A Comprehensive Study for Modern Models: Linking Requirements with Software Architectures

sisay yemata
sisay.yemata@aau.edu.et

A comprehensive study for modern models: linking requirements with software architectures

Sisay Yemata
School of Graduate Studies, Software Engineering Track, Addis Ababa University, Addis Ababa, Ethiopia
sisay.yemata@aau.edu.et

Belachew Regane
School of Graduate Studies, Software Engineering Track, Addis Ababa University, Addis Ababa, Ethiopia
belachew.regane@aau.edu.et

ABSTRACT

Several models recently have been addressed in software engineering for requirements transformation. However, such transformation models have encountered many problems due to the nature of requirements. In the classical transformation modeling, some requirements are discovered to be missing or erroneous at later stages, in addition to major assumptions that may affect the quality of the software. This has created a crucial need for new approaches to requirements transformation. In this paper, a comprehensive study is presented in the main modern models of linking requirements to software architectures. An extensive evaluation is conducted to investigate the capabilities of such modern models to overcome those limitations when transforming requirements, validating their consideration of bringing quality for the software development process. Key research gaps and open issues are discussed, highlighting the possible future directions that can be considered in this field.

Keywords

Quality requirements, software architecture, requirement engineering, software quality, transformation models.

INTRODUCTION

One of the major issues in software systems development today is quality (Dobrica & Niemela, 2002). To bring quality software, the appropriate transformation of requirements is necessary for the early stages of the software development life cycle. "Software Quality comprises all characteristics and significant features of a product or an activity which relate to the satisfaction of a given requirement (Boehm et al., 1976)." Transformation is the means of linking requirements with software architecture and vice versa (Pimentel et al., 2012), whereas; transformation models are the abstract graphical presentation of requirements (Chakraborty et al., 2012). Software Architecture (SA) represents "the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the Principles of its design and evolution” (May 2011). Requirements are expected by stakeholders “why” the developing software system should make (Yu, 2001).

The challenge in software development is to develop software with the right quality levels (J. Bosch & Molin, 1999). To void this challenge, transformation of requirements in the early stages of software development, the life cycle is a crucial task. However, some quality requirements are missing during the transformation of requirements which are the primary driving force for systems and subsystem architectures (Firesmith, 2005) and they heavily influenced by the architecture (J. Bosch & Molin, 1999).

To discover these missing requirements and assumptions, there were classical models that used for the transformation of requirement, such as waterfall and V-shape Software Development Life Cycle Models. In the traditional transformation models software systems should not begin software architecture design until complete, correct and consistent requirements the specification is reached (Liu & Mei, n.d.)]. If some problems were revealed at the architecture phase, there were no mechanisms to discover those missing requirements, not backward transformations (Liu & Mei, n.d.). Backward transformation is the transformation of architecture to requirements, because they are not following iterative way of transformation (Bhuvaneswari & Prabaharan, 2013; Forsberg & Mooz, 1991; Larman & Basili, 2003).

In this paper, a comprehensive study is presented in the main modern models of requirements transformation that helps to discover the missing requirements, in addition to assumptions. Modern transformation models used iterative means of transformation between requirements and architecture that means forward (from requirements to architecture) and backward (from architecture to requirements) transformation. An extensive evaluation is conducted to investigate the capabilities of such modern models to overcome those limitations when transforming requirements, validating their consideration of bringing quality for the software development process. Key research gaps and open issues are discussed, highlighting the possible future directions that can be considered in this field. These modern models are used to transform requirements form requirements to architecture and vice versa (Avgeriou et al., 2011; J. Bosch & Molin, 1999; Yu, 2001, 2001). Some of the models transform from requirement to architecture, while others transform from architecture to requirements by decomposing and composing of problems (Avgeriou et al., 2011). These new transformation models are addressing the focus of the software system researchers and the industry's interest by bringing quality software system (Alebrahim et al., 2011). Therefore, the quality requirements being discovered starting from the initial stages of the System Development Life Cycle (SDLC) give us the following advantages. (i) Discover the missing requirements and undocumented assumptions early, (ii) Minimize the time required, and (iii) Minimize the cost of software development.

Those modern requirement transformation models are categorized into two. The first category is the models are used to transform from the requirement to architecture, this category includes the twin peaks model(Castro et al., 2012; Pimentel et al., 2012), multi-view model, goal-oriented modeling (Hall et al., 2002), scenario-oriented model (Pimentel et al., 2012) and feature-orientation (Liu & Mei, n.d.). The second category of models is used to transform from architecture to requirements, such models include: Feature Solution Graph (FSG) (de Bruin & van Vliet, 2003), problem frame (Alebrahim et al., 2011), Recover Assumption Analysis method (Roeller et al., 2006). Evaluations of those modern models examined based on the developed criteria and research gaps are identified. The category of modern models is presented in the following diagram.

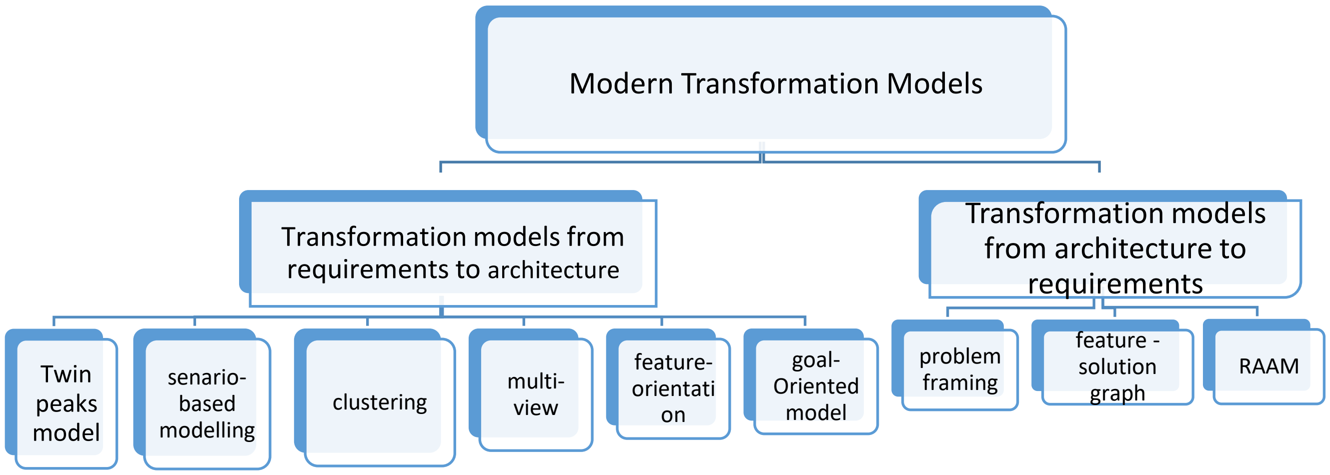

Figure 1. categories of modern transformation models

The remainder of the paper is organized as follows: section 2 transformation models from requirement to architecture, section 3 transformation models from architectural to requirement, section 4 comparisons of transformation models for emerging technologies, section 5 comparisons and evaluations of models, section 6 discussions and research gap, section 7 conclusions and future work, section 8 references.

TRANSFORMATION MODELS FROM REQUIREMENTS TO ARCHITECTURE

Twin Peaks model: It is used to highlights the relationship between requirements and architecture. Requirements describe as a problem and architecture as a solution, in between the requirement and architecture there is a scenario which emphasizes on the incrementally elaborating details in both artifacts (Pimentel et al., 2012). Using model transformation approaches appear as an effective way to generate architectural models from requirements models (Pimentel et al., 2012), and it is used for the co-development of requirements specification and architectural design description (Firesmith, 2005; Forsberg & Mooz, 1991). **Goal-Oriented Modeling** (GOM), there are goals that focus on "why" the system should do it rather than what the system should do. Unlike the traditional requirement transformation models, approaches (Eridaputra et al., 2014). This model is focused on functional goals of the system (Yu, 2001) and it used the two most popular methodologies i* and Knowledge Acquisition in autOmatic Specification (KAOS). **Scenario-based modeling (SBM):** scenarios are written by the user language or natural language during requirement analysis, at the design or architectural level written by the developers in the context of the system as Scenario-Based Modeling (SBM) that used to transform requirements into architectural design and answer "how" and "what" questions (Yu, 2001). Then, by issuing "why" questions referring to these scenarios and it is focusing on the functional requirements (Yu, 2001). **Clustering method:** is used to structuring requirements with respect to their impact on the architecture design process. Such as gaining architecture relevant information from requirements which might not have been discovered during requirements analysis. Identified structures help derive strategies for the implementation of requirements in the architecture and it is used to develop a software system from scratch (Galster, Eberlein, et al., 2013) starting from the individual requirements by appalling the bottom-up approach and this approach treats functional and non-functional requirements equally.

Multi-view model: One difficulty arising in architectural design is the different interests of the stakeholders. This multi-view model is used to transform different stakeholder requirements/interests (Kruchten, 1995). The most well-known model is perhaps the “4+1” view model presented by Rational Software Corporation (Kruchten, 1995). **Feature-Orientation**: it is the model that used for linking requirements to software architecture. First, it discovers the functional requirements and then discovers the non-functional requirements by following the iterative processes (Liu & Mei, n.d.).

TRANSFORMATION MODEL FROM ARCHITECTURE TO REQUIREMENTS

Feature solution graph: first an architecture address only functional requirements, then it is focused on the architecture for capturing architectural knowledge by fragmented architecture that connects quality requirements with solution fragments at the architectural level. The solution fragments captured in this a graph is used to iteratively compose an architecture driven by the quality requirements (de Bruin & van Vliet, 2003). Thus, the quality requirements discovered by decomposing (top-down approach) the reference architecture and composing (bottom-up approach) these requirements.

Problem frame: A problem frame defines the shape of a problem by capturing the characteristics and interconnections of the parts of the world, it is concerned with, and the concerns and difficulties that are likely to arise in discovering its solution (Cox & Phalp, n.d.; Hall et al., 2002). With the problem frame the derivation of the software architecture is starting from the problem diagram and then decomposing it into sub-problems in order to discover the missing quality requirements of the software (Cox & Phalp, n.d.).

Most software development problems are complex, thus problem frame it provides a means of analyzing by decomposing and composing those complex problems. It is also, allowing architectural structures, services, and artifacts to be considered as part of the problem domain (Hall et al., 2002). Most likely it is workable to the new knowledge domain to develop artifacts.

Recover Assumption Analysis method: most of the assumption requirements are missing during requirement specification and revealing at later stages of software development life cycle phases and they may be invalid or the new assumptions contracted with a previous one (Roeller et al., 2006). As the software designer and The architect considers the future requirement is a crucial task. So, Recover Assumption Analysis Method is used to discover those hidden, implicit and undocumented assumptions at early stages of software development phases by gathering requirements from different sources using different requirement gathering methods as stated (Roeller et al., 2006).

In the summary of their usage and category from Requirement to Architecture (R to A) and from architecture to requirement (A to R) transformation of models is presented

.. Model/method	Usage of models	Category
Multiple-view model	Used to address the interest of different aspect of the architecture (Kruchten, 1995).	R to A
Goal-oriented Modeling	Scenarios and agents together to guide the RE to architectural design process (Yu, 2001).	R to A
Scenario-based modeling	Employs iterative evaluation and transformation of the software architecture in order to satisfy the quality requirements (Yu, 2001).	R to A
Twin Peaks model	Single goal model to express both requirements and architectural concerns and approach based on model transformations to derive architectural, structural specifications from system goals (Forsberg & Mooz, 1991; Galster, Mirakhorli, et al., 2013; Pimentel et al., 2012).	R to A
Clustering method	Gaining architecture relevant information from requirements to design the architecture (Galster, Eberlein, et al., 2013).	R to A
Feature-Orientation	Used to map requirements to architecture (Liu & Mei, n.d.).	R to A

Feature solution (FS) graph	Used for composition and decompose software architecture (de Bruin & van Vliet, 2003).	A to R
RAAM	Method to recover assumptions (implicit or undocumented.) from an existing software product(Roeller et al., 2006).	A to R
Problem Frames	Model- and pattern-based method that allows software engineers to take quality requirements into account right from the beginning of the software development process and extend problem frames, allowing architectural structures, services and artifacts to be considered as part of the problem domain (Cox & Phalp, n.d.; Hall et al., 2002).	A to R

Table 1. Transformation models from requirements to architecture and vis versa.

APPLICABILITY OF TRANSFORMATION MODELS FOR EMERGING TECHNOLOGIES

Nowadays, the developments and enhancement of emerging technologies are increasing in computing. The development of emerging technologies has its own advantages and challenges. To address their challenges transforming the requirements to architecture and vis versa is a crucial task. Therefore, from the identified modern transformation models which are applicable in specific technologies requirements transformation is necessary to address the requirements transformation process to identify and incorporate quality requirements to the developed technologies. To do this, the following table shows the applicability of the transformation model to emerging technologies.

Types of emerging technologies	Transformation models								
	Twin Peaks	Multi-view	Clustering	Goal-oriented	Scenario-based	Feature-Orientation	Feature-Solution Graph	Problem frame	RAAM
Used for IoT system requirement transformation	No	No	No	No	No	No	No	No	No
Used for cloud-based system requirement transformation	No	No	No	No	No	No	No	No	No
Used for big data requirement transformation	No	No	No	Yes	No	No	No	No	No
Used for cyber-physical system requirement transformation.	No	No	No	No	No	No	No	No	No

Table 2. comparison of transformation models for emerging technologies

COMPARISON AND EVALUATION OF TRANSFORMATION MODELS

The transformation models are compared and evaluated using the set of criteria which develops based on the comparison and evaluation relevance's.

Comparison/ evaluation criteria	Model/method								
	Transformation models from requirements to architecture					Transformation models from architecture to requirements			
	Twin Peaks	Multi-view	Clustering	Goal-oriented	Scenario-based	Feature-Orientation	Feature-Solution Graph	Problem frame	RAAM
Level of decomposition and composition	No	No	Less	No	No	No	High	High	No
Addressing the range of stakeholder interests	Less	High	Less	Less	Less	Less	Less	Less	Less
Level of addressing NFR	Less	Less	High	Less	Less	High	High	High	High
Level of addressing FR	High	High	High	High	High	High	Less	Less	Less
Time required	Less	High	High	Less	Less	High	High	High	High
Cost	Low	High	High	Low	Low	High	High	High	High
Level of Discover assumptions	Less	Less	Less	Less	Less	Less	Less	Less	High

Table 3. Comparison and evaluation models

DISCUSSION AND RESEARCH GAP

The focus of this discussion is on the presentation of the requirements transformation models which bring quality on the software development. In table 5.1 the main issues were examined that is the comparison of modern transformation models from requirements to software architecture and Vis versa to discover the missing requirements. In addition to undiscovered the missing requirements undiscovered assumptions cause for poor quality software development. In this paper, the new transformation models that used for transforming requirements were compared by using extensive evaluation criteria and presented as follows:

Level of decomposing and composing: the aim of decomposing and composing requirements and software architectures is to discover the missing requirements specially, quality requirements/ non-functional requirements. So, based on the comparison table 5.1 transformation models from architecture to requirements have high capabilities to discover the missing requirements.

Level of addressing non-functional requirements: non-functional requirements also known as quality requirements which have been a high effect on software quality (Yu, 2001). Therefore, software development required transformation models to discover quality requirements/ NFR. Based on the comparison table all transformation models in the second category have a high capability of discovering quality requirements.

Addressing the range of stakeholders' interests: In any of the software project development, different stakeholders who have different interests/ expectations are participating. So, modern transformation models required to respect all stakeholders' interests during the transformation of requirements. Among the transformation models from table 5.1, the multi-view transformation model from the first category has a higher chance to satisfy different stakeholder interests.

Level of addressing Functional Requirements: functional requirements are the goals of the system (Yu, 2001). Transformation models are required to transform those requirements into architecture and discovered the missing requirements. So, it needs modern transformation models in order to discover the missing requirements. The first categories of the transformation models have higher capabilities to discover the functional requirements.

Level of discovering assumptions: in addition to discovering functional and non-functional requirements, discovering assumptions in software development is essential to bring the quality of the software. Thus, from the category table 5.1 only one modern transformation model is presented to discover missing assumptions. Even if, RAAM is used to transform the requirement starting from early-stage up to later stages of software development that requires more cost and time.

Generally, the second category that means transformation models from architecture to requirements has a higher level of discovering assumptions and quality requirements/ non-functional requirements. As a result, it requires more time and cost. Whereas, the first categories of the transformation model more focused on discovering the functional requirements and give less attention to quality requirements. Thus, when we compare requirements to architecture transformation models and architecture to requirements models, requirements to architecture transformation models require less time and cost to discover missing requirements.

Since the quality of a software system is more depends on the non-functional requirement or quality requirements and they are more addressing by transforming architecture (Jan Bosch & Molin, 1999; Dobrica & Niemela, 2002; Yu, 2001). Thus, the second category is giving more focus to addressing the quality of the software system according to the evaluation criteria and the existing kinds of literature.

Research gaps identified from the discussion are: (i) kinds of literature focus only on the structural transformation of requirements, not focus on the behavioral aspect of the transformation models (ii) there are no requirements transformation models for Internet of thing, cloud-based systems, and cyber-physical systems (iii) recovering assumptions before the implementation phase are not considered by more kinds of literature, even if, recover the assumption from the starting phase of System Analysis and Design Life Cycle phase up to implementation stages is addressed. This requires investing in additional cost and time.

CONCLUSION AND FUTURE WORK

There are several models in the area of software engineering that used for transforming requirements to architecture. However, problems exist during the transformation of requirements by the nature of the model during the transforming of requirements. In the classical transformation modeling requirements are missing and uncovered, in addition to this, undiscovered assumptions are affecting the quality of the software. In this work, we presented new models which help overcome those limitations by transforming requirements during the development of the software starting from the early stages of the software development life cycle. Most of the models transform from requirements to architecture, some of them transform from architecture to the requirements in order to recover the missing requirements and assumptions.

Based on the research gap discussed in the discussion section of this paper, the following activities will be addressed in the future work. (i) The behavioral aspect of the transformation models will be presented. (ii) Requirement transformation models will be presented for Internet of things, cloud-based system and cyber physical systems (iii) recover assumptions before the implementation and deployment stages of the software development life cycle will be presented.

REFERENCES

Alebrahim, A., Hatebur, D., & Heisel, M. (2011). A Method to Derive Software Architectures from Quality Requirements. 2011 18th Asia-Pacific Software Engineering Conference, 322–330. https://doi.org/10.1109/APSEC.2011.29

Avgeriou, P., Grundy, J., Hall, J. G., Lago, P., & Mistrík, I. (2011). Relating software requirements and architectures. Springer Science & Business Media.

Bhuvaneswari, T., & Prabaharan, S. (2013). A survey on software development life cycle models. International Journal of Computer Science and Mobile Computing, 2(5), 262–267.

Boehm, B. W., Brown, J. R., & Lipow, M. (1976). Quantitative evaluation of software quality. Proceedings of the 2nd International Conference on Software Engineering, 592–605.

Bosch, J., & Molin, P. (1999). Software architecture design: Evaluation and transformation. Proceedings ECBS'99. IEEE Conference and Workshop on Engineering of Computer-Based Systems, 4–10. https://doi.org/10.1109/ECBS.1999.755855

Bosch, Jan, & Molin, P. (1999). Software architecture design: Evaluation and transformation. Proceedings ECBS'99. IEEE Conference and Workshop on Engineering of Computer-Based Systems, 4–10.

Castro, J., Lucena, M., Silva, C., Alencar, F., Santos, E., & Pimentel, J. (2012). Changing attitudes towards the generation of architectural models. Journal of Systems and Software, 85(3), 463–479. https://doi.org/10.1016/j.jss.2011.05.047

Chakraborty, A., Baowaly, M. K., Arefin, A., & Bahar, A. N. (2012). The role of requirement engineering in software development life cycle. Journal of Emerging Trends in Computing and Information Sciences, 3(5), 723–729.

Cox, K., & Phalp, K. (n.d.). From Process Model to Problem Frame – A Position Paper. 4.

de Bruin, H., & van Vliet, H. (2003). Quality-driven software architecture composition. Journal of Systems and Software, 66(3), 269–284. https://doi.org/10.1016/S0164-1212(02)00079-1

Dobrica, L., & Niemela, E. (2002). A survey on software architecture analysis methods. IEEE Transactions on Software Engineering, 28(7), 638–653.

Eridaputra, H., Hendradjaya, B., & Danar Sunindyo, W. (2014). Modeling the requirements for big data application using goal oriented approach. 2014 International Conference on Data and Software Engineering (ICODSE), 1–6. https://doi.org/10.1109/ICODSE.2014.7062702

Firesmith, D. (2005). Quality Requirements Checklist. The Journal of Object Technology, 4(9), 31. https://doi.org/10.5381/jot.2005.4.9.c4

Forsberg, K., & Mooz, H. (1991). The relationship of system engineering to the project cycle. INCOSE International Symposium, 1(1), 57–65.

Galster, M., Eberlein, A., & Jiang, L. (2013). Structuring Software Requirements for Architecture Design. 2013 20th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS), 119–128. https://doi.org/10.1109/ECBS.2013.14

Galster, M., Mirakhorli, M., Cleland-Huang, J., Burge, J. E., Franch, X., Roshandel, R., & Avgeriou, P. (2013). Views on software engineering from the twin peaks of requirements and architecture. ACM SIGSOFT Software Engineering Notes, 38(5), 40–42. https://doi.org/10.1145/2507288.2507323

Hall, J. G., Jackson, M., Laney, R. C., Nuseibeh, B., & Rapanotti, L. (2002). Relating software requirements and architectures using problem frames. Proceedings Ieee Joint International Conference on Requirements Engineering, 137–144.

Kruchten, P. (1995). Reference: Title: Architectural Blueprints—The "4+ 1" View Model of Software Architecture. IEEE Software, 12, 6.

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. A brief history. Computer, 36(6), 47–56.

Liu, D., & Mei, H. (n.d.). Mapping requirements to software architecture by feature-orientation. 8.

May, I. S. O. (2011). Systems and software engineering–architecture description. Technical Report. ISO/IEC/IEEE 42010.

Pimentel, J., Castro, J., Santos, E., & Finkelstein, A. (2012). Towards Requirements and Architecture Co-evolution. In M. Bajec & J. Eder (Eds.), Advanced Information Systems Engineering Workshops (Vol. 112, pp. 159–170). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-31069-0_14

Roeller, R., Lago, P., & van Vliet, H. (2006). Recovering architectural assumptions. 22.

Yu, L. L. E. (2001). From requirements to architectural design–using goals and scenarios. First International Workshop From Software Requirements to Architectures-STRAW, 1, 22.