# Derivation of Domain-specific Architectural Knowledge Views from Governance and Security Compliance Metadata

Huy Tran, Ioanna Lytra, Uwe Zdun

Software Architecture Research Group
University of Vienna, Austria
E-Mail: firstname.lastname@univie.ac.at

## ABSTRACT

In the area of software architecture, in recent years, modeling design decisions is becoming more and more popular as a means to record architectural knowledge (AK) and capture the rationale of a design. Unfortunately, decision modeling is rather time-consuming and hence often forgotten in practice. In this paper, we propose that this problem can be avoided by utilizing domain-specific information that contains AK. We focus on domain-specific AK in the domain of business governance and security compliance. A novel approach is presented for recording the compliance concerns and extracting the corresponding AK. For this purpose we introduce a compliance meta-model and propose a mapping to the AK. Model-driven techniques are used as a supporting tool for generating AK documentation.

## Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*domain-specific architectural knowledge, decision vaporization, architectural view*

## Keywords

domain-specific; architectural knowledge; architectural design decision, view; compliance; process-driven SOA

## 1. INTRODUCTION

In recent years, software architecture is less and less seen as only the components and connectors constituting a system's principal design, and more and more as a set of principal design decisions governing a system [14, 8]. In this context, the idea to gather the Architectural Knowledge (AK) about a software system gets into the focus of the software architecture community. An important idea in this context is to not only document the components and connectors, but also the design rationale of the architecture. Hence, in this paper we use AK to refer to both the structural architecture information, e.g., modeled using components and connectors, as well as design rationale documentations.

Unfortunately, AK tends to evaporate as software systems evolve, with grave consequences for software development projects [9]. Consider the domain of governance and security compliance: Many security concerns in process-driven SOAs are implemented in many different parts of the system and hence scattered through the process models and service and backend source code. If the AK related to governance and security compliance is not properly documented, it is likely that future changes, unaware of such important design rationale, violate governance measures or security concern implementations.

A number of approaches have been proposed to generically solve this problem, for example based on text templates for AK [20] or based on meta-models to describe the AK [21, 9]. While these approaches work well in general, in the daily business AK capturing is considered as an afterthought or not at all [22]. Retrospective modeling of AK is often seen as a painful additional responsibility without many gains [22]. It is still unclear how to capture AK without introducing efforts that outweigh the benefits [2]. Hence, how to document, maintain, and evolve architectural knowledge in a way that is accepted by the average developer or designer and does get in the way of the daily work is one of the major obstacles for the adoption of AK documentation in practice today. The research works proposed so far manage to accurately document the AK, but they fall short in meeting this criterion. Apart from that, the tools for capturing architectural decisions that have been proposed in the literature so far ([2, 9]) are rather general and none of them focuses on domain-specific Architectural Decisions.

At the first glance, it might seem impossible to solve this obstacle of additional efforts, as any recording of a significant amount of extra information means a significant amount of extra work. In this paper we propose to circumvent this problem by focusing on domain-specific AK that must be recorded anyway by an organization. That is, many organizations must for completely other reasons document some information that contains AK. For example, in the domain of governance and security compliance, many companies have to document a lot of information for complying to regulations, such as Basel II, IFRS, the European Directive 95/46/EC Individual Protection, or the Sarbanes-Oxley


*SAC'13* March 18-22, 2013, Coimbra, Portugal.

Act, Basel II[1], IFRS[2], Tabaksblat[3], European Directive 95/46/EC Individual Protection[4], or the Sarbanes-Oxley Act[5], and in many cases design rationale has to be included in these documentations. If we can make sure that the architectural design rationale is accurately captured in the compliance documentation process, it is likely that AK evaporation can be prevented because the organizations are legally obliged to record this information (at least no extra costs are to be expected). This approach does surely not solve every problem related to AK documentation of governance and security compliance concerns. However, our hypothesis is that our approach has the potential to enable organizations to document a large part of their AK, related to governance and security compliance, at a very low extra cost. This would be a significant improvement to the current situation, and likely lead to a much higher adoption of AK documentation in practice.

The remainder of this paper is as follows. Section 2 introduces a motivating example in the banking sector. Section 3 introduces a Compliance Metadata model based on a view-based, model-driven approach that is used to link AK to system models and describes the mapping of the compliance metadata model to AK. We explain the overall tool architecture in Section 4 and revisit the motivating example in Section 5. Next, we discuss the related work in Section 6 and finally summarize our main contributions.

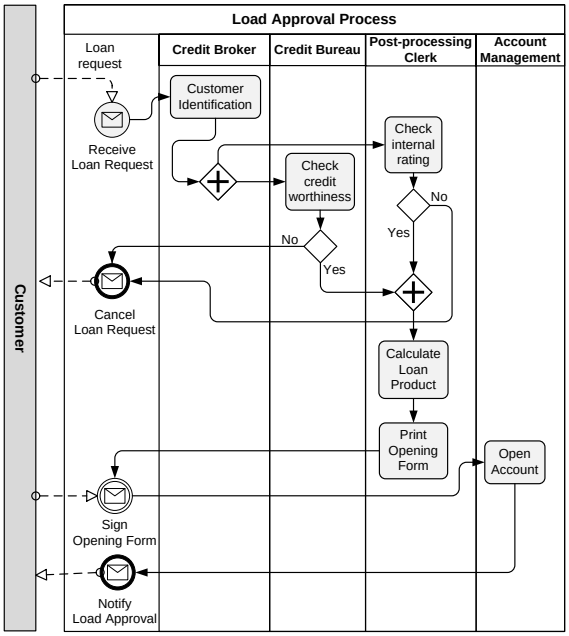


**Figure 1: The loan approval process**

## 2. MOTIVATING EXAMPLE

Let us consider a system for loan approval as a motivating example. The control flow of the main loan approval process is shown in Figure 1. This process has a number of activities that trigger services in various internal and external back-end systems such as the credit worthiness verification in the banking system, the internal rating system, risk and loan calculation system, and so on.

Consider further that the loan approval application needs to follow certain regulations and that any impact of these regulations on the system must be fully documented in order to be able to easily show to auditors the compliance to those regulations. For example, the "*EU Directive 95/46/EC Individual Protection*" requires the prevention of the abuse of individual-related data. To meet this criterion, we can modify the architecture to only use secure connectors, wherever individual-related data is transmitted. Modeling the architecture with secure connectors in the component & connector diagrams documents the architectural changes, but it does not record why these changes have happened, that the different changes of connectors in the architecture are caused by the same requirement, and the consequences of the changes such as slower transmission times. In other words important AK is not documented.

In this example, because of the need to document the prevention of the abuse of individual-related data fully to auditors, any (semi-)formalized solution of recording this information could be used as a source of AK. If this information could be leveraged, the improvement for the AK recording would be twofold: (1) stakeholders are motivated to record this information as it eases their work in other circumstances (the audit) and has hence a perceivable benefit for their work, and (2) managers are willing to invest in this compliance documentation as it would be costly not to convince the auditors of the compliance to the regulations that apply for a software system or even violating them.

[1] http://www.basel-ii-risk.com

[2] http://www1.icaew.co.uk/library/index.cfm?AUB=TB2I_25594

[3] http://www.fortis.com/governance/fortis_reference_codes_tabaksblat.asp

[4] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:HTML

[5] http://www.sec.gov/about/laws/soa2002.pdf

## 3. COMPLIANCE METADATA MODEL AND MAPPING TO AK

### 3.1 Background

In a process-driven, service-oriented architecture (SOA), business functionality is accomplished by using a process engine (or a workflow engine) to enact a business process that comprises coordinated activities invoking various services. A typical business process consists of various tangled concerns such as the control flow, data processing, service and process invocations, event handling, human interactions, transactions, business compliance, and so on. The entanglement of those concerns increases the complexity of process development and maintenance as the number of involved services and processes grow.

In order to deal with this complexity, we exploit the notion of architectural views [7] to describe the various SOA concerns [15] (see Figure 3). The view-based approach, which has been realized in terms of an Eclipse integrated development environment, namely, View-based Modeling Framework (VbMF), is used as a foundation for implementing the proposed Compliance and AK views.

VbMF provides a number of foundational (semi-)

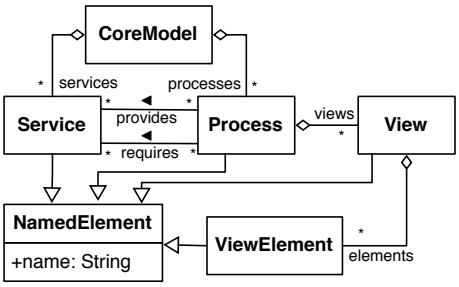


**Figure 2: Core model – the foundation for VbMF's extension and name-based integration**

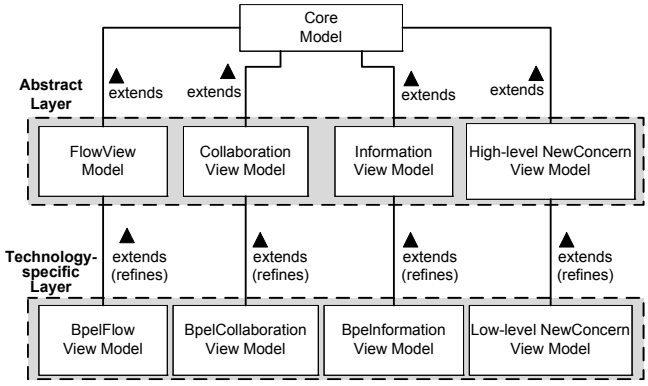


**Figure 3: Overview of the View-based Modeling Framework [15]**

formalizations for representing essential concerns of business processes such as the execution flow, service invocations, and data processing [15] along with view extension, integration, and code generation mechanisms. Thus, VbMF can be extended to capture other concerns, for instance, human interaction [6], transactions, fault handling [15], traceability [19], and so forth. The view extension mechanism shall be used in the subsequent section for devising the Compliance Metadata model. By using the view extension mechanism, a new concern can be integrated into our approach by using a corresponding *New-Concern-View* model that extends the basic concepts of the Core model and defines additional concepts of that concern. As a result, the Core model (see Figure 2) plays an important role in VbMF because it provides the basis for extending and integrating view models as well as establishing and maintaining the dependencies between view models [15, 16, 19, 18, 17].

## 3.2 Compliance Metadata model

We introduce a Compliance Metadata model to record the business compliance information as shown in Figure 4. The elements of this model that extend the *NamedElement* of the VbMF Core model, are used to describe compliance-specific concepts such as controls, risks, compliance documents, and compliance requirements, and so forth. The concepts of the Compliance Metadata model are independent from any application domains. In the context of our work, the Compliance Metadata model shall provide domain-specific AK for the domain of process-driven SOAs for business compliance: It describes which parts of the SOA, i.e. which services, processes, and activities have which roles in the compliance architecture and to which compliance requirements they are linked. This knowledge describes important architectural decisions, for instance, why certain services and processes are assembled in certain architectural configurations. Hence, they are not directly usable for recording AK, but as they are linked (in this model via the *NamedElement* from the Core model) to the architectural elements that compliance addresses, they could be utilized for AK recording. Based on the AK recorded in this model, we develop various transformation templates for generating architecture documentations and configurations.

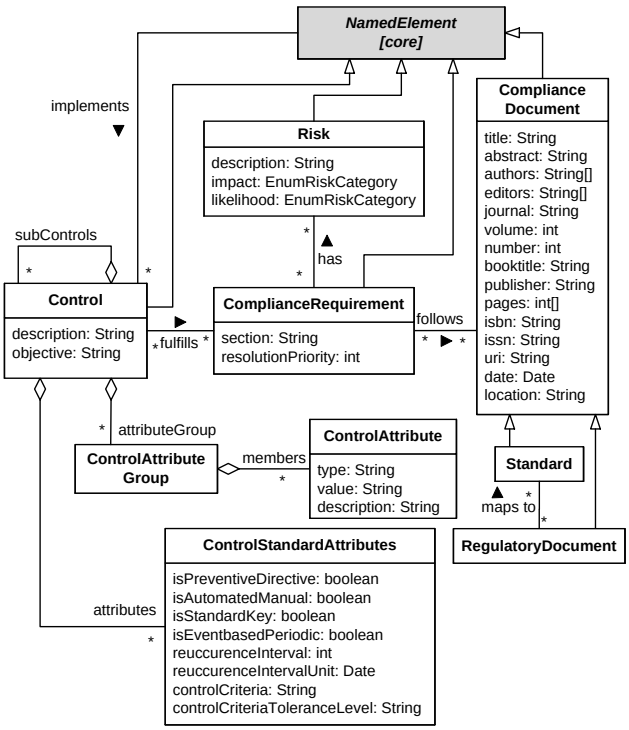


**Figure 4: The Compliance Metadata model**

| Compliance Metamodel Section | AK Text Template Section |
|---|---|
| ComplianceRequirement, ComplianceDocument | Issue |
| Status is not present, but a default can be selected. | Status |
| ComplianceDocument | Assumptions, constraints |
| Control, Attributes | Positions |
| Control, Attributes | Decision |
| Risks | Argument |
| Risks | Implications |
| Links via name-based matching | Related decisions |

**Table 1: Mapping concepts of the Compliance Metadata model to AK**

## 3.3 Mapping Compliance Metadata to AK

In order to capture the AK implied and contained by the compliance regulations, we use the architecture decision description template proposed by Tyree and Akerman [20]. Therefore, the completion is automated through the mapping between the elements of our Compliance Metadata model and the fields of the template. In Table 1, we can see that the elements roughly correspond to each other, hence our approach is feasible. Through this approach we are able to extract compliance documents as well as capture the architectural knowledge related to them, and therefore, avoiding the double effort or potential disagreements.

# 4. TOOL ARCHITECTURE

In Figure 5, we show a rough sketch of the proposed tool architecture and the interplay of the different parts. In this architecture, domain-specific languages (DSLs) are used to record this knowledge in a syntax understandable by the stakeholders who record the knowledge. Different textual and graphical syntaxes for the different architectural views are provided and supported via a tool suite. Model-driven development (MDD) techniques are used to generate (most of) the AK documentation as well as an AK View for the software system. The AK View can be derived from the domain-specific knowledge sources using the mapping illustrated in the previous section and can be realized by exploiting the model-driven transformation and reverse engineering techniques provided in VbMF [15, 16, 17]. Both kinds of AK can be stored in a central AK Repository. System parts that are generated via MDD solutions can be referenced from the AK views using view integration mechanisms [15, 18].

Architectural Decision for CrmSystem service connection

| | |
|---|---|
| Issue | Compliance of CrmSystem Service to the EU Directive 95/46/EC Individual Protection |
| Status | Accepted (default) |
| Assumptions, constraints | EU Directive 95/46/EC Individual Protection |
| Positions | Secure transmission of individual-related data through secure protocol connectors ... [*to be extracted from other model instances*] |
| Decision | Secure transmission of individual-related data through secure protocol connectors |
| Argument | High impact - low likelihood for abuse of individual-related data |
| Implications | High impact - low likelihood for abuse of individual-related data |
| Related decisions | - |

**Table 2: An excerpt of AK extracted from the loan approval process model's instance**

# 5. MOTIVATING EXAMPLE RESOLVED

To illustrate our approach, let us revisit the motivating example - the loan approval application presented in Section 2. As the essential concerns of the loan approval application such as the control flow, data handling, service invocations, and so on, are modeled by using VbMF views as mentioned in Section 3 and in [15], we will concentrate mainly on the compliance concern. Figure 6 depicts an instance of the Compliance Metadata model, which models the

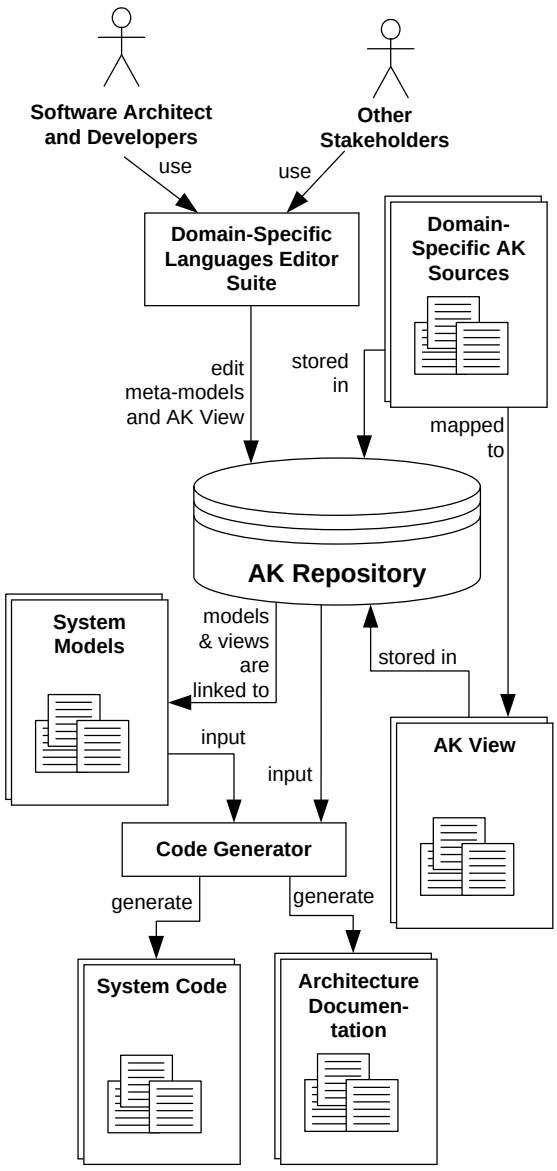


**Figure 5: Tool architecture**

regulation explained above for the loan approval application. This model instance records compliance metadata including a directive from the European Union on the protection of individuals with regard to the processing of personal data. In particular, the compliance control is implemented by the services of the loan approval application.

The *C1* compliance control indicating secure transmissions of personal data annotates a number of SOA elements such as *CreditBureau*, *AccountManagement*, and *CrmSystem* via the name-based matching mechanism [18]. The requirement *CR1* follows the legislative document and is associated with an *AbuseRisk*. The compliance control *C1* associates various *NamedElements* to the corresponding services modeled previously in VbMF.

Table 2 shows the AK documentation derived from this

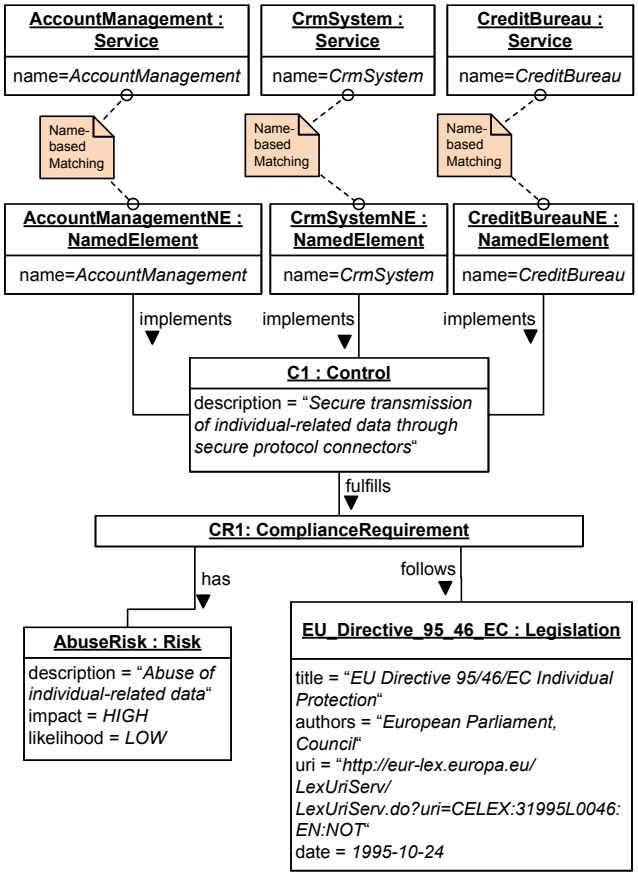


**Figure 6: An excerpt of the loan approval process model's instance**

model. It mainly extracts then information from Figure 6 straightforwardly. Some information, however, also requires a broader focus of many different compliance metamodel instances. For example, from the information in Figure 6 we could only derive a single *Position* entry. If our tool is used for many decisions for the same compliance requirement, we can also extract alternative positions, as they are chosen in other metadata model instances that use these alternative solutions as controls.

As this information is crucial for the project in terms of compliance documentation that is required for auditing purposes, it is likely maintained and kept up-to-date by the developers and users of the system. On the other hand, in this model important AK is also maintained: In particular the requirements for the process and the services that implement the control are recorded. That is, this information can be used to explain the architectural configuration of the process and the services connected via a secure protocols connector. In other words, it provides the compliance-related rationale for the design of this configuration. Moreover, in this particular case this documented AK is likely to be kept consistent with implemented system and the rationale of the architectural decision to use secure protocol connectors does not evaporate.

# 6. RELATED WORK

Much work on better support for codifying the AK has been done in the area of architectural decision modeling. Jansen and Bosch see software architecture as being composed of a set of design decisions [8]. They introduce a generic meta-model to capture decisions, including elements such as problems, solutions, and attributes of the AK. Another generic meta-model that is more detailed has been proposed by Zimmermann et al. [21, 22]. Tyree and Akermann proposed a highly detailed, generic template for architectural decision capturing [20]. A couple of other approaches are summarized in [1]. All these approaches share the problem of a significant extra effort necessary to record AK. Our approach addresses this problem by utilizing domain-specific AK that needs to be recorded for other reasons anyway.

Question, Options, and Criteria (QOC) diagrams [13] raise a design question, which points to the available solution options, and decision criteria are associated with the options. This way decisions can be modeled as such. Kruchten et al. extend this research by defining an ontology that describes the information needed for a decision, the types of decisions to be made, how decisions are being made, and their dependencies [12]. Falessi et al. present the Decision, Goal, and Alternatives framework to capture design decisions [3]. These approaches make decision modeling more formal and precise. However, being more formal and precise than for example the decision templates by [20] also means that these approaches require even more extra work to document the AK completely. Our approach attempts to provide a compromise solution.

Recently, Kruchten et al. extended these ideas with the notion of an explicit decision view [11]. A decision view provides an addition and complement to more traditional sets of architectural views and viewpoints: it gives an explanatory perspective that illuminates the reasoning process itself and not solely its results. Our approach follows this idea to combine the concepts of architectural views and AK to their mutual benefit. But our approach goes significantly beyond the approach by Kruchten et al. by defining not only the high-level views of the 4+1 view model [10], but also detailed technical views to allow for the model-driven generation of the software system.

A number of authors have so far investigated the relationships between patterns (or other reusable AK) and architectural decisions, which our approach makes explicit. Harrison et al. discuss the importance of this relation following qualitative empirical results [5]. Zimmermann et al. [21, 22] propose to integrate patterns in their reusable decision model. Harrison and Avgeriou present a study how patterns, tactics, and decisions interact [4]. These approaches use – like our approach– other AK sources as the basis for AK gathering, but in contrast to our approach generic, reusable knowledge is used. Our approach in contrast uses domain-specific compliance documentations.

The approach presented in this paper facilitates techniques and methods of the view-based modeling framework (VbMF) [15, 16, 17, 19, 18]. In particular, VbMF Core model is derived and extended, on the one hand, to describe concepts in the domain of business compliance. On the other hand, name-based matching view integration [15, 18] is used to link the aforementioned compliance-specific con-

cepts to the system models. The model transformation and the model-driven reverse engineering techniques provided in VbMF are leveraged for extracting and transforming compliance knowledge into corresponding architectural knowledge [15, 16, 17].

# 7. CONCLUSION AND FUTURE WORK

This paper presented a novel approach for capturing AK related to governance and security compliance using DSLs. We introduced a compliance meta-model to record business compliance information and we suggested a mapping to the corresponding AK as well as a tool architecture for recording this knowledge, generating AK documentation and AK Views, storing AK in a central AK repository and referencing system parts generated via MDD solutions from AK views. Future work includes the development of supporting tools to facilitate the experimentation with more complicated compliance rules and to evaluate our approach on a broader scale. Furthermore, the AK tool could be extended to support the capturing of other domain-specific architectural decisions.

## Acknowledgment

We thank the anonymous reviewers for their insightful and constructive comments for improving this paper. This work was partially supported by the EU's Seventh Framework Programme Project INDENICA (`http://www.indenica.eu`), Grant No. 257483.

# 8. REFERENCES

[1] M. A. Babar and P. Lago. Editorial: Design decisions and design rationale in software architecture. *J. Syst. Softw.*, 82:1195–1197, August 2009.

[2] R. Capilla, F. Nava, and C. Carrillo. Effort estimation in capturing architectural knowledge, 2008.

[3] D. Falessi, M. Becker, and G. Cantone. Design decicion rationale: Experiences and steps towards a more systematic approach. *SIG-SOFT Soft. Eng. Notes 31 – Workshop on Sharing and Reusing Architectural Knowledge*, 31(5), 2006.

[4] N. B. Harrison and P. Avgeriou. How do architecture patterns and tactics interact? a model and annotation. *J. Syst. Softw.*, 83:1735–1758, October 2010.

[5] N. B. Harrison, P. Avgeriou, and U. Zdun. Using patterns to capture architectural decisions. *IEEE Softw.*, July 2007.

[6] T. Holmes, H. Tran, U. Zdun, and S. Dustdar. Modeling Human Aspects of Business Processes - A View-Based, Model-Driven Approach. In *European Conf. Model Driven Architecture - Foundations and Applications (ECMDA-FA)*, pages 246–261, Berlin, Germany, 2008. Springer-Verlag.

[7] IEEE. IEEE Std 1471-2000: Recommended Practice for Architectural Description of Software-Intensive Systems, 2000.

[8] A. Jansen and J. Bosch. Software architecture as a set of architectural design decisions. In *WICSA '05: 5th IEEE/IFIP Conf. on Software Architecture*, pages 109–120. IEEE Computer Society, 2005.

[9] A. Jansen, J. van der Ven, P. Avgeriou, and D. Hammer. Tool support for architectural decisions. In *Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture*, Washington, DC, USA, 2007. IEEE Computer Society.

[10] P. Kruchten. The 4+1 View Model of Architecture. *IEEE Softw.*, 12(6):42–50, 1995.

[11] P. Kruchten, R. Capilla, and J. C. Duenas. The decision view's role in software architecture practice. *IEEE Softw.*, 26:36–42, 2009.

[12] P. Kruchten, P. Lago, and H. Vliet. Building up and reasoning about architectural knowledge. In C. Hofmeister, editor, *QoSA 2006 (Vol. LNCS 4214)*, pages 43–58, 2006.

[13] A. MacLean, R. M. Young, V. Bellotti, and T. Moran. Questions, options, and criteria: Elements of design space analysis. *Human-Computer Interaction*, 6(3–4):201–250, 1991.

[14] R. N. Taylor and A. van der Hoek. Software design and architecture the once and future focus of software engineering. In *2007 Future of Software Engineering*, FOSE '07, pages 226–243, Washington, DC, USA, 2007. IEEE Computer Society.

[15] H. Tran, U. Zdun, and S. Dustdar. View-based and Model-driven Approach for Reducing the Development Complexity in Process-Driven SOA. In *Int'l Conf. Business Process and Services Computing (BPSC)*, pages 105–124. Lecture Notes in Informatics (LNI), 2007.

[16] H. Tran, U. Zdun, and S. Dustdar. View-based Integration of Process-driven SOA Models At Various Abstraction Levels. In *Int'l Workshop on Model-Based Software and Data Integration (MBSDI)*, pages 55–66, Berlin, Germany, 2008. Springer CCIS.

[17] H. Tran, U. Zdun, and S. Dustdar. View-Based Reverse Engineering Approach for Enhancing Model Interoperability and Reusability in Process-Driven SOAs. In *Int'l Conf. Software Reuse (ICSR)*, pages 233–244. Springer, 2008.

[18] H. Tran, U. Zdun, and S. Dustdar. Name-based view integration for enhancing the reusability in process-driven SOAs. *Int'l Journal of Business Process Integration and Management*, 5(3):229—-239, 2011.

[19] H. Tran, U. Zdun, and S. Dustdar. VbTrace: using view-based and model-driven development to support traceability in process-driven SOAs. *J. Softw. & Syst. Model.*, 10(1):5–29, Nov. 2011.

[20] J. Tyree and A. Akerman. Architecture decisions: Demystifying architecture. *IEEE Softw.*, 22(19–27), 2005.

[21] O. Zimmermann, T. Gschwind, J. Kuester, F. Leymann, and N. Schuster. Reusable architectural decision models for enterprise application development. In *Quality of Software Architecture (QoSA) 2007*. Springer-Verlag, July 2007.

[22] O. Zimmermann, U. Zdun, T. Gschwind, and F. Leymann. Combining pattern languages and reusable architectural decision models into a comprehensive and comprehensible design method. *IEEE/IFIP Conf. on Software Architecture*, pages 157–166, 2008.