Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Comparing Methodologies for the Transition between Software Requirements and Architectures

Matthias Galster
University of Calgary, Canada
mgalster@ucalgary.ca

Armin Eberlein
American University of Sharjah, UAE
eberlein@ucalgary.ca

Mahmood Moussavi
University of Calgary, Canada
moussam@ucalgary.ca

***Abstract*—The transition from software requirements to software architectures has consistently been one of the main challenges during software development. Various methodologies that aim at helping with this transition have been proposed. However, no systematic approach for assessing such methodologies exists. Also, there is little consensus on the technical and non-technical issues that a transition methodology should address. Hence, we present a method for assessing and comparing methodologies for the transition from requirements to architectures. This method also helps validate newly proposed transition methodologies. The objective of such validations is to assess whether or not a methodology has the potential to lead to better architectures. For that reason, this paper discusses a set of commonly known but previously only informally described criteria for transition methodologies and organizes them into a schema. In the paper we also use our method to characterize a set of 14 current transition methodologies. This is done to illustrate the usefulness of our approach for comparing transition methodologies as well as for validating newly proposed methodologies. Characterizing these 14 methodologies also gives an overview of current transition methodologies and research.**

***Keywords*—software requirements engineering, software architecture, transition, technology assessment, quality assessment**

I. INTRODUCTION

Validation of new methodologies for the transition from software requirements to architectures and comparing them with existing research is essential for assessing the quality of such methodologies. According to Cameron et al. [1] methodologies in software engineering can be compared based on their notation, their rules / underlying processes, and their results. This means that one way of validating new approaches is to assess if the methodology itself meets certain criteria which are considered essential. Another way is to directly compare the result of different methodologies. However, in the software architecture domain it is difficult to directly compare the output of different approaches for the following reasons:

- Methodologies differ in the representation and documentation of the output architecture (e.g., text, figures, sketches).
- Methodologies use different notations. Each notation might support different architectural features (e.g., formal notations, such as SDL, are suitable for control structures, semi-formal notations, such as UML, are suitable for functional structures).
- Different methodologies result in architectures at different levels of granularity (e.g., low-level close to design versus high-level architecture).
- Methodologies address different intents and purposes of the architecture (e.g., architecture as a means for project management and work package definition versus architecture for performance estimation).

One solution to these problems could be model transformation, i.e., transforming output models of one methodology into output models of another methodology. However, this is difficult for two reasons: First, model transformation could make an excellent architecture bad by changing it (i.e., we would not only compare methodologies but also model transformation). Second, an architecture representation does not only include models but also text and other documentation [2]. Moreover, model transformation would not address all of the problems mentioned above (e.g., it would not address the various purposes of architectures).

Not much work has been done on systematic classification and comparison of methodologies for transiting between requirements engineering (RE) and software architectures. The relatively small number of empirical comparisons of transition approaches suggests that empirical comparisons are difficult to perform. Thus, a qualitative approach might be more promising. Moreover, there is not much guidance on desirable characteristics of transition methodologies and their usefulness. Therefore, we decided to survey the state of research to identify a set of commonly known but informally described criteria of transition approaches. We examine criteria for analyzing and comparing transition methods and based on the examination results suggest a set of criteria that focuses on the essence of a transition. These criteria are organized using ideas from the Goal-Question-Metric approach [3].

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

The main objective of this paper is to propose a method to assess and compare different transition methodologies and to demonstrate the use of the method to characterize existing and newly proposed methodologies. Due to the problems stated above we suggest a qualitative method to evaluate existing transition approaches rather than comparing the actual output of approaches. The method focuses on requirements and architecture-relevant aspects rather than on general process properties. In particular, the presented method can be used to

- suggest critical success factors for methodologies that help with the transition from requirements engineering to software architectures,
- categorize and compare existing transition methodologies,
- identify areas in the context of relating requirements engineering and software architectures where more work needs to be done, to point out weaknesses in methodologies and to improve methodologies, and
- validate new research results in the domain of relating requirements and architecture and to show significant differences between transition methodologies.

The rest of the paper is organized as follows: Section II discusses related work. In section III we present our method which then is applied to a set of current transition approaches in section IV. Section V presents conclusions and discusses directions for future research.

II. Related Work

Any attempt to provide a taxonomic comparison based on a comprehensive overview of the state-of-the-art in a particular area of research and practice is normally based on discoveries and conclusions of other researchers and practitioners and other previous surveys [4]. However, to the best of our knowledge no attempt has been made so far that provides a comprehensive assessment of transition methodologies themselves. Instead, previous work on transition approaches usually is limited to short surveys to support the need for a new methodology.

Some methodologies for comparing architecture design methods or architecture modeling approaches have been proposed. Roshandel et al. compare tradeoffs among architecture modeling approaches, focusing on the capability of identifying design effects [5]. The authors compare the use of two architecture description languages to model a system which was initially described in UML. However, no structured comparison is provided. Song and Osterweil present a systematic comparison approach for software design methodologies [6]. They argue that a sophisticated comparison requires building a process model of methodologies to compare and classify components of the methodology. Some of the ideas of this comparison approach are used in our work.

A general model of a design method has been presented by Hofmeister et al. [7]. Similarly, Kazman et al. identify essential components of a software architecture design process [8]. Practical needs of software architects have been identified by Falessi et al. [9]. Some elements of the general design method, as well as some of the essential components of design processes and needs of practitioners are incorporated into our method as a kind of checklist.

Comparison and evaluation methods have been proposed for other domains, such as Architecture Description Languages [10], architecture evaluation and analysis methods [4, 11-13] or requirements specification methods [14]. In [13], Babar and Gorton introduce the dimensions context, stakeholders, contents and reliability to organize attributes that help describe architecture evaluation methods. Even though these methods exist in other domains, there is little consensus on the technical and non-technical issues that a methodology for transiting between requirements and architectures should address.

III. Method to Compare Transition Approaches

A. Fundamentals

We developed a classification and comparison method by discovering commonalities and differences found among existing transition approaches (see Table II) and also by including properties of assessment methods from other domains (see previous section). To a great extent, our framework includes features either supported by any of the existing transition methodologies or reported as desirable by researchers and practitioners.

To identify the components of our method, we have also drawn upon a number of other sources, including previously developed comparison frameworks from other domains, an extensive survey of literature and studies that involved software engineering practitioners (including Hofmeister et al. [7], Bass et al. [14], Kazman et al. [8], Falessi et al. [9], Babar et al. [4], Song and Osterweil [6], Galster et al. [15], and Babar and Gorton [13]). However, we do not claim that we have produced an exhaustive list of features that a comparison schema should have. This schema is quite easily modifiable as is necessary in an area that is still in its inception stage.

We have assessed the suitability of our method in different ways. During the development of the method, a theoretical assessment of criteria was performed by relating each of them to the published literature on comparison frameworks and transition (see Table I). Also, we applied it to a set of current transition methodologies to show its applicability (see Table III).

B. Criteria for Comparison

Our premise is that a transition is intended to create architectures that satisfy requirements and their underlying intent, within schedule and budget. Thus, any transition approach must support the following groups of criteria, which we consider as essential:

- **Group 1:** Criteria on how the methodology fits into the software development process.
- **Group 2:** Criteria related to the artifacts created by the methodology.
- **Group 3:** Criteria related to the methodology's ease of use.
- **Group 4:** Criteria on how the methodology is used.

- **Group 5:** Criteria related to the maturity of the methodology.
- **Group 6:** Criteria related to how software quality attributes are addressed by the methodology.

There might be other criteria that could be considered as well, e.g., interaction management, dependencies among process components, ability to handle various requirements and design notations or scope of design activities. Some of these criteria are implicitly covered by the presented set of criteria (e.g., interaction management) or are too generic (e.g., scope of design activities). Others are not considered as crucial. However, we consider above groups of criteria as crucial to the success of any transition. The key groups of criteria provide a basis on which any transition methodology can be examined.

C. Criteria

A detailed list of the criteria is shown in Table I.

TABLE I. LIST OF CRITERIA

Group	#	Criterion	Question	Metric / value
1	1	Impact on / of RE process	Does methodology impact RE process or is methodology impacted by RE process [15]?	Boolean
	2	Architecture-relevant requirements	Does methodology support identification of architecture-relevant requirements [7], [14]?	Boolean
	3	Business and mission goals	Do requirements stem from business goals or mission goals [7], [14]?	Boolean
	4	Relation to RE methodologies	Is methodology related to existing RE methodologies [15]?	Boolean
	5	Representation	What is the representation for requirements / architecture (tree structure (T), graph (G), text (X)) [15]?	T *xor* G *xor* X
	6	Focus of transition	Does transition focus on RE (RE, e.g., specifying requirements), on architectural aspects (AR, e.g., refining architectural components) or on the actual transition (TR) [15]?	RE *xor* AR *xor* TR
	7	Specificity of domain	Is methodology specific to one application domain [15]?	Boolean
2	8	Requirements documentation	Are requirements expressive (i.e., provide necessary information for architecting, easy to organize architecture-relevant requirements) [14]?	Depending on user
	9	Support for variability	Does the methodology explicitly support variability in requirements [9], [14]?	Boolean
	10	Evaluation approach	Is evaluation and analysis of architecture elements included in the methodology [4], [8]?	Boolean
	11	Creation of architecture candidates	Does the methodology allow the creation of different architecture candidates [7], [15]?	Boolean
	12	Support for code derivation	Does the output of the methodology support the generation of code artefacts [15]?	Boolean
	13	Formality of notations	Does the methodology represent its products in a formal way (F) to allow tool support and consistency / completeness checking, or semi-formal (S) or informal (I) [15]?	F *xor* S *xor* I
	14	Novelty of representation	Does the methodology introduce a new representation / notation?	Boolean
3	15	Skill level necessary to carry out methodology	Are there any special skills or training needed to carry out the methodology effectively [14]?	Boolean
	16	Tool support	Is there any tool support that helps perform the methodology [9], [14]?	Boolean
	17	Human involvement	Does the methodology require human intelligence in performing decisions or can steps be automated [6]?	Boolean
4	18	Iterativeness	Does methodology follow a recursive flow with interations or a waterfall-like flow [15]?	Boolean
	19	Stakeholder participation and communication	As requirements are not always understood by developers, are stakeholders explicitly included in the methodology and participate in prioritizing requirements and setting the focus of the method [8], [15]?	Boolean
	20	Different architectural views	Does the methodology support the creation of different architectural views to address a separation of concerns [9], [15]?	Boolean
	21	Use of knowledge base	Does methodology support reuse of previously gathered knowledge / experience [7], [9]?	Boolean
	22	Abstraction and refinement	Does the methodology include guidelines for refinement / abstraction [9]?	Boolean
	23	Risk management	Does the methodology provide guidelines to recognize and manage risks [9]?	Boolean
	24	Tracing rationale behind decisions	Are decision rationales documented and traceable throughout the methodology [15]?	Boolean
	25	Covered activities	Which of the following classes of activities is supported by the methodology: requirements analysis (R), decision making (D), architecture evaluation (A) [15]?	R *xor* D *xor* A
	26	Use of templates to capture architectureal information	Does methodology provide templates for more consistency across various users and executions (i.e., provide repeatability of gathering and documenting information) [8]?	Boolean
5	27	Evaluation	Has the methodology been evaluated [4], [9]?	Boolean
	28	Previous applications	Has methodology been applied to real-world projects or used in industry [15]?	Boolean
6	29	Use of design primitives or tactics	Does methodology support quality attribute design principles [8]?	Boolean
	30	Use of quality attribute scenarios	Does methodology use quality attribute scenarios, map scenarios onto architecture representations and non-functional properties [8]?	Boolean
	31	Cost / benefit analysis	Does methodology help elicitate costs / benefits associated with architectures [8], [15]?	Boolean

The criteria described in Table I include *descriptive* characteristics and *comparative* characteristics of transition methodologies. Criteria are assessed using an adaptation of the Goal-Question-Metric approach (GQM) [3]. GQM is an approach to software metrics that allows tracing goals or evaluation criteria to data that are intended to make these criteria measurable. Often, GQM is used to identify rationales for defining and adapting techniques or to identify strengths and weaknesses of current methods. We focus on the structure of GQM (i.e., goals as the conceptual level, questions as the operational level, and metrics as the quantitative level), but not on its underlying process (i.e., goal identification, derivation of questions, completeness checks, etc.). In our work we have one goal which is the same for all criteria and refers to transition approaches in general: the determination of the quality and usefulness of a transition approach. The questions are derived from the criteria. Each question has one possible metric assigned to it (e.g., *Boolean* or a *set* of possible values). In addition to plain Boolean information we can also add textual information for further explanation (e.g., for a domain-dependent methodology we could add information in what domain this methodology can be applied). However, due to space limitations this has been omitted in this paper.

IV. APPLICATION

In this section we apply our method to a set of 14 transition approaches. The transition approaches are listed in Table II. Due to space limitations we do not provide any details regarding the surveyed approaches but provide references to their description in literature.

TABLE II. SURVEYED TRANSITION APPROACHES

#	Methodology	Reference(s)
A1	Goal-based transition	[16], [17]
A2	Problem frames	[18], [19]
A3	Use case maps	[20]
A4	Model bridging	[21], [22], [2]
A5	Rule-based decision making	[23]
A6	Architecting requirements	[24]
A7	Object-oriented transition	[25]
A8	Twin Peaks Model	[26]
A9	Patterns	[27]
A10	Multi-objective decision analysis	[28]
A11	Relating functional and architectural specifications	[29]
A12	Automated derivation of agent architectures from specifications	[30]
A13	Solving requirements conflicts and architecture design	[31]
A14	Co-development of requirements and functional architectures	[32]

A. *Results of Comparison*

The results of the assessment are shown in Table III. '✓' means that the criterion is met, '×' that the criterion is not met, and 'n/a' indicates that no reasonable judgment could be made or that a criterion is not applicable for a methodology. 'U' in the column for criterion 8 denotes that this criterion depends on the way the user uses the methodology.

TABLE III. RESULTS OF ASSESSMENT

Method	Criterion																														
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*
A1	✓	✓	✓	✓	G	RE	×	U	✓	×	×	✓	S	×	✓	✓	✓	✓	✓	✓	×	✓	×	✓	R	×	✓	✓	✓	✓	✓
A2	✓	×	×	✓	G	RE	×	U	×	×	×	n/a	S	×	✓	×	✓	×	n/a	✓	✓	×	×	×	R	×	✓	✓	✓	✓	×
A3	✓	×	×	✓	G	RE	×	U	✓	×	×	n/a	S	×	✓	✓	✓	×	✓	×	×	✓	×	×	R	×	✓	✓	×	×	×
A4	✓	×	×	✓	G	AR	×	U	×	×	×	n/a	S	×	✓	✓	✓	×	✓	×	×	✓	×	×	A	✓	×	✓	✓	×	×
A5	✓	×	×	n/a	G	RE	×	U	✓	×	✓	n/a	F	✓	×	n/a	✓	×	✓	×	✓	✓	×	✓	A	×	×	✓	n/a	✓	×
A6	✓	×	✓	n/a	T	TR	×	U	×	×	n/a	n/a	I	✓	×	×	✓	×	✓	×	×	✓	×	✓	R	×	n/a	n/a	×	✓	×
A7	×	×	✓	✓	G	TR	×	U	×	×	✓	✓	F	×	✓	×	✓	×	✓	✓	n/a	✓	×	×	n/a	×	✓	n/a	✓	✓	×
A8	✓	×	×	n/a	G	TR	×	U	✓	×	n/a	n/a	n/a	n/a	×	×	✓	✓	✓	✓	×	✓	×	×	R	×	×	✓	×	✓	×
A9	✓	✓	✓	✓	G	TR	✓	U	✓	×	✓	n/a	F	×	✓	×	✓	×	✓	✓	×	✓	×	✓	n/a	×	✓	×	✓	✓	×
A10	✓	✓	✓	✓	X	TR	×	U	×	✓	✓	×	I	×	✓	n/a	✓	×	✓	✓	n/a	✓	×	✓	D	×	✓	×	×	×	×
A11	✓	×	×	×	G	TR	×	U	×	✓	✓	×	F	×	✓	×	✓	✓	✓	✓	×	✓	×	n/a	D	×	✓	n/a	×	×	×
A12	✓	×	✓	✓	G	TR	✓	U	✓	×	×	n/a	I	×	✓	✓	✓	✓	✓	✓	×	✓	×	✓	D	×	✓	n/a	×	×	×
A13	✓	✓	✓	✓	X	TR	✓	U	✓	✓	✓	×	I	×	×	×	✓	✓	✓	✓	×	×	×	×	R	×	✓	✓	✓	✓	✓
A14	✓	✓	✓	✓	G	TR	×	U	×	×	×	×	S	×	×	×	✓	✓	✓	×	×	✓	×	✓	R	×	✓	×	×	✓	×

B. *Summary of Applying the Method*

In the introduction we claimed that our method can be used for four purposes:

- Suggest critical success factors for transitions between requirements and architectures: We achieved this through a set of criteria, organized in six groups. These criteria highlight critical aspects that any transition methodology should address.
- Categorize and compare existing work on the transition between requirements and architectures: Table III shows the results of such a comparison. Fig. 1 illustrates how criteria are met by current methodologies (criteria 5, 6, 8, 13 and 25 are not

plotted as they are not Boolean criteria). The comparison revealed several features supported by most current methodologies (e.g., stakeholder involvement). Also, all approaches require human input (criterion 17), an indicator that automatic transitions might not be feasible. On the other hand, the survey also highlights a number of issues which existing methods do not sufficiently address (e.g., tracing decision rationales or templates for capturing architectural knowledge). Moreover, for criterion 12 (generation of code fragments) we cannot judge if this feature is supported or not by several approaches due to a lack of description in related literature. Combined with the fact that only two approaches support this feature this leads to the conclusion that considering code derivation does not seem to be a priority when developing transition methodologies. Based on these observations we can see that the proposed method provides an important advance towards answering questions regarding the features a good transition approach should support and how to compare and assess transition approaches.

- Identify areas where more work needs to be done in order to make advances in this area: Table III and Fig. 1 show that many approaches do not use templates to capture architectural information or do not sufficiently support tracing design rationales. However, these aspects seem to be important for practitioners [9]. Also, as we can see in the case of criterion 8, the expressiveness of requirements always depends on the user but is not prescribed by the methodology. These areas could be directions for future research.
- Validate new research results in the problem domain against key factors and show differences between transition methodologies: This issue goes back to the original problem of not being able to directly compare the outcome of transition methodologies. We can use the proposed method as a checklist to assess new methodologies based on the set of essential criteria for transition methodologies and to identify how well a new methodology fulfills these criteria.

C. *Additional Use of the Method in Software Development*

In addition to the use of the method mentioned in the previous section, its set of attributes and resulting comparisons are useful for the following: First, the method itself can act as checklist when developing new methodologies for the transition between requirements and architectures. Second, applying our method to comparing existing methodologies helps select a proper transition method in a particular context. However, a selection is only based on certain criteria, mostly from Group 3 (ease of use) and 5 (maturity). Criteria in Group 1, 2 and 4 might be less appropriate selection criteria as they refer more to the question of how to actually apply methodologies.

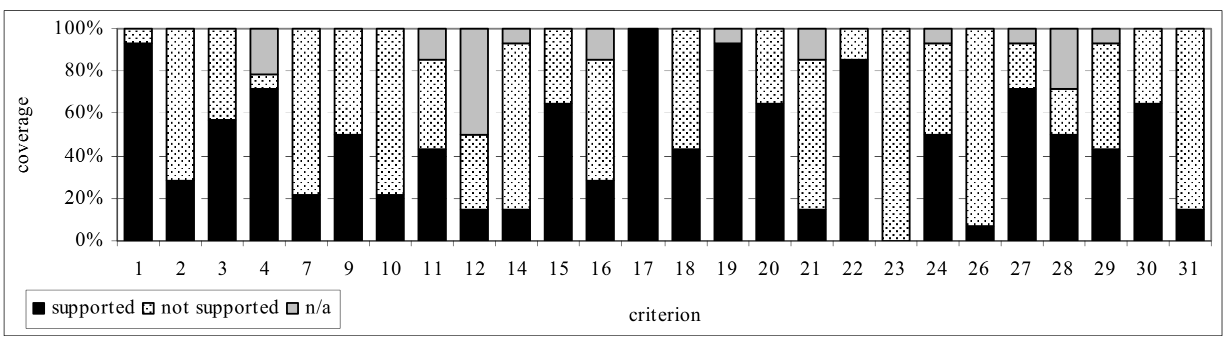

Figure 1. Plot of boolean criteria to show how well critera are met by current methodologies

V. Conclusions and Future Work

The transition from requirements to architectures is a critical step in the software development cycle. Having a systematic approach is a necessity to ensure that the process is usable by a broad range of stakeholders and leads to architectures of good quality. When new methodologies are suggested it is important to assess their usefulness. Thus, the main contribution of this paper is a method for assessing, classifying and comparing approaches that help in the transition from software requirements to architectures. This method has been developed by discovering similarities and differences between existing transition approaches and by surveying existing literature on comparison approaches in other domains. We have also demonstrated how the proposed schema can be used to identify the essential features that a good transition approach should provide and to identify gaps in current methodologies. We believe these issues indicate some of the areas where future research should be concentrated. We do not suggest that our schema is complete in its existing form. Rather, we expect this schema to be modified and extended in future as a result of our ongoing research.

References

[1] J. R. Cameron, A. Campbell, and P. T. Ward, "Comparing Software Development Methods: Example," Information and Software Technology, vol. 33, pp. 386-402, July-August 1991.

[2] H. Kaindl and J. Falb, "From Requirements to Design: Model-driven Transformation or Mapping," in Proc. First International Workshop on Model Reuse Strategies (MoRSe 2006), Warsaw, Poland, 2006, pp. 29-32.

[3] V. Basili, G. Caldiera, and D. Rombach, "The Goal Question Metric Approach," in Encyclopedia of Software Engineering. vol. 1, J. J. Marciniak, Ed. New York, NY: John Wiley & Sons, 1994, pp. 528-532.

[4] M. A. Babar, L. Zhu, and R. Jeffery, "A Framework for Classifying and Comparing Software Architecture Evaluation Methods," in Proc. 2004 Australian Software Engineering Conference, Melbourne, AUS, 2004, pp. 309-318.

[5] R. Roshandel, B. Schmerl, N. Medvidovic, D. Garlan, and D. Zhang, "Understanding Tradeoffs Among Different Architectural Modeling Approaches," in Proc. Fourth Working IEEE/IFIP Conference on Software Architecture, Oslo, Norway, 2004, pp. 47-56.

[6] X. Song and L. J. Osterweil, "Experience with an Approach to Comparing Software Design Methodologies," IEEE Transactions on Software Engineering, vol. 20, pp. 364-384, May 1994.

[7] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, and P. America, "Generalizing a Model of Software Architecture Design from Five Industrial Approaches," in Proc. 5th Working IEEE/IFIP Conference on Software Architecture, Pittsburgh, PA, 2005, pp. 77-88.

[8] R. Kazman, L. Bass, and M. Klein, "The Essential Components of Software Architecture Design and Analysis," Journal of Systems and Software, vol. 79, pp. 1207-1216, August 2006.

[9] D. Falessi, G. Cantone, and P. Kruchten, "Do Architecture Design Methods Meet Architects' Needs?," in Proc. The Working IEEE/IFIP Conference on Software Architecture, Mumbai, India, 2007, pp. 44-53.

[10] N. Medvidovic and R. N. Taylor, "A Classification and Comparison Framework for Software Architecture Description Languages," IEEE Transactions on Software Engineering, vol. 26, pp. 70-93, 2000.

[11] R. Kazman, L. Bass, M. Klein, T. Lattanze, and L. Northrop, "A Basis for Analyzing Software Architecture Analysis Methods," Software Quality Journal, vol. 13, pp. 329-355, November 2005.

[12] M. A. Babar and B. Kitchenham, "Assessment of a Framework for Comparing Software Architecture Analysis Methods," in Proc. 11th International Conference on Evaluation and Assessment in Software Engineering, Keele, England, 2007.

[13] M. A. Babar and I. Gorton, "Comparison of Scenario-Based Software Architecture Evaluation Methods," in Proc. 11th Asia-Pacific Software Engineering Conference, Busan, Korea, 2004, pp. 600-607.

[14] L. Bass, J. Bergey, P. Clements, P. Merson, I. Ozkaya, and R. Sangwan, "A Comparison of Requirements Specification Methods from a Software Architecture Perspective," SEI CMU, Pittsburgh, Technical Report CMU/SEI-2006-TR-013, 2006.

[15] M. Galster, A. Eberlein, and M. Moussavi, "Transition from Requirements to Architecture: A Review and Future Perspective," in Proc. 7th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2006), Las Vegas, NV, 2006, pp. 9-16.

[16] A. v. Lamsweerde, "From System Goals to Software Architecture," in LNCS 2804 - Formal Methods For Software Architectures, M. Bernardo and P. Inverardi, Eds. Berlin / Heidelberg: Springer Verlag, 2003, pp. 25-43.

[17] M. Bradozzi and D. E. Perry, "From Goal-Oriented Requirements to Architectural Prescriptions: The Preskriptor Process," in Proc. The Second International Workshop on Software Requirements and Architectures (STRAW '03) at ICSE'03, Portland, OR, 2003.

[18] J. G. Hall, M. Jackson, R. C. Laney, B. Nuseibeh, and L. Rapanotti, "Relating Software Requirements and Architectures using Problem Frames," in Proc. IEEE Joint International Requirements Engineering Conference (RE'02), Essen, Germany, 2002, pp. 137-144.

[19] L. Rapanotti, J. G. Hall, M. Jackson, and B. Nuseibeh, "Architecture-driven Problem Decomposition," in Proc. 12th International Requirements Engineering Conference, Kyoto, Japan, 2004, pp. 80-89.

[20] D. Amyot and G. Mussbacher, "Bridging the requirements/design gap in Dynamic Systems with Use Case Maps (UCMs)," in Proc. 23rd International Conference on Software Engineering, Toronto, Canada, 2001, pp. 743-744.

[21] N. Medvidovic, P. Gruenbacher, A. Egyed, and B. W. Boehm, "Bridging Models Across the Software Lifecycle," The Journal of Systems and Software, vol. 68, pp. 199-215, 2003.

[22] D. Liu and H. Mei, "Mapping requirements to software architecture by feature-orientation," in Proc. The Second International Workshop on Software Requirements and Architectures (STRAW '03) at ICSE'03, Portland, OR, 2003.

[23] W. Liu and S. Easterbrook, "Eliciting Architectural Decisions from Requirements using a Rule-based Framework," in Proc. The Second International Workshop on Software Requirements and Architectures (STRAW '03) at ICSE'03, Portland, OR, 2003, pp. 94-99.

[24] W. Liu, "Architecting Requirements," in Proc. Doctoral Consortium at RE'04, Kyoto, Japan, 2004.

[25] H. Kaindl, "Difficulties in the Transition from OO Analysis to Design," IEEE Software, vol. 16, pp. 94-102, September / October 1999.

[26] B. Nuseibeh, "Weaving Together Requirements and Architectures," IEEE Software, vol. 34, pp. 115-117, March 2001.

[27] L. Xu, H. Ziv, T. A. Alspaugh, and D. J. Richardson, "An Architectural Pattern for Non-functional Dependability Requirements," Journal of Systems and Software, vol. 79, pp. 1370-1378, October 2006.

[28] L. Xu, "MODA - Multiple Objective Decision Analysis: Balancing Quality Attributes in Software Architectures," in Proc. Doctoral symposium at the 30th International Conference on Software Engineering, Leipzig, Germany, 2008, pp. 1019-1022.

[29] F. Corradini, P. Inverardi, and A. L. Wolf, "On Relating Functional Specifications to Architectural Specifications," Science of Computer Programming, vol. 59, pp. 171-208, February 2006.

[30] C. H. Sparkman, S. DeLoach, and A. L. Self, "Automated Derivation of Complex Agent Architectures from Analysis Specifications," in LNCS 2222 - Agent-Oriented Software Engineering II, M. J. Wooldridge, G. Weiss, and P. Ciancarini, Eds. Berlin / Heidelberg: Springer Verlag, 2001, pp. 278-296.

[31] A. Herrmann, B. Paech, and D. Plaza, "ICRAD: An Integrated Process for the Solution of Requirements Conflicts and Architectural Design," International Journal of Software Engineering and Knowledge Engineering, vol. 16, pp. 917-950, December 2006.

[32] K. Pohl and E. Sikora, "The Co-Development of System Requirements and Functional Architecture," in Conceptual Modelling in Information Systems Engineering, J. Krogstie, A. L. Opdahl, and S. Brinkkemper, Eds. Berlin: Springer Verlag, 2007, pp. 229-246.