between the meta-meta- and meta-model of the decision. It is reloaded from the repository each time the application starts,

Tags – represents a classification tags' tree,

Position types – a collection of decision type instances specific to the decision meta-model, i.e. **Positive**, **Negative**, and **Open**,

Artifacts – maps to the meta-model entities that comprise the body of the decision model, i.e., **Issue** and **Alternative**.

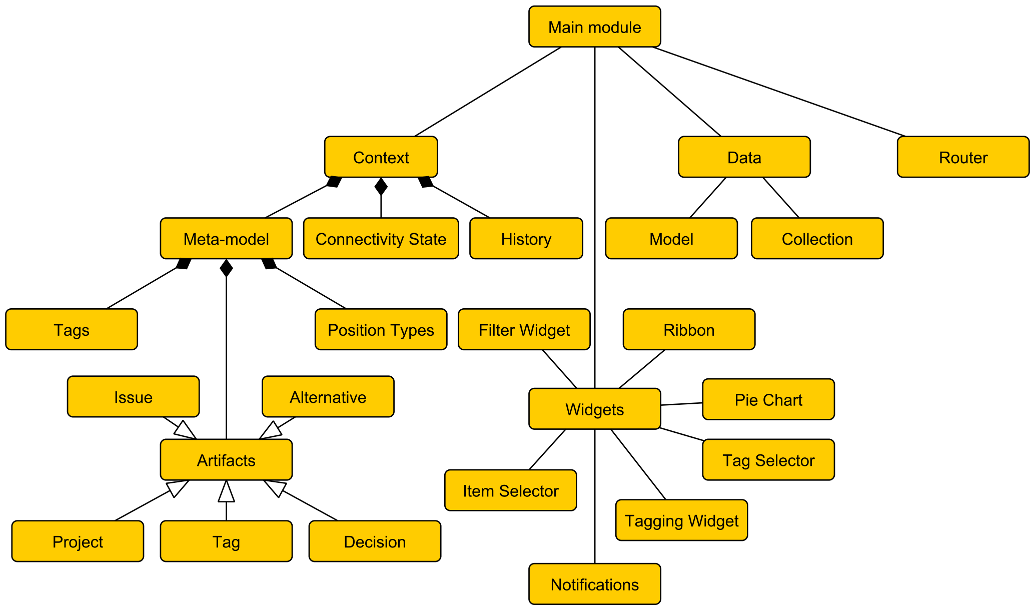


Figure 6.11. Decomposition of the main module of the SAW user interface

The decision module can be decomposed into the units presented in Figure 6.12. Apart from the core components common to all the modules (Context, Router), it comprises the following:

Issue list – a main panel widget providing a wire-frame for the view listing design issues. Typically the design issues cover the design space of an individual project,

Issue list Item – models individual design issues within the **Issue list** widget,

Issue details – implements a main panel widget comprising a list of design alternatives related to the design issue,

Alternative compact – models properties of a single design alternative within the **Issue details** view,

Alternative details – handles individual design alternatives with its incoming and outgoing relations.

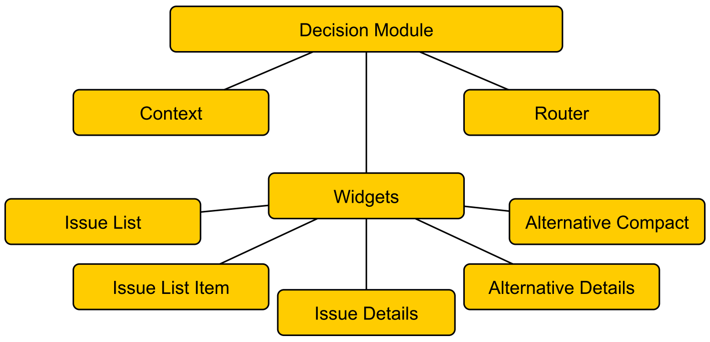


Figure 6.12. Decomposition of the decision module of the SAW user interface

Rationale: Adoption of modular, rich-client web application architecture enables clear separation of concerns between the modules and makes it possible to deliver a low-latency live experience of distributed collaborative work.

6.6.3 Graph-based Decision Space Meta-Meta-Model

Observing the variety of available architecture decision management tools, we noticed that typically they are bound to a single decision meta-model (see Chapter 3). We perceived it as a limitation and decided to address it by increasing the level of abstraction for modeling decision spaces. Accordingly we distinguish three levels of decision model abstractions:

Decision model – represents concrete decision space elements including design issues, design alternatives, positions, projects, etc.

Decision meta-model – describes architecture model items and the relations among them. Typically it reflects one of the contemporary architecture decision models (see Section 3.1) and optionally extends it to fit the needs of a particular application domain.

Decision meta-meta-model – a top-level of abstraction that operates on a graph structure using nodes and edges to represent meta-model and model entities.

Rationale: The application of this three-level abstraction structure enables the SAW to offer a high degree of flexibility to a decision meta-model without compromising any of its features essential for providing a high degree of Situational Awareness (see Section 6.1). This is possible because internally the SAW makes very minimal assumptions about the decision meta-model.

6.6.4 Node Graph Observer and Notification System

In order to deliver high user awareness of the shared design space, a suitable data replication mechanism is needed. We have implemented a lightweight notification mechanism that distributes identifiers of altered graph nodes so that clients can reload node data if needed. In cases when 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 6.13). The notification system is very general and has also been used to implement the view pointer broadcast feature.

Rationale: Live collaboration o graph data structure requires an efficient mechanism for handling change notifications. We have decided to adopt an observer pattern because it lazily loads data about nodes that are within the interest of particular observer.

6.6.5 Smart Client-Side Graph Caching

One of the implications of implementing the model and (most of) application logic on the client-side is that interface between the client and the server is fairly fine-grained. In the SAW, the majority of the API calls concerns either fetching node

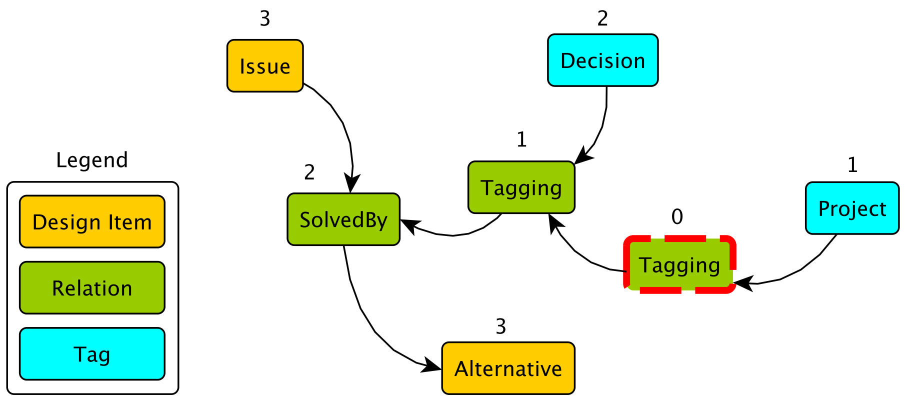


Figure 6.13. Event propagation over the shared graph model. Views can subscribe to observe changes within a certain distance of their model elements

information or information about incoming and outgoing relations. In practice, in the short term, only small part of the design space is influenced by the alterations, so a substantial number of node and edge fetch calls can be cached efficiently. A web-browser is equipped with a cache mechanism designed to efficiently cache content of the web-resources. In the case of the SAW, the browser-cache serves its purpose, but only after executing a server call that returned HTTP status 304–Not modified[4]. In situations with a very large number of fine-grained calls, the server-side can be easily overloaded. To this end, we have implemented a persistent, HTML5 local storage based, client-side caching mechanism. The cache uses the notification subsystem to invalidate altered cache entries. Each design space alteration (and the notification related to it) is uniquely identified with a monotonic timestamp so that clients reconnecting to the server can fetch a list of the nodes and edges altered since the last session.

Rationale: We have decided to introduce client-side caching for the decision model because it improved application start times and significantly reduced server load during the peak times, such as the beginning of the design workshop.

6.6.6 Deployment

The Software Architecture Warehouse deployment package has been prepared as a portable virtual machine image (Oracle VirtualBox[5]). For the purpose of

[4]`http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`
[5]`https://www.virtualbox.org/`

the evaluation it was instantiated on an ordinary Dell Optiplex 780 desktop machine with an Intel Core2 Quad (3GHz) processor and 8 GB of system memory, running Microsoft Windows 7. The particular choice of operating system for the virtual machine host was made to demonstrate the feasibility of deployment in the typical corporate environment. The virtualized image runs an up-to-date release of Ubuntu Linux (10.04) with a minimum of 512 MBytes of memory (2GB+ recommended) and bridged networking access. The connectivity with the rich-web browser client requires access to the standard HTTP port (80), as well as an additional link port 8080 providing push-message bus access.

The set-up serving both a publicly available demo-instance and instances used internally for the in-class evaluation was running multiple, parallel instances of the VM-image on the aforementioned hardware (see Figure 6.14). In order to overcome firewall-imposed connectivity limitations, a HTTP-reverse proxy was used to provide accessibility from the public network.

Rationale: We have decided for the virtual-machine based deployment because it significantly eased multi-server deployment as well as enabled easy portability in ad-hoc deployment scenarios.

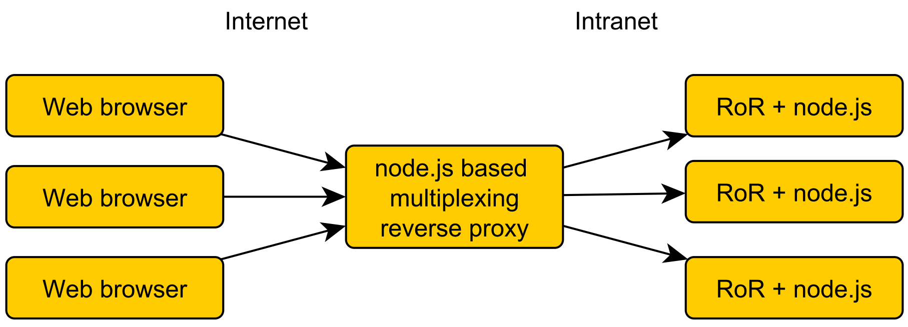


Figure 6.14. A deployment view of the Software Architecture Warehouse in a network

6.7 Summary

In this chapter, we provided a concise walkthrough of the features and the architectural design of the Software Architecture Warehouse. We delivered a number of high- and low-level views of the system modularization. We have also presented a number of unique features such as graph-based decision meta-modeling and graph-node observer notification, that make SAW stand out of the competition by providing high degree of liveness combined with a blackboard-style collaboration.

In the next chapter, we introduce an off-line analysis tool designed to provide insight into the dynamics of the architecture design workshops and analyze the structure of the recorded decision spaces.

Chapter 7

The Analyzer

In this chapter, we pick up the research questions that we have scoped and defined in Chapter 4. Addressing them requires a definition of qualitative and quantitative measures of properties of the architecture position making process. To this end, we investigate how the argumentation viewpoint, that we have proposed in Chapter 5 can be used. We apply the Goal Question Metric (GQM) methodology by Basili et al. [BCR94] to investigate the collaborative architecture position making process.

7.1 Goals

One of the virtues of architecture positions is their consistency. In order to evaluate it, one would need to see how aligned the positions of multiple stakeholders are. In the happy case, all positions would have their positions aligned. Decisions with colliding alternatives would indicate that the design process is still underway. Decisions with no positions would reveal white spots in the design that would require further effort.

The goal, therefore, is to assess the level of consensus of a position model involving multiple position makers. We are interested in studying both the current (static) level of consensus as well as its dynamics, in order to observe how the current level has been reached over time.

This is a pre-requisite to achieving a high level of team situational awareness, since without agreement among all participants on what is being perceived, on how to understand it, and most importantly on what the expected consequences of a position are, the likelihood of making a position collaboratively will be very low.

7.1.1 Questions

We state the three following questions that refer to the goal stated in the previous section; with particular emphasis and focus on assessing the consensual qualities of architectural positions.

Question 1: How aligned are the positions?

As introduced in the previous section, one of the virtues of architecture positions is their consistency. Due to the wicked nature of the software architecture design problem, it is not feasible to make a comprehensive assessment of position model completeness and consistency. Nevertheless, we find it perfectly feasible to assess completeness and consistency in terms of the degree of alignment within the argumentation viewpoint of the position model.

Question 2: How volatile is the consensus on the positions?

Building upon the previous question, it would be interesting to see how a proportion of the position states have been changing within a particular timespan of a design workshop or even over the course of the whole duration of the project.

Question 3: How democratic are the positions?

There exists a variety of collaborative position making strategies [Kva00]. These range from an authoritarian, centralized, single-handed process, through collaborative positions based on consensus building, finish with the so-called "design by committee", where each stakeholder is required to participate (see Section 2.2.2). Each of these strategies has particular expectations about the number of stakeholders and their degree of participation as they contribute their positions in the context of a particular design position.

7.1.2 Metrics

In [NP11] we introduced a range of metrics that broadly addressed the structure and dynamics of the software architecture position models. Here we recall two of the metrics we drafted earlier (structural) and introduce new ones (content and argumentation) focused on the consensus assessment goal and building upon the previously introduced argumentation viewpoint. A summary of the metrics that we present in this Section can be found in Table 7.7.

For clarity, we have adopted the following template to introduce the metric definitions:

ID – an identifier that we are going to use throughout this paper to refer to a given metric;

Name – the informal definition of the metric's purpose and semantics,

Parameters – if the calculation of the metric can be parameterized, then we list the parameters to be supplied;

Domain – lists the types of position items that a given metric can be applied to (e.g., Decisions, Issues or Alternatives);

Scale – specifies the nature of the metric result, as proposed in [Ste46] (e.g., Ratio vs. Ordinal);

Range – defines the possible values of the metric;

The metrics that we propose in the following sections are divided into categories reflecting their nature. We distinguish structural, and content related metrics.

Structural Metrics

We selected two of the structural metrics introduced in [NP11], which give an indication of both the number of positions that will need to be taken within a design workshop (or in general, a given project) and the effort required to reach a complete choice. In Figure 7.1 we present an example of a design space with three (partially overlapping) projects with a number of design issues and alternatives. The Table 7.1 contains values calculated for metrics 1 and 2 over this design space.

Metric 1 – Issue count
Domain: *Project*, Scale: *Ratio*, Range: *[0,N]*

A position space typically consists of numerous design issues. Some of the issues can be reused in multiple projects. This metric represents the number of design issues within a particular project.

Metric 2 – Alternative count
Domain: *Project, Issue*, Scale: *Ratio*, Range: *[0,N]*

A design issue can be addressed by choosing from among multiple design alternatives. This metric counts the number of such alternatives within the context of a particular design issue. Its value (if greater than 1) can be used to estimate the effort required to make a complete choice over the issue. Other interesting observations can be achieved by aggregating the results of this metric for a set of design issues, for example, those contained within a project.

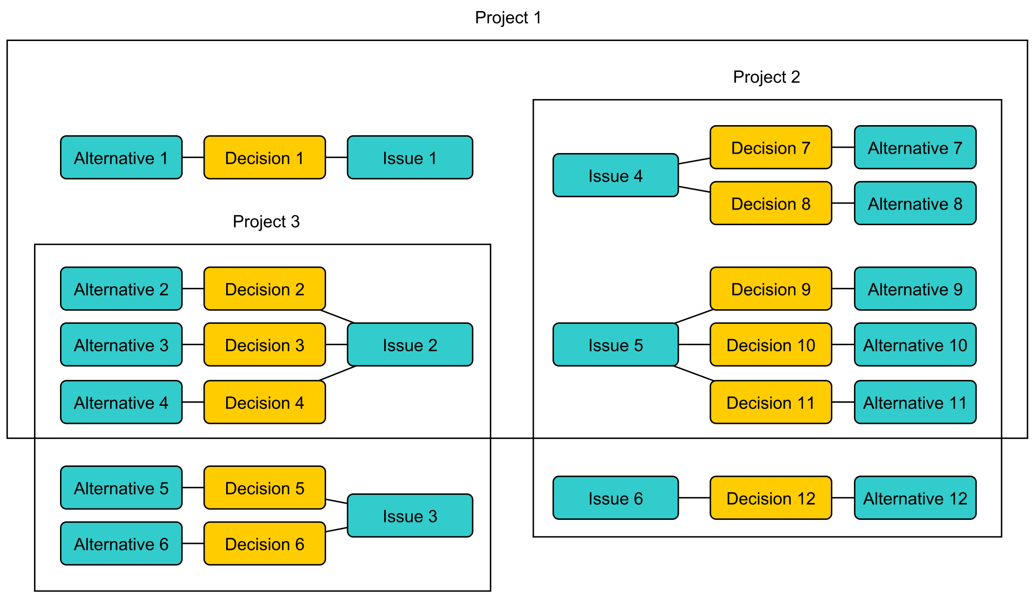


Figure 7.1. An example of design space comprising three projects. Modeled according to the position argumentation viewpoint

Scope	Metric 1 - Issue count	Metric 2 - Alternative count
Project 1	4	9
Project 2	3	6
Project 3	2	5
Issue 1	-	1
Issue 2	-	3
Issue 3	-	2
Issue 4	-	2
Issue 5	-	3
Issue 6	-	1

Table 7.1. Values for metrics 1 and 2 calculated for sample design space presented in Figure 7.1

Content Metrics

Content-related metrics focus on aspects related to the core of the position items, i.e., their attributes (like name, authorship meta-data, and timestamps), and their dynamics (the assumption is that all changes to the position model are logged).

Metric 3 – Relative number of contributors
Domain: *Project, Issue, Alternative*, Scale: *Ratio*, Range: *%*
Assuming that each position item is labeled by the user who has created it, it is interesting to observe how many users have contributed new items within the context of the project, issue, or alternative respectively. In order to mitigate the effect of diverse design team sizes, we propose to analyze the number of contributors relative to the total number of design team members.

Metric 4 – Relative number of position makers
Domain: *Issue, Decision*, Scale: *Ratio*, Range: *%*
Like Metric 3, this metric counts the number of design team members involved in expressing positions on a particular position. Again, as in Metric 3, this metric calculates the number of position makers relative to the total number of design team members.

Metric 5 – Activity timespan
Domain: *Project, Issue, Alternative, Position*, Scale: *Ratio*, Range: *[0,N]*
Decision items are created, updated and, in some cases, eventually deleted from the design space. Their lifecycle involves a series of events that correspond to state modifications by some of the architects. As the *activity timespan* of an item we define the duration between the first event (typically the creation of the item) and the last recorded event corresponding to the item.

Metric 6 – Time since last change
Domain: *Issue, Alternative, Position*, Scale: *Ratio*, Range: *[0,N]*
This metric calculates the time elapsed since the last recorded event related to the particular position item. The time reference would be either the current time, for the live measurements, or the end of the time-frame for a time-boxed experiment.

```
2075    creation        User_1
2075    state   no alternatives
2081    update  User_1  6
2082    update  User_1  7
2089    update  User_1  14
2089    blured  User_1  10
2111    state   no positions
2111    relation created         from: Decision_18       to: Issue_9 (no)       User_1
2111    position        (no)    User1   no positions
2135    relation created         from: Deicision_19      to: Issue_9 (no)       User_1
2135    position        (no)    User1   no positions
2135    state   incomplete
2139    position        Positive        User1   aligned Decision_19
2317    position        Negative        User2   aligned Decision_18
2317    state   complete
```

Figure 7.2. An example of an anonymized event log recorded for a design issue

Metric	Absolute value	Relative value
3. Relative number of contributors	3	3/9=33%
4. Relative number of position makers	2	2/9=22%
5. Activity timespan	2317-2075=242 [s]	-
6. Time since the last change	5400-2317=3083 [s]	-

Table 7.2. Values for metrics 3 and 6 calculated for a sample design issue presented in Figure 7.2

In Figure 7.2 we present an anonymized listing of events recorded with the Software Architecture Warehouse for a design issue. Each row represents an individual event. The first column specifies an event time-stamp expressed in a number of seconds elapsed since the start of the design workshop. It is followed by the type of a recorded event and the name of a user that generated it. In Table 7.2 we have collected values calculated for content metrics (3-6). Due to the nature of relative metrics (3 and 4), in order to interpret the results, it is important to know that in the case of the particular design workshop there was nine participants. For metric 6, it is important to know that in this particular case, the duration of the design workshop was 90 minutes, which is equivalent to 5400 seconds.

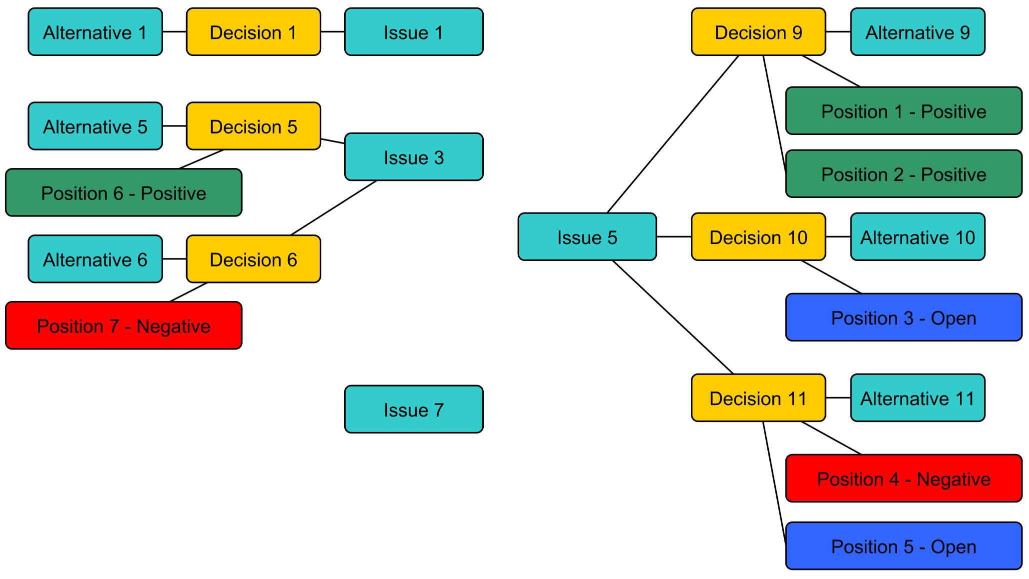


Figure 7.3. An example of a position space with two design issues, six alternatives, and six positions

Argumentation Metrics

The following metrics relate to the argumentation viewpoint (see Figure 5.3 in Section 5.1).

Metric 7 – Position type count
Parameter: *position type*, Domain: *Decision*, Scale: *Ratio*, Range: *[0,N]*

The position meta-model specifies the types of positions that users state about an architecture position. In the particular meta-model that we adopted, we distinguish: positive (accept), negative (reject), and open (neutral) positions. This metric takes a particular position type as a parameter and returns the number of positions of this type that were contributed to the position.

Metric 8 – Consensus state
Domain: *Decision*, Scale: *Ordinal*, Range: *{no positions, aligned, colliding, sealed}*

The positions contributed to the position can be aligned in a variety of ways. As introduced earlier in Section 5.1.1, we distinguish the following elementary consensus (alignment) states:
no positions – when the number of positions is equal to zero,
aligned – all positions in the context of the position are of the same type,
colliding – positions of more than one type exist

sealed – a single position was chosen to settle the argument and seal the position, preventing the addition of further positions.

A position with *aligned* or *sealed* positions is said to be decided for a particular position type (for example decided *positive*).

This metric aggregates over all position types to compute the current consensus state of a particular position based on the previously defined rules.

Metric 9 – Decision consensus state count

Parameter: *consensus state*, Domain: *Issue*, Scale: *Ratio*, Range: *[0,N]*

Building upon Metric 8, this metric counts the number of positions with positions aligned in the given consensus state. The metric is applicable to a design issue, but it may also reveal interesting features when its values are aggregated over the broader, project-level scope.

Metric 10 – Choice state

Domain: *Issue*, Scale: *Issue*, Range: *{no alternatives, no positions, incomplete, conclusive, inconclusive, warring}*

A design issue under consideration, depending on the design alternatives addressing it and their consensus state (compare with Metric 8 and see Section 5.1.1), can take one of the following choice states:

no alternatives – there are no alternatives addressing the given issue,

no positions – there are alternatives addressing the issue, but all corresponding positions are without positions,

incomplete – there are positions addressing positions about the design issue, but there is at least one position in which positions are not *aligned*,

conclusive – all positions are in the *aligned* consensus state, precisely one is decided *positive*, and all others are decided *negative*,

inconclusive – there is more than one position decided as *positive* or *open*,

warring – all positions are aligned with the *negative* position type.

The last three choice states are together referred to as **complete** choices. This metric aggregates the values of Metric 9 over all positions/alternatives associated with a given issue.

In Figure 7.3 we present a sample design space with four design issues, six design alternatives, and seven positions. For positions from this design space, we have calculated values of metrics 7 (Position type count) and 8 (Consensus state), and summarized them in Table 7.3. Further we have calculated values for metric 9 (Decision consensus state count) and 10 (Choice state), and collected them in Table 7.4.

Metric	**7. Position type count**			**8. Consensus state**
Scope	**Positive**	**Negative**	**Open**	
Decision 1	0	0	0	no positions
Decision 5	1	0	0	no positions
Decision 6	0	1	0	aligned
Decision 9	2	0	0	aligned
Decision 10	0	0	1	aligned
Decision 11	0	1	1	colliding

Table 7.3. Values for metrics 7 and 8 calculated for a sample decision space presented in Figure 7.3

Metric	**9. Decision consensus state count**			**10. Choice state**
Scope	**no positions**	**aligned**	**colliding**	
Issue 1	1	0	0	no positions
Issue 3	0	2	0	conclusive
Issue 5	0	2	1	inconclusive
Issue 7	0	0	0	no alternatives

Table 7.4. Values for metrics 9 and 10 calculated for a sample decision space presented in Figure 7.3

Metric 11 – Relative consensus state timespan
Parameter: *consensus state*, Domain: *Decision*, Scale: *Ratio*, Range: *%*

Within its lifecycle, the design position traverses some of the consensus states specified in the context of the Metric 8. For the purpose of investigating the dynamics of the position process, it is interesting in particular to observe the timing of the transitions between these states. This metric accepts a *consensus state* as a parameter and calculates the amount of time a particular position has spent in this state. The calculated value is returned as a relation to the overall timespan of a particular design position (Metric 5).

Metric 12 – Relative choice state timespan
Parameter: *consensus state*, Domain: *Issue*, Scale: *Ratio*, Range: *%*

The design issue lifecycle consists of choice states defined in the context of Metric 10. This metric, given a particular choice state as a parameter, and analogously to Metric 12, calculates the relative amount of time that the design issue has spent in it.

Metric 13 – Time since last position
Domain: *Issue, Decision*, Scale: *Ratio*, Range: *[0,T]*

This metric calculates the absolute amount of time (in seconds) elapsed since the last position (see Section 5.1) was stated in the context of a particular design position.

Metric 14 – Consensus state transition count
Domain: *Decision*, Scale: *Ratio*, Range: *[0,N]*

The consensus state on a given design position traverses multiple states (defined for Metric 8). This metric calculates the number of state transitions. The reason for an increase in the number of consensus state transitions could be, for example, that newly added and/or revoked positions would make the state flip between *aligned* and *colliding*.

Metric 15 – Choice state transition count
Domain: *Issue*, Scale: *Ratio*, Range: *[0,N]*

Like Metric 14, this metric calculates the number of state transitions of the choice state (as defined in the context of Metric 10) for a given design issue.

Metric **Scope**	**Issue 10**	**Issue 11**
3. Relative number of contributors	67% (6/9)	22% (2/9)
4. Relative number of position makers	55% (5/9)	22% (2/9)
12. Relative choice state timespan		
- no alternatives:	3% (102s)	9% (476s)
- no positions:	1% (30s)	0% (11s)
- incomplete	94% (2821s)	17% (876s)
- complete	2% (46s)	73% (3741s)
15. Choice state transition count	5	6

Table 7.5. Values for metrics 3, 4, 12, and 15 calculated for a sample set of design issues in Figure 7.4 and Figure 7.5

```
1379    creation        User4
1379    state   no alternatives
1481    position        (no)    User5   no positions
1481    relation created         from: Decision_15       to: Issue_10 (no)      User5
1481    state   no positions
1484    update  User5   105
1490    update  User5   111
1490    update  User5   111
1511    relation created         from: Decision_16       to: Issue_10 (no)      User5
1511    state   incomplete
1511    position        (no)    User5   no positions
1522    position        Negative        User5   aligned Decision_15
1524    relation created         from: Decision_17       to: Issue_10 (no)      User5
1524    position        (no)    User5   no positions
1562    position        Negative        User5   aligned Decision_15
1598    update  User5   219
1606    update  User5   227
1606    position        Positive        User6   aligned Decision_16
1606    update  User5   227
1607    update  User5   228
1608    update  User5   229
1611    position        Positive        User5   aligned Decision_16
1631    position        Positive        User5   aligned Decision_16
1646    state   complete
1646    position        Positive        User5   aligned Decision_17
1662    position        Negative        User6   aligned Decision_15
1692    position        (no)    User4   no positions
1692    state   incomplete
1702    relation created         from: Decision_18       to: Issue_10 (no)      User4
1713    position        Positive        User6   aligned Decision_17
1747    position        (no)    User4   no positions
1748    relation created         from: Decision_19       to: Issue_10 (no)      User4
1820    position        Negative        User5   colliding Decision_17
2069    position        Negative        User7   aligned Decision_15
2099    position        Negative        User7   aligned Decision_15
2158    position        Positive        User7   aligned Decision_16
```

Figure 7.4. An anonymized event log recorded for a design issue 20

```
1483    creation        User_8
1483    state   no alternatives
1491    update  User_8  8
1634    update  User_8  151
1959    relation created         from: Decision_20       to: Issue_11 (no)      User_8
1959    position        (no)    User_8  no positions
1959    state   no positions
1970    state   incomplete
1970    relation created         from: Decision_21       to: Issue_11 (no)      User_8
1970    position        (no)    User_8  no positions
2739    position        Negative        User_9  aligned Decision_20
2843    state   complete
2843    position        Open    User_8  aligned Decision_21
3096    position        AntiOpen        User_8  no positions
3096    state   incomplete
3099    position        Positive        User_8  aligned Decision_21
3099    state   complete
```

Figure 7.5. An anonymized event log recorded for a design issue 21

In Figure 7.4 and Figure 7.5 we provide two examples of anonymized event logs collected for two different design issues (11 and 12). Every row in Figures starts with a time-stamp expressing a number of seconds elapsed since the beginning of the design workshop. Following fields are specific to the event type. In Table 7.5 we collected values Metrics 3, 4, 12 and 15, calculated these design issues.

```
1058    creation        User_1
1058    relation created         from: Alternaitve_20    to: Issue_12 (no)      User_1
1058    position        (no)    User_1  no positions
1060    focused User_1
1071    update  User_1  13
1072    update  User_1  14
1075    update  User_1  17
1076    update  User_1  18
1106    update  User_1  48
1112    blured  User_1  52
1144    position        Negative        User_2  aligned Decision_30
1204    position        Positive        User_1  colliding Decision_30
1396    position        AntiNegative    User_2  aligned
```

Figure 7.6. An anonymized event log recorded for a design decision 30

```
3511    creation        User_1
3511    relation created         from: Alternative_21    to: Issue_13 (no)      User_1
3511    position        (no)    User_1  no positions
3514    update  User_1  3
3515    update  User_1  4
3521    update  User_1  10
3521    update  User_1  10
3646    position        Positive        User_1  aligned Decision_31
3713    position        Positive        User_2  aligned Decision_31
3820    position        Negative        User_3  colliding Decision_31
```

Figure 7.7. An anonymized event log recorded for a design decision 31

Metric	**Scope**	**Decision 30**	**Decision 31**
11. Relative consensus time:			
	- no positions:	2% (86s)	5% (135s)
	- aligned	94% (4251s)	6% (174s)
	- colliding	4% (192s)	89% (2558s)
13. Time since last position		59% (3191s)	47% (2558s)
14. Consensus state transition count		3	2

Table 7.6. Values for metrics 11, 13 and 14 calculated for two sample positions based on event logs presented in Figure 7.6 and Figure 7.7

In Figure 7.6 and Figure 7.7, we provide anonymized event logs recorded for design decisions 30 and 31. As in the case of design issues, every row in Figures starts with a time-stamp and is followed with event specific fields. In Table 7.6 we summarize values of metrics 11, 13 and 14 calculated for these decisions.

7.1.3 Interpretation Model

In this section we, show how to answer the three questions stated in Section 7.1.1 with the help of the metrics defined in Section 7.1.2. In our interpretation model, we distinguish two types of input: first order, which directly influences the answer, and second order, which impacts the precision and the uncertainty of the interpretation.

Question 1: How aligned are the positions?

In order to assess the alignment of the positions in the context of a position, one needs to see what the types of the positions are. The elementary number of the positions of a particular type can be measured with the use of Metric 7 (Position type count), but the interpretation of those numbers is done with the use of Metric 8 (Consensus state). In the happy case, all positions would have their positions aligned on a single position type. Decisions with colliding alternatives would indicate that the design process is still underway. Decisions without positions would be potentially white spots in the design that would require further attention. Therefore, in terms of alignment preference the *aligned* consensus state is the most desirable, whereas the *colliding* consensus state is the least desirable.

Analogously, in the context of a design issue, for the sake of position model completeness, the most desirable choice states are *complete*, in particular, the *conclusive* choice, followed by the *inconclusive* and *warring* choices. The *incomplete* and *no positions* choices are strong indicators of the lack of completeness of a position model, which is a pre-requisite for assessing its alignment/consensus.

The second-order effects on position alignment assessments come from the sample size being evaluated. For example, in the context of a design issue, a conclusive choice can be achieved with only one alternative, but this outcome is relatively weak compared to the situations in which multiple alternatives were taken into consideration, and only one was eventually accepted (see Metric 2). Similarly, in the context of a particular position, the aligned consensus state can be achieved with as little as one position. Therefore, we find it relevant to take into account the total number of positions (see Metric 7), and even more importantly, the relative number of unique position contributors (see Metric 3). Clearly, at least two positions (by different contributors) are necessary before one can start evaluating their alignment.

Id	Name	Parameter	Domain	Type	Scale	Range	Questions
1	Number of issues	-	Project	structural	Ratio	N	3
2	Number of alternatives	-	Issue	structural	Ratio	N	3
3	Relative number of contributors (creating items)	-	Issue, Alternative, Decision	content	Ratio	%	1,2
4	Relative number of editors (doing updates)	-	Issue, Alternative, Decision	content	Ratio	%	2,3
5	Time since last change	-	Issue, Alternative, Decision	content	Ratio	T	2
6	Activity time	-	Issue, Alternative, Decision	content	Ratio	N	2,3
7	Position count	position type	Issue, Alternative, Decision	argumentation	Ratio	N	1,2,3
8	Decision count with particular consensus state	consensus state	Issue	argumentation	Ratio	N	1
9	Consensus state of particular decision	-	Decision	argumentation	Ordinal	{}	1
10	Choice state of particular issue	-	Issue	argumentation	Ordinal	{}	1
11	Time spent in particular consensus state	consensus state	Decision	argumentation	Ratio	%	2
12	Time spent in particular choice state	choice state	Issue	argumentation	Ratio	%	2
13	Time since last position was stated	-	Issue, Alternative, Decision	argumentation	Ratio	T	2
14	Number of transitions of the consensus state	-	Decision	argumentation	Ratio	N	2
15	Number of transitions of the choice state	-	Issue	argumentation	Ratio	N	2

Table 7.7. Summary of the metrics related to the three questions concerning the consensus goal

Question 2: How volatile is the consensus on the positions?

In the ideal case, the positions would gradually and monotonically move from the *no positions* state into the *aligned* consensus state. In practice, position dynamics can be much more volatile, which implies both a high number of state transitions and some time spent arguing to find a way of leaving the undesired *colliding* state (see Metrics 14 and 11). Ideally, the preferred position dynamics would consist of most of the time being spent in the *aligned* state.

Analogously in the context of a design issue, the desirable choice state traversal would go directly from the initial *no alternatives* state, through *no positions*, and briefly through *incomplete*, before finally reaching the *conclusive* choice state. Any additional choice transitions and time spent in undesirable choice states would represent an increase in the overall position volatility (see Metrics 12 and 15 respectively).

Another factor contributing to position volatility is the age of the particular position item. We find it useful to measure two age components of an item, that is, the activity time (Metric 5) and the time elapsed since the last change (Metric 6). Based on the assumption that past volatility has influence over future position item dynamics, we would argue that a short activity time and/or a short span of time since the last change, by extrapolation are likely to coincide with more activity around the item. Conversely, if a particular item did not show any activity for a long time, we would expect it to remain stagnant.

Analogously to Question 1, the second-order influence on the volatility of a position item comes from the size of the sample under investigation. Particular positions are more likely to exhibit volatility if the total number of positions and contributors is low (Metrics 3, 4, and 7).

We can speculate that a high level of volatility is a strong indicator of an immature, controversial and ad-hoc design.

Question 3: How democratic are the positions?

As a rule of thumb, we can say that in the happy case stakeholder involvement would match the chosen position making-strategy. In particular, for the *collaborative* position making strategy, we would expect a high participation rate (Metric 3) and a relatively large number of design issues with a complete choice state. In cases in which the position making is still in progress, a number of inconclusive and warring choice states would be expected (Metric 10). The *competitive* strategy can be characterized by a relatively high number of conclusive position choices (Metric 10) and by a participation rate that is limited to a few or a single position maker (Metrics 3 and 4). The *authoritative* position making strategy has similar traits, but additionally we would expect a number of positions in the

Feature	**Data Warehouse (DW)**	**Analyzer**
Collection	OnLine Transaction Processing System (OLTP)	SAW web-client and -server
Staging	Extraction Translation and Loading (ETL)	Analyzer stage 1
Analysis	Data Marts	Analyzer stage 2
Reporting	External System	External (LaTeX+pgfplots)

Table 7.8. Mapping data warehouse and Analyzer features

sealed consensus state.

As in the case of the previous questions, there are a number of second-order influences. In particular, a small number of stakeholders, or a single position maker in the extreme case, would make it impossible to speak about any kind of collaborative position making strategy. Also, having a small number of considered design issues, a possibly having only a single alternative to choose from, can pose a significant threat to the significance of the answer.

7.2 Analysis Workflow

The analysis of the architecture design workshop dynamics, pictured in Figure 7.8, starts with data acquisition, proceeds with two analysis stages, and finishes with a graphical representation of the outcome (see Table 7.8 and compare with Table 6.1). As execution of the analysis process takes a significant amount of time, for the sake of development and debugging efficiency, the core of the analysis has been split into two, decoupled stages. After completion of the first stage of analysis, the intermediate data ((micro-)Metric Matrix CSV files) is saved, so that it can be used in multiple runs of the second stage. The second stage of analysis produces a number of CSV files (one per metric) that are parsed by PGFplots[1] to embed charts into the LaTeX documents.

In the following sections, we provide a thorough walkthrough of the data analysis workflow.

7.2.1 Data Acquisition

In order to answer the questions stated in the previous Sections, we focused our observations on the architecture workshops supported with the Software

[1]`http://pgfplots.sourceforge.net/`

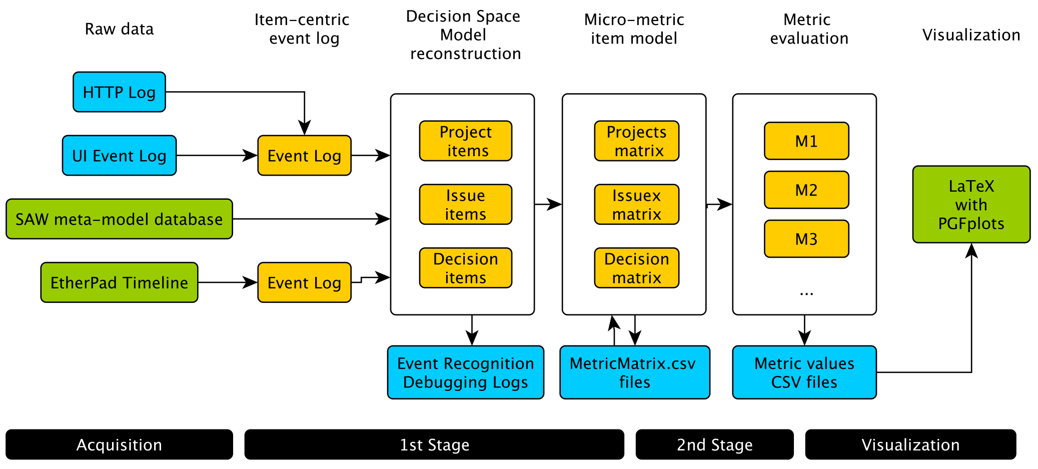


Figure 7.8. Six step design workshop analysis workflow with text-based data (blue), in-memory representation (yellow) and other data sources (green).

Architecture Warehouse, using a collaborative, free-form text editor (EtherPad, see Section 3.5) as a baseline. Therefore, the data used for the analysis had two distinctive origins.

The Software Architecture Warehouse records the process of architecture position making from three sources:

HTTP requests – Software Architecture Warehouse being a web application communicates with its server back-end over the HTTP protocol. The server is configured to log each request including credentials and an optional payload.

Client UI events – The rich-client application front-end of the SAW is event-driven. The high-level events encapsulating information about the actions triggered by the user are scrupulously passed over to the server. The event description typically encapsulates information about the user action context such as active user, project, visible design issue, etc.

Client DOM capture – Inspired by the web user behavior tracking functionality provided by the SMT2[2] suite, we have equipped the SAW with a mechanism that not only tracks the mouse pointer position within the web-application

[2]`https://code.google.com/p/smt2/`

but performs a periodic snap-shot of the complete DOM model of the web-application and delivers it to the central logging system.[3]

The architecture position making process supported by the EtherPad required a different instrumentation approach. In order to analyze the dynamics of the process, we used so-called timeline slider functionality, which allowed us to identify the precise time of creation and/or editing of particular design issues, alternatives, and positions. The authorship of the position model items was identified thanks to the fact that EtherPad color-codes content provided by the individual users. The transformation of the EtherPad position process transcript into the stream of events, as well as the assignment of unique identifiers to the recorded position items, was done manually.

7.2.2 Event Identification

The data collected by the instrumentation of the architecture workshop is not suitable for use directly with the system of metrics that we proposed. In order to achieve the right abstraction level, we processed the raw log data into a log of events that concern particular position model items. The event logs were collected for each architecture design workshop individually. Each registered event was described according to the following attributes:

Time-stamp – at first, the full date and time of the event occurrence is registered. In the pre-processing stage, the absolute time-stamps are turned into times relative to the beginning of the design workshop.

Item Type – determines the type of the position item, as declared in the position meta-model

Item ID – uniquely identifies the position model item. In the case of data collected from the SAW, the IDs are assigned automatically; in the case of the EtherPad; this is done manually.

Action – one of the CRUD actions, or position meta-model specific actions such as **Relate**, or **Decide**

User – a unique identifier of the user involved in the action

Content – extra information about the event, such as item type for the **Create** action or position type for the **Decide** action

[3]Implemented extensions are available under: `https://github.com/ian7/smt3`

Time-stamp	Item Type	Item ID	Action	User	Content	Parameter
09:01:57	Issue	102	create	Fabio		
09:02:09	Issue	102	update	Fabio		Single DB for all?
09:02:13	Alternative	4	create	Fabio		
09:02:13	Alternative	4	relate	Fabio	Issue	102
09:02:13	Issue	103	create	Cesare		
09:02:14	Issue	103	update	Cesare		Distributed DB?
09:02:40	Alternative	4	update	Fabio		a DB to store the catalogue data and one for the user and their orderds
09:03:13	Alternative	5	create	Rasel		
09:03:14	Alternative	5	relate	Rasel	Issue	102
09:03:14	Alternative	5	update	Rasel		no
09:03:20	Alternative	5	decide	Rasel	Open	can be good to separate sensitive data (customer credit card number not in same database ad catalog/order)
09:03:26	Alternative	4	update	Fabio		a DB to store the catalogue data and one for the user (address, e-mail,...) and their orderds

Table 7.9. An example set of raw events recorded during the design workshop (Run 3) assisted by the use of the EtherPad (see Table A.1)

Parameter – payload for the **Update** action, position rationale for the **Decide** action, or identifier for the tip of the **Relate** action, etc.

In Table 7.9, we provide an extract of the raw event log recorded at the beginning of the design workshop supported by the use of the EtherPad. In the extract, we can recognize creation and editing of two design issues (IDs: 102 and 103) and two design alternatives (IDs: 4 and 5) related to the first issue. There is also one (Open) position provided by the user.

7.2.3 Decision Model Structure Recognition

In the next step of the analysis process, we determined the elementary position model structure, by identifying events related to the individual items and tracing the relations between them. The position model structure is essential for recovering position item context, such as the reference between the position and issue-to-alternative relation, etc. Within the position model structure, the choice-state of individual positions were analyzed first, so that the aggregated choice state could be inferred in the context of the design issue. The recovery of the position model structure is essential and cannot be replaced with the snapshot of the final state of the position model. This is due to the fact that the model does not grow monotonically and thus intermediate states of the model can be significantly different from the final outcome.

At this point, the analyzer collects all events identified in the previous stage to instantiate their object representation. Each event object instance has access to the previous events registered for a particular position model item, making it possible to identify and track each temporal state within the life-cycle of the position item. In Figure 7.9 we provide a textual representation of an example of the event list identified for a design issue. Each entry starts with a time-stamp

```
3502  creation  marcin@sonyx.net
3502  state no alternatives
3506  update  marcin@sonyx.net  4
3534  relation created   from: 51778257da300c56d5000007  to: 51778237da300c57f1000004 (no)  marcin@sonyx.net
3534  state no positions
3534  position  (no)  marcin@sonyx.net  no positions
3552  state incomplete
3552  position  (no)  marcin@sonyx.net  no positions
3552  relation created   from: 51778269da300c56a7000002  to: 51778237da300c57f1000004 (no)  marcin@sonyx.net
5224  position  Positive  User2 aligned 51778257da300c56c4000002
5326  position  Negative  User2 aligned 51778269da300c57f1000005
5326  state complete
```

Figure 7.9. An example of an event log recorded for a design issue with two alternatives and two positions

relative to the beginning of a design workshop and continues with a logging message tailored to the specific event type. In the event listing, it can be noticed how the choice **state** of the design issue changes, as alternatives and positions are added, from **no alternatives**, through **no positions** and **incomplete** finishing with **complete**.

7.2.4 Micro-Metrics

The metrics that we have defined in Section 7.1.2 operate at a relatively high level, therefore they cannot be calculated directly from the data (events) recorded during the architecture design workshops. In order to fill this gap, we devised a set of micro-metrics suitable for estimating properties of each individual position model element and mapping back to the Metrics (see Table 7.10). We have implemented micro-metrics ranging from type-agnostic, such as the measurement of the item life-span, to type-specific, such as project, issue, and alternative related activity. The micro-metrics are evaluated individually and persisted in the CSV file specific to the item type. In Table A.3 we present captured alternatives with the respective micro-metric values calculated.

In the following sub-sections we are going to document a number of micro-metrics using the following notation:

Micro-Metric Name {Optional Parameter} [returned value type] –
brief description and references.

7.2.5 Item Identification Micro-Metrics

Project [text] – An identifier of the position space to which a given element belongs (see Project micro-metrics in Section 7.2.5)

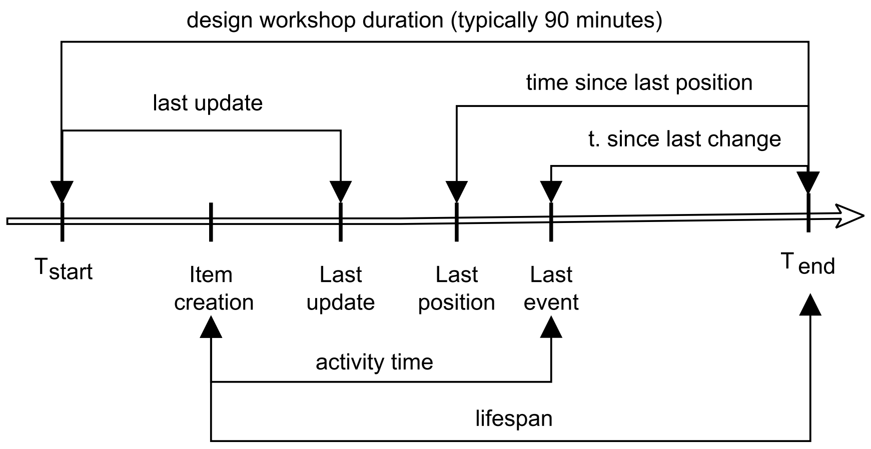


Figure 7.10. A schematic lifecycle of the architecture design workshop with events specific to the item dynamics micro-metrics marked on the timeline

ID [text] – A unique identifier for the position space element, as recognized during Event Identification (see Section 7.2.2)

Destroyed [0,1] – Due to the fact that position space is undergoing very dynamic transformations during the position making process, some of its elements are going to be destroyed in the process. This micro-metric returns one if the given element no longer exists in the final position space, zero otherwise

Item Dynamics Micro-Metrics

The following metrics examine the time-dynamics of position items. It needs to be mentioned that some of the item events, such as **Last update**, **Last position**, and **Last event**, are depicted separately in the Figure 7.10, but in practice can occur in at the same time, for example, when an update or position happens to be the last event related to the position item.

Update Count [0..N] – number of individual updates to the item description

Last Update [seconds] – number of seconds elapsed between the beginning of the design workshop and the last update event related to a given item

Activity time [seconds] – a timespan in seconds between the first event (creation) and the last event for given item

-	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15
Project ID	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
ID	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Destroyed	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Update Count	-	-	-	-	x	x	-	-	-	-	-	-	-	-	-
Last Update	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Activity Time	-	-	-	-	x	-	-	-	-	-	-	-	-	-	-
Lifespan	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Last Change	-	-	-	-	-	x	-	-	-	-	-	-	x	-	-
Contributors	-	-	x	-	x	x	-	-	-	-	-	-	x	-	-
Time since last position	-	-	-	-	x	-	-	-	-	-	-	-	x	-	-
Position Count	-	-	-	-	x	-	x	x	-	-	-	-	-	-	-
Positive	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-
Positive Revoked	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-
Negative	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-
Negative Revoked	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-
Open	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-
Open Revoked	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-
Final State	-	-	x	x	-	-	-	x	-	-	x	-	-	x	-
Final Decision	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-
Editors	-	-	-	-	x	x	-	-	-	-	-	-	x	-	-
Deciders	-	-	-	x	x	x	-	-	-	-	-	-	x	-	-
Time in No Positions	-	-	-	-	-	-	-	-	-	-	x	-	-	-	-
Time in Aligned	-	-	-	-	-	-	-	-	-	-	x	-	-	-	-
Time in Colliding	-	-	-	-	-	-	-	-	-	-	x	-	-	-	-
Consensus State Changes	-	-	-	-	-	-	-	-	-	-	-	-	-	x	-
Alternatives Count	-	-	-	-	x	x	-	-	x	x	-	-	x	-	-
Alternatives in No Positions	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-
Alternatives in Colliding	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-
Alternatives in Aligned	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-
Alternatives in Sealed	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-
Final Choice	-	-	x	x	-	-	-	-	-	x	-	x	-	-	x
Choice State Changes	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x
Time in No Alternatives	-	-	-	-	-	-	-	-	-	-	-	x	-	-	-
Time in No Positions	-	-	-	-	-	-	-	-	-	-	-	x	-	-	-
Time in Incomplete	-	-	-	-	-	-	-	-	-	-	-	x	-	-	-
Time in Complete	-	-	-	-	-	-	-	-	-	-	-	x	-	-	-
Project ID	x	x	-	-	-	-	-	-	-	-	-	-	-	-	-
Issues Created	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Alternatives Created	x	x	-	-	-	-	-	-	-	-	-	-	-	-	-
Issues Edited	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Alternatives Edited	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Positive Positions Created	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-
Negative Positions Created	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-
Open Positions Created	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-
Decisions Revoked	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-

Table 7.10. Summary of coupling between micro-metrics (first column) and metrics (M1-M15). See Section 7.1.2 and Section 7.2.4 for extensive description of metrics and micro-metrics

Lifespan [seconds] – measures the time elapsed between the first recorded event related to a given item and the end of the design workshop

LastChange [seconds] – measures the time elapsed since the last recorded event of the item and the end of the design workshop

Editors [0..N] – the number of unique users that edited the description of a particular item

Contributors [0..N] – the number of unique users that did any action that altered the state, by editing or relating to given item

Time since last position [seconds] – the time elapsed between the last position related to a given item and the end of the design workshop. Returns -1 in cases where there are no positions

Argumentation Micro-Metrics

The following micro-metrics concern an architectural position that relates to one design **Issue** and one design **Alternative**, with multiple **Positions** related to it. Table A.3 provides an example of argumentation micro-metric values calculated for a selected range of design positions that we acquired during the evaluation.

Position Count [0..N] – the total number of positions ever recorded for the decision. This includes position revoke events

Compacted Position Count [0..N] – the number of non-revoked positions at the end of a workshop. Hereafter referred as to *final*

{Positive,Negative,Open} [0..N] – the number of non-revoked positions of the respective type (Positive, Negative, Open)

{Positive,Negative,Open} Revoked [0..N] – the number of revoked positions of the respective type (Positive, Negative, Open)

Final State [consensus state] – final state of consensus on the position. It implements the definition of Metric 8 (**Consensus State**) defined in Section 7.1.2

Final Decision [{Positive, Negative, Open}] – if the **Final State** of positions is **aligned**, then this micro-metric returns the type of the aligned positions, otherwise the value is undefined

Deciders [0..N] – counts the number of unique users that have contributed positions to a given position

Time in {No Positions, Aligned, Colliding} [seconds] – through its lifecycle, a position traverses a number of consensus states as defined in Section 5.1.1. This micro-metric calculates the amount of time (in seconds) that a particular position has spent in given consensus state

Consensus State Changes [0..N] – this micro-metric counts the number of transitions between the consensus states of particular positions (see Section 5.1.1)

Issue Micro-Metrics

The following micro-metrics concern a number of **Decisions** and **Alternatives** related to a particular design **Issue**, together with relevant **Positions**. In Table A.2, we provide an extract from the data that we acquired during the evaluation.

Alternatives Count [0..N] – counts the number of design alternatives related to a given design issue,

Alternatives in {No Positions, Colliding, Aligned, Sealed} [0..N] – counts the number of alternatives with positions in a particular consensus state (see Section 7.1.2) in the context of a design issue,

Final Choice [choice state] – returns the final choice state of a design issue, as per the definition of the issue's lifecycle (see Section 5.1.1),

Choice State Changes [0..N] – similar to the **Consensus state changes** for a design position, this micro-metric calculates the number of choice state transitions within the lifecycle of a design issue (see Section 5.1.1),

Time in {No Alternatives, No Positions, Incomplete, Complete} [seconds] – analogous to the micro-metric measuring the time that a position spent in a particular consensus state (see Section 7.2.5), this micro-metric returns the time (in seconds) that a design issue has spent in a given choice state,

Deciders [0..N] – counts the total number of unique position makers who have contributed their positions to positions related to a given design issue

Project Micro-Metrics

In this section, we introduce micro-metrics that concern whole position spaces. The snapshot of the dataset that we acquired during our evaluation is presented in the Appendix. The position spaces recorded in parallel with the EP and the SAW carry distinctive IDs but belong to a single run. In order to make this mapping explicit, we have augmented the Table A.1 with two extra columns identifying the **Exercise Run** and the **Source** of the data (see Section 8.2). In cases in which imperfect workshop conditions made it impossible to record the exercise run in a way that provided comparable data for EP and SAW, the position space was excluded from the analysis (**not used** row in Table A.1).

ProjectID [text] – a unique identifier for a position space

{Issues, Alternatives} Created [0..N] – counts the design issues/alternatives within a position space

{Issues, Alternatives} Edited [0..N] – counts the design issues/alternatives in a design space that have been edited at least once

{Positive, Negative, Open} Positions Created [0..N] – counts the total number of positions collected within a particular position space

Decisions Revoked [0..N] – counts the total number of revoked positions in a position space

7.3 Summary

In this chapter, we described the design of the software that we implemented for the purpose of analyzing empirical data collected during the architecture design workshops. Using the Goal Question Metric approach, we identified collaborative position making goals, we then defined questions addressing them, and we specified metrics that can be used to quantify the observation. Finally, we provided details about the implementation of the aforementioned metrics in the analyzer.

In the next chapter, we gather core research ingredients that we have introduced in the previous three chapters. We examine the application of the analytical framework presented in this chapter on positions modeled with use of argumentation viewpoint implemented in the Software Architecture Warehouse, and we compare the results with the positions recorded using a free-form collaborative text editor (EtherPad).

Chapter 8

Evaluation

In this chapter, we present the results of the formative and empirical evaluation of the Software Architecture Warehouse and the Collaborative Architecture Decision Metrics suite. We perform the evaluation on the data recorded during the in-class design workshops. Finally, we conclude with an interpretation of these results and an investigation of the main threats to their validity.

8.1 Formative Evaluation

The purpose of performing a formative evaluation was to orient and guide our research at its early stage. Our main interest in this phase was to address the problem of collaborative architecture design decision consensus (see Section 4.2.2). This lead us to our first research question, that is: how to support collaborative software architecture decision making? During our investigation, we came across the concept of situational awareness (see Section 2.3.3). We have devoted three formative evaluation cycles over the course of two years to research how Software Architecture Warehouse can be used to increase team situational awareness during architecture design workshops. Different releases of the SAW have been used in more than 50 collocated 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 roles of software architects (including that of lead architect), software developers and other stakeholders, such as customers or end-users of the systems being designed. The SAW has also been used in distributed design workshops over conference calls and in hybrid workshops with some collocated participants and others connecting remotely. The feedback received has helped us to refine the support for team situational awareness and improve the tool usability and scalability.

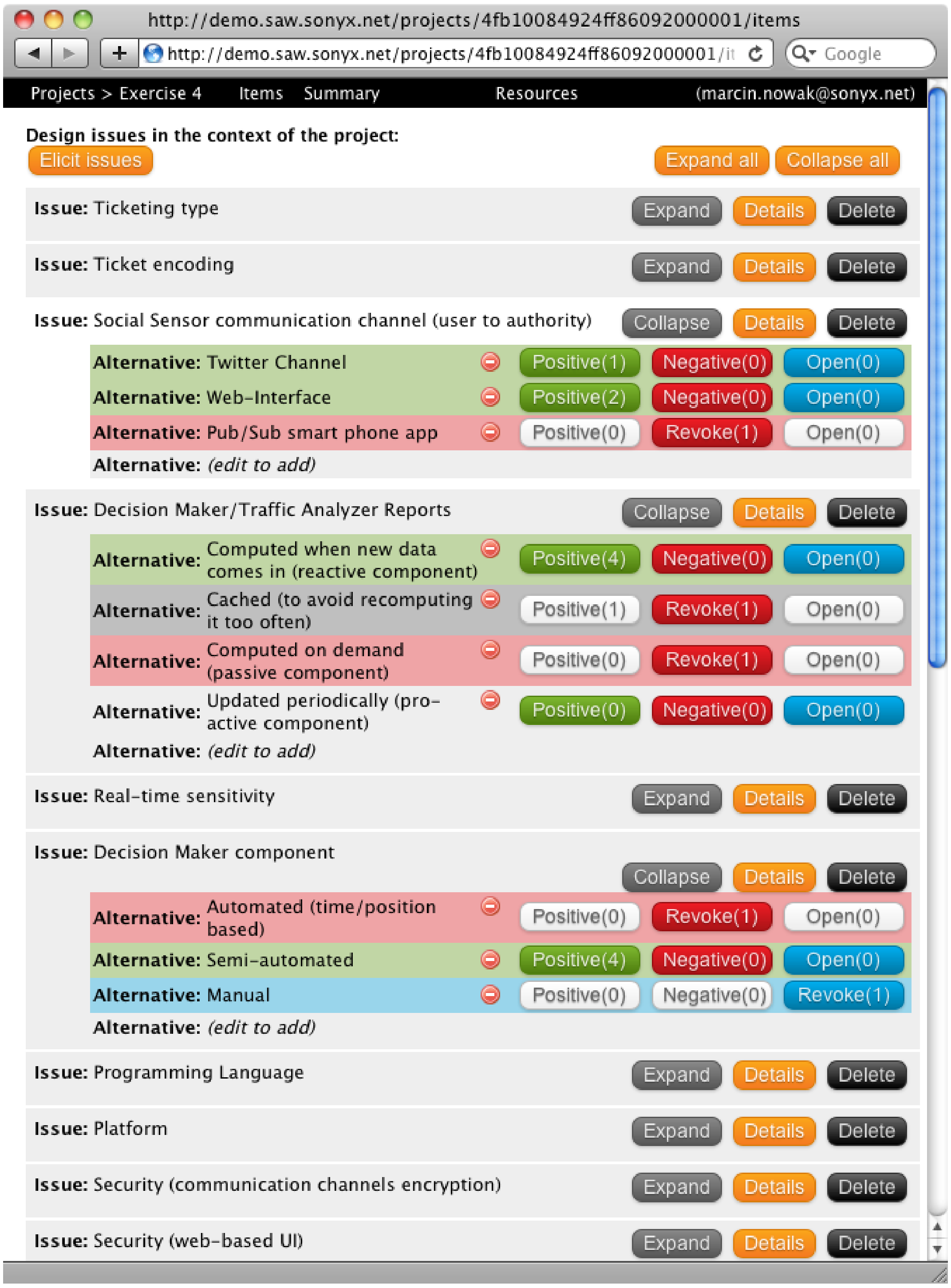
We have observed that the usage patterns and load may vary greatly in intensity over a design workshop session (which average 90 minutes in length), making the real-time performance requirement very challenging to achieve without sufficient resources on the server-side and over an unreliable network. 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 up of up to 100 issues (with five 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 who have accumulated some experience with the tool's user interface can be effective at capturing the discussion while leading it. In other cases, we had to resort to recruiting minute takers (or scribes) who 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 of contributing their input to 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 became aware of the presence of the additional communication channel.

One of the most significant observations that we have made was that adopting the SAW as an extra communication channel set a very high demands for performance of its web user interface, in particular for the responsiveness (see Section 3.4). The first prototypes of the SAW that were based on the typical MVC-based web application style were not keeping up with the fast dynamics of the design workshops. The situation was particularly dire for the views that aggregated information about multiple decision items undergoing rapid change (see Figure 8.1).

The feature of broadcasting pointers (see Section 6.4.6) 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 were able to 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 on each alternative (see Figure 8.2). At a more


http://demo.saw.sonyx.net/projects/4fb10084924ff86092000001/items
http://demo.saw.sonyx.net/projects/4fb10084924ff86092000001/it
Google
Projects > Exercise 4
Items
Summary
Resources
(marcin.nowak@sonyx.net)
Elicit issues
Expand all
Collapse all

Design issues in the context of the project:

Issue: Ticketing type — Expand | Details | Delete

Issue: Ticket encoding — Expand | Details | Delete

Issue: Social Sensor communication channel (user to authority) — Collapse | Details | Delete

Alternative: Twitter Channel	Positive(1)	Negative(0)	Open(0)
Alternative: Web-Interface	Positive(2)	Negative(0)	Open(0)
Alternative: Pub/Sub smart phone app	Positive(0)	Revoke(1)	Open(0)
Alternative: *(edit to add)*			

Issue: Decision Maker/Traffic Analyzer Reports — Collapse | Details | Delete

Alternative: Computed when new data comes in (reactive component)	Positive(4)	Negative(0)	Open(0)
Alternative: Cached (to avoid recomputing it too often)	Positive(1)	Revoke(1)	Open(0)
Alternative: Computed on demand (passive component)	Positive(0)	Revoke(1)	Open(0)
Alternative: Updated periodically (pro-active component)	Positive(0)	Negative(0)	Open(0)
Alternative: *(edit to add)*			

Issue: Real-time sensitivity — Expand | Details | Delete

Issue: Decision Maker component — Collapse | Details | Delete

Alternative: Automated (time/position based)	Positive(0)	Revoke(1)	Open(0)
Alternative: Semi-automated	Positive(4)	Negative(0)	Open(0)
Alternative: Manual	Positive(0)	Negative(0)	Revoke(1)
Alternative: *(edit to add)*			

Issue: Programming Language — Expand | Details | Delete

Issue: Platform — Expand | Details | Delete

Issue: Security (communication channels encryption) — Expand | Details | Delete

Issue: Security (web-based UI) — Expand | Details | Delete

Figure 8.1. A view of the list of design issues in the project in the early prototype of the Software Architecture Warehouse

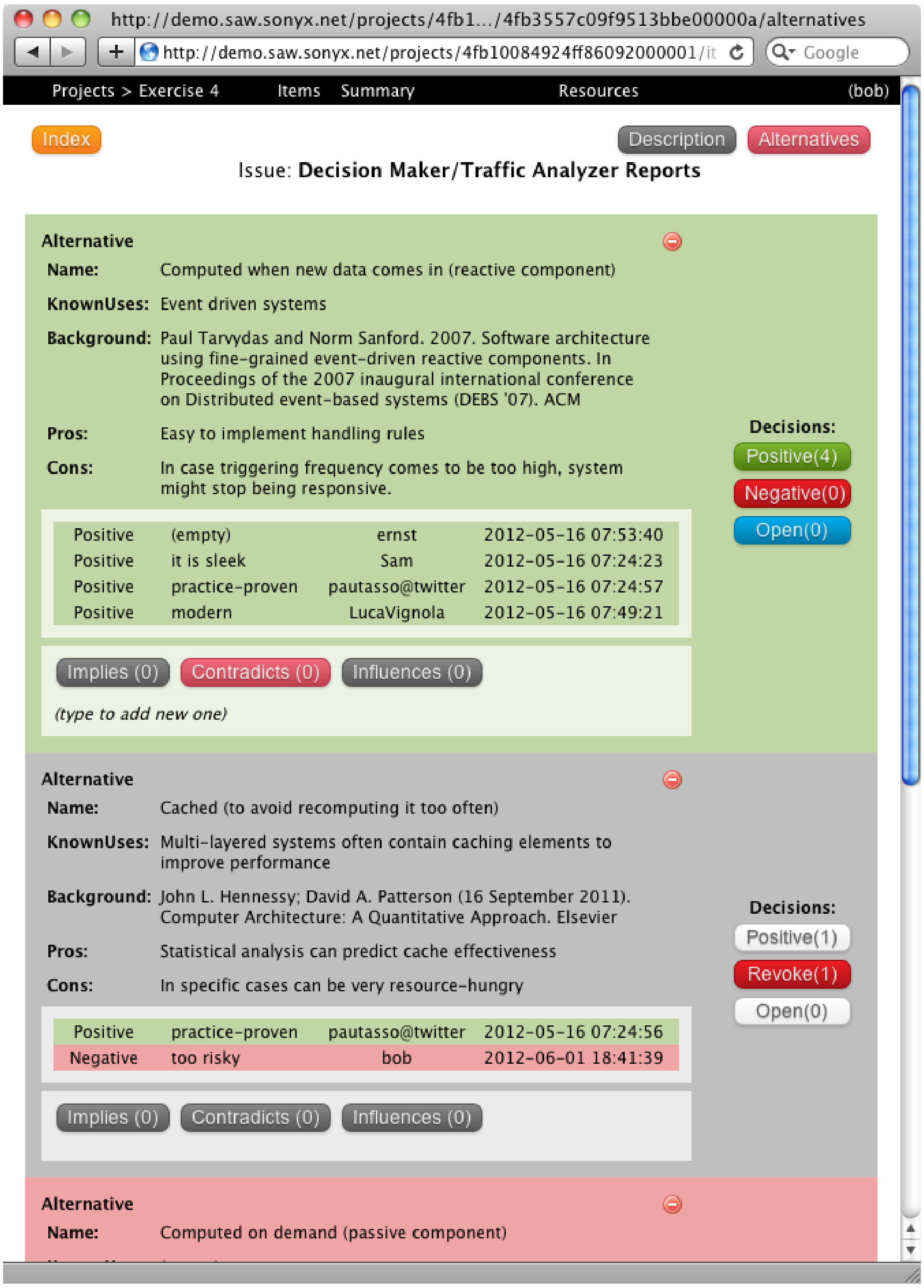
fine-grained level, users were also able to enter the rationale for their positions. This required additional time and effort and was met with some resistance. In particular, not all users can immediately and independently provide a rationale for their positions and prefer to wait for others to express their viewpoints and piggy back their position on the previous ones. Finally, we added the ability to revoke positions since people needed to be able to change their minds as the consensus building process was taking place (see Figure 8.3).

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 (see Figure 8.3). This was used to track the progress of the workshops. This way the tool can provide a separate list of open issues that need to be decided upon; this list keeps shrinking during the closed phase of the discussion, thereby providing all participants with a sense of accomplishment while the list of sealed and decided issues grows.

We have also observed that the SAW added an extra communication channel to the discussion in such a way that workshop participants could contribute to the design space without interrupting the ongoing discussion. Similarly, some participants who 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.

8.2 Empirical Evaluation

The empirical data was collected during the exercises accompanying the Software Architecture and Design class at the University of Lugano during the Spring Semester of 2013. The experiments were conducted over nine design workshops, each lasting for 90 minutes. The participants received a design exercise description (about one page long) a day before an exercise was to take place so that they had enough time to familiarize themselves with the domain and the requirements. Usually, an exercise description was not exhaustive enough to proceed with detailed design, so there was some time invested in the clarifications, which required making decisions about the interpretation of the requirements. The goal of the exercises was to produce a sketch of a software architecture while keeping track of the architectural decisions involved.



http://demo.saw.sonyx.net/projects/4fb1.../4fb3557c09f9513bbe00000a/alternatives
http://demo.saw.sonyx.net/projects/4fb10084924ff86092000001/it
Google
Projects > Exercise 4
Items
Summary
Resources
(bob)
Index
Description
Alternatives

Issue: **Decision Maker/Traffic Analyzer Reports**

Alternative

Name:	Computed when new data comes in (reactive component)
KnownUses:	Event driven systems
Background:	Paul Tarvydas and Norm Sanford. 2007. Software architecture using fine-grained event-driven reactive components. In Proceedings of the 2007 inaugural international conference on Distributed event-based systems (DEBS '07). ACM
Pros:	Easy to implement handling rules
Cons:	In case triggering frequency comes to be too high, system might stop being responsive.

Positive	(empty)	ernst	2012-05-16 07:53:40
Positive	it is sleek	Sam	2012-05-16 07:24:23
Positive	practice-proven	pautasso@twitter	2012-05-16 07:24:57
Positive	modern	LucaVignola	2012-05-16 07:49:21

Alternative

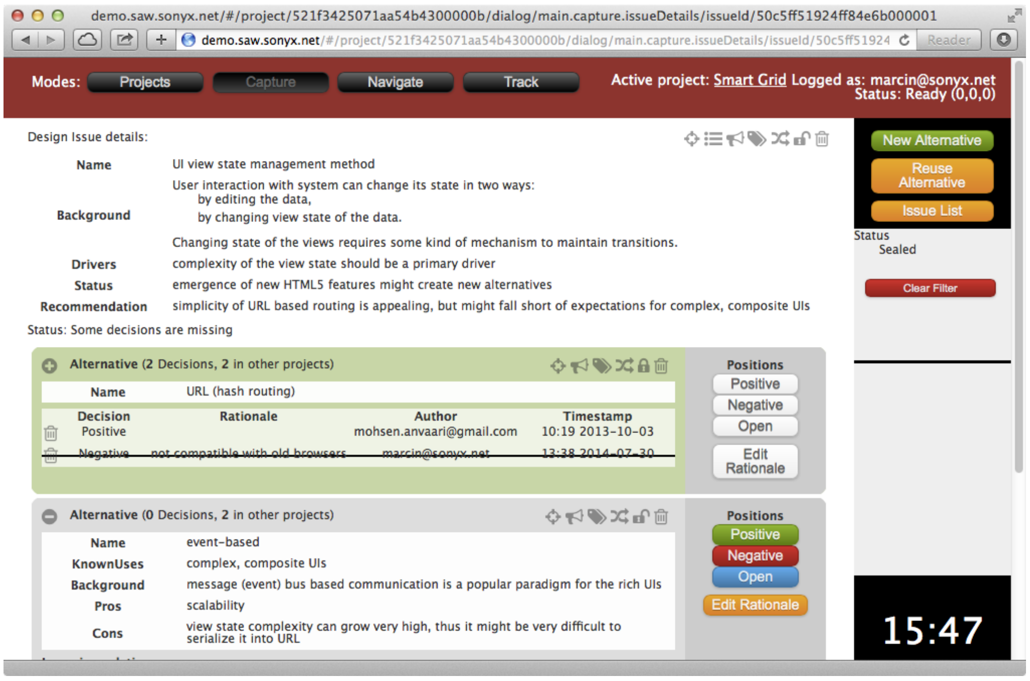
Name:	Cached (to avoid recomputing it too often)
KnownUses:	Multi-layered systems often contain caching elements to improve performance
Background:	John L. Hennessy; David A. Patterson (16 September 2011). Computer Architecture: A Quantitative Approach. Elsevier
Pros:	Statistical analysis can predict cache effectiveness
Cons:	In specific cases can be very resource-hungry

Positive	practice-proven	pautasso@twitter	2012-05-16 07:24:56
Negative	too risky	bob	2012-06-01 18:41:39

Alternative

Name: Computed on demand (passive component)

Figure 8.2. The SAW prototype presenting the Issue Detail View with full information about design alternatives in the context of a particular design issue. Decisions on the alternatives are presented together with user-specified rationale and are color-highlighted based on the consensus state.



demo.saw.sonyx.net/#/project/521f3425071aa54b4300000b/dialog/main.capture.issueDetails/issueId/50c5ff51924ff84e6b000001

Design Issue details:

Name	UI view state management method
Background	User interaction with system can change its state in two ways: by editing the data, by changing view state of the data. Changing state of the views requires some kind of mechanism to maintain transitions.
Drivers	complexity of the view state should be a primary driver
Status	emergence of new HTML5 features might create new alternatives
Recommendation	simplicity of URL based routing is appealing, but might fall short of expectations for complex, composite UIs

Status: Some decisions are missing

Alternative (2 Decisions, 2 in other projects)

Name	URL (hash routing)

Decision	Rationale	Author	Timestamp
Positive		mohsen.anvaari@gmail.com	10:19 2013-10-03
~~Negative~~	~~not compatible with old browsers~~	~~marcin@sonyx.net~~	~~13:38 2014-07-30~~

Alternative (0 Decisions, 2 in other projects)

Name	event-based
KnownUses	complex, composite UIs
Background	message (event) bus based communication is a popular paradigm for the rich UIs
Pros	scalability
Cons	view state complexity can grow very high, thus it might be very difficult to serialize it into URL

Status
Sealed

Figure 8.3. The contemporary version of SAW presenting the Issue Detail View with a sealed design alternative related to two design positions. One of the positions was revoked by the user (overstrike)

8.2.1 Participants

The number of participants in our experimental workshops was 18. We decided to create two independent groups in order to increase the participants' involvement in the design workshops. The exercises were scheduled so that two groups would perform the (same) exercise in parallel in two physically separated rooms. Each design group included a head architect who held responsibility for the design and for consulting the rest of group. In the situation when system requirements require clarifications, one of the assistants would step in as an oracle (or customer). The role of the head architect would rotate among the design team every week so that everyone gained the experience of leading the design exercise. Prior to beginning the exercises, the participants went through the theoretical part of the Software Architecture lecture and acquired necessary training in architecture and decision modeling. Their skills were evaluated during a written mid-term examination. The results of the mid-term were used to select head architects for the parallel groups in a way such that their knowledge and level of understanding would be comparable, thus enabling a side-by-side comparison of the two groups' performance.

8.2.2 Baseline and Observation

The goal of the experiments is to evaluate the utility of structured architecture decision management for the process of collaborative software architecture design. In order to provide a valid baseline for our experimentation, we decided to use a generic collaborative text editor - the EtherPad[1], hereafter referred to as EP (for details see Section 3.5). We decided to use it because it provides a low-latency collaboration experience, it has proven useful in small teams. While it does not set any constraints on the free form of the captured textual content, it clearly marks the authorship of every text editing operation.

8.2.3 Data Collection

Given that two experimental groups were given different software tools to work with, different methods of data acquisition were required. In the case of the EtherPad group, the group dynamics were recorded thanks to the time-line feature, which allows users to roll back and forth every change of the content. The authorship of particular text snippets can be easily recognized thanks to the color-coding of the users' comments.

The performance of the group using the Software Architecture Warehouse was recorded in three ways. First, most importantly, the SAW produced an architecture decision space persisted as a graph inside the document-based database. A significant limitation of this record is that it does contain only the final state of the decision space. Of course, all the decision space items have their creation and modification time-stamps recorded, but there is very little that can be said about the dynamics of the change. For that reason, in the final analysis, the decision space graph was used mostly as the source for the decision meta-model. In order to alleviate this limitation, the server-backend access log was used as a second data source. Most importantly, it contains a chronological log of all requests that can be used to reproduce and analyze the complete dynamics of the decision space throughout the exercise. For the details, we refer the reader to Section 7.2.

8.3 Results

In this section, we present an application of the metrics defined in Section 7.1.2 to the decision dataset we have collected. Each metric is presented with a brief interpretation of the outcome.

[1]An open-source online editor providing collaborative editing in real-time `http://etherpad.org/`

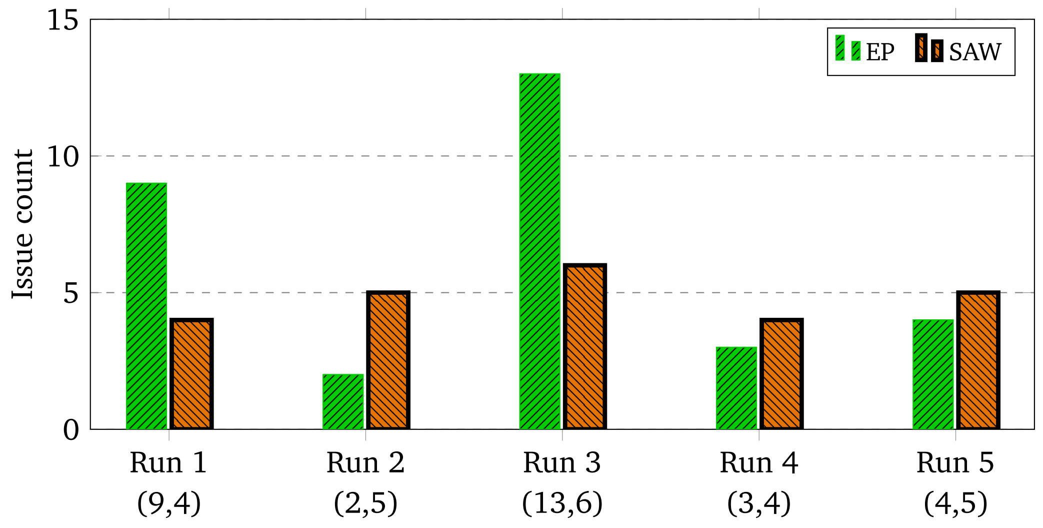


Figure 8.4. Number of issues counted in the consecutive design workshop runs. In parentheses, we provide the issue count for the EP and the SAW respectively (M1)

Metric 1 – Issue count
Domain: *Project*, Scale: *Ratio*, Range: *[0,N]*

A decision space typically consists of numerous design issues. Some of the issues can be reused in multiple projects. This metric represents the number of design issues within a particular project.

In Figure 8.4 we present the number of issues collected in five runs of the experiment. The average number of issues per project is slightly higher for the EP (6.2) than for the SAW (4.8), but it also has a higher standard deviation (4.17 and 0.75 respectively).

The population of the design issues collected within the design workshops appears to be about right or even slightly high, considering the imposed time limit of 90 minutes.

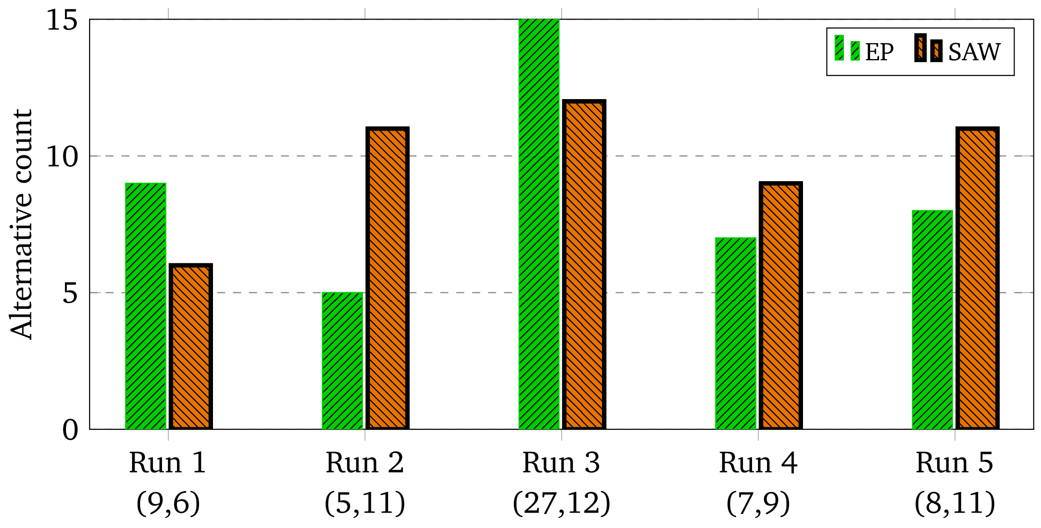


Figure 8.5. Number of alternatives counted in the consecutive design workshop runs. In parentheses, we provide the issue count for the EP and the SAW respectively (M2)

Metric 2 – Alternative count
Domain: *Issue*, Scale: *Ratio*, Range: *[0,N]*

A design issue can be addressed by choosing from among multiple design alternatives. This metric counts the number of such alternatives within the context of a particular design issue. Its value (if greater than 1) can be used to estimate the effort required to make a complete choice about the issue. Other interesting observations can be achieved by aggregating the results of this metric over a set of design issues, for example, those contained within a project.

Analogously to the previous metric, in Figure 8.5 we present the distribution of the number of design alternatives collected within the experiment runs. The average number of alternatives is very similar for the EP and the SAW (11.2 and 9.8). Similarly to Metric 1, the EP is characterized by much higher standard deviation (8.9 for the SAW and 2.3 for the EP).

Figure 8.6 presents the distribution of the number of design alternatives per design issue within the population of design issues. It is noticeable that for the EP most design issues have only two alternatives, whereas for the SAW the same number of issues have two and three alternatives. The average number of alternatives per issue for the EP and the SAW is very similar (1.66 vs. 1.82) as well as their standard deviation (0.81 and 1.28).

The distribution of the number of alternatives per design issue with their peaks between two and three is within reasonable expectations.

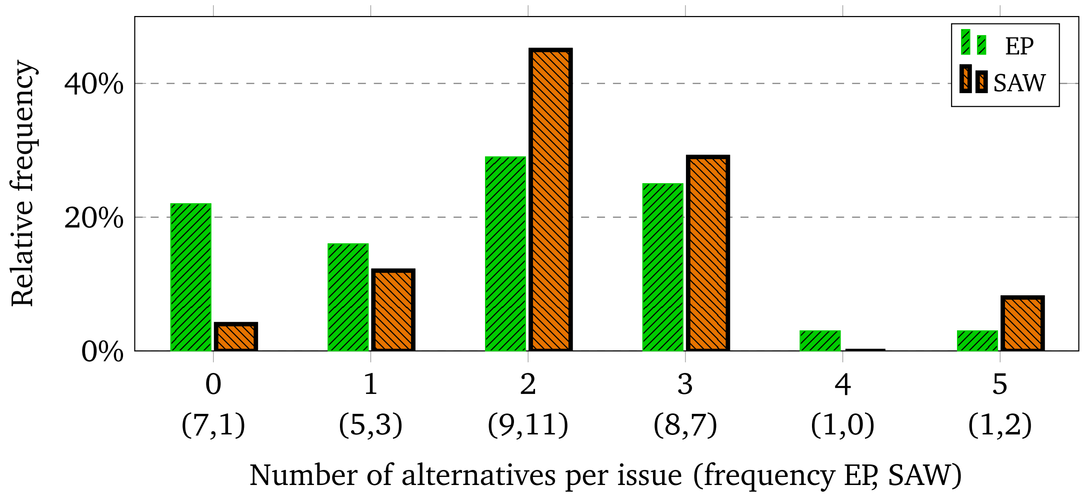


Figure 8.6. Histogram of the number of alternatives per design issue. Bars are scaled proportionally to the total number of issues (EP:31, SAW:24). The absolute population size of the categories is provided in parentheses (M2)

Metric 3 – Relative number of contributors

Domain: *Issue, Alternative*, Scale: *Ratio*, Range: *%*

Assuming that each decision item is labeled by the user who has created it, it is interesting to observe how many users have contributed new items within the context of the project, issue, or alternative respectively. In order to mitigate the effect of a diverse design team size, we propose to analyze the number of contributors relative to the total number of design team members.

Metric 3 is defined as the number of contributors (see Section 7.2.4) relative to the size of the design team. Here, for the sake of clarity, we are going to be operating on the absolute numbers[2]. The histogram presented in Figure 8.7 shows that the majority of design issues and alternatives involved only a single user. Categories with more than seven contributors are empty and not included in the chart.

In Figure 8.8 it can be seen that the consensus state is related to the number of contributors in a way such that alternatives with more than two contributors always have positions, and even more interestingly, that the proportion between colliding and aligned alternatives grows with the number of contributors. Figure 8.9 shows a similar connection between design issue choice state and the number of contributors. In particular, in our dataset, design issues with more than three contributors always have positions stated. The relation to the complete/incomplete choice states is not clearly visible.

[2]In the experiments the team size was 9 (both for – the SAW and the EP).

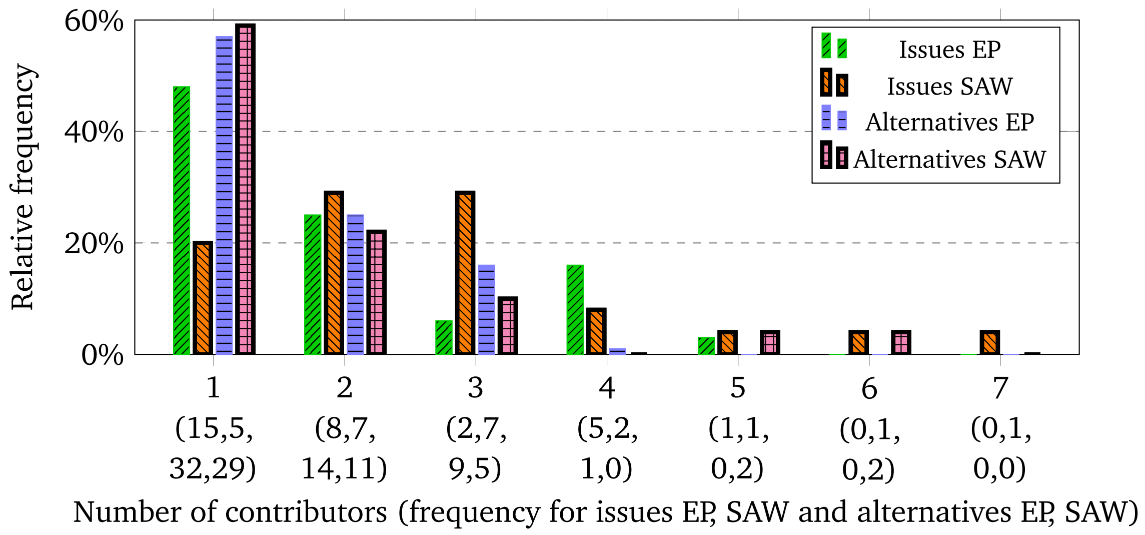


Figure 8.7. Histogram of the number of contributors working on design issues (EP, SAW) and alternatives (EP, SAW) respectively. Numbers in parentheses reflect respective category counts (M3)

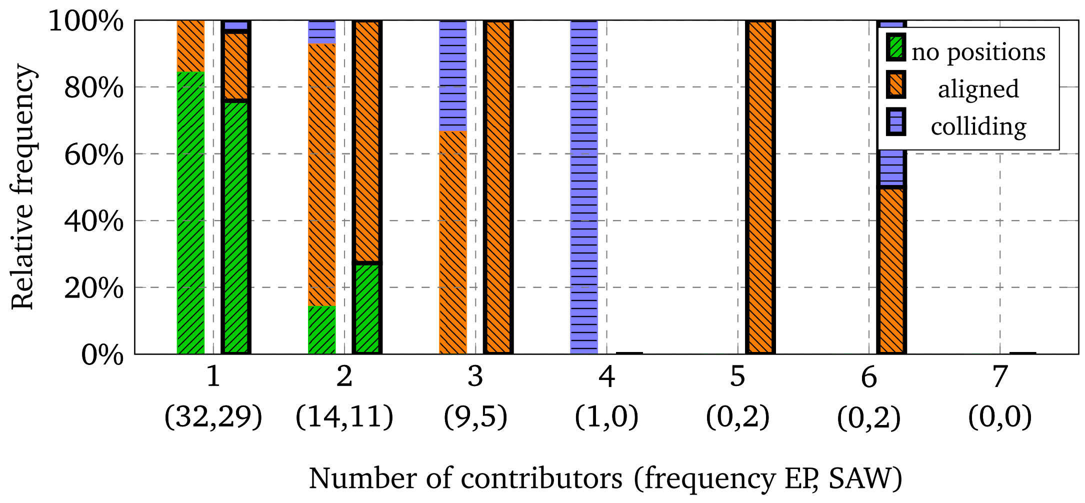


Figure 8.8. Proportion of the consensus states in relation to the number of contributors for the EP (left bar) and the SAW (right bar). Numbers in parentheses represent respective alternative counts in the category (M3)

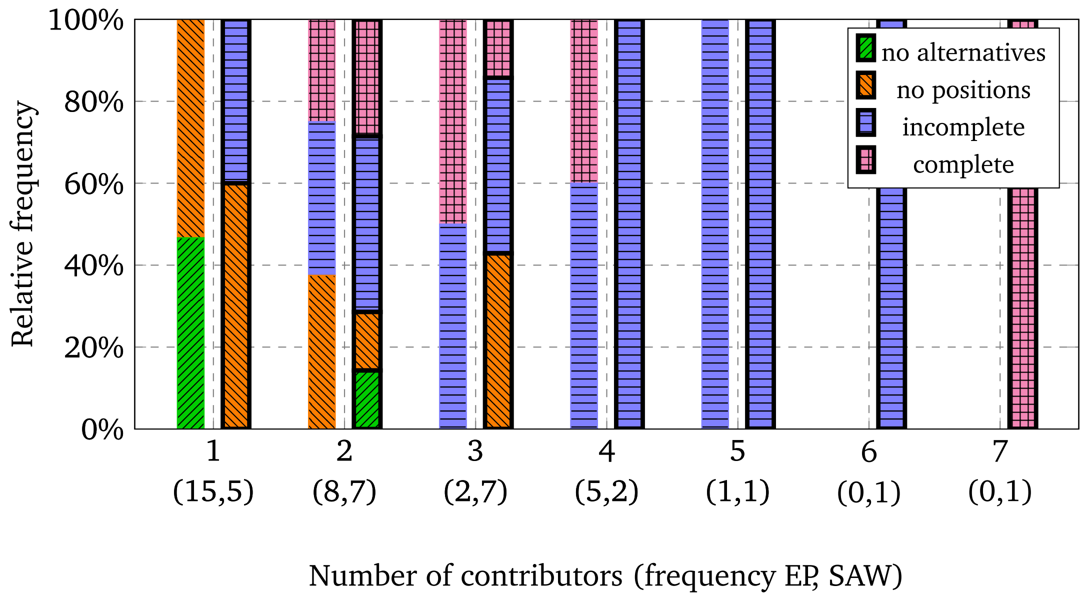


Figure 8.9. Proportion of the choice states in relation to the number of contributors for EP (left bar) and SAW (right bar). Numbers in parentheses represent respective issue counts in the category (M3)

Metric 4 – Relative number of decision makers

Domain: *Issue, Decision*, Scale: *Ratio*, Range: *%*

Like Metric 3, this metric counts the number of design team members involved in expressing positions about a particular decision. Again, as in Metric 3, this metric calculates the number of decision makers relative to the total number of design team members.

The histogram of the number of decision makers presented in Figure 8.11 shows that the majority of decision items (both issues and decisions) had no positions recorded, and thus there were no deciders. It seems that the number of occurrences in the categories declines with the growing number of deciders. The decline pattern is similar for decisions recorded with EP and SAW (lower chart), with a slightly larger number of decisions for more than three deciders. The distribution seems to be different for the design issues recorded with the EP and the SAW (upper chart). For the EP, more than half of the design issues had no positions recorded, whereas, for the SAW, the first three bins collected roughly a third of the population each. The *long-tail* pattern for the SAW is visible for categories representing more than four deciders.

In Figure 8.10 we see that there is no clear relation between the number of decision makers and the consensus state of alternatives. Surprisingly, we have

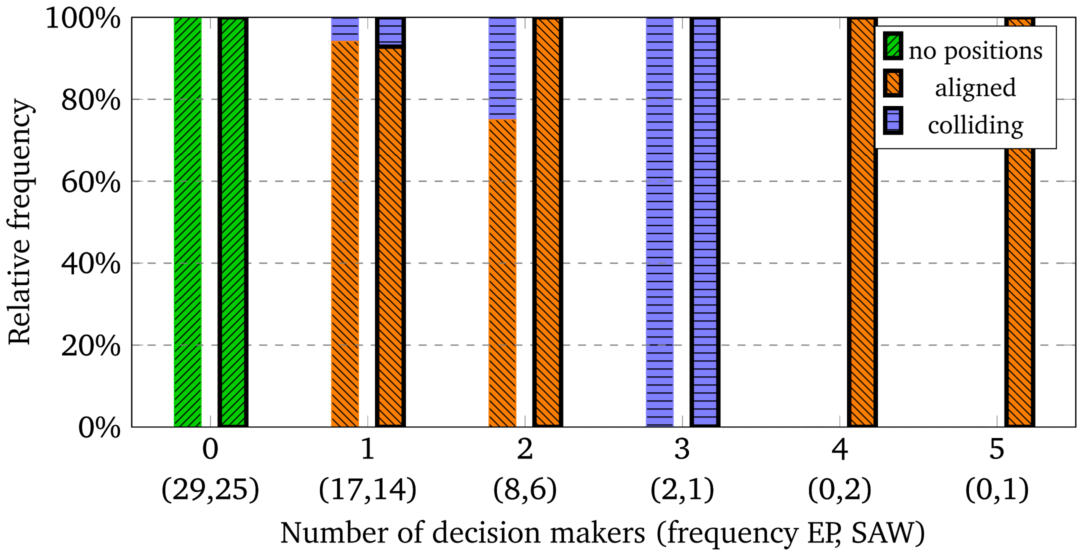


Figure 8.10. Proportion of the choice states with respect to the number of the decision makers in EP (left bar) and SAW (right bar). Numbers in parentheses represent issue counts in the category (M4)

found single cases of decisions with only one decision maker that have stated contradicting positions. Another surprising observation is that all alternatives with over three deciders are aligned. The intuition suggests that with a large number of deciders, chances for collision are higher, as in the case of the category with three deciders.

The number of contributors and decision makers (Metrics 3 and 4, Figures 8.7 and 8.11 respectively) who are involved in the work on a particular decision item, however, is worryingly low. At the same time, the proportion of the decisions with *colliding* positions grows with the number of contributors, which in turn indicates that the argumentation model alignment is far from perfect.

Figure 8.12 shows that both for the EP and the SAW, the majority issues without any decisions has some proposed alternatives. For categories with one or more decision makers, the majority of the issues is in the incomplete choice state. Surprisingly, the one design issue characterized by the highest number of deciders has a completed choice.

Unfortunately, the population size in the categories representing a high participation of deciders is rather small; therefore, both consensus and choice state analysis in this categories is likely to be very noisy.

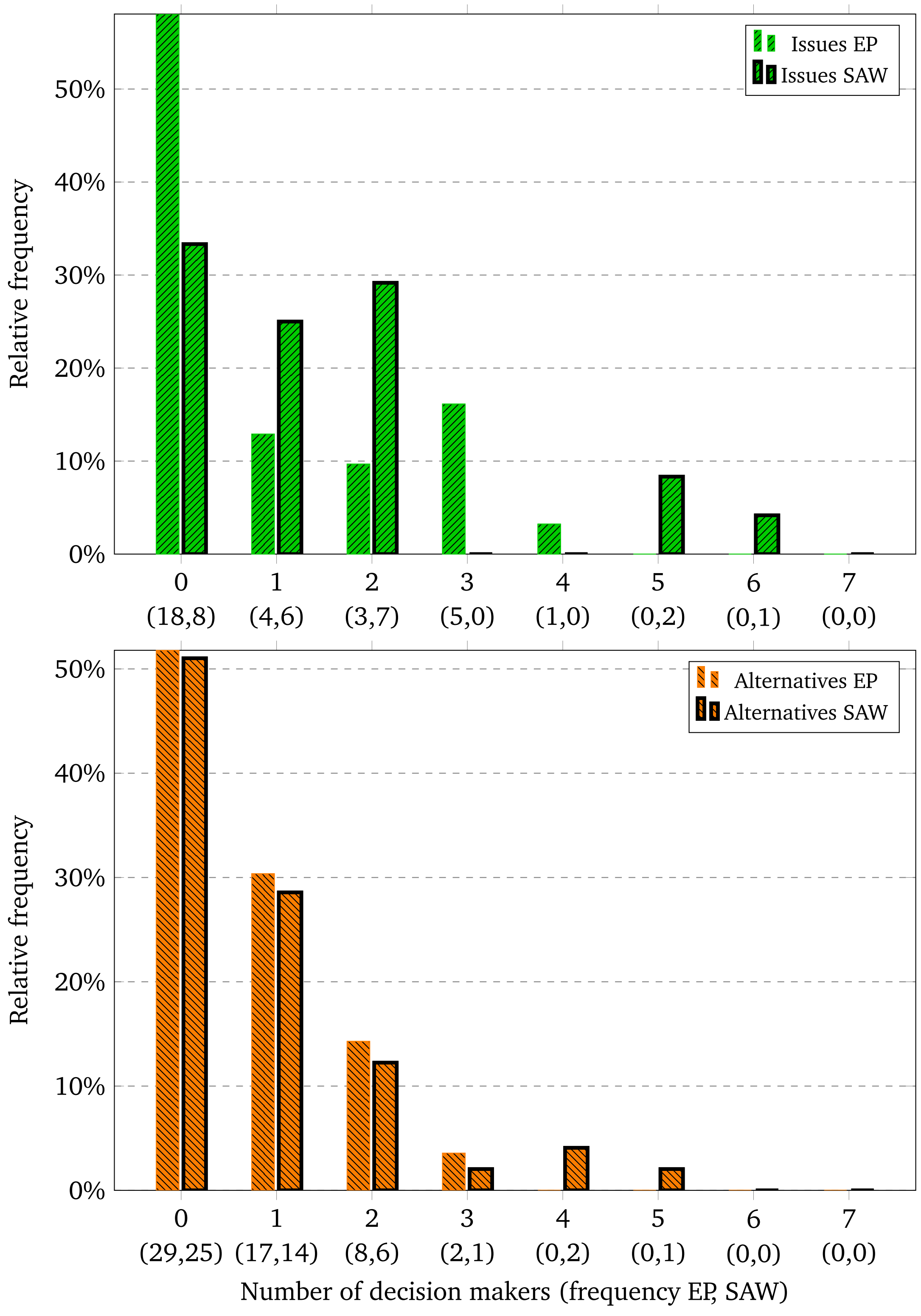


Figure 8.11. Histogram of the decision makers count for the design issues (upper chart) and alternatives (lower chart) for the EP and the SAW (M4)

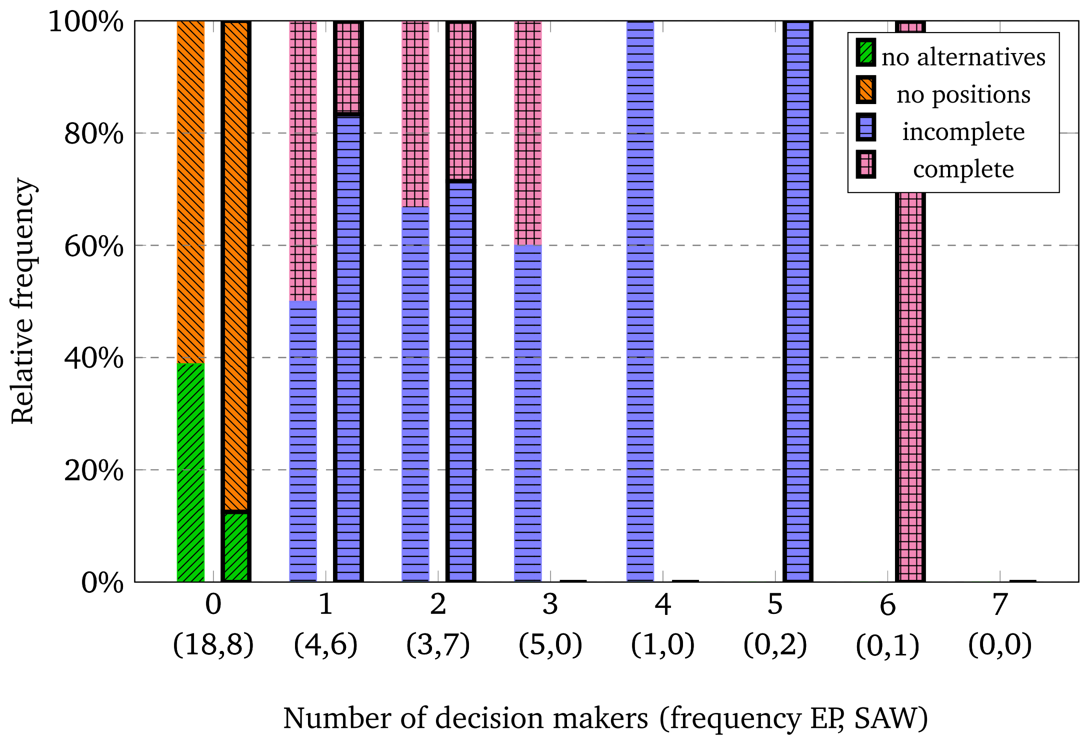


Figure 8.12. Proportion of the choice states to the number of the decision makers for the EP (left bar) and the SAW (right bar) respectively. Numbers in parentheses represent the issue counts in the category (M4)

Metric 5 – Activity timespan
Domain: *Issue, Alternative*, Scale: *Ratio*, Range: *[0,N]*

Decision items are created, updated and, in some cases, eventually deleted from design space. Their lifecycle involves a series of events that correspond to state modifications by some of the architects. As the *activity timespan* of an item we define the duration between the first event (typically the creation of the item) and the last recorded event corresponding to the item.

The histogram of the decision item activity timespan presented in Figure 8.13 shows that there is a slight disparity between the level of activity over time of design issues and design alternatives between the EP and the SAW users. It appears that users of the SAW tend to work on the items for slightly longer than users of EP, especially within the timespan range between 20 and 70 minutes. Conversely, the vast majority of both issues and alternatives for EP has an activity timespan shorter than 10 minutes.

In Figure 8.14 we have charted a number of design issues' characteristics in relation to their activity timespan. For the first three characteristics (**alternatives count**, **editors count**, and **deciders count**), apart from the trend exposed by the histogram (Figure 8.13), there is no striking pattern emerging from the chart. In the chart exposing **update count** there is a disparity between the data collected for the EP and the SAW. We believe that this effect comes from the difference in the data acquisition method (see Section 7.2.1), and thus is not relevant for comparison of the workshop dynamics differences for the SAW and the EP.

Analogously, in Figure 8.15 we charted the characteristics of all design alternatives. This figure also confirms the trend observed earlier in the histogram. Additionally, a population of alternatives with a high (four or more) **position count** can be observed for the SAW. For the **editors count** and **deciders count** we see no clearly visible trend. As before, **updates count** shows a significant disparity, but to a lesser degree than in the case of design issues.

In order to expose trends existing in the data scattered in Figures 8.14 and 8.15 we have decided to introduce a cut-off threshold that eliminates issues and alternatives characterized by a short **activity timespan**. The rationale for this is that a long **activity timespan** can be a good indicator of the importance of a particular design issue or alternative.

In order to inspect the characteristics of design decisions and design issues, we have decided to classify them based on their activity time. In Figure 8.13 it can be seen that a major share of design issues and alternatives (decisions) were active for a rather short time (fewer than 10 minutes). For our further analysis, we have applied heuristics assuming that decision items characterized by a longer activity time are of higher value for the design. In fact, Figures 8.15 and 8.15 show that

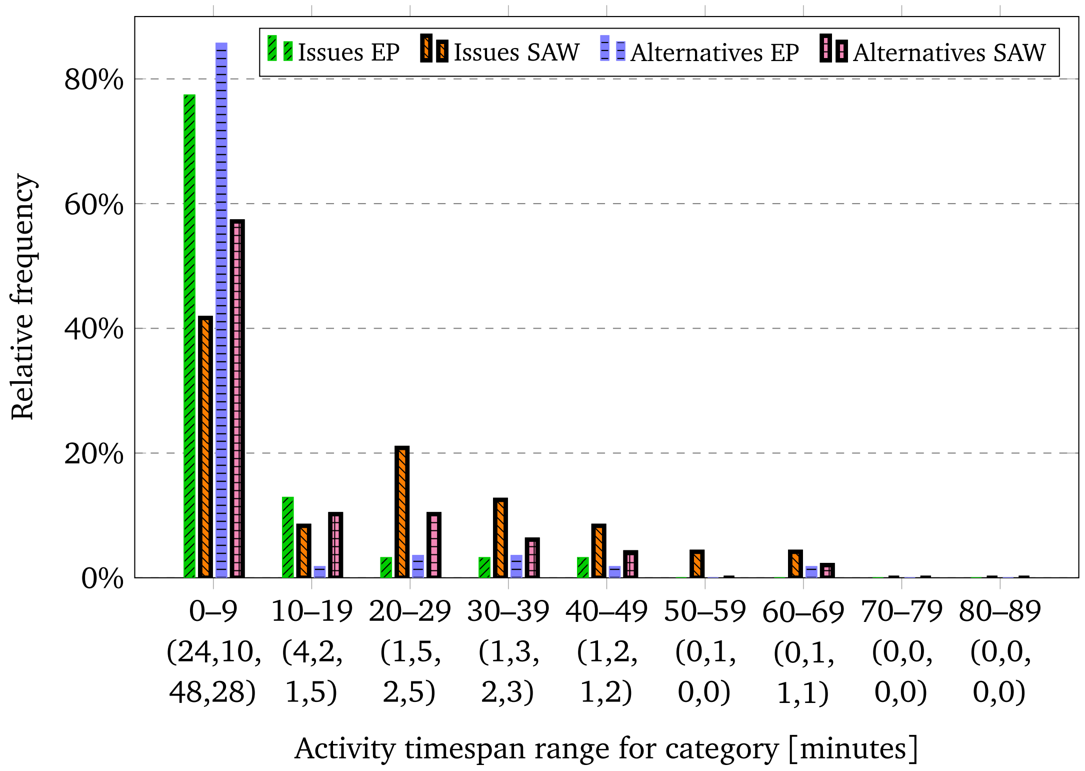


Figure 8.13. Histogram of the activity timespan for design issues and alternatives. Numbers in parentheses represent frequencies for Issues (EP, SAW) and Alternatives (EP, SAW) given the category (M5)

the characteristics of decision items, with longer activity times are significantly different from those active only for a short time. As we could not decide upon the fixed cut-off threshold between short-lived and long-lived items, we have studied both populations in relation to a moving cut-off time (see Figure 8.16). It can be observed that the overall count (cut-off time value 0) of both the issues and the alternatives is favorable for the EP. Conversely, for higher values of the cut-off time (greater than 10 minutes), the counts of the design issues and the alternatives are significantly higher for the SAW.

In Figure 8.17 it can be observed that for the SAW the average number of design alternatives per issue increases with growing activity time (high cut-off values). We have not observed the similar trend for design issues recorded with the EP.

In Figure 8.16 we have plotted a relation between the cut-off time and respective issue and alternative population sizes. As hinted at earlier in the histogram (Figure 8.13), it is visible that the overall population size for both issues and

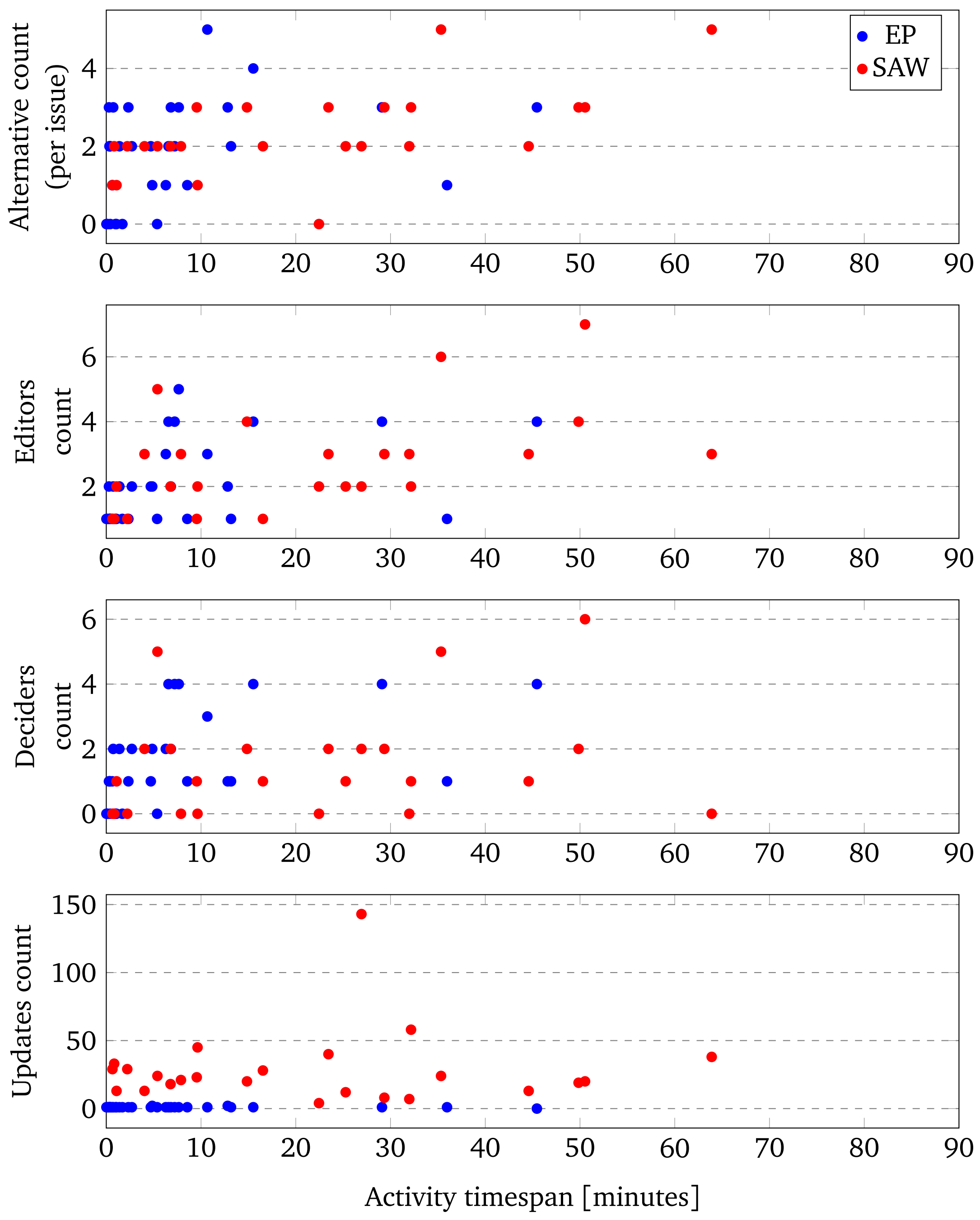


Figure 8.14. Properties of the design issues plotted the against activity timespan for the EP and the SAW (M5)

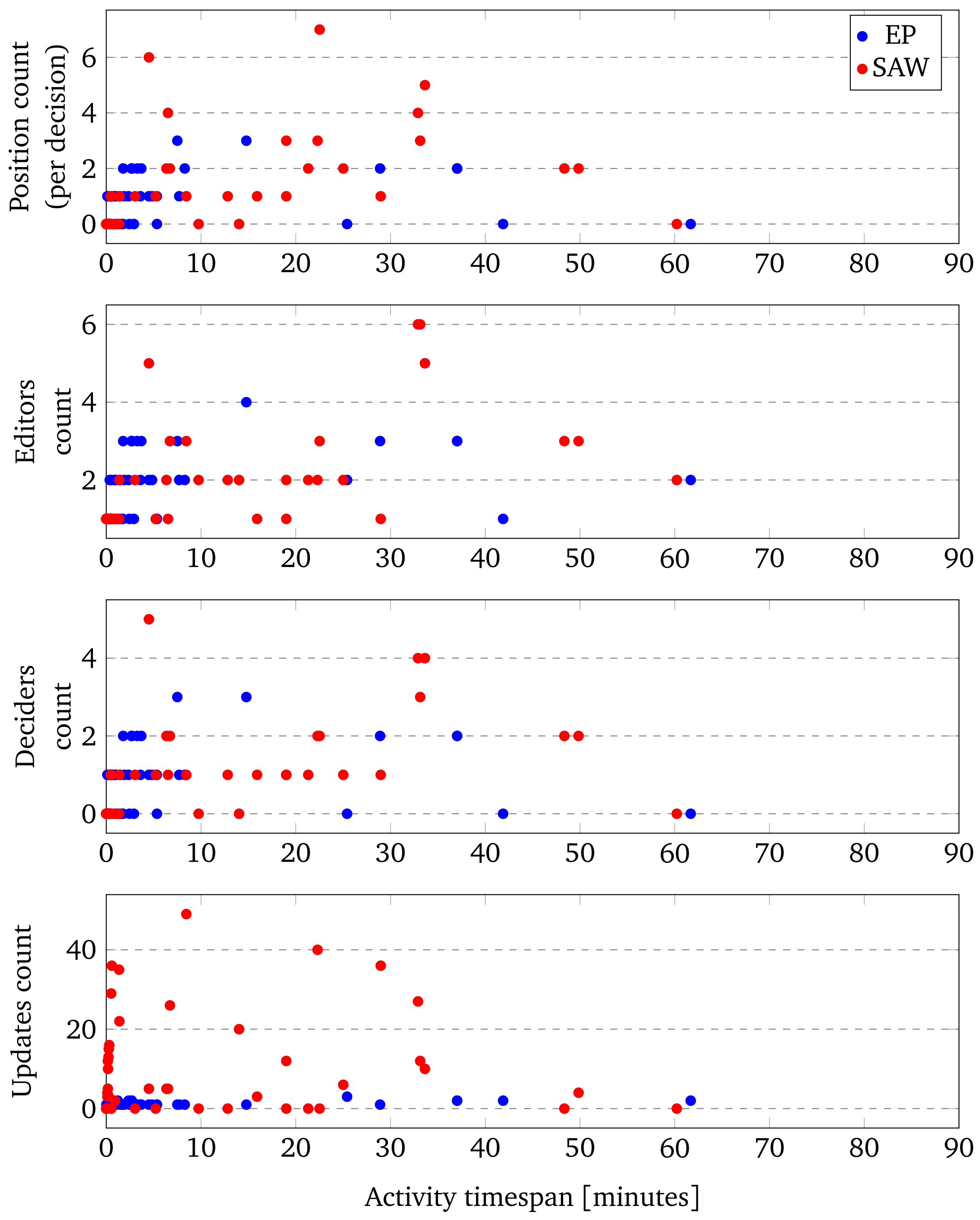


Figure 8.15. Properties of the design alternatives plotted against the activity timespan for the EP and the SAW (M5)

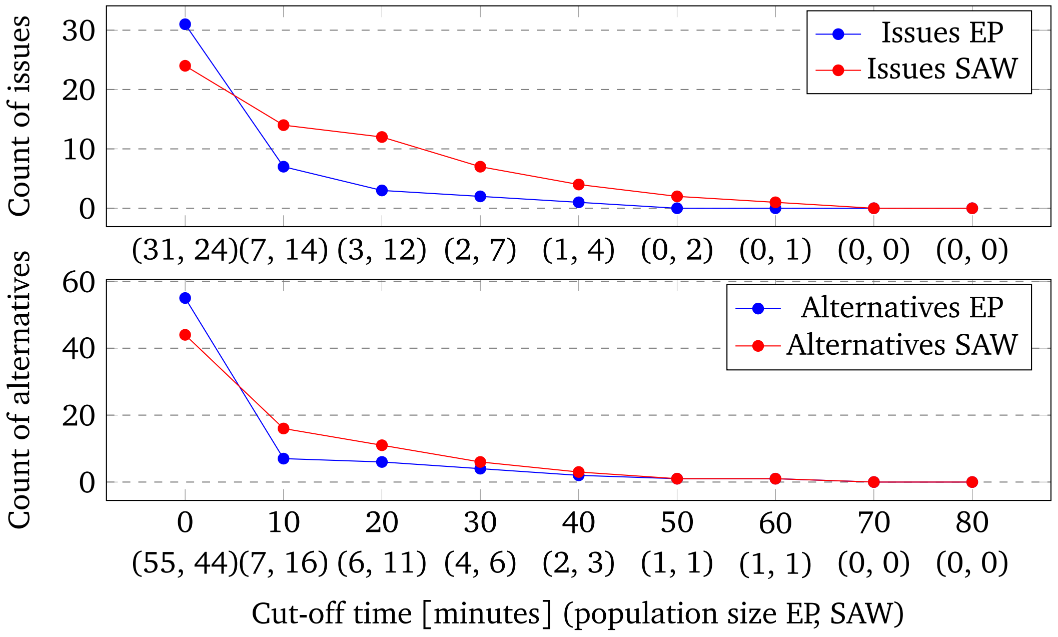


Figure 8.16. Number of issues and alternatives depending on the cut-off threshold time. In parentheses we provide counts for the EP and the SAW respectively. (M5)

alternatives is larger in the case of the EP than the SAW (cut-off value 0). Conversely, for the higher cut-off time values, the population counts for the SAW are consistently higher than for the EP. Using a variable cut-off time we, have investigated a number of issues' and alternatives' characteristics, as presented in Figure 8.17. The data points in this figure represent averages over the population of decision items with the **activity time** over the given cut-off threshold.

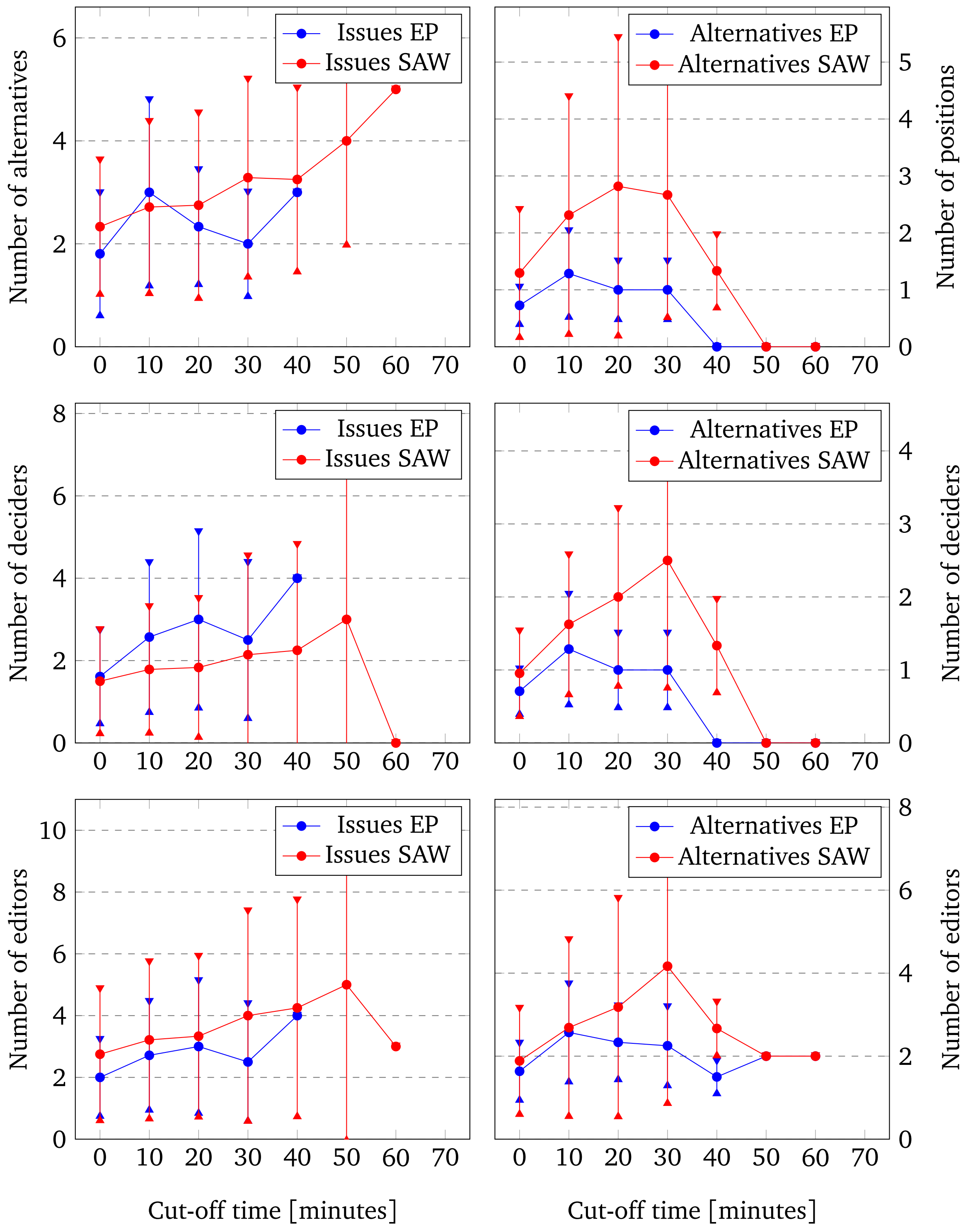


Figure 8.17. Properties of design issues and alternatives plotted against the population cut-off time for the EP and the SAW. Error bars represent calculated $\sigma/2$ (M5)

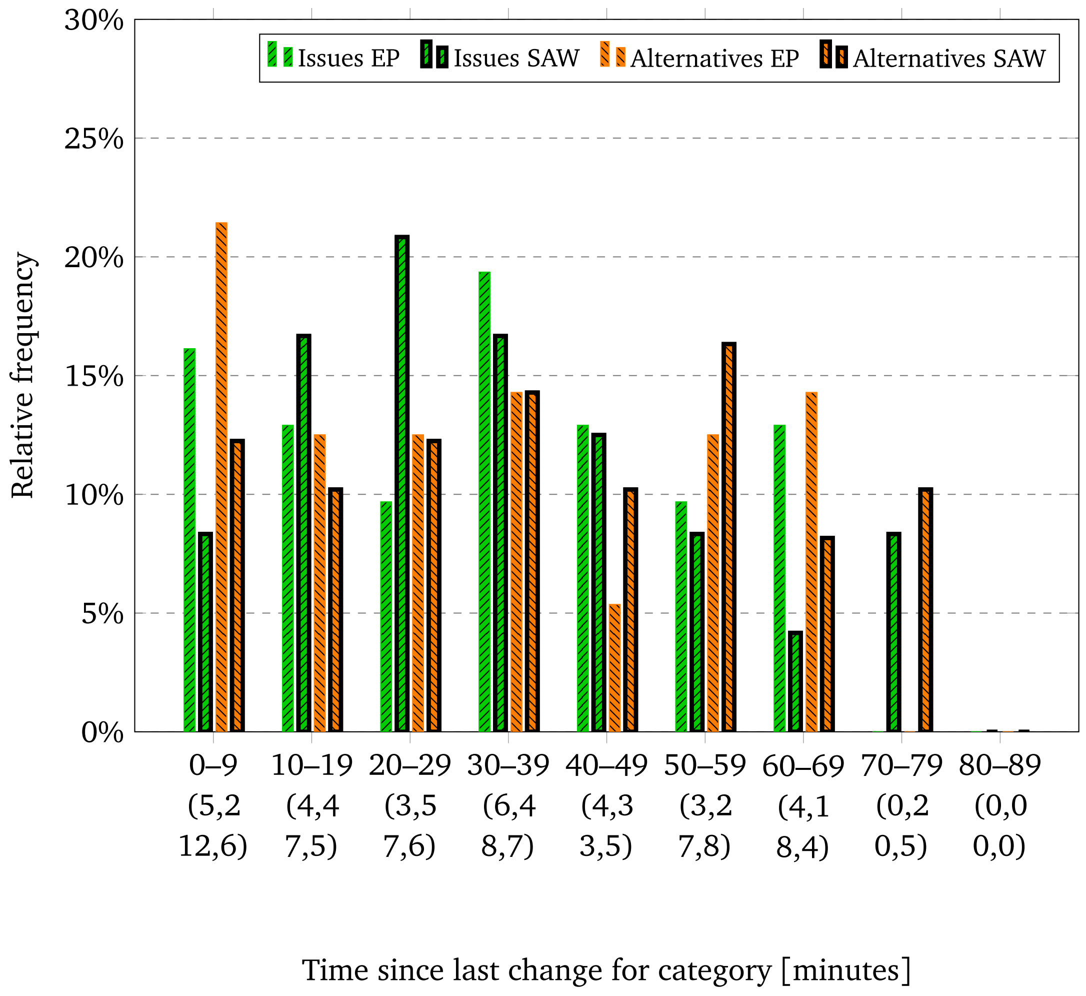


Figure 8.18. Time since last change – histogram (10-minute intervals). Very recently edited items are on the left, while old items are towards the right. (M6)

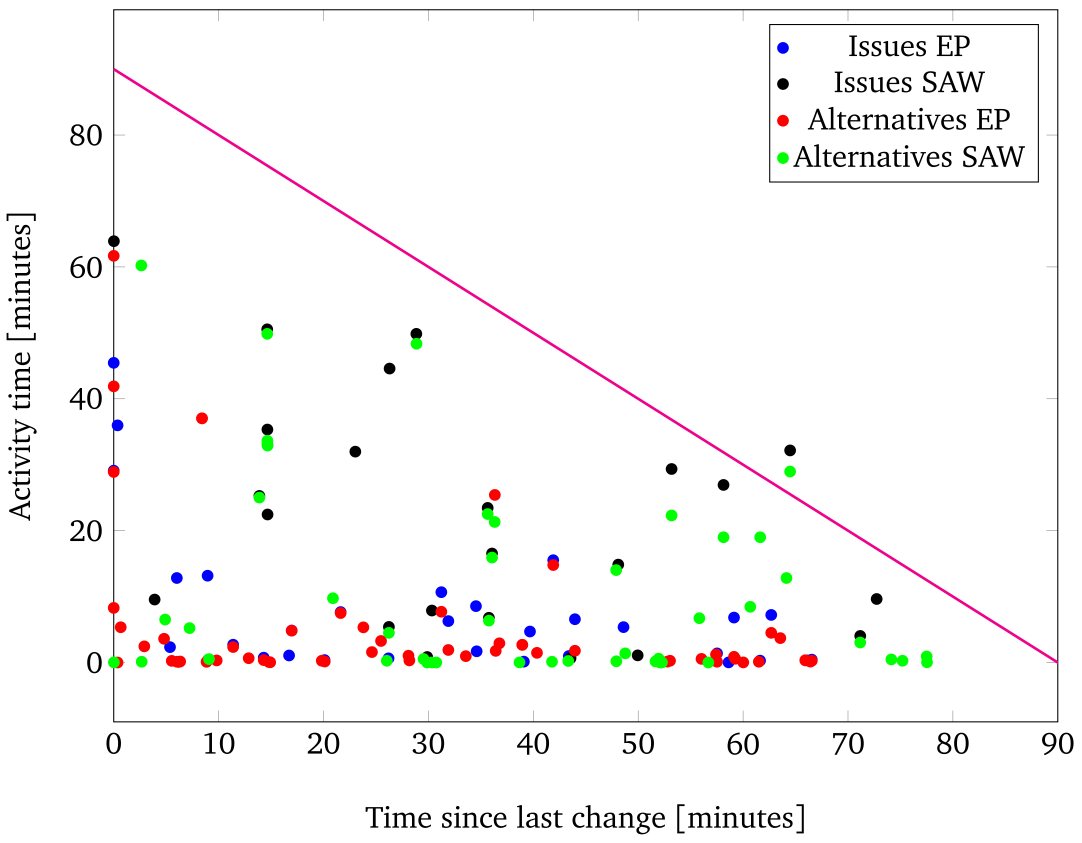


Figure 8.19. Activity time plotted against the time of the last change for issues and alternatives (EP, SAW) (M5, M6). The red line in Figure marks the maximum possible activity time that could occur during the design workshop.

Metric 6 – Time since last change
Domain: *Issue, Alternative, Position*, Scale: *Ratio*, Range: *[0,N]*

This metric calculates the time elapsed since the last recorded event related to the particular decision item. The time reference would be either the current time for live measurements, or the end of the time-frame for a time-boxed experiment.

The histogram of the **time since last change** for the design issues and alternatives (Figure 8.18) reveals that within the first 20 minutes of the workshop (right part of the chart), there is more activity done by the users supported by SAW, than those using EP. Otherwise, the histogram presents a rather even distribution of the **time since last change**, in particular when compared with **activity time** (see Figure 8.13).

In Figure 8.19 we can observe precisely the distribution of the **activity time** for issues and alternatives in relation to the **time since last change**. The straight line cutting thought the chart is a maximum possible activity time for the item created at a particular moment of the design workshop, with an assumption of the 90-minute-long session. An outlier, laying over the maximum activity time, that consists of one SAW issue and one SAW alternative can be spotted. We have decided to include it in the dataset as it made an important contribution to the design. It can be observed that there are a number of items with the **time since last change** close to zero and non-zero **activity time**. This can be interpreted as last-minute changes that were done just before the design workshop finished.

Figures 8.20 and 8.21 present an analysis analogous to the one we presented for the activity time. The even distribution of nodes on the scatter charts confirms the rather even pace of the design workshop. Again the **update count** data reveals a significant disparity between the records for the EP and the SAW, which cannot be interpreted because of the differences in the data acquisition (see Metric 5, Figures 8.14 and 8.15, and Section 7.2.1).

An analysis performed with Metric 6 (time since the last change) shows that the designers' activity was distributed very evenly throughout the time of the design workshop (see Figure 8.18).

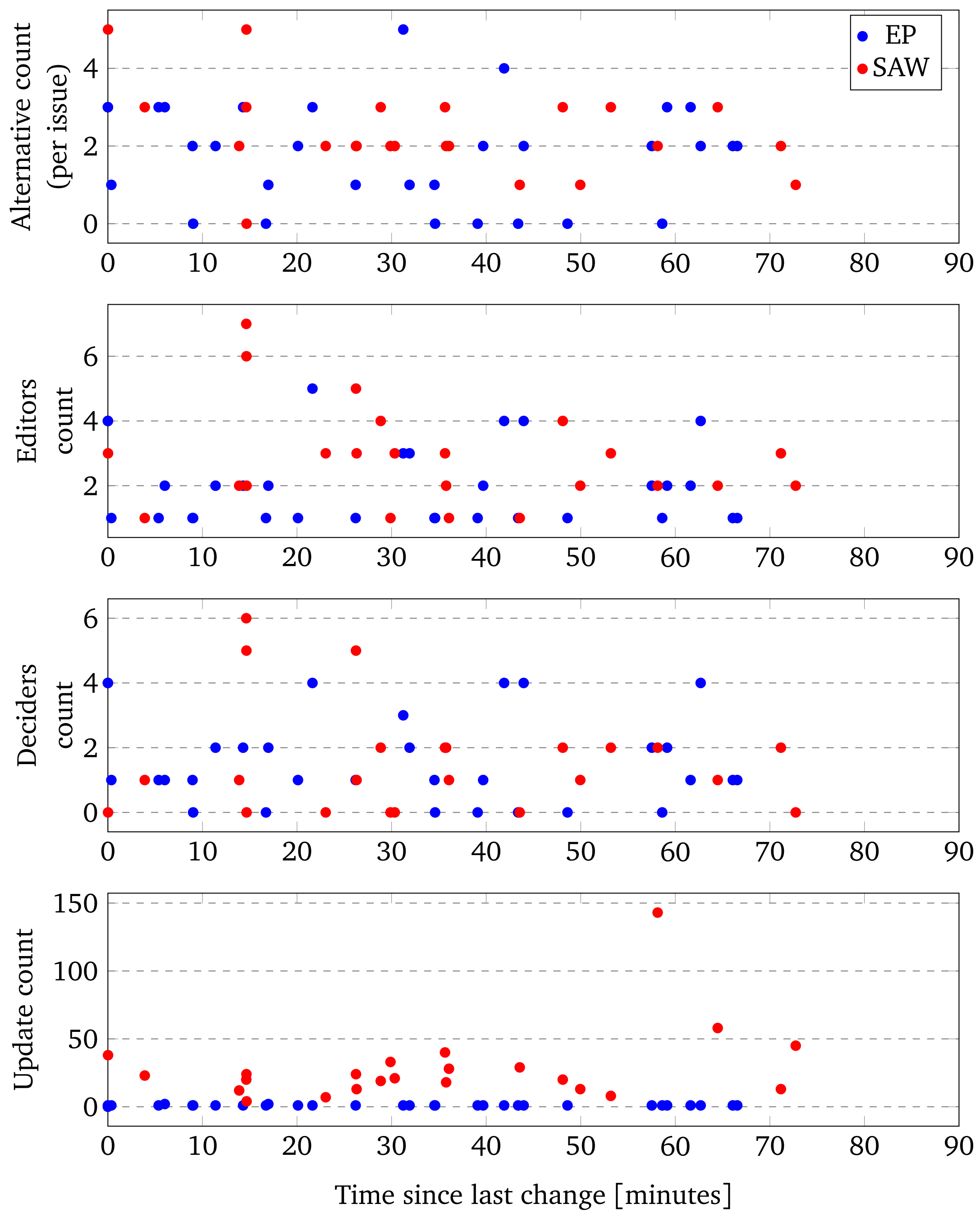


Figure 8.20. Properties of design issues plotted against the time since last change for the EP and the SAW (M6)

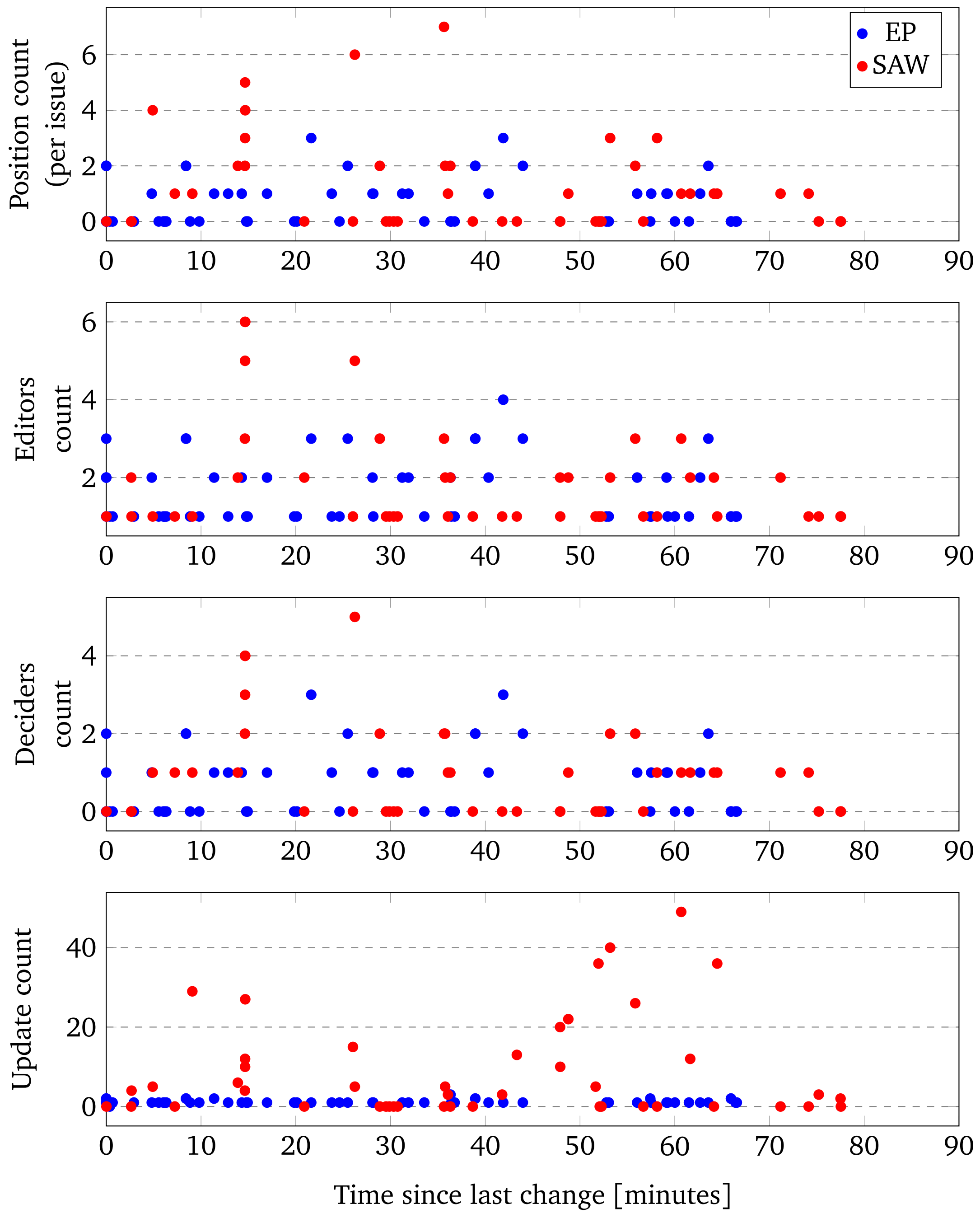


Figure 8.21. Properties of design alternatives plotted against the time since the last change for EP and SAW (M6)

Metric 7 – Position type count
Parameter: *position type*, Domain: *Decision*, Scale: *Ratio*, Range: *[0,N]*

The decision meta-model specifies the types of the positions that users state about the architecture decision. In the particular meta-model that we adopted, we distinguish: positive (accept), negative (reject), and open (neutral) positions. This metric takes a particular decision type as a parameter and returns the number of positions of this type that were contributed to the decision.

In Figure 8.22 we present the number of positions recorded in the consecutive experimental runs. It is noticeable that this number varies greatly, in particular for the EP. Figure 8.23 shows that more than half of the decisions have no positions related to them. In the following categories, it is noticeable that the relative frequency of decisions decreases for categories with a growing number of positions. For the SAW, a long-tail effect, similar to the one observed for Metric 5 (**number of decision makers**), can be observed. In Figure 8.24 we show the distribution of the choice states within the categories. Surprisingly, for categories with four or more positions, are all in the **aligned** choice state.

This result can be contrasted against Metric 7 (**position type count** – see Figure 8.23), showing that more than half (51% for both the EP and the SAW) of the decisions have no positions. This is a strong indicator that the decision model is far from being complete, from a consensus point of view. It can be observed that for categories representing between one and three decisions, the relative frequency of decisions for the EP is higher. For the SAW, this effect is compensated in categories with four or more positions.

In Figure 8.24 it can be seen that decisions with no positions, by definition are in a **no positions** consensus state, whereas those with one position are always **aligned**. A category with two positions opens the chance of position collision, which is reflected in the mixed composition of **aligned** and **colliding** consensus states in this category. Interestingly, all decisions that have three positions are in a **colliding** state. This might suggest that within the limits of the design workshop there was not enough time for decision makers to revisit and reconsider their positions to reach a consensus. Surprisingly, all decisions with more than four positions are in the **aligned** state. This can be the result of timid team members following the lead of the main architect and reconfirming his judgment. Within the categories in which comparable data is available, the difference in the relative frequencies of consensus state between the EP and the SAW is not significant enough to draw conclusions.

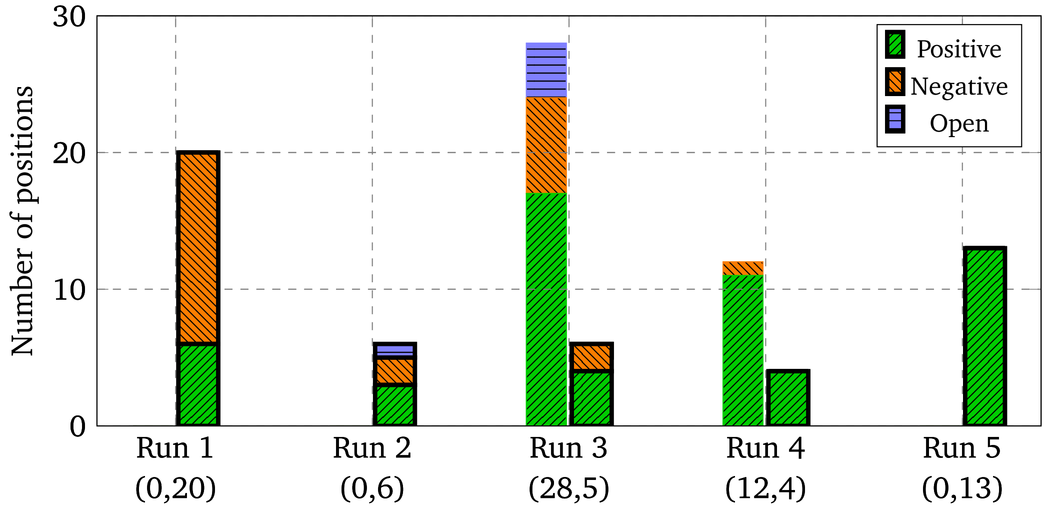


Figure 8.22. Number of positions with distinctive types, in consecutive experimental runs for the EP (left bar) and the SAW (right bar) respectively (M7)

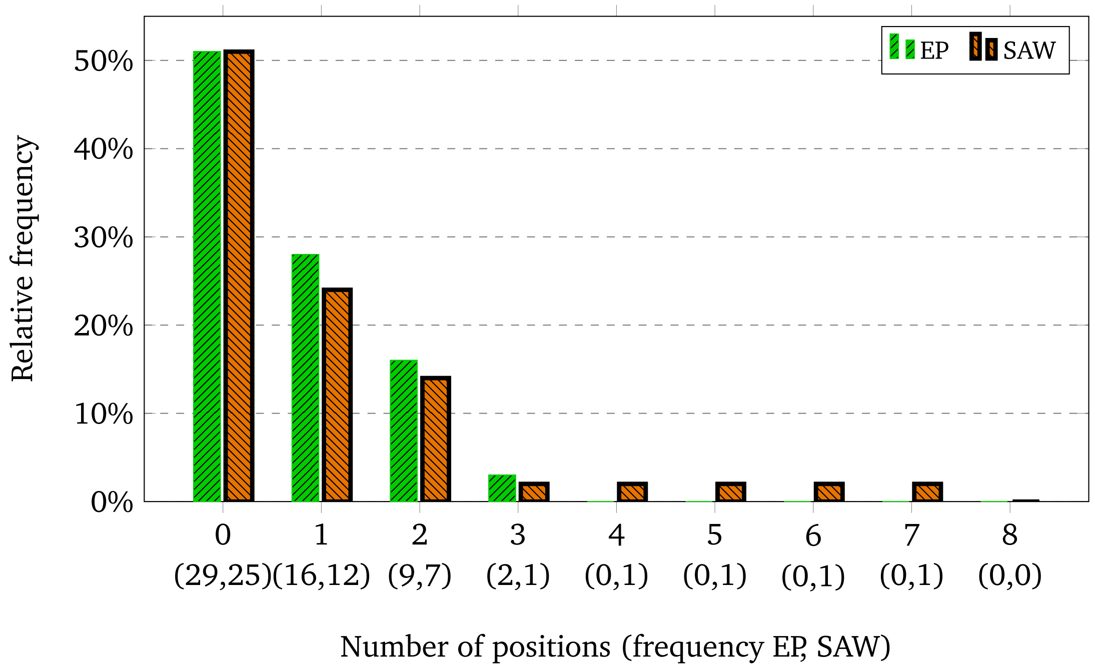


Figure 8.23. Histogram for number of positions per decision (alternative). In parentheses we provide population count in the categories for the EP and the SAW respectively (M7)

Metric 8 – Consensus state

Domain: Decision Scale: Ordinal Range: {no positions, aligned, colliding, sealed}

The positions contributed to the decision can be aligned in a variety of ways. As introduced earlier in Section 5.1.2, we distinguish the following elementary consensus (alignment) states:

no positions – when the number of positions is equal to zero,

aligned – all positions in the context of the decision are of the same type,

colliding – positions of more than one type exist,

sealed – a single position was chosen to settle the argument and seal the decision, preventing the addition of further positions (not represented in the dataset).

Decisions with *aligned* or *sealed* positions are said to be decided for a particular decision type (for example decided *positive*). This metric aggregates over all position types the previous Metric 7 to compute the current consensus state of a particular decision based on the previously defined rules.

In Figure 8.24 we can see how decision consensus depends on the number of positions on the decision. Concerning the first two categories, these are derived directly from the state definitions (0 positions = no positions; 1 position = aligned by definition, see Section 7.1.2). As expected two or more positions allow for conflicts. Still, as already observed for the previous Metric 7 (**position type count**), decisions with a large number of positions (4-7) are all aligned; this might be an artifact caused by the small size of the population in our sample.

The total number of analyzed decisions was 105, with 56 from the EP and 49 from the SAW. The overall share of decisions without positions was 51% (EP: 52%, SAW: 51%), another 7% (EP: 9%, SAW: 4%) had a colliding set of positions, and finally 42% (EP: 39%, SAW: 45%) had aligned positions.

Considering the **consensus state** of the decisions (Metric 8) presented in Figure 8.24, it can be noticed that both for the EP and the SAW the relative number of decisions with aligned positions is higher than the relative number of positions in a colliding state.

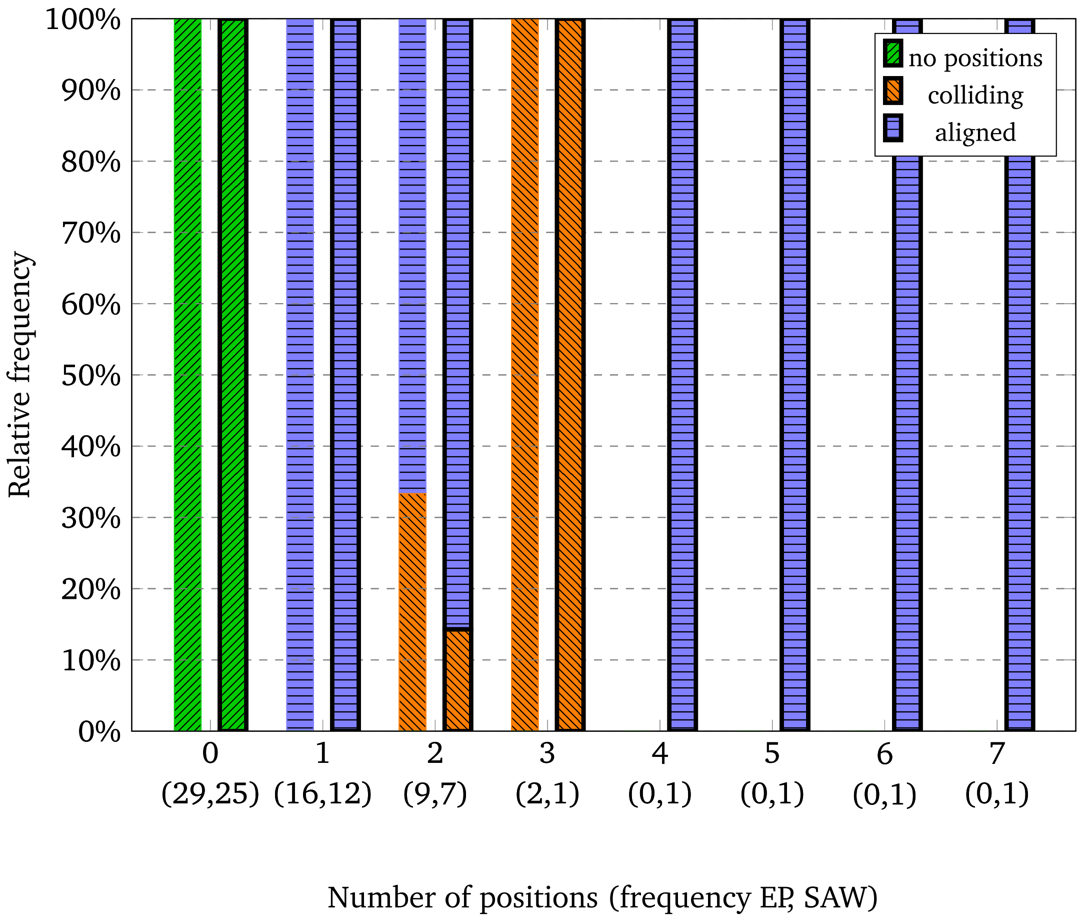


Figure 8.24. Proportion of the consensus states in relation to the number of positions. Two bars for each category represent values for the EP (left bar) and the SAW (right bar) respectively (M8)

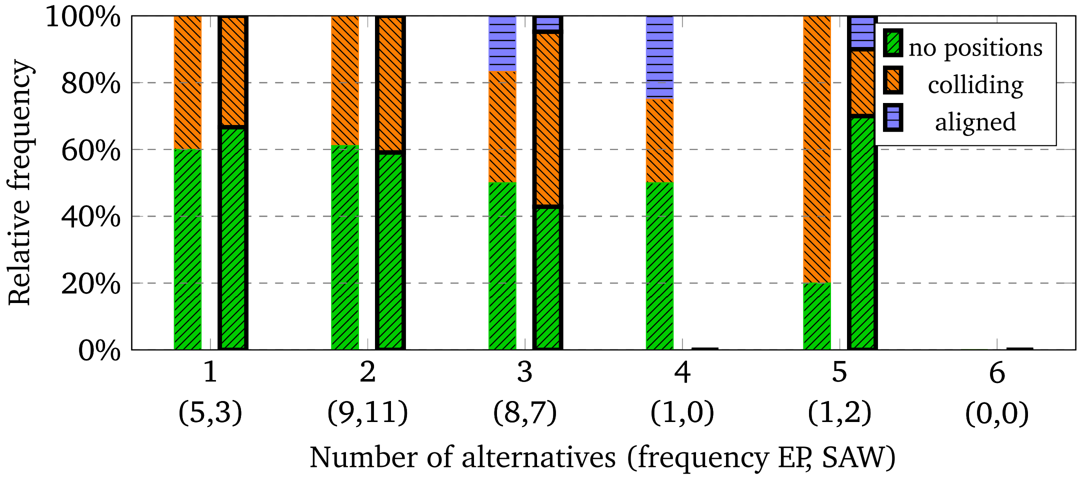


Figure 8.25. Proportion of the consensus states for alternatives in relation to the number of alternatives related to a design issue for EP (left bar) and SAW (right bar). Alternative counts in the respective category are provided in parentheses (M9)

Metric 9 – Decision consensus state count

Parameter: *consensus state*, Domain: *Issue*, Scale: *Ratio*, Range: *[0,N]*

Building upon Metric 8, this metric counts the number of decisions with positions aligned in the given consensus state. The metric is applicable to a design issue, but it may also reveal interesting features when its values are aggregated over a broader, project-level scope.

The number of decisions that have reached a given consensus state can be observed in relationship to the number of alternatives that need to be chosen. In Figure 8.25 we can see that independently of the number of alternatives, a significant number of alternatives (and thus decisions) remain without positions. Interestingly, it can be observed that decisions with aligned positions occur only for design issues with three or more alternatives.

Analyzing the results of Metric 9 (Figure 8.25), we noticed that, surprisingly, the share of decisions with **no positions** decreases with the number of alternatives related to the design issue. Moreover, the decisions with an aligned set of positions occur for the design issues with three and more alternatives. This might suggest that some design issues draw less designers' attention and, therefore, are related to fewer alternatives and decisions over these alternatives are often not aligned. Other issues, drawing more designers' attention, are related to more alternatives and their decisions are more aligned.

Metric 10 – Choice state
Domain: *Issue*, Scale: *Ratio*, Range: *{no alternatives, no positions, incomplete, conclusive, inconclusive, warring}*

A design issue under consideration, depending on the design alternatives addressing it and their consensus state (see Metric 8), can take one of the following choice states:

no alternatives – there are no alternatives addressing the given issue,

no positions – there are alternatives addressing the issue, but all corresponding decisions are without positions,

incomplete – there are positions addressing the issue's decisions, but there is at least one decision for which the positions are not *aligned*,

conclusive – all decisions are in the *aligned* consensus state, precisely one is decided *positive*, and all others are decided *negative*,

inconclusive – there is more than one decision decided as *positive* or *open*,

warring – all decisions are aligned with the *negative* position type.

The last three choice states are all referred to as **complete** choices. Also, in this case, Metric 10 aggregates the values of Metric 9 over all decisions associated with a given issue.

In Figure 8.26 we show how the choice state depends on the number of alternatives related to the design issue. As per definition (see Section 5.1.2), issues in the first category (0) are in the **no alternatives** choice state. For issues with one alternative (category 1) are the majority of issues have **no positions**, whereas the remainder (by definition) is in the complete choice state. For design issues with two or more alternatives, it is noticeable that a **complete** choice is less prevalent and replaced by an **incomplete** choice. Interestingly, our analysis has found a case of one design issue with five design alternatives but no positions related to it.

Moreover, the **choice state** (Metric 10) presented in Figure 8.26 shows that the relative number of incomplete choices grows with the number of alternatives. This might indicate that, due to time pressure, decision makers could more easily afford to tackle the issues with fewer alternatives, which are typically easier to address, than the issues with a larger number of alternatives, that require more consensus building effort.

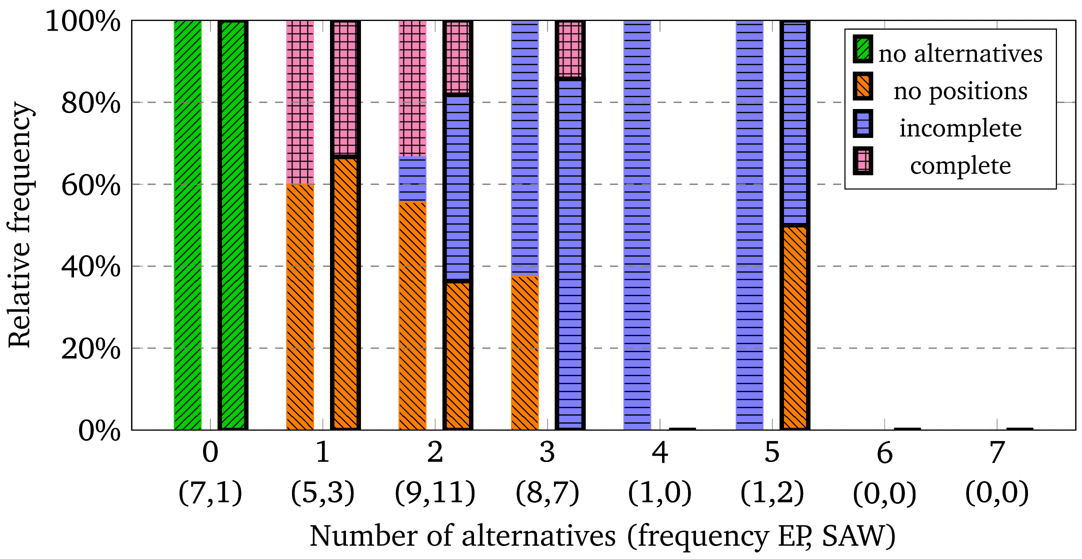


Figure 8.26. Proportion of the choice states in relation to the number of alternatives for the EP (left bar) and the SAW (right bar). Issue counts for the categories are provided in parentheses (M10)

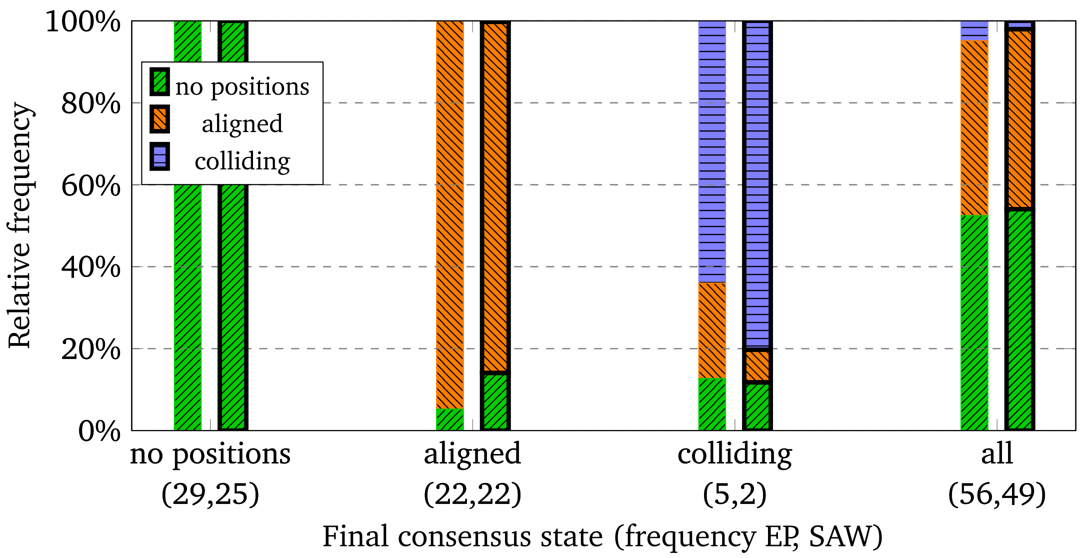


Figure 8.27. Relative amount of time that decisions spent in the consensus state, in relation to their final consensus state for the EP (right bar) and the SAW (left bar). Respective decision counts for each consensus state are provided in parentheses (M11)

Metric 11 – Relative consensus state timespan
Parameter: *consensus state*, Domain: *Decision*, Scale: *Ratio*, *Range: %*

Within its lifecycle, the design decision traverses some of the consensus states specified in the context of Metric 8. For the purpose of investigating the dynamics of the decision-making process, it is interesting in particular to observe the timing of the transitions between these states. This metric accepts a *consensus state* as a parameter and calculates an amount of time a particular decision has spent in this state. The calculated value is returned as a relation to the overall time-span of a particular design decision (Metric 5).

The amount of time that each decision spends in a given consensus state can be observed depending on the final consensus state that it will eventually reach. In Figure 8.27 we can observe that decisions whose final consensus state is **no positions** are not visiting any other states (compare Metric 14). Similarly, with decisions in an **aligned** state, the only other visited state is **no positions**. Finally, the decisions with the **colliding** final state remain for some time in both the **no positions** and **aligned** states. The general observation that can be drawn from the fact that the state in which decisions spent most of time is consistent with their final consensus state decisions change their consensus state mostly towards the beginning of their lifecycle.

Analyzing the values of relative consensus timespan (Metric 11), it is visible that the final consensus state of the decision correlates with the consensus state that decision occupies the majority of the time. This can be straightforward interpreted as good for the decisions in the *aligned* state, but also means that *colliding* decisions remain colliding forever. In fact, analyzing Metric 14, we haven't found any case of the decision that would turn from the *colliding* state back to *aligned*.

Metric 12 – Relative choice state timespan
Parameter: *consensus state*, Domain: *Issue*, Scale: *Ratio*, *Range: %*

The design issue lifecycle consists of choice states defined in the context of Metric 10. This metric, given a particular choice state as a parameter, and analogously to Metric 12, calculates the relative amount of time that the design issue has spent in it.

Also, in this case, we show the amount of time that each issue spends in a given choice state as a function of the final choice state of the issue. As pictured in Figure 8.28, the design issues that have a final state of **no alternatives**, **no positions**, or **complete** show no surprising behavior (see Metric 11). An interesting behavior can be observed in the bar representing the **incomplete** final choice state since it shows that 5% of the time was spent in the **complete** choice state. This can be a result of having reached the **complete** choice state before a new design alternative was proposed, which would flip the choice state back to **incomplete**. Similarly, to the observation made for Metric 11, judging by the correspondence of the state with the largest timespan and the final choice state, it is apparent that issues change their choice state mostly towards the beginning of their lifecycle.

An inspection of the choice state timespan (Metric 12) and state transition count (Metric 15) shows very similar behavior, namely, that in most of the cases issues spent a major part of their timespan in their final choice state.

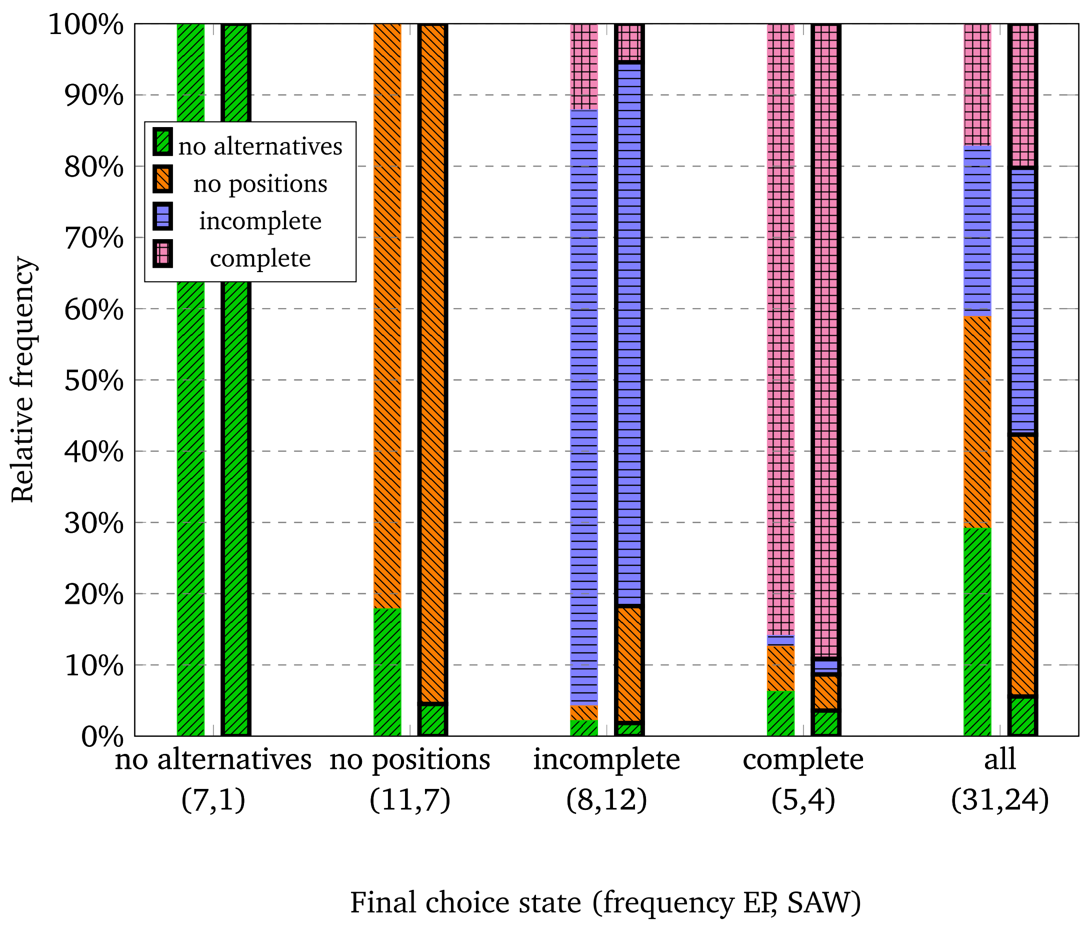


Figure 8.28. Relative amount of time that issues spent in the choice state, in relation to their final choice state for the EP (left bar) and the SAW (right bar). Issue counts in respective categories are provided in parentheses (M12)

Metric 13 – Time since last position
Domain: *Issue, Decision*, Scale: *Ratio*, Range: *[0,T]*

The design issue lifecycle consists of choice states defined in the context of Metric 10. This metric, given a particular choice state as a parameter, and analogously to Metric 12, calculates the relative amount of time that the design issue has spent in it.

The time since a position was expressed does not appear uniformly distributed (Figure 8.29). As already observed, the majority of alternatives (and thus decisions) have no positions related to them (see Metric 7 – **position count**, Figure 8.23 in particular). The number of design issues with **no positions** appears to be significantly higher for the EP (58%) than for the SAW (33%). In most of the remaining time-bins, the relative frequency of design issues is higher for the SAW than for the EP.

It seems that very few positions were expressed at the beginning of the workshop (**time since last position** over 80 minutes), as well as at the end of the workshop (**time since last position** under 10 minutes). A slight peak of occurrences exists in the range between 30 and 40 minutes.

There is no clear pattern emerging in the scatter charts showing the relationship between the **alternative count** and **deciders count** for design issues (Figure 8.30). In the charts presenting the relations of the **position count** and **deciders count** with the **time since last position** (Figure 8.31), a large number of high-ranking occurrences can be observed in the aforementioned time-range (30-40 minutes).

In Figure 8.33 we present an analysis of consensus states in relation to the **time since last position**. By definition, decisions without positions are in the **no positions** state. For all other categories, it can be observed that the vast majority of decisions have positions in the **aligned** consensus state. We could not find a consistent trend in the relation between the **time since last position** and the consensus state.

Similarly, but for design issues, in Figure 8.32 we present the relation between the **time since last position** and the choice state of the design issue. Per definition **no alternatives** and **no positions** choice states are possible only for design issues without user positions. All other categories show a mixture of **incomplete** and **complete** choice states. However, no clear trend or pattern is emerging in the chart.

Finally, an inspection of the time when the last positions are stated (Metric 13, Figure 8.29) shows that the time-distribution of decisions made throughout the design workshop is relatively even, with a slight bump around the mid-point of the workshop. There is no significant growth of last-position counts towards the end of the workshop, which means that the decision making was not rushed towards the end.

Metric 14 – Consensus state transition count
Domain: *Decision*, Scale: *Ratio*, Range: *[0,N]*

The consensus state on a given design decision traverses multiple states (defined for Metric 8). This metric calculates the number of state transitions. The reason for an increase in the number of consensus state transitions would be, for example, that newly added and/or revoked positions would make the state flip between *aligned* and *colliding*.

Figure 8.34 presents a histogram of consensus state transitions. The numbers in parentheses represent the size of the population, in terms of the number of decisions. Decision creation is accounted for as the first state change. The investigation of the relation between the number of consensus state changes and the final consensus state shows that all decisions in the *no positions* state have exactly one transition, decisions in the *aligned* state have exactly two transitions, and decisions in the *colliding* states have precisely three transitions. In the context of the decision life-cycle (see Section 5.1.2, and Figure 5.6), this implies that the consensus state was not particularly volatile during the design workshops.

Metric 15 – Choice state transition count
Domain: *Issue*, Scale: *Ratio*, Range: *[0,N]*
Like Metric 14, this metric calculates the number of state transitions of the choice state (as defined in the context of Metric 10) for a given design issue.

The design issue creation leaves it in the **no alternatives** choice state. The first transition occurs when an alternative is related to the design issue, thus changing its choice state to **no positions**. Depending on the number of alternatives, with the first position stated, the design issue transitions either to the **complete** state, in cases in which there is only one alternative, or to the **incomplete** choice state if there are multiple alternatives. Following that, further changes in the positions can make the choice state travel between any of the aforementioned states.

Analogously to Metric 14 (consensus state transition count), but referring to the design issue life-cycle (Figure 5.7), in Figure 8.35 we present a histogram of the choice state transitions within the population of the design issues. In this figure, it can be observed that the majority of the design issues for the SAW has between one and three transitions, whereas for the EP most of the design issues have zero or one transition.

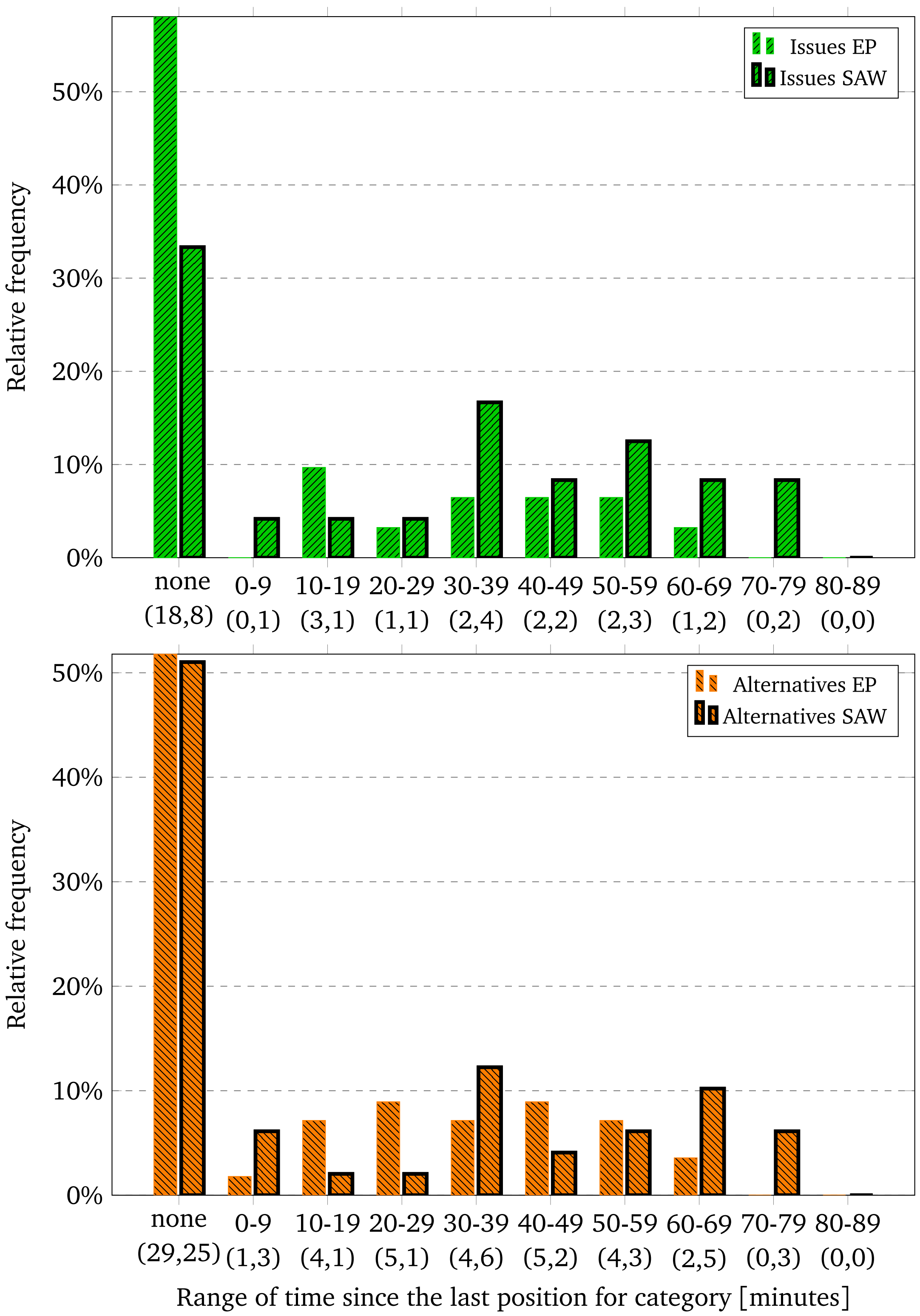


Figure 8.29. Histogram of time elapsed since the last position for issues (upper chart) and alternatives (lower chart) in the EP and the SAW. Issue and alternative counts for respective categories are provided in parentheses (M13)

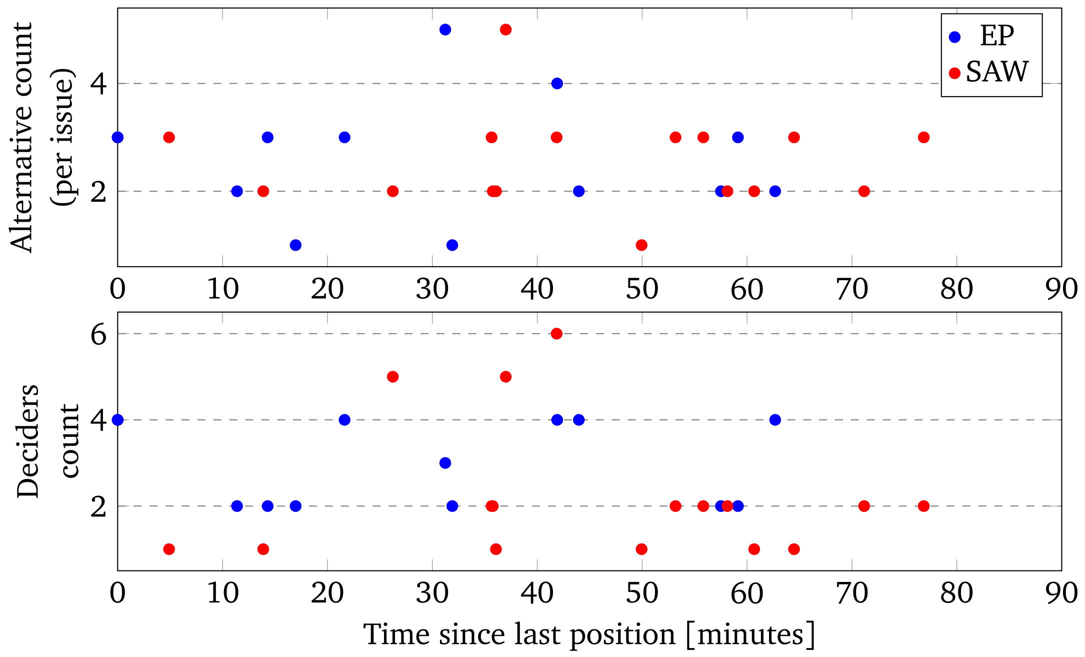


Figure 8.30. Number of alternatives (per issue) and number of deciders plotted for design issues in relation to the time since the last position for the EP and the SAW (M13)

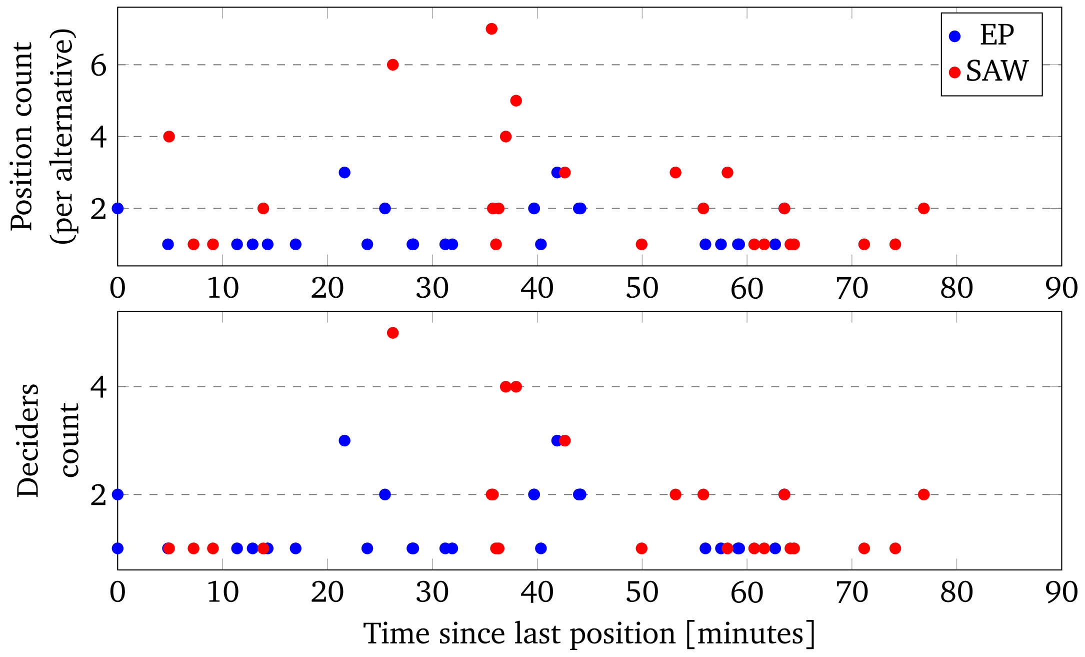


Figure 8.31. Number of positions (per decision) and number of deciders plotted for design alternatives in relation to the time since the last position for the EP and the SAW (M13)

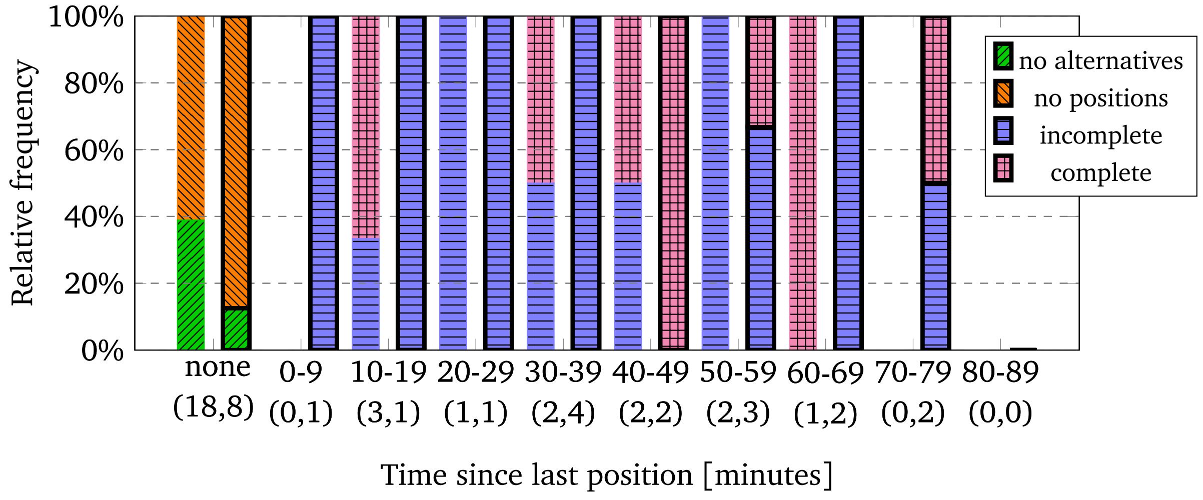


Figure 8.32. A proportion of the design issue choice states in relation to the time elapsed since the last position for the EP (left bar) and the SAW (right bar). In parentheses we provide the absolute count of issues in particular categories (M13)

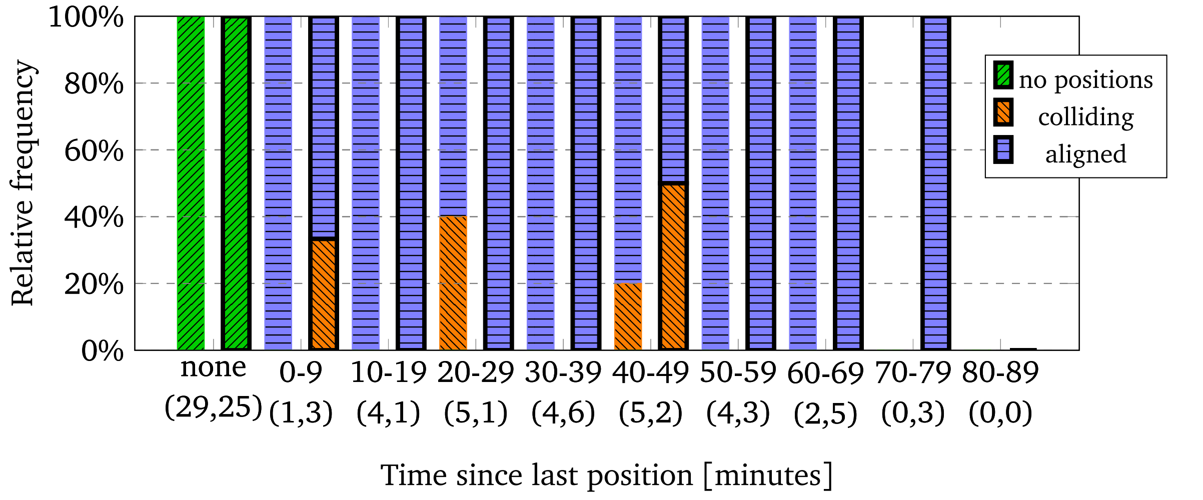


Figure 8.33. A proportion of the alternative states in relation to the time elapsed since the last position for the EP (left bar) and the SAW (right bar). In parentheses we provide the absolute count of alternatives in particular categories (M13)

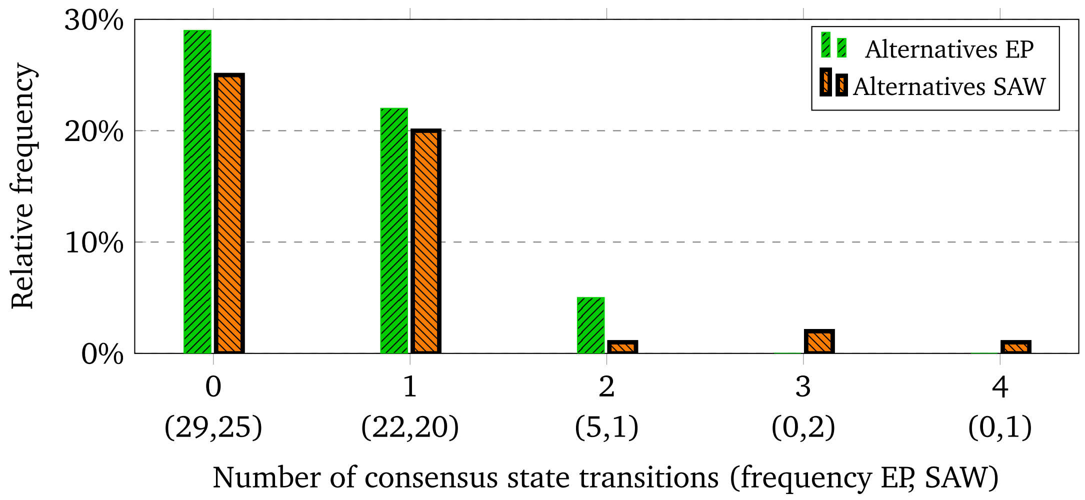


Figure 8.34. Histogram of the number of consensus state transitions for the alternatives. Alternative counts in respective categories are provided in parentheses (M14)

Figure 8.36 presents the proportions of the final choice state in relation to the number of transitions. From the observation that the choice state **no alternatives** applies only to issues with one choice state change, it can be inferred that no alternatives nor positions were deleted in the process. Similarly, from the fact that the **no positions** choice state applies only to design issues with two choice state transitions, it can be inferred that no positions were revoked. The fact that some design issues have more than four state changes (the minimum to reach a complete choice) indicates that the choice state has flipped at least once during the time-frame of the design workshop.

An inspection of the choice state timespan (Metric 12) and state transition count (Metric 15) shows very similar behavior, namely, that in most of the cases, the issues spent a major part of their timespan in their final choice state. There is a slight difference in behavior, however, insofar as a minor amount of design issues (7%) actually turned from the *complete* into the *incomplete* choice state. Observing Metric 15 in Figure 8.35 one can notice that the maximal observed number of choice state transitions reached five, which means at most two choice state flips. This indicates that choice volatility was not particularly high.

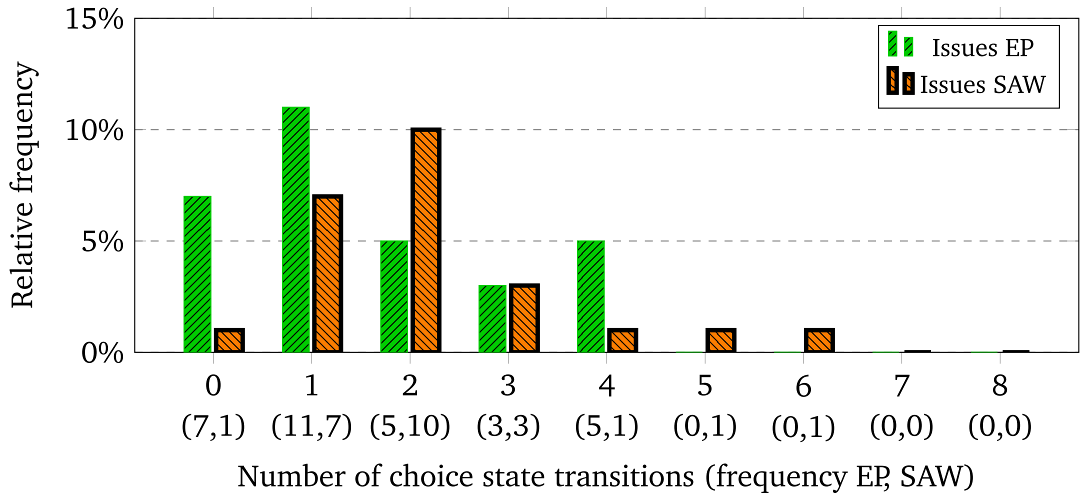


Figure 8.35. Histogram for number of choice state transitions of issues. Issue counts in respective categories are provided in parentheses (M15)

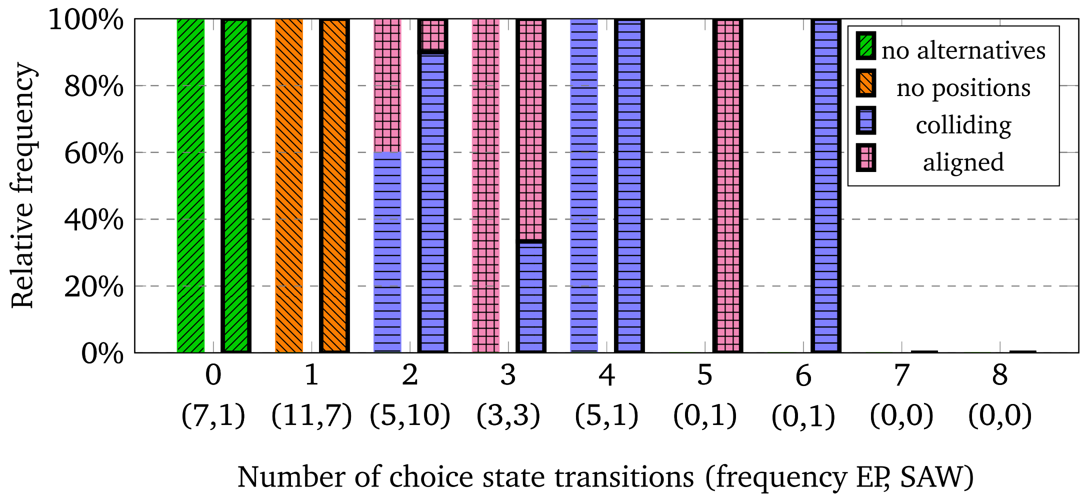


Figure 8.36. Proportion of the final choice states in relation to the number of choice state transitions for the EP (left bar) and the SAW (right bar). Issue counts for respective categories are provided in parentheses (M15)

8.4 Interpretation

In this section revisit the interpretation model that we have introduced in Section 7.1.3 and apply it to the presented results. In order to progress systematically, we follow the three questions that we stated in Section 7.1.1.

8.4.1 Question 1: How aligned are the decisions?

Observing results of Metric 7 (see Figure 8.22) it can be noticed that the position alignment for the decisions with few positions (2-3) is very similar for SAW and EP. The result for decisions with more than three positions that we recorded for SAW shows that they are all in the aligned consensus state (see Figure 8.24). These decisions have most likely drew the most of attention. The fact that we did not record any decisions with more than three positions for EP can be interpreted as the preference of (decision) quantity over quality of argumentation.

In the context of the design issue, the alignment of individual decisions impacts the choice state of a design issue. Seeing a relationship between a number of alternatives and consensus state of them (see Figure 8.25), it can be noticed that the proportion of consensus states does not change significantly and does not depend on the tool type (EP, SAW). It can be also observed that the decisions with aligned consensus state occur only for issues with more than two alternatives. This can be interpreted as focusing on the complex issues that happens independently of used tooling.

8.4.2 Question 2: How volatile is the consensus over the decisions?

Analyzing the **consensus state timespan** (Metric 11, see Figure 8.27) we have observed quite a linear progression of decisions through the consensus state machine. That is, we did not observe any decision returning from the colliding state back to the aligned state. Combined with rather short activity timespan (Metric 5, see Figure 8.13), this can be interpreted as immaturity and incompleteness of the decision model. The flat distribution of the **time since the last change** (Metric 6, see Figure 8.18) suggests that architects kept capturing new decisions all through the design workshop. We expect that given more time for considerations, the architects would revisit and refine their decisions effectively increasing observed volatility.

In terms of impact of tooling on the consensus volatility, it can be noticed that population of design issues and alternatives, with activity time over 10 minutes and in particular in range over 20 minutes, is larger for SAW than its equivalent for

EP. (Metric 5, see Figure 8.16). In Figure 8.17 can be observed that the **position count** for the design alternatives registered by the SAW is significantly higher than the number registered by the EP. A similar trend is visible for the number of **deciders** for design alternatives. Surprisingly, a converse dynamic is visible for the number of **deciders** for design issues. A much less pronounced difference between the number of **editors** is visible both for issues and alternatives. Looking at the **alternative count**, it is visible that for the SAW the number increases slightly with the cut-off time, whereas for the EP value oscillates only slightly.

We interpret this as an effect of SAW helping architects to focus on specific topics for longer, without increasing consensus volatility. Unfortunately, as can be observed in Figure 8.16 the categories' population for long cut-off times are rather low; therefore, the standard deviation calculated for the items in the category is large. Without more data points, it is difficult to draw significant conclusions from observations made here.

8.4.3 Question 3: How democratic are the decisions?

Observing stakeholder involvement in capturing decisions (Metric 3, Figure 8.7) it can be noticed that the number of issues and alternatives captured and edited by more than four architects is significantly higher for SAW than for EP. Similarly, the participation in decision making (Metric 4, see Figure 8.11) is significantly higher for saw when looking at design issues, but not when looking at design alternatives. This needs to be interpreted in relation to the number of alternatives per issue (Metric 2, see Figure 8.6) that is significantly higher for SAW than for EP.

We read this as a sign for low degree of participation in both decision capturing and making, with a slight advantage for decisions captured with SAW. We found it quite surprising that decisions involving a large number of decision makers are dominantly aligned. This observation reconfirms the observation we've made while analyzing other questions, which is that more effort has been invested in particularly complex issues, leaving other ones with less attention.

In terms of decision making strategy, we did not identify a single stakeholder that would dominate the decision making process, therefore definitively cannot speak about the authoritative strategy. On the other hand, as pointed out earlier, the participation rate was relatively low, so we cannot speak about fully collaborative design. Finally, we did not observe a high rate of conclusive choices, characteristic for the competitive design strategy. Consequently, we interpret the dynamics of the architecture workshops as a phase of a mixture between competitive and collaborative strategies.

8.5 Summary

In our exploratory study, we have compared the structure and dynamics of decision models recorded during architecture workshops supported by two tools: the Software Architecture Warehouse and the EtherPad. Both tools provide low-latency support for collaboration in small teams and are suitable for assisting remote and collocated design workshops. The core difference between the SAW and the EP lies in the way the decision model structure is imposed on the recorded data. The SAW uses an explicit decision meta-model, whereas the EP relies on an implicit one (see Section 3.3.3). The breadth of characteristics covered by the framework of metrics that we devised allowed us to investigate the quality of the collaborative architecture decision making process in a comprehensive, multi-dimensional manner.

In reference to our thesis (see Section 4.3), the results that we presented suggest that in fact there exists an noticeable difference in the structure and dynamics of the decision models recorded during the workshops assisted by the respective tools. In the following sections, we thoroughly discuss the threats to the validity of our observations.

8.6 Threats to Validity

The quasi-experiment that we conducted for the purpose of our study is fundamentally burdened by limitations to its validity. Threats to its validity come from three sources [CS63, SCC01], namely, internal, construct, and external.

8.6.1 Internal Validity

The internal threats to validity are threats against the utility of the experiment for demonstrating the causality that is promised by our thesis (see Section 4.3). The variable in the experimental set-up that we have prepared is the degree to which the structure of the recorded decision model is forced on the design team by the tool assisting the decision making process. In order to estimate the influence of the variable on the dynamics of the design process, we have established two set-ups. The base-line set-up provided designers with free-form collaboration features, whereas the investigated scenario included the Software Architecture Warehouse operated within a predefined decision model structure. In order to assure the internal validity of the experiment/observation, it was crucial that both set-ups provided with conditions differentiated only by the investigated variable. To this end, we have devised a number of means to guarantee fair comparison of the groups, working in both set-ups. The first major threat to the comparability of the set-ups could be having non-even populations of designers. We have mitigated this problem by using a key to assign students to either group. In order to assure a balance of skills in both groups we have used the score of a mid-term exam and mixed teams in such a way that the total score set was even.

Another strategy that we have used to assure the comparability of the results was to perform design exercise sessions at exactly the same time so that no information could be leaked between the groups. Additionally, in order to avoid the situations in which a particular design topic matched the expertise of given group, we have repeated the design exercises with a range of systems to be designed. Finally, the participants were not informed beforehand which set-up they were going to participate in, thus implementing the blinding principle (see [Sel11][Chapter 8]).

A significant threat to the internal validity of our observations comes from the small population counts that make it difficult or impossible to draw conclusions with high statistical significance. This is particularly the case for the investigations of the decision items with a long activity time (Metric 5). With this in mind, our results need to be interpreted as the results of an exploratory research. In order to provide high statistical confidence, our analysis would require a significantly larger dataset.

8.6.2 Construct Validity

In order to quantify the qualities of the decision model and the dynamics of the decision making process, we have devised a number of metrics (see Section 7.1.2). The goal of the evaluation that we have performed is to demonstrate the applicability of the argumentation decision metrics to the data recorded during the architecture design workshops.

The metrics provide the raw input that needs to be interpreted. In Section 7.1.3 we have provided an interpretation model suitable for analyzing both set-ups individually and for distilling differences between them. The combination of well-defined metrics together with the interpretation model assures construct validity.

8.6.3 External Validity

The external threats to validity question the potential for generalizing the results obtained in the experimental set-up to the in-situ environment. This is, in fact, a serious issue that many experiments in software engineering are facing.

van Heesch and Avgeriou in [vHA10] and [vHA11] discuss in detail differences between so-called Naive and Mature architecting. In fact, our experimental conditions are close to the Naive architecting mode. We cannot claim anything about the transferability of these results to the Mature architecting mode. Performing similar experiments on the population of “Mature” architects was unfeasible for a number of reasons. First of all, the time of experienced software architects is a very valuable resource. Second, the level of expertise and experience among the profession architects is very diverse. It would be very difficult, if not impossible, to create two design teams with comparable potential to perform our experiments.

Another external threat to validity comes from the fact that we have used a dummy set of exercises instead of real-world problems. We have made a conscious decision about it, because this way we were in full control of the scope of the problem and the feasibility of the solution. In fact, the exercise execution time constraints and logistics such as parallel execution would have made it very difficult to use real-world projects.

Chapter 9

Conclusions

Software systems are encompassing our daily life. Many of them are designed to make it easier by transferring complexity from real-life into the data and processes inside the system. As much as this internal complexity can be hidden from the end-user, it is inevitably absorbed by its design. Software architecture, being an abstraction over the software design, helps to divide and conquer design complexity, but does not remove it altogether. In fact, software architecture design sets very high requirements on the architects' work quality, because of the high impact that it has on the final shape of the system. The breadth and depth of the expertise required to deliver an architecture design often requires the effective collaboration of multiple domain specialists.

The modern way of looking at software architecture is to consider it as a set of principal design decisions [JB05]. This implies that the quality of architecture design translates directly to the quality of architecture design decisions. This observation has guided our investigation of how to estimate and assure high quality of collaborative architecture decisions. We have followed a heuristic for decision quality that relies on decision makers being 1) **smart** and 2) **well informed**. The collaborative decisions introduce a third factor, namely **consensuality**. Our research on architecture decision argumentation modeling and the dynamics of the decision making process is paving the way towards a better understanding of the quality of collaborative architecture decisions and thereby, ultimately, towards better software design.

9.1 Summary

In this dissertation, we thoroughly explored the design space of collaborative software architecture decision making support tools, and within this space we identified a desirable combination of decisions that later served for us as a foundation for the design of the Software Architecture Warehouse. Next, we reviewed a number of architecture decision meta-models and observed that none of them offers support for detailed decision argumentation modeling. Therefore, we have devised an argumentation viewpoint extension to the ISO 42010 architecture description standard. The Software Architecture Warehouse was tailored to fully support the argumentation viewpoint. Within the framework of the argumentation viewpoint we identified a decision consensus lifecycle and a design issue choice state lifecycle. Following that, we specified a framework of fifteen metrics suitable for estimating structure and dynamics of architecture decision models with argumentation. Next, we implemented and applied the framework of decision argumentation metrics to the experimental data acquired during a series of in-class architecture design workshops.

9.2 Contributions

This dissertation makes several contributions that we summarize as follows:

Design Space of Tools SupportingCollaborative Software Architecture Decisions

We surveyed seventeen tools suitable for supporting software architects in the process of collaborative design decision making. The tools ranged from generic, real-time collaborative editors to specialized software architecture modeling suites. As a result, we have obtained a design space with seven design issues and nineteen design alternatives that constitutes a reusable asset for the design of future generation tools.

Architecture Decision Argumentation Viewpoint

We defined an argumentation viewpoint extension to the ISO 42010 architecture description standard. Its main goal is to provide precise representation for designer positions and thereby make it possible to explicitly represent the consensus of design decisions. On the basis of the viewpoint positions, we defined a decision consensus lifecycle and a design issue choice state machine.

Architecture Decision Argumentation Metrics

Within the framework of the decision argumentation viewpoint, we defined a suite of fifteen metrics suitable for estimating structure and dynamics of software architecture decisions, with particular focus on decision argumentation.

The Software Architecture Warehouse

We implemented a web application designed to support collaborative architecture decision making for architecture design workshops in collocated and distributed configurations. The Software Architecture Warehouse fully implements the decision argumentation viewpoint. Additionally we implemented an analyzer that realizes the argumentation metrics that we defined earlier.

In-class Evaluation Study

We conducted a series of in-class architecture design workshops in a configuration that allowed us to collect parallel design logs for two teams, one assisted by the SAW, the other one by the EtherPad. We used the collected data to showcase the application of the Architecture Argumentation Metrics.

9.3 Limitations

In our work on the Software Architecture Warehouse, we limited the scope of managed design artifacts to architecture decisions. We decided to do so in order to focus our research, development and evaluation effort. This approach has proven to be sufficient in the context of the heavily time-constrained, in-class design workshops, but our observations might not transfer well into the context of a large industrial organization that needs to effectively manage a broader spectrum of architecture design artifacts.

In particular simultaneous work on architecture design and decisions, in a fashion similar to twin-peaks model, can potentially change dynamics of architecture workshops. We anticipate a slower pace of workshops and more in-depth discussion within the design domain. We can speculate that such conditions would lead to more pronounced benefits of using structured decision in the live, collaborative setting.

Secondly, we have conducted the evaluation in the sterile, academic conditions, that do not perfectly resemble the typical architecture design workshop conditions in an industrial setting. Our participants are unexperienced and overly uniform. Our experience in industrial environment shows that that typical participants of an architecture workshop are much more diversified in terms of expertise and experience. We have also identified that experienced experts are often strongly opinionated; therefore we anticipate more authoritarian dynamics.

Thirdly, as we already mentioned, the design workshops that we have investigated were limited time-limited by the duration of a single exercise session. In the industrial setting, the design stakeholders typically invest way more effort and time into the architecture design. We expect an increase of time spent in both in a solitary mode - in the form of workshop preparation and post-workshop clean-up, as well as in collaborative modes - multiple rounds of workshops and high latency expert opinion exchange.

Finally, the engagement level of participants in our evaluation was relatively low; resulting in decision spaces that with rather few design decisions that have a significant number of recorded positions. This often made it difficult to interpret metric outcomes and draw conclusions with statistical confidence. We are confident that decision-centric representation of software architecture and, in particular, argumentation viewpoint are going be recognized and gain acceptance of professional software architects; therefore we believe that accumulation of larger number of decision spaces will eventually provide conclusive evidence of observations presented in this thesis.

9.4 Lessons Learned

Through five years, that we have devoted to research and development, we have gathered a unique insight into the state of the art and practice. In this section, we collect lessons that we have learned. We trust that the reader will find our experience valuable and that it will foster progress in software architecture decisions research and practice.

Standing on the shoulders of giants
Pretty early, we have learned that there exists a plethora of architecture decision modeling standards (see Section 3 and there is no point in proposing yet another standard to rule them all. Instead, for we have decided to increase level of abstraction and meta-model decision meta-model, effectively making it possible to apply our considerations to virtually any decision model. This decision was reflected in the design of the Software Architecture Warehouse (see Section 6.6.3).

Limiting engineering scope and focusing the effort
We have taken rather practical, engineering approach to researching software architecture decisions. It means that we have spent a significant amount of time on design, development and broadly understood software engineering. It has been a great journey to make, but with hindsight, we can say that due to limitations in quality and availability of resources it is very hard to produce high quality of software covering broad functionality.

In particular when working with modern and fast changing ecosystem, such as web technologies.

Early engagement with the industry
Our experience shows that gaining an interest of industrial partner is not an easy task. Nevertheless, in spite of the fact that confronting industry was very challenging, it was also very rewarding, as we have received invaluable input that has guided our research.

End-to-end automation of data analysis
We have invested a significant amount of effort in the development of analysis framework that we documented in Chapter 7. Very soon we have realized that changes to the Analyzer often require a re-run of the complex data analysis workflow (see Figure 7.8). An approach that we found successful in managing changes was to introduce a continuous integration principle to data analysis and to eliminate all manual steps from the process.

9.5 The Road Ahead

This dissertation opens the way for the new research in several directions:

User-centric Metrics for the Argumentation Analytics
The analysis of the data collected during the design in-class workshops was focused mostly on the structure and dynamics of the decision model. We see a research potential in characterizing profiles of the design team members depending on the role that they play in the design workshop.

Research on Distributed Architecture Design Team Dynamics
All the design workshops that we conducted took place in a collocated configuration. We expect that a set-up in which some or all of the design team members are telecommuting would show significantly different dynamics from those we have observed. This is particularly tempting because the required infrastructure (i.e. the Software Architecture Warehouse) is fully featured to support this mode of operation.

The dynamics of Professional Design Teams
The evaluation that we have performed relied on subjects that were very homogeneous in terms of their knowledge, but rather inexperienced in the software architecture design. We predict that the dynamics of a team

consisting of professional architects with diverse skills and experience would be significantly different.

Standalone, offline deployment of the Software Architecture Warehouse
The current deployment pattern for the SAW assumes reliable connectivity to the central server. This requirement can be easily fulfilled in office conditions, but might prove to be very difficult to be satisfied on the go. In practice design architects often review and rework their decisions out of the office, without reliable connectivity. The architecture of the SAW (rich web-client) is ready to support this mode of operation, but we see a significant challenge in the non-linear synchronization of the underlying knowledge graph model.

Decision Making Guidance
Due to its warehouse-like characteristics, such as multi-tenancy and its decision meta-model agnosticism, the Software Architecture Warehouse is a perfect tool for serving as an accumulator of design experience. When appropriately aggregated, such design experience can provide great input for the guidance of future decisions. Due to the highly sensitive nature of design decisions, a new, strong authorization model would need to be implemented in the SAW.

Live Monitoring
The current implementation of the Software Architecture Warehouse supports live monitoring for only a few metrics of the decision space under design. We see big potential in providing decision makers with comprehensive live monitoring of the decision space characteristics.

Appendix A

The Dataset

In this appendix, we provide a concise, tabular summary of the experimental data collected during the evaluation process.

Column names in Tables A.1, A.2, A.3, and 7.10 refer to micro-metrics defined in Section 7.2.4. The original workshop event records are available publicly under `https://github.com/ian7/saw/tree/navigate/analysis/logs-unweaved`.

Exercise Run	Source	Project	Destroyed	Lifespan	LastChange	Issues Created	Issues Edited	Issues Destroyed	Alternatives Created	Alternatives Edited	Alternatives Destroyed	Positive Decisions Created	Negative Decisions Created	Open Decisions Created	Decisions Revoked
Run 1	EP	ex4	0	3238	0	9	9	0	9	9	0	0	0	0	0
Run 2	EP	ex5	0	2064	0	2	2	0	5	5	0	0	0	0	0
Run 3	EP	ex7	0	3957	0	13	13	0	27	27	0	14	6	4	0
Run 4	EP	ex8	0	4194	0	3	2	0	7	7	0	7	1	0	0
Run 5	EP	ex9	0	4019	0	4	4	0	8	7	0	0	0	0	0
Run 1	SAW	51765a68da300c1849000001	0	5900	0	5	5	1	9	6	4	3	4	0	0
Run 2	SAW	5187d355da300c37f7000001	0	5371	0	5	5	0	11	6	0	2	4	1	1
not used	SAW	518a833eda300c484c000001	0	7804	0	26	9	11	18	9	0	2	1	1	1
Run 3	SAW	51925958da300c697d000001	0	5587	0	6	6	0	13	9	1	4	4	1	2
Run 4	SAW	5193a637da300c3002000001	0	5563	0	5	4	1	7	4	0	2	0	0	0
Run 5	SAW	519a3a22da300c3793000001	0	5938	0	5	5	0	13	8	2	5	0	0	0

Table A.1. Micro-metric values matrix for selected decision spaces

Project	ID	Destroyed	Lifespan	LastChange	Editors	Alternatives Count	Alternatives in NoPositions	Alternatives in Colliding	Alternatives in Aligned	Alternatives in Sealed	Final Choice	ChoiceState Changes	Time In NoAlternatives	Time In NoPositions	Time In Incomplete	Time In Complete	Deciders	Update count	Last update	Activity Time	Contributors	Deciders2	Time since last decision
517652dfda300c1675000001	51777f76da300c5671000003	0	4580	2705	2	3	1	1	1	0	incomplete	3	28	3	4549	0	1	18	2812	1875	2	1	2705
517652dfda300c1675000001	5177813bda300c57f1000001	0	4127	4029	1	0	0	0	0	0	no alternatives	1	4127	0	0	0	0	1	3266	98	1	0	-1
517652dfda300c1675000001	51778237da300c57f1000004	0	3875	2051	2	2	0	0	2	0	complete	4	32	18	1774	2051	1	28	3523	1824	2	1	2051
517652dfda300c1675000001	517783b9da300c57af000001	0	3489	2431	2	2	1	0	1	0	incomplete	4	71	772	2431	215	1	21	3943	1058	2	1	2646
517652dfda300c1675000001	51778d86da300c57f1000008	0	980	55	1	3	3	0	0	0	incomplete	3	107	818	55	0	0	23	6426	925	1	0	-1
517652dfda300c1675000001	51779096da300c5adf000001	0	196	130	1	0	0	0	0	0	no alternatives	1	196	0	0	0	0	44	7243	66	1	0	-1
51765a68da300c1849000001	517a37dbda300c67dc000001	1	6867	2011	4	2	0	0	2	0	complete	4	4692	3	57	2115	3	29	7193	4856	4	3	2109
51765a68da300c1849000001	517a4368da300c691f000001	0	3910	878	7	3	1	0	2	0	complete	4	32	9	88	3781	6	20	5488	3032	7	6	2511
51765a68da300c1849000001	517a46f7da300c6b31000001	0	2999	879	6	5	2	1	2	0	incomplete	5	102	30	2821	46	5	24	6608	2120	6	5	2220
51765a68da300c1849000001	517a4870da300c6c85000001	1	2622	2595	1	0	0	0	0	0	no alternatives	1	2622	0	0	0	0	23	6779	27	1	0	-1
51765a68da300c1849000001	517a49fbda300c6b43000006	0	2227	880	2	0	0	0	0	0	no alternatives	1	2227	0	0	0	0	4	7176	1347	2	0	-1
51765a68da300c1849000001	517a4b44da300c6c55000001	0	1898	1574	5	2	1	0	1	0	complete	3	54	24	0	1820	5	24	7515	324	5	5	1574
5187d355da300c37f7000001	5189f4dada300c286d000003	0	5443	2452	4	3	2	0	1	0	incomplete	3	68	21	5354	0	2	19	699	2991	4	2	5332
5187d355da300c37f7000001	5189f6afda300c286d000006	0	4974	2299	3	2	1	0	1	0	incomplete	4	105	507	2299	2063	1	13	1131	2675	3	1	4362
5187d355da300c37f7000001	5189fa67da300c2a50000001	0	4022	2103	3	2	2	0	0	0	incomplete	3	164	12	3846	0	0	7	2222	1919	3	0	-1
5187d355da300c37f7000001	5189fcf0da300c2a32000001	0	3373	3334	1	1	1	0	0	0	no positions	2	38	3335	0	0	0	29	2742	39	1	0	-1
5187d355da300c37f7000001	518a0425da300c2ca3000001	0	1528	955	1	3	0	1	2	0	incomplete	7	60	62	1297	109	1	23	5131	573	1	1	1015
518a833eda300c484c000001	518cb237da300c3c1f000003	0	6125	3685	6	2	1	1	0	0	incomplete	3	28	26	6071	0	5	6	1688	2440	6	5	3685
518a833eda300c484c000001	518cb2c6da300c3d05000003	0	5982	5982	1	0	0	0	0	0	no alternatives	1	5982	0	0	0	0	0	1822	0	1	0	-1
518a833eda300c484c000001	518cb334da300c3ceb000001	1	5872	5499	2	0	0	0	0	0	no alternatives	1	5872	0	0	0	0	0	1932	373	2	0	-1
518a833eda300c484c000001	518cb345da300c3c8b000001	0	5855	5855	1	0	0	0	0	0	no alternatives	1	5855	0	0	0	0	0	1949	0	1	0	-1
518a833eda300c484c000001	518cb347da300c3c28000004	0	5853	5853	1	0	0	0	0	0	no alternatives	1	5853	0	0	0	0	0	1951	0	1	0	-1
518a833eda300c484c000001	518cb361da300c3c16000005	1	5827	5736	2	0	0	0	0	0	no alternatives	1	5827	0	0	0	0	23	2018	91	2	0	-1
518a833eda300c484c000001	518cb3a1da300c3c16000006	1	5763	5740	1	0	0	0	0	0	no alternatives	1	5763	0	0	0	0	0	2041	23	1	0	-1
518a833eda300c484c000001	518cb3a3da300c3c70000003	1	5761	5728	1	0	0	0	0	0	no alternatives	1	5761	0	0	0	0	0	2043	33	1	0	-1
518a833eda300c484c000001	518cb3a5da300c3c28000006	1	5760	5726	1	0	0	0	0	0	no alternatives	1	5760	0	0	0	0	0	2044	34	1	0	-1
518a833eda300c484c000001	518cb3a5da300c3cf4000001	0	5759	3774	3	2	1	0	1	0	incomplete	3	67	3	5689	0	1	17	2126	1985	3	1	5443
518a833eda300c484c000001	518cb3a5da300c3cf4000002	0	5759	5759	1	0	0	0	0	0	no alternatives	1	5759	0	0	0	0	0	2045	0	1	0	-1
518a833eda300c484c000001	518cb3a5da300c3c16000007	1	5759	5725	1	0	0	0	0	0	no alternatives	1	5759	0	0	0	0	0	2045	34	1	0	-1
518a833eda300c484c000001	518cb3a6da300c3ccf000003	0	5758	5758	1	0	0	0	0	0	no alternatives	1	5758	0	0	0	0	0	2046	0	1	0	-1
518a833eda300c484c000001	518cb3adda300c3d05000004	1	5751	5723	1	0	0	0	0	0	no alternatives	1	5751	0	0	0	0	0	2053	28	1	0	-1
518a833eda300c484c000001	518cb489da300c3c28000008	0	5531	5531	1	0	0	0	0	0	no alternatives	1	5531	0	0	0	0	0	2273	0	1	0	-1
518a833eda300c484c000001	518cb567da300c3d05000007	1	5309	4855	2	0	0	0	0	0	no alternatives	1	5309	0	0	0	0	0	2495	454	2	0	-1
518a833eda300c484c000001	518cb6b0da300c3d0500000b	1	4980	4851	2	0	0	0	0	0	no alternatives	1	4980	0	0	0	0	0	2824	129	2	0	-1
518a833eda300c484c000001	518cb6b0da300c3cf4000008	1	4980	4851	2	0	0	0	0	0	no alternatives	1	4980	0	0	0	0	0	2824	129	2	0	-1
518a833eda300c484c000001	518cb6b9da300c3c28000009	1	4971	4849	2	0	0	0	0	0	no alternatives	1	4971	0	0	0	0	0	2833	122	2	0	-1
518a833eda300c484c000001	518cb749da300c3c70000004	0	4827	3474	2	1	1	0	0	0	no positions	2	1352	3475	0	0	0	8	2988	1353	2	0	-1
518a833eda300c484c000001	518cb811da300c3c70000005	0	4627	4406	4	2	2	0	0	0	incomplete	3	104	18	4505	0	0	14	3263	221	4	0	-1
518a833eda300c484c000001	518cb868da300c3c70000007	0	4540	4437	1	3	3	0	0	0	incomplete	3	26	65	4449	0	0	13	3278	103	1	0	-1
518a833eda300c484c000001	518cb8bada300c3ccf000005	0	4458	4458	1	0	0	0	0	0	no alternatives	1	4458	0	0	0	0	0	3346	0	1	0	-1
518a833eda300c484c000001	518cbb9bda300c3cf400000c	0	3721	3518	2	2	2	0	0	0	no positions	2	203	3518	0	0	0	15	4262	203	2	0	-1

Project	ID	Destroyed	Lifespan	LastChange	Editors	Alternatives Count	Alternatives in NoPositions	Alternatives in Colliding	Alternatives in Aligned	Alternatives in Sealed	Final Choice	ChoiceState Changes	Time In NoAlternatives	Time In NoPositions	Time In Incomplete	Time In Complete	Deciders	Update count	Last update	Activity Time	Contributors	Deciders2	Time since last decision
518a833eda300c484c000001	518cbcafda300c3cfd000006	0	3445	2943	3	6	6	0	0	0	incomplete	3	37	29	3379	0	0	13	4385	502	3	0	-1
518a833eda300c484c000001	518cc0aada300c3c2800000e	0	2426	2239	2	1	0	0	1	0	complete	3	107	24	0	2295	2	18	5393	187	2	2	2239
51925958da300c697d000001	51932ecdda300c1678000001	0	5799	3869	2	3	2	0	1	0	incomplete	3	93	99	5607	0	1	58	868	1930	2	1	3869
51925958da300c697d000001	51932f88da300c1666000003	0	5612	5479	1	2	2	0	0	0	incomplete	3	114	19	5479	0	0	29	1084	133	1	0	-1
51925958da300c697d000001	51933184da300c168a000005	0	5104	3488	2	2	0	0	2	0	complete	6	476	11	876	3741	2	143	1634	1616	2	2	3488
51925958da300c697d000001	5193321cda300c1768000002	0	4952	3191	3	3	1	0	2	0	incomplete	3	114	309	4529	0	2	8	1707	1761	3	2	3191
51925958da300c697d000001	51933226da300c1678000003	0	4942	4364	2	1	1	0	0	0	no positions	2	237	4705	0	0	0	45	2222	578	2	0	-1
51925958da300c697d000001	519333d4da300c1666000009	0	4512	4270	3	2	0	0	2	0	complete	4	36	24	182	4270	2	13	2257	242	3	2	4270
5193a637da300c3002000001	5195f29ada300c2853000002	0	3777	2886	4	3	2	0	1	0	incomplete	3	24	38	3715	0	2	20	1807	891	4	2	3350
5193a637da300c3002000001	5195f60bda300c2a50000001	1	2896	2888	1	0	0	0	0	0	no alternatives	1	2896	0	0	0	0	0	2667	8	1	0	-1
5193a637da300c3002000001	5195f82eda300c2a50000003	0	2349	833	2	2	1	0	1	0	incomplete	3	15	12	2322	0	1	12	3251	1516	2	1	833
5193a637da300c3002000001	5195f866da300c2baa000002	0	2293	1820	3	2	2	0	0	0	incomplete	3	448	25	1820	0	0	21	3708	473	3	0	-1
5193a637da300c3002000001	5195fa28da300c2c31000001	0	1843	1793	1	2	2	0	0	0	no alternatives	1	1843	0	0	0	0	33	3746	50	1	0	-1
519a3a22da300c3793000001	519c6c39da300c7f59000001	0	3835	1	3	5	5	0	0	0	incomplete	3	64	1520	2251	0	0	38	2139	3834	3	0	-1
519a3a22da300c3793000001	519c6d5ada300c0fdb000001	0	3546	2139	3	3	1	0	2	0	incomplete	3	56	32	3458	0	2	40	2436	1407	3	2	2139
519a3a22da300c3793000001	519c6ee0da300c7f59000002	0	3156	2164	1	2	1	0	1	0	incomplete	3	37	10	3109	0	1	28	2815	992	1	1	2164
519a3a22da300c3793000001	519c6f3eda300c0263000001	0	3062	2997	2	1	0	0	1	0	complete	3	53	12	0	2997	1	13	2886	65	2	1	2997
519a3a22da300c3793000001	519c713bda300c7f7d000008	0	2553	2146	2	2	1	0	1	0	incomplete	3	26	13	2514	0	2	18	3405	407	2	2	2146
ex4	ex4-101	0	3238	2916	1	0	0	0	0	0	no alternatives	1	3238	0	0	0	0	1	322	322	1	0	-1
ex4	ex4-102	0	2663	2381	2	2	2	0	0	0	incomplete	3	150	132	2381	0	1	1	576	282	2	0	-1
ex4	ex4-103	0	2662	2603	1	0	0	0	0	0	no alternatives	1	2662	0	0	0	0	1	635	59	1	0	-1
ex4	ex4-104	0	2585	2072	1	1	1	0	0	0	no positions	2	513	2072	0	0	1	1	1133	513	1	0	-1
ex4	ex4-105	0	1610	1572	1	1	1	0	0	0	no positions	2	38	1572	0	0	1	1	1663	38	1	0	-1
ex4	ex4-106	0	1327	537	1	2	2	0	0	0	incomplete	3	719	71	537	0	1	1	1928	790	1	0	-1
ex4	ex4-107	0	1130	361	2	3	3	0	0	0	incomplete	3	755	6	369	0	1	2	2862	769	2	0	-1
ex4	ex4-108	0	1067	1003	1	0	0	0	0	0	no alternatives	1	1067	0	0	0	0	1	2235	64	1	0	-1
ex4	ex4-109	0	566	541	1	0	0	0	0	0	no alternatives	1	566	0	0	0	0	1	2697	25	1	0	-1
ex5	ex5-102	0	1228	1206	1	2	2	0	0	0	incomplete	3	12	10	1206	0	1	1	846	22	1	0	-1
ex5	ex5-103	0	462	322	1	3	3	0	0	0	incomplete	3	73	42	347	0	1	1	1671	140	1	0	-1
ex7	ex7-101	0	3957	3548	2	3	1	0	2	0	incomplete	3	355	2	3600	0	2	1	351	409	2	2	3548
ex7	ex7-102	0	3534	3451	2	2	1	0	1	0	incomplete	3	16	60	3458	0	2	1	435	83	2	1	3451
ex7	ex7-103	0	3518	3517	1	0	0	0	0	0	no alternatives	1	3518	0	0	0	0	1	440	1	1	0	-1
ex7	ex7-104	0	3445	2514	4	4	2	1	1	0	incomplete	5	44	6	3393	2	4	1	555	931	4	3	2514
ex7	ex7-105	0	2514	1874	3	5	1	0	4	0	incomplete	5	5	11	2413	85	3	1	1447	640	3	2	1874
ex7	ex7-106	0	2354	2346	1	0	0	0	0	0	no alternatives	1	2354	0	0	0	0	1	1611	8	1	0	-1
ex7	ex7-107	0	2291	1914	3	1	0	0	1	0	complete	3	263	114	0	1914	2	1	1834	377	3	1	1914
ex7	ex7-108	0	2178	2076	1	0	0	0	0	0	no alternatives	1	2178	0	0	0	0	1	1881	102	1	0	-1
ex7	ex7-109	0	1757	1298	5	3	0	1	2	0	incomplete	4	9	0	1666	82	4	1	2203	459	5	4	1298
ex7	ex7-110	0	1746	0	4	3	0	2	1	0	incomplete	5	12	9	1662	63	4	1	2223	1746	4	3	0
ex7	ex7-111	0	1309	1018	2	1	0	0	1	0	complete	2	0	291	0	1018	2	2	2673	291	2	1	1018
ex7	ex7-112	0	902	858	2	3	2	0	1	0	incomplete	3	5	5	892	0	2	1	3058	44	2	1	858
ex7	ex7-113	0	845	683	2	2	0	0	2	0	complete	4	21	13	128	683	2	1	3128	162	2	2	683
ex8	ex8-101	0	4194	3761	4	2	0	0	2	0	complete	4	159	4	270	3761	4	1	159	433	4	3	3761
ex8	ex8-102	0	3032	2638	4	2	0	0	2	0	complete	4	287	18	79	2648	4	1	1449	394	4	3	2638

Project	ID	Destroyed	Lifespan	LastChange	Editors	Alternatives Count	Alternatives in NoPositions	Alternatives in Colliding	Alternatives in Aligned	Alternatives in Sealed	Final Choice	ChoiceState Changes	Time In NoAlternatives	Time In NoPositions	Time In Incomplete	Time In Complete	Deciders	Update count	Last update	Activity Time	Contributors	Deciders2	Time since last decision
ex8	ex8-103	0	2727	0	4	3	0	1	2	0	incomplete	5	0	79	387	2261	4	0	3682	2727	4	3	0
ex9	ex9-101	0	4019	3993	1	2	2	0	0	0	incomplete	3	10	16	3993	0	1	1	9	26	1	0	-1
ex9	ex9-102	0	3985	3965	1	2	2	0	0	0	no positions	2	9	3976	0	0	1	1	42	20	1	0	-1
ex9	ex9-103	0	3976	3697	2	3	3	0	0	0	incomplete	2	0	275	3701	0	1	1	309	279	2	0	-1
ex9	ex9-104	0	2180	22	1	1	1	0	0	0	no positions	2	2158	22	0	0	1	1	3992	2158	1	0	-1

Table A.2. Micro-metric values matrix for the design issues

Project	ID	Destroyed	Lifespan	LastChange	Position Count	Compacted Position Count	Positive	PositiveRevoked	Negative	NegativeRevoked	Open	OpenRevoked	Final State	Final Decision	Deciders	Time In NoPositions	Time In Aligned	Time In Colliding	Time In Sealed	Consensus State Changes	Editors	Update count	Last update	Activity Time	Contributors	Deciders2	Time since last decision
5193a637da300c3002000001	4faa1da8924ff84f2600000e	0	1804	1772	0	0	0	0	0	0	0	0	no positions	n/a	0	0	0	0	0	0	1	0	3759	32	1	0	-1
51765a68da300c1849000001	4faa1da8924ff84f2600000e	0	1839	1254	0	0	0	0	0	0	0	0	no positions	n/a	0	0	0	0	0	0	2	0	7539	585	2	0	-1
518a833eda300c484c000001	4faa1da8924ff84f2600000e	0	3525	3525	0	0	0	0	0	0	0	0	no positions	n/a	0	0	0	0	0	0	1	0	4279	0	1	0	-1
5193a637da300c3002000001	4faa1dad924ff852c2000008	0	1793	1793	0	0	0	0	0	0	0	0	no positions	n/a	0	0	0	0	0	0	1	0	3770	0	1	0	-1
517652dfda300c1675000001	51777f92da300c56d5000002	0	4552	2705	2	2	1	0	0	0	1	0	colliding	n/a	1	1487	360	2705	0	2	2	0	2825	1847	2	1	2705
517652dfda300c1675000001	51777f95da300c56cc000001	0	4549	4376	1	1	0	0	1	0	0	0	aligned	Negative	1	173	4376	0	0	1	2	0	2828	173	2	1	4376
517652dfda300c1675000001	51777fabda300c56cc000002	0	4527	2756	0	0	0	0	0	0	0	0	no positions	n/a	0	4527	0	0	0	0	1	22	2940	1771	1	0	-1
517652dfda300c1675000001	51778257da300c56d5000007	0	3843	2153	1	1	1	0	0	0	0	0	aligned	Positive	1	1690	2153	0	0	1	2	0	3534	1690	2	1	2153
517652dfda300c1675000001	51778269da300c56a7000002	0	3825	2051	1	1	0	0	1	0	0	0	aligned	Negative	1	1774	2051	0	0	1	2	0	3552	1774	2	1	2051
517652dfda300c1675000001	51778400da300c57a6000002	0	3418	2646	1	1	0	0	1	0	0	0	aligned	Negative	1	772	2646	0	0	1	2	0	3959	772	2	1	2646
517652dfda300c1675000001	517787dbda300c57f1000007	0	2431	2370	0	0	0	0	0	0	0	0	no positions	n/a	0	2431	0	0	0	0	1	37	4986	61	1	0	-1
517652dfda300c1675000001	51778df1da300c5af3000001	1	873	867	0	0	0	0	0	0	0	0	no positions	n/a	0	873	0	0	0	0	1	0	6504	6	1	0	-1
517652dfda300c1675000001	51778e66da300c5afd000001	0	756	0	0	0	0	0	0	0	0	0	no positions	n/a	0	756	0	0	0	0	1	7	7376	756	1	0	-1
517652dfda300c1675000001	51779123da300c5baf000001	0	55	41	0	0	0	0	0	0	0	0	no positions	n/a	0	55	0	0	0	0	1	9	7335	14	1	0	-1
51765a68da300c1849000001	517a4388da300c6aaa000001	1	3878	1080	6	6	6	0	0	0	0	0	aligned	Positive	6	20	3858	0	0	1	7	4	5508	2798	7	6	2511
51765a68da300c1849000001	517a4391da300c6a81000003	0	3869	878	2	2	0	0	2	0	0	0	aligned	Negative	2	38	3831	0	0	1	3	4	5514	2991	3	2	3814
51765a68da300c1849000001	517a43b4da300c6b1d000001	1	3834	3791	0	0	0	0	0	0	0	0	no positions	n/a	0	3834	0	0	0	0	2	0	5544	43	2	0	-1
51765a68da300c1849000001	517a475dda300c6c1c000002	0	2897	879	5	5	0	0	5	0	0	0	aligned	Negative	4	41	2856	0	0	1	5	10	6505	2018	5	4	2279
51765a68da300c1849000001	517a477bda300c6c29000001	0	2867	879	3	3	2	0	1	0	0	0	colliding	n/a	3	135	174	2558	0	2	6	12	6521	1988	6	3	2558
51765a68da300c1849000001	517a4788da300c6c46000001	0	2854	880	4	4	4	0	0	0	0	0	aligned	Positive	4	82	2772	0	0	1	6	27	6553	1974	6	4	2220
51765a68da300c1849000001	517a4830da300c6c61000005	1	2686	2170	0	0	0	0	0	0	0	0	no positions	n/a	0	2686	0	0	0	0	2	0	6692	516	2	0	-1
51765a68da300c1849000001	517a4867da300c6b31000005	1	2631	2175	0	0	0	0	0	0	0	0	no positions	n/a	0	2631	0	0	0	0	2	0	6747	456	2	0	-1
51765a68da300c1849000001	517a4a2fda300c6c6a000003	0	2175	2130	1	1	0	0	1	0	0	0	aligned	Negative	1	45	2130	0	0	1	2	0	7203	45	2	1	2130
51765a68da300c1849000001	517a4a32da300c6c29000002	0	2172	2109	2	2	2	0	0	0	0	0	aligned	Positive	2	57	2115	0	0	1	3	0	7206	63	3	2	2109
51765a68da300c1849000001	517a4b7ada300c6c61000006	0	1844	1574	6	6	0	0	6	0	0	0	aligned	Negative	5	24	1820	0	0	1	5	5	7550	270	5	5	1574
5187d355da300c37f7000001	5189f51eda300c2888000003	0	5375	5373	0	0	0	0	0	0	0	0	no positions	n/a	0	5375	0	0	0	0	1	0	717	2	1	0	-1
5187d355da300c37f7000001	5189f533da300c2923000001	0	5354	2453	2	2	2	0	0	0	0	0	aligned	Positive	2	10	5344	0	0	1	3	0	738	2901	3	2	5332
5187d355da300c37f7000001	5189f718da300c2968000001	0	4869	4362	1	1	0	0	1	0	0	0	aligned	Negative	1	507	4362	0	0	1	3	49	1263	507	3	1	4362
5187d355da300c37f7000001	5189fafcda300c287e000006	0	3873	3838	0	0	0	0	0	0	0	0	no positions	n/a	0	3873	0	0	0	0	1	36	2252	35	1	0	-1
5187d355da300c37f7000001	5189fb0bda300c2a87000001	0	3858	3857	0	0	0	0	0	0	0	0	no positions	n/a	0	3858	0	0	0	0	1	0	2234	1	1	0	-1
5187d355da300c37f7000001	5189fb17da300c2a87000002	0	3846	3846	0	0	0	0	0	0	0	0	no positions	n/a	0	3846	0	0	0	0	1	0	2246	0	1	0	-1
5187d355da300c37f7000001	5189fd16da300c2b9c000001	0	3335	3321	0	0	0	0	0	0	0	0	no positions	n/a	0	3335	0	0	0	0	1	13	2770	14	1	0	-1
5187d355da300c37f7000001	518a0122da300c2a32000002	0	2299	2283	0	0	0	0	0	0	0	0	no positions	n/a	0	2299	0	0	0	0	1	15	3807	16	1	0	-1
5187d355da300c37f7000001	518a0461da300c2ca3000003	0	1468	1155	1	1	0	0	1	0	0	0	aligned	Negative	1	313	1155	0	0	1	1	0	4624	313	1	1	1155
5187d355da300c37f7000001	518a049fda300c2dcd000001	0	1406	1015	4	2	1	0	0	1	1	0	colliding	n/a	1	282	109	1015	0	4	1	5	4846	391	1	1	1015
5187d355da300c37f7000001	518a050cda300c2dba000002	0	1297	1266	1	1	0	0	1	0	0	0	aligned	Negative	1	31	1266	0	0	1	1	29	4820	31	1	1	1266
518a833eda300c484c000001	518cb253da300c3c16000002	0	6097	3685	8	6	5	0	0	0	1	1	colliding	n/a	5	39	16	6042	0	2	5	4	1730	2412	5	5	3685
518a833eda300c484c000001	518cb26dda300c3c1f000004	0	6071	5493	0	0	0	0	0	0	0	0	no positions	n/a	0	6071	0	0	0	0	2	5	1742	578	2	0	-1
518a833eda300c484c000001	518cb3e8da300c3b6f000009	0	5692	5589	0	0	0	0	0	0	0	0	no positions	n/a	0	5692	0	0	0	0	1	3	2214	103	1	0	-1
518a833eda300c484c000001	518cb3ebda300c3c82000003	0	5689	5443	1	1	0	0	1	0	0	0	aligned	Negative	1	246	5443	0	0	1	2	1	2210	246	2	1	5443
518a833eda300c484c000001	518cb879da300c3c31000008	0	4523	4523	0	0	0	0	0	0	0	0	no positions	n/a	0	4523	0	0	0	0	1	0	3281	0	1	0	-1

Project	ID	Destroyed	Lifespan	LastChange	Position Count	Compacted Position Count	Positive	PositiveRevoked	Negative	NegativeRevoked	Open	OpenRevoked	Final State	Final Decision	Deciders	Time In NoPositions	Time In Aligned	Time In Colliding	Time In Sealed	Consensus State Changes	Editors	Update count	Last update	Activity Time	Contributors	Deciders2	Time since last decision
518a833eda300c484c000001	518cb882da300c3c1f00000c	0	4514	4186	0	0	0	0	0	0	0	0	no positions	n/a	0	4514	0	0	0	0	1	29	3617	328	1	0	-1
518a833eda300c484c000001	518cb88bda300c3cfd000003	0	4505	4403	0	0	0	0	0	0	0	0	no positions	n/a	0	4505	0	0	0	0	2	0	3299	102	2	0	-1
518a833eda300c484c000001	518cb8c3da300c3d0500000c	0	4449	4449	0	0	0	0	0	0	0	0	no positions	n/a	0	4449	0	0	0	0	1	0	3355	0	1	0	-1
518a833eda300c484c000001	518cb8cfda300c3c1f00000d	0	4437	4063	0	0	0	0	0	0	0	0	no positions	n/a	0	4437	0	0	0	0	1	10	3740	374	1	0	-1
518a833eda300c484c000001	518cbc66da300c3c16000009	0	3518	3518	0	0	0	0	0	0	0	0	no positions	n/a	0	3518	0	0	0	0	1	0	4286	0	1	0	-1
518a833eda300c484c000001	518cbc91da300c3ceb000004	0	3475	3418	0	0	0	0	0	0	0	0	no positions	n/a	0	3475	0	0	0	0	1	61	4386	57	1	0	-1
518a833eda300c484c000001	518cbcd4da300c3c7000000a	0	3408	3407	0	0	0	0	0	0	0	0	no positions	n/a	0	3408	0	0	0	0	1	0	4396	1	1	0	-1
518a833eda300c484c000001	518cbcf1da300c3d0500000f	0	3379	3378	0	0	0	0	0	0	0	0	no positions	n/a	0	3379	0	0	0	0	1	0	4425	1	1	0	-1
518a833eda300c484c000001	518cbd35da300c3b6f00000d	0	3311	3282	0	0	0	0	0	0	0	0	no positions	n/a	0	3311	0	0	0	0	1	19	4518	29	1	0	-1
518a833eda300c484c000001	518cbe39da300c3c3100000a	0	3051	3051	0	0	0	0	0	0	0	0	no positions	n/a	0	3051	0	0	0	0	1	0	4753	0	1	0	-1
518a833eda300c484c000001	518cbe50da300c3cf400000e	0	3028	3028	0	0	0	0	0	0	0	0	no positions	n/a	0	3028	0	0	0	0	1	0	4776	0	1	0	-1
518a833eda300c484c000001	518cbea4da300c3c2800000d	0	2944	2899	0	0	0	0	0	0	0	0	no positions	n/a	0	2944	0	0	0	0	1	51	4902	45	1	0	-1
518a833eda300c484c000001	518cc115da300c4081000001	0	2319	2239	2	2	2	0	0	0	0	0	aligned	Positive	2	24	2295	0	0	1	2	0	5485	80	2	2	2239
51925958da300c697d000001	51932f2ada300c1666000002	0	5706	5625	0	0	0	0	0	0	0	0	no positions	n/a	0	5706	0	0	0	0	1	35	961	81	1	0	-1
51925958da300c697d000001	51932f8dda300c175e000001	0	5607	3869	1	1	1	0	0	0	0	0	aligned	Positive	1	1738	3869	0	0	1	1	36	1056	1738	1	1	3869
51925958da300c697d000001	51932ffada300c1680000003	0	5498	5479	0	0	0	0	0	0	0	0	no positions	n/a	0	5498	0	0	0	0	1	16	1106	19	1	0	-1
51925958da300c697d000001	5193300dda300c1666000004	0	5479	5469	0	0	0	0	0	0	0	0	no positions	n/a	0	5479	0	0	0	0	1	12	1117	10	1	0	-1
51925958da300c697d000001	519330dcda300c1728000002	1	5272	5215	0	0	0	0	0	0	0	0	no positions	n/a	0	5272	0	0	0	0	1	3	1367	57	1	0	-1
51925958da300c697d000001	5193328eda300c1768000003	0	4838	3698	1	1	0	0	1	0	0	0	aligned	Negative	1	1140	3698	0	0	1	2	12	1770	1140	2	1	3698
51925958da300c697d000001	51933313da300c1678000004	0	4705	4650	0	0	0	0	0	0	0	0	no positions	n/a	0	4705	0	0	0	0	1	2	1936	55	1	0	-1
51925958da300c697d000001	51933360da300c1666000007	0	4628	3488	3	1	1	0	0	0	0	1	aligned	Positive	1	887	3741	0	0	3	1	0	1959	1140	1	1	3488
51925958da300c697d000001	5193336bda300c1850000001	0	4617	3848	1	1	0	0	1	0	0	0	aligned	Negative	1	769	3848	0	0	1	2	0	1970	769	2	1	3848
51925958da300c697d000001	519333c3da300c168a000006	0	4529	3191	3	1	1	0	0	1	0	0	aligned	Positive	2	86	4251	192	0	3	2	40	2106	1338	2	2	3191
51925958da300c697d000001	519333c4da300c1768000004	0	4528	4512	0	0	0	0	0	0	0	0	no positions	n/a	0	4528	0	0	0	0	1	3	2074	16	1	0	-1
51925958da300c697d000001	519333f8da300c175e000005	0	4476	4448	1	1	1	0	0	0	0	0	aligned	Positive	1	28	4448	0	0	1	1	0	2111	28	1	1	4448
51925958da300c697d000001	51933410da300c17ff000003	0	4452	4270	1	1	0	0	1	0	0	0	aligned	Negative	1	182	4270	0	0	1	2	0	2135	182	2	1	4270
5193a637da300c3002000001	5195f2b2da300c2a07000001	0	3753	3350	2	2	2	0	0	0	0	0	aligned	Positive	2	131	3622	0	0	1	3	26	1838	403	3	2	3350
5193a637da300c3002000001	5195f2d8da300c2a07000002	0	3715	2874	0	0	0	0	0	0	0	0	no positions	n/a	0	3715	0	0	0	0	2	20	1866	841	2	0	-1
5193a637da300c3002000001	5195f615da300c2a58000001	0	2886	2875	0	0	0	0	0	0	0	0	no positions	n/a	0	2886	0	0	0	0	1	10	2684	11	1	0	-1
5193a637da300c3002000001	5195f83dda300c2ba1000001	0	2334	833	2	2	2	0	0	0	0	0	aligned	Positive	1	1477	857	0	0	1	2	6	3258	1501	2	1	833
5193a637da300c3002000001	5195f849da300c2bb5000001	0	2322	2321	0	0	0	0	0	0	0	0	no positions	n/a	0	2322	0	0	0	0	1	0	3241	1	1	0	-1
5193a637da300c3002000001	5195fa26da300c2ba1000002	0	1845	1845	0	0	0	0	0	0	0	0	no positions	n/a	0	1845	0	0	0	0	1	0	3718	0	1	0	-1
5193a637da300c3002000001	5195fa3fda300c2a50000005	0	1820	1820	0	0	0	0	0	0	0	0	no positions	n/a	0	1820	0	0	0	0	1	0	3743	0	1	0	-1
519a3a22da300c3793000001	519c6c79da300c019f000001	0	3771	158	0	0	0	0	0	0	0	0	no positions	n/a	0	3771	0	0	0	0	2	0	2167	3613	2	0	-1
519a3a22da300c3793000001	519c6d92da300c0fdb000002	0	3490	2139	7	7	7	0	0	0	0	0	aligned	Positive	2	857	2633	0	0	1	3	0	2448	1351	3	2	2139
519a3a22da300c3793000001	519c6db2da300c022e000001	0	3458	2179	2	2	2	0	0	0	0	0	aligned	Positive	1	1190	2268	0	0	1	2	0	2480	1279	2	1	2179
519a3a22da300c3793000001	519c6debda300c024e000001	0	3401	3401	0	0	0	0	0	0	0	0	no positions	n/a	0	3401	0	0	0	0	1	0	2537	0	1	0	-1
519a3a22da300c3793000001	519c6f05da300c7f59000003	0	3119	2164	1	1	1	0	0	0	0	0	aligned	Positive	1	955	2164	0	0	1	1	3	2826	955	1	1	2164
519a3a22da300c3793000001	519c6f0fda300c7f7d000006	0	3109	3099	0	0	0	0	0	0	0	0	no positions	n/a	0	3109	0	0	0	0	1	5	2838	10	1	0	-1
519a3a22da300c3793000001	519c6f73da300c7f7d000007	0	3009	2926	1	1	1	0	0	0	0	0	aligned	Positive	1	12	2997	0	0	1	2	22	3008	83	2	1	2997
519a3a22da300c3793000001	519c7155da300c02e0000005	0	2527	2146	2	2	2	0	0	0	0	0	aligned	Positive	2	233	2294	0	0	1	2	5	3420	381	2	2	2146
519a3a22da300c3793000001	519c7162da300c019f000003	0	2514	2507	0	0	0	0	0	0	0	0	no positions	n/a	0	2514	0	0	0	0	1	3	3430	7	1	0	-1
519a3a22da300c3793000001	519c7269da300c019f000004	1	2251	344	0	0	0	0	0	0	0	0	no positions	n/a	0	2251	0	0	0	0	1	11	3702	1907	1	0	-1

Project	ID	Destroyed	Lifespan	LastChange	Position Count	Compacted Position Count	Positive	PositiveRevoked	Negative	NegativeRevoked	Open	OpenRevoked	Final State	Final Decision	Deciders	Time In NoPositions	Time In Aligned	Time In Colliding	Time In Sealed	Consensus State Changes	Editors	Update count	Last update	Activity Time	Contributors	Deciders2	Time since last decision
519a3a22da300c3793000001	519c727eda300c02d7000003	1	2231	327	0	0	0	0	0	0	0	0	no positions	n/a	0	2231	0	0	0	0	1	13	3727	1904	1	0	-1
519a3a22da300c3793000001	519c7a8dda300c057c000001	0	167	160	0	0	0	0	0	0	0	0	no positions	n/a	0	167	0	0	0	0	1	4	5777	7	1	0	-1
519a3a22da300c3793000001	519c7b32da300c05a0000001	0	2	1	0	0	0	0	0	0	0	0	no positions	n/a	0	2	0	0	0	0	1	0	5936	1	1	0	-1
ex4	ex4-1	0	2513	0	0	0	0	0	0	0	0	0	no positions	n/a	1	2513	0	0	0	1	1	2	3238	2513	1	0	-1
ex4	ex4-2	0	2381	2206	0	0	0	0	0	0	0	0	no positions	n/a	1	2381	0	0	0	1	1	1	1032	175	1	0	-1
ex4	ex4-3	0	2072	2014	0	0	0	0	0	0	0	0	no positions	n/a	1	2072	0	0	0	1	1	1	1224	58	1	0	-1
ex4	ex4-4	0	1572	1477	0	0	0	0	0	0	0	0	no positions	n/a	1	1572	0	0	0	1	1	1	1761	95	1	0	-1
ex4	ex4-5	0	608	589	0	0	0	0	0	0	0	0	no positions	n/a	1	608	0	0	0	1	1	1	2649	19	1	0	-1
ex4	ex4-6	0	537	531	0	0	0	0	0	0	0	0	no positions	n/a	1	537	0	0	0	1	1	1	2707	6	1	0	-1
ex4	ex4-7	0	375	370	0	0	0	0	0	0	0	0	no positions	n/a	1	375	0	0	0	1	1	1	2868	5	1	0	-1
ex4	ex4-8	0	369	362	0	0	0	0	0	0	0	0	no positions	n/a	1	369	0	0	0	1	1	1	2876	7	1	0	-1
ex4	ex4-9	0	361	40	0	0	0	0	0	0	0	0	no positions	n/a	1	361	0	0	0	1	1	1	3198	321	1	0	-1
ex5	ex5-4	0	1216	1207	0	0	0	0	0	0	0	0	no positions	n/a	1	1216	0	0	0	1	1	1	857	9	1	0	-1
ex5	ex5-5	0	1206	1190	0	0	0	0	0	0	0	0	no positions	n/a	1	1206	0	0	0	1	1	1	874	16	1	0	-1
ex5	ex5-6	0	389	380	0	0	0	0	0	0	0	0	no positions	n/a	1	389	0	0	0	1	1	1	1684	9	1	0	-1
ex5	ex5-7	0	347	332	0	0	0	0	0	0	0	0	no positions	n/a	1	347	0	0	0	1	1	1	1732	15	1	0	-1
ex5	ex5-8	0	322	175	0	0	0	0	0	0	0	0	no positions	n/a	1	322	0	0	0	1	1	1	1889	147	1	0	-1
ex7	ex7-1	0	3602	3601	0	0	0	0	0	0	0	0	no positions	n/a	1	3602	0	0	0	1	1	1	356	1	1	0	-1
ex7	ex7-10	0	2509	2421	1	1	1	0	0	0	0	0	aligned	Positive	2	88	2421	0	0	2	2	1	1452	88	2	1	2421
ex7	ex7-11	0	2498	2337	2	2	2	0	0	0	0	0	aligned	Positive	3	41	2457	0	0	2	3	2	1620	161	3	2	2382
ex7	ex7-11	0	2498	2337	2	2	2	0	0	0	0	0	aligned	Positive	3	41	2457	0	0	2	3	2	1620	161	3	2	2382
ex7	ex7-12	0	2336	1874	1	1	1	0	0	0	0	0	aligned	Positive	2	462	1874	0	0	2	2	1	1662	462	2	1	1874
ex7	ex7-13	0	2291	2185	0	0	0	0	0	0	0	0	no positions	n/a	1	2291	0	0	0	1	1	1	1772	106	1	0	-1
ex7	ex7-14	0	2028	1914	1	1	1	0	0	0	0	0	aligned	Positive	2	114	1914	0	0	2	2	1	2043	114	2	1	1914
ex7	ex7-15	0	1748	1428	1	1	1	0	0	0	0	0	aligned	Positive	1	320	1428	0	0	2	1	1	2209	320	1	1	1428
ex7	ex7-16	0	1748	1686	1	1	1	0	0	0	0	0	aligned	Positive	2	62	1686	0	0	2	2	1	2209	62	2	1	1686
ex7	ex7-17	0	1748	1298	3	3	0	0	2	0	1	0	colliding	n/a	3	352	82	1314	0	3	3	1	2209	450	3	3	1298
ex7	ex7-18	0	1734	0	2	2	1	0	1	0	0	0	colliding	n/a	3	89	1645	0	0	3	3	1	2232	1734	3	2	0
ex7	ex7-19	0	1725	1529	2	2	1	0	1	0	0	0	colliding	n/a	3	133	63	1529	0	3	3	1	2238	196	3	2	1529
ex7	ex7-2	0	3600	3548	1	1	0	0	1	0	0	0	aligned	Negative	2	52	3548	0	0	2	2	1	365	52	2	1	3548
ex7	ex7-20	0	1710	1691	1	1	0	0	1	0	0	0	aligned	Negative	1	19	1691	0	0	2	1	1	2248	19	1	1	1691
ex7	ex7-22	0	1309	1018	1	1	1	0	0	0	0	0	aligned	Positive	2	291	1018	0	0	2	2	1	2654	291	2	1	1018
ex7	ex7-23	0	897	895	0	0	0	0	0	0	0	0	no positions	n/a	1	897	0	0	0	1	1	1	3062	2	1	0	-1
ex7	ex7-24	0	892	887	0	0	0	0	0	0	0	0	no positions	n/a	1	892	0	0	0	1	1	1	3070	5	1	0	-1
ex7	ex7-25	0	880	858	1	1	1	0	0	0	0	0	aligned	Positive	2	22	858	0	0	2	2	1	3080	22	2	1	858
ex7	ex7-26	0	824	683	1	1	1	0	0	0	0	0	aligned	Positive	2	141	683	0	0	2	2	2	3247	141	2	1	683
ex7	ex7-27	0	811	772	1	1	0	0	0	0	1	0	aligned	Open	1	39	772	0	0	2	1	1	3157	39	1	1	772
ex7	ex7-3	0	3589	3555	1	1	0	0	1	0	0	0	aligned	Negative	1	34	3555	0	0	2	1	1	375	34	1	1	3555
ex7	ex7-4	0	3518	3445	0	0	0	0	0	0	0	0	no positions	n/a	1	3518	0	0	0	1	1	2	512	73	1	0	-1
ex7	ex7-5	0	3458	3451	1	1	0	0	0	0	1	0	aligned	Open	1	7	3451	0	0	2	1	1	500	7	1	1	3451
ex7	ex7-6	0	3401	2514	3	3	2	0	0	0	1	0	colliding	n/a	4	201	211	2989	0	3	4	1	561	887	4	3	2514
ex7	ex7-7	0	3395	3362	1	1	1	0	0	0	0	0	aligned	Positive	2	33	3362	0	0	2	2	1	567	33	2	1	3362
ex7	ex7-8	0	3198	3182	0	0	0	0	0	0	0	0	no positions	n/a	1	3198	0	0	0	1	1	1	775	16	1	0	-1

Project	ID	Destroyed	Lifespan	LastChange	Position Count	Compacted Position Count	Positive	PositiveRevoked	Negative	NegativeRevoked	Open	OpenRevoked	Final State	Final Decision	Deciders	Time In NoPositions	Time In Aligned	Time In Colliding	Time In Sealed	Consensus State Changes	Editors	Update count	Last update	Activity Time	Contributors	Deciders2	Time since last decision
ex7	ex7-9	0	3175	3167	0	0	0	0	0	0	0	0	no positions	n/a	1	3175	0	0	0	1	1	1	790	8	1	0	-1
ex8	ex8-1	0	4035	3813	2	2	2	0	0	0	0	0	aligned	Positive	3	194	3841	0	0	2	3	1	163	222	3	2	3813
ex8	ex8-2	0	4031	3761	1	1	1	0	0	0	0	0	aligned	Positive	2	270	3761	0	0	2	2	1	166	270	2	1	3761
ex8	ex8-3	0	2745	2638	2	2	2	0	0	0	0	0	aligned	Positive	3	90	2655	0	0	2	3	1	1466	107	3	2	2638
ex8	ex8-4	0	2727	505	2	2	2	0	0	0	0	0	aligned	Positive	3	79	2648	0	0	2	3	2	3689	2222	3	2	2647
ex8	ex8-4	0	2727	505	2	2	2	0	0	0	0	0	aligned	Positive	3	79	2648	0	0	2	3	2	3689	2222	3	2	2647
ex8	ex8-5	0	504	288	1	1	1	0	0	0	0	0	aligned	Positive	2	216	288	0	0	2	2	1	3697	216	2	1	288
ex8	ex8-6	0	497	0	2	2	1	0	1	0	0	0	colliding	n/a	2	380	117	0	0	3	2	1	3710	497	2	1	0
ex9	ex9-1	0	4009	3993	0	0	0	0	0	0	0	0	no positions	n/a	1	4009	0	0	0	1	1	1	26	16	1	0	-1
ex9	ex9-2	0	3993	3985	0	0	0	0	0	0	0	0	no positions	n/a	1	3993	0	0	0	1	1	1	34	8	1	0	-1
ex9	ex9-3	0	3976	2180	0	0	0	0	0	0	0	0	no positions	n/a	1	3976	0	0	0	1	2	5	1839	1796	2	0	-1
ex9	ex9-3	0	3976	2180	0	0	0	0	0	0	0	0	no positions	n/a	1	3976	0	0	0	1	2	5	1839	1796	2	0	-1
ex9	ex9-3	0	3976	2180	0	0	0	0	0	0	0	0	no positions	n/a	1	3976	0	0	0	1	2	5	1839	1796	2	0	-1
ex9	ex9-4	0	3701	0	0	0	0	0	0	0	0	0	no positions	n/a	1	3701	0	0	0	1	2	2	4019	3701	2	0	-1
ex9	ex9-5	0	3697	3690	0	0	0	0	0	0	0	0	no positions	n/a	1	3697	0	0	0	1	1	1	329	7	1	0	-1
ex9	ex9-6	0	22	22	0	0	0	0	0	0	0	0	no positions	n/a	1	22	0	0	0	1	1	0	3997	0	1	0	-1

Table A.3. Micro-metric values matrix for the design alternatives

Bibliography

[AAE+14] Tim Aerdts, Stefan Arians, Vadim Emrich, Michael Klingen, Ben Ripkens, and Theo Rutten. Open decision repository, 2014. `https://code.google.com/p/opendecisionrepository/`.

[AKL+07] Paris Avgeriou, Philippe Kruchten, Patricia Lago, Paul Grisham, and Dewayne E. Perry. Sharing and reusing architectural knowledge - architecture, rationale, and design intent. In *29th International Conference on Software Engineering (ICSE 2007)*, pages 109–110, 2007.

[ANP12] Saeed Aghaee, Marcin Nowak, and Cesare Pautasso. Reusable decision space for mashup tool design. In Simone Diniz Junqueira Barbosa, José Creissac Campos, Rick Kazman, Philippe A. Palanque, Michael D. Harrison, and Steve Reeves, editors, *ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS'12, Copenhagen, Denmark - June 25 - 28, 2012*, pages 211–220. ACM, 2012.

[BCK03] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, third edition, 2003.

[BCMM08] Janet E. Burge, John M. Carroll, Raymond McCall, and Ivan Mistrìk. *Rationale-Based Software Engineering*. Springer, 2008.

[BCR94] Victor Basili, Gianluigi Caldiera, and Dieter H. Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.

[BDLvV09] Muhammad Ali Babar, Torgeir Dingsøyr, Patricia Lago, and Hans van Vliet. *Software Architecture Knowledge Management - Theory and Practice*. Springer, 2009.

[BGBG95] Ronald M. Baecker, Jonathan Grudin, William Buxton, and Saul Greenberg. *Readings in Human-Computer Interaction: Towards the Year 2000*. Morgan-Kaufmann, 1995.

[BGJ05] Muhammad Ali Babar, Ian Gorton, and D. Ross Jeffery. Capturing and using software architecture knowledge for architecture-based software development. In *2005 NASA / DoD Conference on Evolvable Hardware, 29 June - 1 July 2005, Washington, DC*, 2005.

[BHS07a] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing*. Wiley, 2007.

[BHS07b] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-Oriented Software Architecture Volume 5: On Patterns and Pattern Languages*. Wiley, 2007.

[BM05] Felix Bachmann and Paulo Merson. Experience using the web-based tool wiki for architecture documentation. Technical Report SEI-2005-TN-041, Carnegie Mellon University, Software Engineering Institute, 2005.

[BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. Wiley, 1996.

[Boe00] Barry W. Boehm. Requirements that handle ikiwisi, cots, and rapid change. *IEEE Computer*, 33(7):99–102, 2000.

[Boo06] Grady Booch. On design. Blog, March 2006. `https://www.ibm.com/developerworks/community/blogs/gradybooch/entry/on_design?lang=en`.

[Bos04] Jan Bosch. Software architecture: The next step. In *First European Workshop on Software Architecture*, volume 3047 of *Lecture Notes in Computer Science*, pages 194–199. Springer, St Andrews, UK 2004.

[Bro10] Frederick P. Brooks. *The Design of Design: Essays from a Computer Scientist*. Addison-Wesley, 2010.

[Bur05] Janet E. Burge. *Software Engineering Using design RATionale*. PhD thesis, Worcester Polytechnic Institute, 2005.

[Cap09] Rafael Capilla. Embedded design rationale in software architecture. In *Joint Working IEEE/IFIP Conference on Software Architecture 2009 and European Conference on Software Architecture 2009, WICSA/ECSA 2009, Cambridge, UK, 14-17 September 2009*, pages 305–308. IEEE, 2009.

[CB88] Jeff Conklin and Michael L. Begeman. gibis: a hypertext tool for exploratory policy discussion. In *Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, CSCW '88, pages 140–152, 1988.

[CDN10] Rafael Capilla, Juan C. Dueñas, and Francisco Nava. Viability for codifying and documenting architectural design decisions with tool support. *Journal of Software Maintenance*, 22(2):81–119, 2010.

[CdVL10] Viktor Clerc, Edwin de Vries, and Patricia Lago. Using wikis to support architectural knowledge management in global software development. In *SHARK '10: Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge*, pages 37–43, 2010.

[Cho06] Chun Wei Choo. *The Knowing Organization: How Organizations Use Information to Construct Meaning, Create Knowledge and Make Decisions.* Oxford University Press, second edition edition, 2006.

[Cle07] Paul C. Clements. An economic model for software architecture decisions. In *ESC '07: Proceedings of the First International Workshop on The Economics of Software and Computation*, 2007.

[Cle11] Viktor Clerc. *Architectural Knowledge Management in Global Software Development*. PhD thesis, Free University of Amsterdam, 2011.

[CLvV07] Viktor Clerc, Patricia Lago, and Hans van Vliet. The architect's mindset. In Sven Overhage, Clemens A. Szyperski, Ralf Reussner, and Judith A. Stafford, editors, *Software Architectures, Components, and Applications, Third International Conference on Quality of Software Architectures, QoSA 2007, Medford, MA, USA, July 11-23, 2007, Revised Selected Papers*, volume 4880 of *Lecture Notes in Computer Science*, pages 231–249. Springer, 2007.

[CNPD06] Rafael Capilla, Francisco Nava, Sandra Pérez, and Juan C. Dueñas. A web-based tool for managing architectural design decisions. *ACM SIGSOFT Software Engineering Notes*, 31(5), 2006.

[Con68] Melvin E. Conway. How do committees invent? *Datamation magazine*, 1968.

[Con05] Jeff Conklin. *Dialogue Mapping: Building Shared Understanding of Wicked Problems*. Wiley, 1st edition, 2005.

[Coy05] Richard Coyne. Wicked problems revisited. *Design Studies*, 26(1):5 – 17, 2005.

[CS63] Donald T. Campbell and Julian C. Stanley. *Experimental and quasi-experimental designs for research*. Houghton Mifflin Company, 1963.

[CSM08] Xiaofeng Cui, Yanchun Sun, and Hong Mei. Towards automated solution synthesis and rationale capture in decision-centric architecture design. In *Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture WICSA '08*, pages 221–230, 2008.

[Dal11] Kimiz Dalkir. *Knowledge Management in Theory and Practice*. MIT Press, second edition, 2011.

[dBFL$^+$07] Remco de Boer, Rik Farenhorst, Patricia Lago, Hans van Vliet, Viktor Clerc, and Anton Jansen. Architectural knowledge: Getting to the core. In *Proceedings of the 3rd Internationale Conference on the Quality of Software Architectures (QoSA)*, pages 197–214. Springer LNCS, 2007.

[DeM86] Tom DeMarco. *Controlling Software Projects: Management, Measurement, and Estimates*. Prentice Hall, 1986.

[dGJKK12] Thijmen de Gooijer, Anton Jansen, Heiko Koziolek, and Anne Koziolek. An industrial case study of performance and cost design space exploration. In David R. Kaeli, Jerry Rolia, Lizy K. John, and Diwakar Krishnamurthy, editors, *Third Joint WOSP/SIPEW International Conference on Performance Engineering, ICPE'12, Boston, MA, USA - April 22 - 25, 2012*, pages 205–216. ACM, 2012.

[Dur11] Zoya Durdik. An architecture-centric approach for goal-driven requirements elicitation. In *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13rd European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9, 2011*, pages 384–387, 2011.

[EC09] Peter Eeles and Peter Cripps. *The Process of Software Architecting*. Pearson, 2009.

[Ede05] Amnon H. Eden. Strategic versus tactical design. In *38th Hawaii International Conference on System Sciences (HICSS-38 2005), 3-6 January 2005, Big Island, HI, USA*, 2005.

[EG89] Clarence A. Ellis and Simon J. Gibbs. Concurrency control in groupware systems. In James Clifford, Bruce G. Lindsay, and David Maier, editors, *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989.*, pages 399–407. ACM Press, 1989.

[EHK06] Amnon H. Eden, Yoram Hirshfeld, and Rick Kazman. Abstraction classes in software design. *IEEE Software*, 153:163–182, 2006.

[EK03] Amnon H. Eden and Rick Kazman. Architecture, design, implementation. In Lori A. Clarke, Laurie Dillon, and Walter F. Tichy, editors, *Proceedings of the 25th International Conference on Software Engineering (ICSE-03), May 3-10, 2003, Portland, Oregon, USA*, pages 149–159. IEEE Computer Society, 2003.

[End95] Mica R. Endsley. Thoward a theory of sitaution awareness in dynamic systems. *Human Factors*, 37(1):32–64, March 1995.

[End00] Mica R. Endsley. Theoretical underpinnings of situation awareness: a critical review. In M. R. Endsley and D. J. Garland, editors, *Situation Awareness Analysis and Measurement*, pages 1–24, Mahwah, NJ, USA, 2000. Lawrence Erlbaum Associates.

[FCKK11] Davide Falessi, Giovanni Cantone, Rick Kazman, and Philippe Kruchten. Decision-making techniques for software architecture design: A comparative survey. *ACM Computing Surveys*, 43(4):33, 2011.

[FdB09] Rik Farenhorst and Remco C. de Boer. *Architectural Knowledge Management: Supporting Architects and Auditors*. PhD thesis, Vrije Universiteit Amsterdam, 2009.

[FJFH09] Rik Farenhorst and Hans van Vliet Johan F. Hoorn, Patricia Lago. What architects do what they need to share knowledge. Technical report, VU University Amsterdam, 2009.

[FLvV07a] Rik Farenhorst, Patricia Lago, and Hans van Vliet. Eagle: Effective tool support for sharing architectural knowledge. *Int. J. Cooperative Inf. Syst.*, 16(3/4):413–437, 2007.

[FLvV07b] Rik Farenhorst, Patricia Lago, and Hans van Vliet. Prerequisites for successful architectural knowledge sharing. In *18th Australian Software Engineering Conference (ASWEC 2007), April 10-13, 2007, Melbourne, Australia*, pages 27–38, 2007.

[Fow02] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002.

[FS99] Martin Fowler and Kendall Scott. *UML Distilled: A Brief Guide to the Standard Object Modeling Language (2nd Edition)*. Addison-Wesley, 1999.

[FWC93] J. Favela, A. Wong, and A. Charkavatrthy. Supporting collaborative engineering design. *Engineering with computers*, 9:125–132, 1993.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[GIZ08] David Greenspan, Aaron Iba, and J.D. Zamfirescu. Etherpad, 2008. `http://etherpad.org/`.

[Goo13] Google. Google Docs, 2013. `http://docs.google.com`.

[Gru94] Jonathan Grudin. Computer-supported cooperative work: History and focus. *IEEE Computer*, 27(5):19–26, 1994.

[GS94] David Garlan and Mary Shaw. An introduction to software architecture. Technical report, Software Engineering Institute, Carnegie Mellon University, 1994.

[Han05] Sven Ove Hansson. *Decision Theory: A Brief Introduction*. Department of Philosophy and the History of Technology, Royal Institute of Technology (KTH), Stockholm, 2005.

[HGS$^+$13] Matthias Heinrich, Franz Josef Grüneberger, Thomas Springer, Philipp Hauer, and Martin Gaedke. Gawi: A comprehensive workspace awareness library for collaborative web applications. In Florian Daniel, Peter Dolog, and Qing Li, editors, *13th International*

Conference on Web Engineering, ICWE 2013, Aalborg, Denmark, July 8-12, 2013. Proceedings, volume 7977 of *Lecture Notes in Computer Science*, pages 482–485. Springer, 2013.

[HKN$^+$07] Christine Hofmeister, Philippe Kruchten, Robert L. Nord, Henk Obbink, Alexander Ran, and Pierre America. A general model of software architecture design derived from five industrial approaches. *Journal of Systems and Software*, 80(1):106 – 126, 2007.

[HL08] Lile Hattori and Michele Lanza. On the nature of commits. In *23rd IEEE/ACM International Conference on Automated Software Engineering - Workshop Proceedings (ASE Workshops 2008), 15-16 September 2008, L'Aquila, Italy*, pages 63–71. IEEE, 2008.

[HP96] Randy Y. Hirokawa and Marshall Scott Poole. *Communication and Group Decision Making*. SAGE Publications, Inc, 2nd edition, 1996.

[HRLT09] Carol L. Hoover, Mel Rosso-Llopart, and Gil Taran. *Evaluating Project Decisions: Case Studies in Software Engineering*. Addison-Wesley, 2009.

[HW03] Gregor Hophe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003.

[HWBYZ13] Gregor Hohpe, Rebecca Wirfs-Brock, Joseph W. Yoder, and Olaf Zimmermann. Twenty years of patterns' impact. *IEEE Software*, 30(6):88, 2013.

[Inm02] W. H. Inmon. *Building the Data Warehouse*. Wiley, 2002.

[ISO91] ISO 9126. Information technology - software product evaluation - quality characteristics and guidelines for their use, 1991.

[ISO11] ISO 42010. ISO/IEC 42010 – Systems and software engineering – architecture description, 2011.

[Jac95] Michael Jackson. *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*. Addison-Wesley Professional, 1995.

[JAvdV09] Anton Jansen, Paris Avgeriou, and Jan Salvador van der Ven. Enriching software architecture documentation. *Journal of Systems and Software*, 82(8):1232 – 1248, 2009.

[JB05] Anton Jansen and Jan Bosch. Software architecture as a set of architectural design decisions. In *Fifth Working IEEE / IFIP Conference on Software Architecture (WICSA 2005), 6-10 November 2005, Pittsburgh, Pennsylvania, USA*, pages 109–120, 2005.

[Jos09] Andrew Josey. *TOGAF Version 9 Enterprise Edition*. The Open Group, January 2009.

[JvdVAH07] Anton Jansen, Jan Salvador van der Ven, Paris Avgeriou, and Dieter K. Hammer. Tool support for architectural decisions. In *Sixth Working IEEE / IFIP Conference on Software Architecture (WICSA 2007), 6-9 January 2005, Mumbai, Maharashtra, India*, page 4. IEEE Computer Society, 2007.

[KCD09] Philippe Kruchten, Rafael Capilla, and Juan Carlos Dueas. The decision view's role in software architecture practice. *IEEE Software*, 26(2):36–42, 2009.

[KG14] Heiko Koziolek and Thomas Goldschmidt. Tool-driven technology transfer to support software architecture decisions. In *Software Engineering 2014, Fachtagung des GI-Fachbereichs Softwaretechnik, 25. Februar - 28. Februar 2014, Kiel, Deutschland*, volume 227 of *LNI*, pages 159–164. GI, 2014. `https://decisions.codeplex.com/`.

[KJ04] Michael Kircher and Prashant Jain. *Pattern-Oriented Software Architecture Volume 3: Patterns for Resource Management*. Wiley, 2004.

[KKC00] Rick Kazman, Mark Klein, and Paul Clements. ATAM: Method for architecture evaluation. Technical Report CMU/SEI-2000-TR-004, CMU SEI, August 2000.

[Kle99] Gary Klein. *Sources of Power*. MIT Press, 1999.

[KLvV06] Philippe Kruchten, Patricia Lago, and Hans van Vliet. Building up and reasoning about architectural knowledge. *Quality of Software Architectures*, pages 43–58, 2006.

[KOS06] Philippe Kruchten, J. Henk Obbink, and Judith A. Stafford. The past, present, and future for software architecture. *IEEE Software*, 23(2):22–30, 2006.

[Kru95] P.B. Kruchten. The 4+1 view model of architecture. *Software, IEEE*, 12(6):42–50, Nov 1995.

[KS08] Daniel Kroening and Ofer Strichman. *Decision Procedures, An Algorithmic Point of View*. Springer, 2008.

[Kva00] Thomas Kvan. Collaborative design: what is it? *Automation in construction*, 9(4):409–415, 2000.

[Lag09] Patricia Lago. *Establishing and Managing Knowledge Sharing Networks*, chapter 7, pages 113–130. Springer, 2009.

[LC01] Bo Leuf and Ward Cunningham. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley Professional, 2001.

[Lee89] Jintae Lee. Decision representation language (DRL) and its support environment. *A.I. Working Paper No. 325*, 1989.

[Lee97] Jintae Lee. Design rationale systems: Understanding the issues. *IEEE Intelligent Systems*, 12(3):78–85, 1997.

[LG86] Barbara Liskov and John Guttag. *Abstraction and specification in program development*. MIT Press, 1986.

[LJA09] Peng Liang, Anton Jansen, and Paris Avgeriou. Knowledge architect: A tool suite for managing software architecture knowledge. Technical report, Software Engineering and Architecture (SEARCH) Group University of Groningen, 2009.

[LJA10] Peng Liang, Anton Jansen, and Paris Avgeriou. *Collaborative Software Engine*, chapter Collaborative Software Architecting through Knowledge Sharing, pages 343–367. Springer, 2010.

[LM06] Michele Lanza and Radu Marinescu. *Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Springer, 2006.

[LTZ13] Ioanna Lytra, Huy Tran, and Uwe Zdun. Supporting consistency between architectural design decisions and component models through reusable architectural knowledge transformations. In Khalil Drira, editor, *Software Architecture - 7th European Conference (ECSA2013), Montpellier, France, July 1-5*, volume 7957 of *LNCS*, pages 224–239, 2013.

[Mah06] Michael Mahemoff. *Ajax Design Patterns*. O'Reilly Media, 2006.

[MT00] Nenad Medvidovic and Richard N. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Trans. Software Eng.*, 26(1):70–93, 2000.

[MTK+14] Christian Manteuffel, Dan Tofan, Heiko Koziolek, Thomas Goldschmidt, and Paris Avgeriou. Industrial implementation of a documentation framework for architectural decisions. In *2014 IEEE/IFIP Conference on Software Architecture, WICSA 2014, Sydney, Australia, April 7-11, 2014*, pages 225–234. IEEE Computer Society, 2014.

[MZ11] Christoph Miksovic and Olaf Zimmermann. Architecturally significant requirements, reference architecture, and metamodel for knowledge management in information technology services. In *Proceedings of the 2011 Ninth Working IEEE/IFIP Conference on Software Architecture*, WICSA '11, pages 270–279, Washington, DC, 2011.

[NHC07] Rajiv Nag, Donald C. Hambrick, and Ming-Jer Chen. What is strategic management, really? inductive derivation of a consensus definition of the field. *Strategic Management Journal*, 28(9):935–955, 2007.

[NP11] Marcin Nowak and Cesare Pautasso. Goals, questions and metrics for architectural decision models. In *Proceedings of the 6th Workshop on Sharing and reusing architectural knowledge SHARK '11*, Waikiki, HA, 2011.

[NP13] Marcin Nowak and Cesare Pautasso. Team situational awareness and architectural decision making with the software architecture warehouse. In Khalil Drira, editor, *7th European Conference on Software Architecture, ECSA 2013, Montpellier, France, July 1-5, 2013. Proceedings*, volume 7957 of *LNCS*, pages 146–161. Springer, 2013.

[NPZ10] Marcin Nowak, Cesare Pautasso, and Olaf Zimmerman. Architectural decision modeling with reuse: Challenges and opportunities. In *Proceedings of the 5th Workshop on Sharing and reusing architectural knowledge SHARK '10*, Cape Town, South Africa, 2010.

[NT95] Ikujiro Nonaka and Hirotaka Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995.

[Nus01] Bashar Nuseibeh. Weaving together requirements and architectures. *IEEE Computer*, pages 115–119, 2001.

[oEE00] Institute of Electrical and Electronics Engineers. Recommended practice for architectural description of software-intensive systems, 2000.

[Par09] David Lorge Parnas. Document based rational software development. *Knowl.-Based Syst.*, 22(3):132–141, 2009.

[PB88] C. Potts and G. Bruns. Recording the reasons for design decisions. In *Proceedings of the 10th International Conference on Software Engineering*, pages 418 –427, apr 1988.

[PIPWJ08] Witold Pedrycz, Nikhil Ichalkaranje, Gloria Phillips-Wren, and Lakhmi C. Jain. *Intelligent Decision Making: An AI-Based Approach*. Springer, 2008.

[PW92] Dewayne E. Perry and Alexander L. Wolf. Foundations for the study of software architecture. *SIGSOFT Softw. Eng. Notes*, 17:40–52, October 1992.

[PZL08] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. big web services: Making the right architectural decision. In *17th International World Wide Web Conference WWW2008*, pages 805–814, Beijing, China, April 2008.

[Rit72] Horst Rittel. On the Planning Crisis: Systems Analysis of the 'First and Second Generations'. *Bedriftskonomen*, 34:390–396, 1972.

[Rit06] Tom Ritchey. General morphological analysis a general method for non-quantified modelling. *Swedish Morphological Society*, 2002–2006.

[Rob00] Nancy Roberts. Wicked problems and network approaches to resolution. *International Public Management Review*, 1, 2000.

[ROJ90] Spencer Rugaber, Stephen B. Ornburn, and Richard J. Leblanc Jr. Recognizing design decisions in programs. *IEEE Software*, 1990.

[RW73] Horst W. J. Rittel and Melvin M. Webber. Dilemmas in a general theory of planning. *Policy Science*, 4:155–169, 1973.

[SCB14] Stephan Sehestedt, Chih-Hong Cheng, and Eric Bouwers. Towards quantitative metrics for architecture models. In Anna Liu, John Klein, and Antony Tang, editors, *WICSA Companion*, page 5. ACM, 2014.

[SCC01] William R. Shadish, Thomas D. Cook, and Donald T. Campbell. *Experimental and Quasi-experimental designs for generalized casual inference*. Houghton Mifflin Co., 2001.

[SEI10] CMU SEI. Software engineering institute, 2010. Community Software Architecture Definitions `http://www.sei.cmu.edu/architecture/start/community.cfm`.

[Sel11] Howard J. Seltman. *Experimental Design and Analysis*. Pearson Prentice Hall, Upper Saddle River, New Jersey, 2011. `http://www.stat.cmu.edu/~hseltman/309/Book/PrefTOC.pdf`.

[SHL08] Weiming Shen, Qi Hao, and Weidong Li. Computer supported collaborative design: Retrospective and perspective. *Computers in Industry*, 59(9):855–862, 2008.

[Sim97] Herbert A. Simon. *Administrative Behavior: A Study of Decision-making Processes in Administrative Organizations: A Study of Decision-making Processes in Administrative Organisations*. The Free Press, 1997.

[SLK09a] Mojtaba Shahin, Peng Liang, and Mohammad-Reza 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.

[SLK09b] Mojtaba Shahin, Peng Liang, and Mohammad Reza Khayyambashi. A survey of architectural design decision models and tools. Technical Report SBU-RUG-2009-SL-01, Sheikh Bahaei University & University of Groningen, Jun 2009.

[SNH95] Dilip Soni, Robert L. Nord, and Christine Hofmeister. Software architecture in industrial applications. In Dewayne E. Perry, Ross Jeffrey, and David Notkin, editors, *17th International Conference on Software Engineering, Seattle, Washington, USA, April 23-30, 1995, Proceedings.*, pages 196–207. ACM, 1995.

[SSRB00] Douglas Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann. *Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects*. Wiley, 2000.

[SSS$^+$01] Albert Selvin, Simon Buckingham Shum, Maarten Sierhuis, Jeff Conklin, Beatrix Zimmermann, Charles Palus, Wilfred Drath, David Horth, John Domingue, Enrico Motta, and Gangmin Li. Compendium: Making meetings into knowledge events. In *Knowledge Technologies*, Austin, TX, March 2001.

[Ste46] S. S. Stevens. On the theory of scales of measurement. *Science*, 103:677–680, 1946.

[SZP07] Nelly Schuster, Olaf Zimmermann, and Cesare Pautasso. Adkwik: Web 2.0 collaboration system for architectural decision engineering. In *Proc. of the International Conference on Software Engineering & Knowledge Engineering (SEKE'2007)*, 2007.

[TA05] Jeff Tyree and Art Akerman. Architecture decisions: Demystifying architecture. *IEEE Software*, 22(2):19–27, 2005.

[Tan90] Steven L. Tanimoto. Viva: A visual language for image processing. *J. Vis. Lang. Comput.*, 1(2):127–139, June 1990.

[Tan07] Antony Tang. *A Rationale-based Model for Architecture Design Reasoning*. PhD thesis, Swinburne University of Technology, February 2007.

[Tau06] Gadi Taubenfeld. *Synchronization Algorithms and Concurrent Programming*. Pearson / Prentice Hall, 2006.

[TGA13] Dan Tofan, Matthias Galster, and Paris Avgeriou. Difficulty of architectural decisions - a survey with professional architects. In *ECSA*, volume 7957 of *Lecture Notes in Computer Science*, pages 192–199. Springer, 2013.

[TJH07] Antony Tang, Yan Jin, and Jun Han. A rationale-based architecture model for design traceability and reasoning. *Journal of Systems and Software*, 80(6):918–934, 2007.

[TMD09] Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy. *Software Architecture - Foundations, Theory and Practice*. Wiley, 2009.

[Tra95] Will Tracz. *Confessions of a Used Program Salesman*. Addison-Wesley, 1995.

[vHA10] Uwe van Heesch and Paris Avgeriou. Naive architecting - understanding the reasoning process of students - a descriptive survey. In Muhammad Ali Babar and Ian Gorton, editors, *Software Architecture, 4th European Conference, ECSA 2010, Copenhagen, Denmark, August 23-26, 2010. Proceedings*, volume 6285 of *Lecture Notes in Computer Science*, pages 24–37. Springer, 2010.

[vHA11] Uwe van Heesch and Paris Avgeriou. Mature architecting - a survey about the reasoning process of professional architects. In *9th Working IEEE/IFIP Conference on Software Architecture, WICSA 2011, Boulder, Colorado, USA, June 20-24, 2011*, pages 260–269. IEEE Computer Society, 2011.

[vHAH12a] Uwe van Heesch, Paris Avgeriou, and Rich Hilliard. A documentation framework for architecture decisions. *Journal of Systems and Software*, 85(4):795–820, 2012.

[vHAH12b] Uwe van Heesch, Paris Avgeriou, and Rich 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.

[vL08] Axel van Lamsweerde. Requirements engineering: from craft to discipline. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2008, Atlanta, Georgia, USA, November 9-14, 2008*, pages 238–249, 2008.

[Wol93] Michael Wolfe. New architectures for wisiwyswiwswys (what i see is what you see when i want to see what you see). In Clyde W. Holsapple and Andrew B. Whinston, editors, *Recent Developments in Decision Support Systems*, volume 101 of *NATO ASI Series*, pages 361–379. Springer Berlin Heidelberg, 1993.

[Zac87] John A. Zachman. A framework for information systems architecture (abstract of tutorial). In Salvatore T. March, editor, *Entity-Relationship Approach, Proceedings of the Sixth International Conference on Entity-Relationship Approach*, page 7. North-Holland, 1987.

[Zdu09] Uwe Zdun. Capturing design knowledge. *IEEE Software*, pages 25–27, March/April 2009.

[Zha02] Lei Zhang. *Knowledge Graph Theory and Structural Parsing*. PhD thesis, University of Twente, 2002.

[Zim09] Olaf Zimmermann. *An Architectural Decision Modeling Framework for Service-Oriented Architecture Design*. PhD thesis, Universität Stuttgart, 2009.

[Zim11] Olaf Zimmermann. Architectural decisions as reusable design assets. *IEEE Software*, 28(1):64–69, 2011.

[ZKL$^+$09] Olaf Zimmermann, Jana Koehler, Frank Leymann, Ronny Polley, and Nelly Schuster. Managing architectural decision models with dependency relations, integrity constraints, and production rules. *Journal of Systems and Software*, 82(8):1249 – 1267, 2009.