Towards Decision Centric Repository of Architectural Knowledge

Bartosz Michalik and Jerzy Nawrocki

Poznan University of Technology, Institute of Computing Science,
ul. Piotrowo 2, 60-965 Poznań, Poland
{Bartosz.Michalik, Jerzy.Nawrocki}@cs.put.poznan.pl

Abstract. Architectural design and design decisions are the key components of architectural knowledge. However, concerns, rationales, and risks should be also captured to prevent knowledge vaporization. So, how to deal with architectural knowledge in incremental knowledge refinement? We believe that usage of the knowledge repository system can support architecture evolution. In this paper, a model of knowledge repository is presented. In this model, the decision-centric approach is complemented with the architectural views to support indirect inter-relations between design decisions. Support for agile development was one of the key aspects of the model design, therefore knowledge vaporisation might be reduced.

Keywords: Architectural knowledge management, Architectural design decisions, Architectural description

1 Introduction

Software architecture has become an important concept in research and industry. Generally speaking software architectural description is a response to the customer concerns (requirements). Thus, this description serves as a primary vehicle for communication among stakeholders [1]. Architectural knowledge is a more general concept. The design decisions, reasoning behind them, and dependencies as well as architectural risks or trade-offs compose it. All this information help us to better understand the nature of the system being built. In addition, management of the knowledge helps to mitigate the risk of project failure, as various problems can be discovered in the early phase of software development.

However, design of software architecture is an iterative and a collaborative task. Often work on architecture begins when only a part of the requirements set is specified. Moreover, requirements can be modified at every stage of the development process. This is especially visible in the agile approaches to the software development.

Although multiple models, method and tools were proposed by researchers they failed confrontation with practice [2]. We believe that the benefits of the architectural knowledge acquisition can be observed when the functional knowledge management tool is available.

In this paper the model of architectural knowledge management repository is presented. In the model the design decisions are complemented with the architectural perspectives. It captures the architectural knowledge artifacts stated above. In addition we try to address the following problems:

- How to maintain the consistency of architectural descriptions in iterative architecture refinement?
- How to manage changes in requirements or architectural decisions?
- How to assess impact of change?

This paper is organized as follows. First, the current research in the field of architectural knowledge is discussed. Then, in Section 3 the existing models are compared. Next, in Section 4, architecture knowledge repository model is presented. In Section 5 the Ksantypa system is taken to illustrate some features of the model. Finally, the paper is concluded with the discussion of future works.

2 Related Work

One of the most popular architecture description (AD) models is probably the 4+1 views model proposed by Kruchten [3]. In that approach architecture is described using 5 views: logical (functional requirements), development (software module organisation), process (system performance, distribution, integrity), physical (system topology) and scenarios (cooperation of components presented in previous views). Recently, basing on the 4+1 views, the "3+1 views in 3D" was proposed [4], which brings the idea of presenting the software architecture in 3-dimensional space.

Multiple perspectives are also used by Clements [1] in the "Views and Beyond" approach. The work suggests components and connector (C&C) perspective for documenting the runtime entities and their interactions.

Finally, views are very important part of architecture model proposed in IEEE 1471 [5]. In the model architectural description consist of multiple views from which, each conform to given viewpoint. Architectural description identifies customers concerns and provides rationales.

Although perspective base approach became a reference point for model presented in IEEE 1471, the drift toward design decisions can be observed in its successor ISO 42010 [6]. In this proposal design description is defined as "a choice made that addresses one or more architecture related concerns and affects (directly or indirectly) the architecture."

More precise definition was suggested by Bosch [7]. In addition to foregoing definition, design decision is a set containing "rationales, the design rules, design constrains and additional requirements". Bosch defines software architecture as a set of the architectural design decisions. Every decision can bring a modification to the architectural model presented in C&C view. In his proposal the new view for presenting design decisions was provided. The need for capturing design decisions was also presented in earlier works [8,1,9]. Multiple works discuss design decision dependencies [10,2] and traceability [11].

Architecture knowledge management requires a tool support. The evaluation shows that there is a gap between tools for capturing rationale, software architecture and requirements [12]. The author claims that the gap can be closed with the tools that support design decisions concept. Several design decision oriented tools [13,14,15] are available. However, those are only research tools and to our best knowledge no industrially used tool exists.

3 Differences Between Description Models

The usage of architectural knowledge (AK) can follow several scenarios: incremental architectural review, review for specific concern, evaluation of impact, getting a rationale, study the chronology, adding or changing a decision etc. [16]. Therefore, architectural knowledge should be captured, described and stored in a form easy to interpret and extract. Architectural knowledge management system should support stakeholders in designing architecture that meets their business needs. However, architecture knowledge refinement is iterative task. During this process stakeholders face imprecise and unstable requirements. They may make wrong choices. Therefore traceability and consistency management should be supported in AK models.

The view (perspective) base approach presents architecture as a whole. With the use of views, different aspects of design are documented. However, researchers expose some drawbacks of this approach. Problems with conveying changes, traceability, documenting implications are identified [17]. Moreover design rules and constrains can be easily violated in view based models [7]. Another drawback is that the stakeholders' concerns are not easy to identify in this approach, as design decisions are not explicitly stated.

For documenting architecture in Model Driven Architecture(MDA) view based approach can be used. In MDA [18] architecture is presented at two levels of details. Platform Independent Model (PIM) describes a system without implementation details and Platform Specific Model (PSM) with full knowledge of the final implementation platform. From architectural knowledge management point of view this model helps to separate technical details from construction ones.

Design decision based approach directly addresses the architectural concerns. In other words requirements (architectural *What?*) immediately meet solutions (architectural *How?*). In this model all decisions are interrelated, so design space can be explicitly explored [7]. This also provides space for precise rationale capturing. Agile documenting is supported because an architect can focus on currently defined requirements and build appropriate solutions. Some papers present the architecture as a set of related design decisions. However this approach does not facilitate overall look on architecture. Moreover in incremental architecture refinement maintainability problems can occur.

Design decisions model provides coarse-grained view on the architecture whereas view model provides fine-grained overview. We believe that advantages of both the view and decision based approaches can be combined. In our model design decisions are first class entities, yet views are also used.

4 Architectural Knowledge Description Model

To our best knowledge, first attempt to combine architectural perspective and design decisions was published by Bosch [7]. Archium meta-model consists of *architecture model* (based on C&C view) and *design decisions model*. Each design decision contains *design fragment*, which describes an architecture modification (*architectural fragment* and *delta*) provided with the decision. However this model supports only two types of decisions proposed by Kruchten [19] .

In our model structural, behavioral and technical design decisions are supported. Model of architectural knowledge repository architectural description can be seen as two related layers(see Fig. 1). They present the design decision space and *architectural frame* (C&C view is used). Architectural frame provides additional axis on which relation between decisions can be observed and provides view on logical structure of the system.

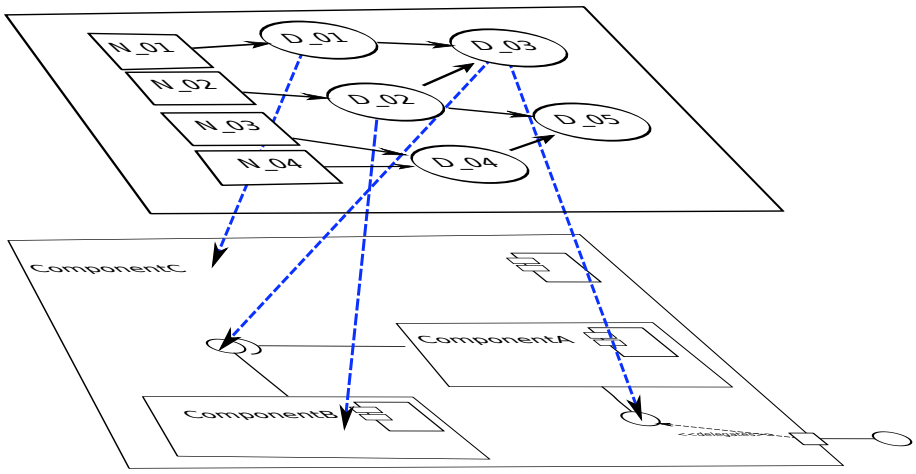

Fig. 1. Decisions mapped on architectural frame. In the top layer boxes represent requirements and ellipses design decisions. In the bottom architectural components are presented using UML components notation.

4.1 Design Decisions

To capture requirements we have adopted Bosch model, although design decision (DD) are treated as a part of architectural description in our proposal. We define design decisions using following elements:

Problem defines stakeholders' concern. Problems are referred by design decisions however this reference can be optional as DD may be consequence of others. Decision can solve multiple problems or be partial solution to one of them. It can address various problems but from our point of view non-functional and functional requirements are the most important. To handle non-functional requirements ISO25000 model [20] is recommended.

Rationale is the reasoning behind the decision being taken. It describes *why* specific solution was provided.

Decision type. Three types of design decisions were considered. Apart from suggested documentation manners textural representation of design decisions can be used in each type.

- Structural decisions are mostly used to describe components and connections captured in *architectural frame* (AF)(see Subsect. 4.2). Furthermore, they are used to document other structural aspect of the system as the deployment or project organisation.
- Behavioral decisions convey information about the dynamic aspects of the system. To document decisions of this type, process perspective can be used.
- Technical decision presents the software components used to solve the problem or integration guidance. This type of decisions together with architectural frame can be interpreted as PSM.

Description provides information about the solution for the given concern. Both, textual and view based descriptions are in use. This description can contain design guidance, rules and constrains.

Alternatives describes alternative solutions that were considered during the system design. Again, alternatives can be described with the use of views or textural information.

State contains information about the status of decision. It can be employed for maintainability purposes and change impact or repository consistency analysis. We suggest usage of four states *proposed*, *pending*, *approved* and *obsolete* although other sets of states can be used [14,19]. During knowledge repository refinement obsolete decision can be transformed to other decision alternative or detached from decisions graph. Decision can be marked pending after architecture evaluation (when conflict was discovered) or when related decision was changed. For architecture analysis one of the architecture analysis method can be taken [21].

Maintainability information. This information can be used for documentation maintenance purpose. In description multiple attributes can be used but date of change and decision owner should be specified.

Version. Each decision is versioned. With the use of version evolution of given decision is recorded. Version is described with following attributes: change author, change date and version number.

Each decision in knowledge repository can be interrelated with others. Decisions can be related directly (with the use of dependence and refinement connectors) or indirectly (with the use of architectural frame). Indirect relations are explained in following section of the article. Two types of direct relation are specified. Decision can be a consequence (*is consequence of* relation) of other decisions or refine one of them (*is refinement of*) (see Fig. 2). In addition, each decision can be bound with a risk in architecture. There are also sensitivity and trade-off points [22] which can be related to

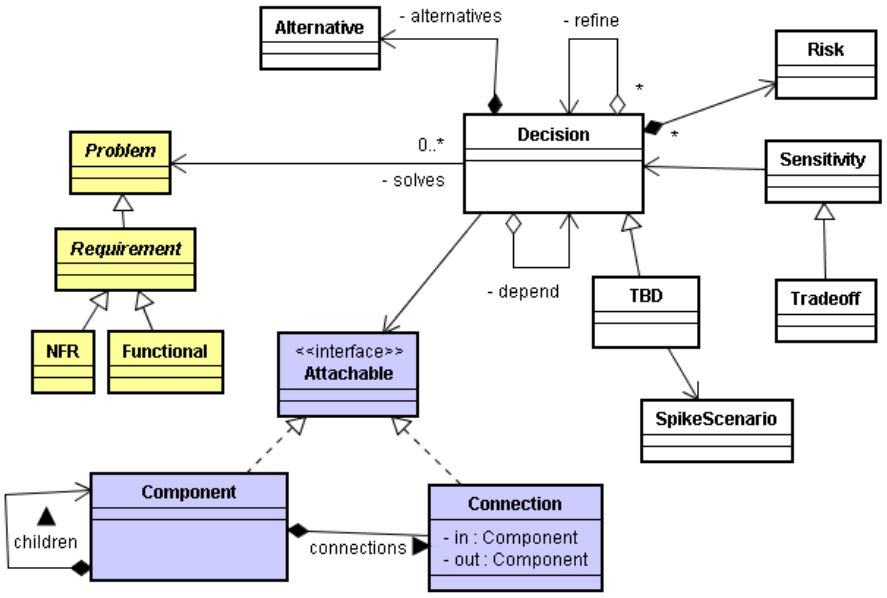

Fig. 2. Architectural knowledge repository model. Relation between requirements, design decisions and architectural frame.

one or many decisions. Therefore, results of architecture analysis can be embedded into architectural knowledge model and used for further analysis.

The importance of decision can be assessed with use of decision attributes and relations. It can be seen as a combination of importance of requirements which it satisfies. When decision space is explored, one should understand *why* the given decision was taken. The motivation for formation of the decision consist of: decision's rationale, alternatives and problem which is solved.

Model describes problem and architectural spaces (see Fig. 2). Binding between architectural components and requirements is embedded in design decisions space.

In conclusions, design decision is a central point of our model. The decision space provides solutions to business needs reported by stakeholders. This layer address all problems directly and binds the requirements with the component model of the architecture. Additional artifact as risks, sensitivity and trade-off points are embedded in the layer. Decisions are interrelated on two axis: direct (with the use of decisions connectors) and indirect (with the use of architectural frame).

4.2 Architectural Frame

The architectural frame (AF) presents overall view on the designed system. The second usage of this perspective in our architectural knowledge management system model is to show another axis of decisions relations. To model AF Component and Connector

view was implemented. In this view, components are used to model processing units and datastores. Connectors for interactions mechanisms [1]. The C&C view provides mechanisms to capture complex architectures. The components and connectors are bound with the use of ports (components interfaces) and roles (connectors interfaces). The components have a composite structure. In other words each component can contain other components. Additionally, each component and connection is versioned in our model. Moreover components are contained in main module called *System*.

To document C&C view three notations were considered:

- **ACME** [23] is an architecture description language (ADL). ACME is built on a core ontology of seven types of entities for architectural representation. It treats architecture as an annotated graph of components and connectors. Both language object types can be attached with a set of properties. Apart from textural representation, graphical modeling is possible with a use of Acme Studio.
- **UML Real-Time profile** [24] (UML-RT) was originally developed for telecommunication industry. The profile provides a natural home for expressing runtime structures, and supplies a semantic mapping to UML. Modeling with use of this profile is supported by commercial tools.
- **UML Components diagram** [25] was designed to support components modeling in UML. Each component can provide or require specific interface. These interfaces are used for modeling of communication between components. Multiple commercial and non-commercial modeling tools are available.

All presented languages can be used to describe architectural frame in our model, however to model Ksantypa 3 (see Section 5) UML Components diagram was used.

In proposed model to each component or connection multiple design decisions (DD) might be attached. This feature provides a way of mapping decision directly on logic structure of the system. Consequently system requirements also are mapped. There are two types of links between *DD* and *AF*. The *direct link* connects design decision with a specific component or connector. The *inheritable link* indicates that component and its internal elements can be influenced by given design decision. If more precise linkage should be provided, *direct links* must be used to point all affected parts of the model.

4.3 Architectural Knowledge Repository

When this model of repository is used several scenarios mentioned in previous section can be applied. In following paragraphs some of them are presented in details:

Incremental architectural refinement. In this scenario architectural knowledge is continuously updated. Multiple artifacts can be delivered by architect or after architecture or requirements analysis meetings. Support for agile development methods is also provided. Modules for existing requirements can be designed in details whereas decisions and components refinement for unstable or general requirements may be postponed to the time when requirements are enhanced. Thanks to versioning chronological aspect of changes can be captured. Additionally, new artifact are easy to assemble with existing knowledge. Design decision space can be supplemented with new decisions.

As component is composite it can be easily refined. In addition links between model elements in both layers can be used for consistency management. Each decision can be reattached to more specific component when it appears. Therefore, the decision scope can be stated accordingly to current needs and requirements knowledge.

Evaluation of change impact. When change is introduced its impact should be evaluated. There are changes in component model, problem and decision space possible. Changes in architectural frame are in most cases a result of decision space modification, although changes in decision space or requirements should be carefully reviewed. As binding between decision and problem spaces is handled in the decisions layer, impact of requirements change can be determined using decision-requirements graph. The impact change can be observed in two dimensions: direct decisions interrelation and architectural frame dependencies.

Exploration of decision space. We can explore decision space using various criteria. Firstly, we can track relation between decisions. Secondly, we can use states or types to extract decisions subsets. Moreover we can browse decisions which provides solutions to specified groups of requirements. For example all decision concerning security issues may be easily identified. Finally, architectural components can be used to identify related decisions.

Combining this criteria we can build complex queries to repository which is hard to achieve with the flat knowledge repository models. *"Find approved decisions concerning performance issues which influence data abstraction component."* can be given as example.

Getting a rationale. As model support rationale capturing this scenario is easy to conduct. Rationales and motivations for given decisions are maintained and can be easily found.

In brief, presented knowledge repository model supports incremental architecture refinement. Additionally, consistency of architectural knowledge can be maintained from very beginning. Model is equipped with mechanisms that simplify traceability and analysis of data. The exploration of space can be performed with use of dependencies or queries. In addition data mining of historical data (on other project in repository) is possible. Views are used to capture solution to customer concerns but the scope of information presented in the view depend on the decision attachment place. For example deployment issues may be presented on the level of whole system or for specific submodule.

5 Ksantypa Case

In the previous section model of the architectural knowledge management system was presented. In this section application of this model to the real system architecture description is described. For this purpose the Ksantypa system administration module was used. First, system concept is presented, after which part of the architectural model of the system is discussed.

5.1 Introduction

Each year Poznan University of Technology serves thousands of applications. There recruitment process is two phased. In first, the required documents are provided by candidates. Then recruitment committees creates rank lists of candidates for each specialisation applicants can choose. In the last step accepted candidates confirm their choices and become students. Unfortunately this process is time consuming for both sites. Applicants must visit university at least twice and spend their time in long queues. This can be big obstacle for candidates living far from Poznan. On the other hand process occupy university human resources. Moreover it is error prone, as dean offices employees transcribe application forms manually to existing system.

To solve this problems Ksantypa system was introduced. Ksantypa is an e-recruitment system developed by Poznan University of Technology. It is designed to simplify recruitment process at the university. In this year third version of the system is developed. The Ksantypa3 is designed to serve all departments at the university and provides tools for automatic rank list generation with the regards to applicants marks and preferences in global manner. The only step which is performed in traditional way is a provision of required documents by accepted applicants. Therefore, candidates handle the data by themselves and must appear at the university only once, to finalize recruitment process. System can be seen as two cooperating parts: candidates web portal and recruitment administration module. In the following section administration module is considered and will be referred as *K3* system.

Administration module supports dean office employees, recruitment committees members and system administrator. It provides functions for: defining a recruitment for given semester for all departments and types of studies, examination handling, validating application documents, ranking applicants and qualifying students. To finalize students qualification cooperation with other university system is required. In addition cooperation with banking systems is required to handle students fees. Apart from above, multiple non-functional requirements were specified. The most important concern changeability, security, performance and usability issues.

5.2 Architecture Discussion

To describe architecture of K3 architectural description model proposed in this paper was used. Following considerations deal with the intermediate version of specification. To simplify discussion, only part of design decision space is presented. K3 consist of two main modules and was written in Java. *XanUI* is written with the use of GWT. This module is responsible for handling user interaction. *XanServ* consist of several modules which communication is based on OSGi specification [26]. The design decision space is described in diagram 3. At this point we need to mention that this is not a proposal of notation. Both part of diagram was generated with Graphviz tool to illustrate relations existing in model. In the *a)* part interrelation between requirements (only non-functional requirements are shown for diagram simplification) and design decisions are presented. Requirements are represented with the green boxes. The full arrows represents *is consequence of* relations. The *is refinement of* is introduced with empty arrow. This relation is used between decisions *D_03* and *D_05*.

The *D_03* deals with the communication between XanServ modules.

D_03 – communication between modules is based on OSGi specification.

Problem. *N_25* components must exchangeable without the need of restarting the production server.

Rationale. OSGi was designed to handle dynamic exchange of modules. In addition components can be loosely coupled and communicate with the use only specified interfaces (OSGi secure visibility of module content in the environment).

Decision type – behavioral.

Alternatives. As an alternative implementation of self hosted dynamic module loading was considered. However this solution is not compatible with *D_04* (use of Springframework as a middleware layer).

Decision *D_05* is a refinement of *D_03* and describes the usage of Spring Dynamic Modules as an OSGi implementation. It is also a consequence of usage Springframework library in the system.

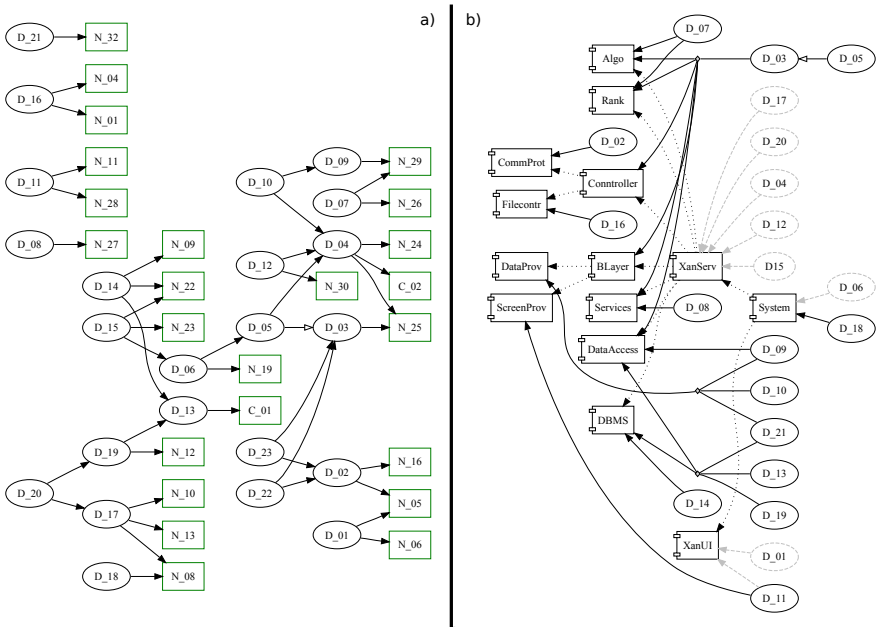

Fig. 3. Ksantypa design decision dependencies. Diagram a) decision space, b) decisions relations with the use of architectural frame.

In the *b)* part of diagram relation between design decisions with the use of architectural frame are presented. In the diagram architectural components are represented with boxes and their containment relation with dotted arrows. Next, decisions are represented

with ellipses. The bindings between decisions and components are draw with the use of dashed (for *inheritable links*) or regular (for *direct links*) arrows. Relations between decisions and connectors are also possible but was not marked in the diagram. As was mentioned before design decisions attached to the same component (or connector) are related. Therefore *D_19* and *D_21* are related.

D_19 (usage of VPD mechanism) was taken to secure data access on the database level. However *D_21* concerns usage of shared database space for Ksantypa administration module and candidates web portal. As portal was not designed for the uses of VPD mechanism integration risk was discovered which was recorded on discussed relation's axis.

In this section case study for model usage was described. Although description of K3 architecture is not complete some characteristics of the knowledge repository model can be observed. Model enable designer to use both view and decision descriptions. This methods are complementary especially when architectural frame is taken into account. With the use of AF additional relations between requirements can be observed. Therefore additional risks may be discovered.

6 Conclusions

Architecture is a result of multiple decision being taken. However, it is shaped by the requirements stated for the system. With the use of our model the binding between requirements and architecture is maintained. Requirements are addressed with the use of design decisions which are first class entities in our approach.

The second important part of the model is *architectural frame* (AF). It provides overall view on a component structure in built system. Additionally, AF provides a space which interrelates design decisions. The analysis of those indirect dependencies allow to find additional risks, trade-offs and conflicts in decisions space. AF has the composite structure, which facilitates an incremental knowledge refinement. Therefore, designers can focus on key problems specifying in detail some modules. It can be done without description consistency loss.

Design decisions can be interdependent as well as bound with the requirements. The analysis of these dependencies simplifies the change risk and impact evaluation. In addition, decisions are documented with a number of attributes. With the use of dependencies and attributes, decisions space can be explored in multiple dimensions. With simple filtering technical decisions space is available. Adding another criteria technical decisions concerning performance issues are identified. Moreover, we belief that historical knowledge mining can provide solutions to current problems.

To conclude, with the use of presented model the architectural knowledge artifacts can be captured. Their interrelations empower multidimensional knowledge exploration and traceability.

As the repository which uses this model is under development the future work includes finalization of the project and its industrial validation. In addition, the filed of integration multimedia data from architectural meeting with existing knowledge will be explored. Another challenge is extension of the model to support product lines. The

introduction of formal methods for consistency management in our model is also a problem awaiting answers.

References

1. Clements, D.: Documenting Software Architectures: Views and Beyond. Addison-Wesley Professional (2002)
2. Kruchten, P., Lago, P., van Vliet, H.: Building up and Reasoning About Architectural Knowledge. Quality of Software Architectures, pp. 43–58 (2006)
3. Kruchten, P.: The 4+ 1 View Model of architecture. Software, IEEE 12(6), 42–50 (1995)
4. Kennaley, M.: "The 3+1 Views of Architecture (in 3d)": An Amplification of the 4+1 Viewpoint Framework. In: Proc. of the 7th Working IEEE/IFIP Conference on Software Architecture (WICSA 2008), pp. 299–302. IEEE Computer Society (2008)
5. IEEE 1471:2000–Recommended Practice for Architectural Description of Software Intensive Systems. (2000)
6. ISO/IEC: ISO/IEC 42010 (IEEE P42010), Systems and Software Engineering – Architecture Description (WD3) (2008)
7. Jansen, A., Bosch, J.: Software Architecture as a Set of Architectural Design Decisions. In: Proc. of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA 2005), pp. 109–120. IEEE Computer Society (2005)
8. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley Professional (2003)
9. Hofmeister, C., Nord, R., Soni, D.: Applied Software Architecture. Addison-Wesley Professional (1999)
10. Tang, A., Jin, Y., Han, J.: A Rationale-based Architecture Model for Design Traceability and Reasoning. Journal of Systems and Software 80(6), 918–934 (2007)
11. Wang, Z., Sherdil, K., Madhavji, N.H.: Acca: An Architecture-centric Concern Analysis Method. In: Proc. of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA 2005), pp. 99–108. IEEE Computer Society (2005)
12. Jansen, A., Bosch, J.: Evaluation of Tool Support for Architectural Evolution. In: Proc. of the 19th IEEE International Conference on Automated Software Engineering (ASE 2004), pp. 375–378 IEEE (2004)
13. Babar, M., Gorton, I.: A Tool for Managing Software Architecture Knowledge. In: Proc. of the 2nd Workshop on Sharing and Reusing Architectural Knowledge, Rationale, and Design Intent, p. 11-. IEEE (2007)
14. Capilla, R., Nava, F., Pérez, S., Dueñas, J.: A Web-based Tool for Managing Architectural Design Decisions. ACM SIGSOFT Software Engineering Notes, 31(5) (2006)
15. Jansen, A., Van der Ven, J., Avgeriou, P., Hammer, D.: Tool Support for Architectural Decisions. In: Proc. of the 6th Working IEEE/IFIP Conference on Software Architecture (WICSA 2007), pp. 4–. IEEE Computer Society (2007)
16. Kruchten, P., Lago, P., van Vliet, H., Wolf, T.: Building up and Exploiting Architectural Knowledge. In: Proc. of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA 2005), pp. 291–292. IEEE Computer Society (2005)
17. Tyree, J., Akerman, A.: Architecture Decisions: Demystifying Architecture. Software, IEEE 22(2), 19–27 (2005)
18. Kleppe, A., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture–Practice and Promise. Addison-Wesley Professional (2003)
19. Kruchten, P.: An Ontology of Architectural Design Decisions in Software Intensive Systems. In: Proc. of the 2nd Groningen Workshop on Software Variability, pp. 54–61 (2004)

20. Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. ISO/IEC 25000:2005(E) (2005)
21. Michalik, B., Nawrocki, J., Ochodek, M.: 3-step Knowledge Transition: a Case Study on Architecture Evaluation. In: Proc. of the 30th International Conference on Software Engineering, pp. 741–748. ACM, New York, NY, USA (2008)
22. Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures. Addison-Wesley Boston, MA (2002)
23. Garlan, D., Monroe, R., Wile, D.: Acme: An Architecture Description Interchange Language. In: Proc. of the 1997 Conference of the Centre for Advanced Studies on Collaborative Research, pp. 169–183. IBM Press (1997)
24. Herzberg, D.: UML-RT as a Candidate for Modeling Embedded Real-time Systems in the Telecommunication Domain. In: Proc. of the 2nd International Conference on The Unified Modeling Language: Beyond the Standard, pp. 330–338. Springer, Heidelberg (1999)
25. Bjerkander, M., Kobryn, C.: Architecting Systems with UML 2.0. IEEE Software 20(4), 57–61 (2003)
26. Alliance, O.: Osgi Service Platform, Core Specification Release 4. Draft, (July 2005)