VFAST Transactions on Software Engineering
http://vfast.org/journals/index.php/VTSE@ 2016 ISSN(e): 2309-3978;ISSN(p): 2411-6246
Volume 4, Number 1, January-December, 2016

pp. 27-33

SURVEY: DEALING NON-FUNCTIONAL REQUIREMENTS AT ARCHITECTURE LEVEL

SYED ROOHULLAH JAN[1], FAZLULLAH KHAN [1], MHHAMAMD TAHIR[1], SHAHZAD KHAN[1], FARMAN ULLAH[2]
[1] Department of Computer Sciences, Abdul Wali Khan University Mardan, Khyber Pakhtunkhwa, Pakistan
[2] Department of Computer Sciences, Bacha Khan University Mardan, Khyber Pakhtunkhwa, Pakistan
Email: {fazlullah, roohullahsyed, muhamamdtahir, shahzad} @akum.edu.pk

Revised August 2016

ABSTRACT. Non-functional requirements (NFRs) are being addressed by the architecture. NFRs are not focused properly as functional requirements (FRs) are dealt and focused. FRs are being taken under consideration at the early stage of software process development (such as architectural level). Usually, the NFRs are being focused at the end of the project, which does not fulfill the desired qualities. Early design decision is very important to achieve a strong connection between design and requirements, quality of a system and a consistent software product. Architecture and NFRs constraint each other therefore, they should be treated together. Runtime NFRs (such as performance, security and fault tolerance) and some of those which are not runtime (such as maintainability) should be considered at the architectural level. This paper presents a survey that emphasizes the integration of NFRs and architectural. We have analyzed the reported techniques on the basis of our evaluation criteria and have presented a comparison.

Keywords*:* Non-functional requrements, maintaibaility, performance, security, faut tolerance.

1. **Introduction.** Architecture is the high level design of a system. It is started after immediately the requirements phase. NFRs are the abstract requirements of a system that is to be built. Non-functional requirements are often called a system features. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements".[13] NFRs are rarely taken under consideration in most of the software development processes[22]. They are rarely considered due to their very high abstraction level, less support of tool and languages, complexity, and informality [3].In Addition in the software engineering, a tight relationship nonfunctional requirement (NFRs) and of software architectures (SAS) exists between [21]. NFRs have high abstraction level because they cannot be expressed quantitatively, such as performance, fault tolerance, availability, maintainability, etc. we can't write a test case to verify a system's " reliability or the absence of security vulnerabilities." Non-functional requirements are proved to be the reason of most modern applications failure [2,] [13]. Functional requirements are being incorporated into software architecture at early stages of process, at the end of process all of them are implemented to satisfy the requirements defined at early stages [1]. Most of the methods, techniques, languages and tools are introduced to capture the functional requirements. NFRs are not taken into account at the early stage of the software development process that affects the system qualities. More over Non-functional requirements are proved to be the reason of most modern applications failure. The continuing stream of business system disasters being reported in the press are rarely the result of a functional defect.[13]. Early design decision is required to strengthen the connection between requirements and design. Examples of design decisions are the decisions such as "we shall separate user interface from the rest of the application to make both user interface and application itself more easily modifiable" [18] [5] NFRs affect different activities and roles related to the software development process. One of the strongest links is with software architecture, especially architectural decision-making: NFRs often influence the system architecture more than functional requirements do[14] For instance, Zhu and Gorton state that "the rationale behind each

architecture decision is mostly about achieving certain NFRs" [15]Chung and Leite claim that "[NFRs] play a critical role during system development, serving as selection criteria for choosing among myriads of alternative designs and ultimate implementations"[16] and Ozkaya et al. say that "business goals and their associated quality attribute requirements strongly influence a system's architecture"[17]

Architecture addresses the characteristics of the system. The architectural decision taken for an application affects the NFRs. If performance is critical issue, we need to localize critical operations within a small number of subsystems with minimized inter communication. If security is demanded, we need to use a layered architecture. Similarly, if safety is important, we need to deploy small number of subsystems. If the focus of the application is on the maintainability, we need to use fine-grain, self-contained and replaceable components. NFRs cannot be evaluated without looking at the system as a whole because NFRs are described as attributes of the system that contributes to the overall quality of the product. This is the evident of the fact that the NFRs are very complex. On the other hand it is also evident that non-functional requirements are also very important since they contribute to the overall quality of the resulting system [11][19].

Nowadays it is demanded to careful capture and models the requirements and architectural design in the early stage of the software process development. It is due to the increasing size, complexity, distribution and heterogeneity [6]. Achieving the quality of the product is very important. Various attributes are generally considered important for obtaining a software design of good quality-various "illities" (maintainability, portability, testability, and traceability), various nesses (correctness, robustness)[19]. An interesting distinction is the one between quality attributes at run-time (performance, security, availability functionality, usability), those not at run-time (modifiability, portability, reusability, integrability and testability) and those related to the architecture's intrinsic qualities (conceptual integrity, correctness, completeness and build ability) [12]. FRs, NFRs must be realized through the architecture [4] [8].Furthermore NFRs is an Important step to close the gap between requirement engineering and Software engineering [24-28].

Non-functional requirements compete and conflict each other (such as maintainability and performance). Using large-grain components improves performance but reduces maintainability. Similarly, introducing redundant data improves availability but makes security more difficult. If both performance and maintainability is required, we need a compromised solution. For example to compromise with availability, we need to sacrifice in rush hours and include some down time. A tradeoff is needed to involve finding an optimal solution. The development of large software systems is the need of almost every organization. Because of the involvement of the non-functional requirements, software architecture design has become an important step in large software development [10]..

This paper presents a survey that emphasizes the integration of NFRs and architectural. Architecture and NFRs constraint each other therefore, they should be treated together. The survey is made on the bases of certain evaluation criteria. This evaluation criteria is defined in the parameters shape. The rest of the paper is as follow: Section 2 discusses the existing techniques of integrating the NFRs and architecture in detail. In section 3 we introduce the evaluation criteria through which we analyze the existing techniques of integrating the NFRs and architecture. A thorough analysis of the existing techniques is presented in section 4. Finally, section 5 concludes the paper.

2. Techniques for Integrating Non-functional Requirements and Architecture. Software architecture is the first step after the requirements elicitation of software development process. The basic purpose of the architecture is to ensure the productivity of the functional requirements (FRs) at the end of the product. The non-functional requirements (NFRs) are usually focused at the end of the product due to which the qualities of the system are affected. A number of techniques are reported to deal both the FRs and NFRs together in order to achieve the functionalities and qualities at the end of the product. The focus of technique may vary from each other, for example one technique may provide some practical solution, whereas another technique may only argue the issue or provide a framework or an approach to cater the issue of integration. Furthermore all the techniques may not applicable to cater all the NFRs (such as 'illities', nesses, those at runtime and those not at runtime). In order to get an insight into the existing literature on the issue of integrating architecture and non-functional requirements (NFRs), we carried out a thorough survey and critical analyses of relating software architecture and non-functional requiremnts techniques as discussed below in the following section.

2.1. Incorporating Non-Functional Requirements into Software Architectures. Nelson et al [1] presented an approach in which transactional and non-functional requirements (NFRs) are formally incorporated into dynamic software architecture. An appointment system is also proposed in order to demonstrate how this approach can be utilized in a real application. Furthermore, three NFRs properties have been chosen to focus

which are safety, availability and performance.

2.2 Use-Case And Scenario-Based Approach To Represent Nfrs And Architectural Policies. C Lopes and H Astudillo [2] present a use case-based approach to describe NFRs. The approach is based on the concepts of architectural policies. This approach is used to employ use cases and scenarios to describe non-functional requirements, relate them to specific functional use case features, and serve as input to the architecture elaboration process.

2.3 A Framework For Building Non-Functional Software Architecture. Nelson et al [3] propose a Parmenides framework that defines how to deal with NFRs within the software development process. The framework describes the NFRs, their refinement, mapping into actual implementation, and integration with functional requirements (FRs). The framework defines precisely how NFRs are expressed and integrated into an architectural-based development. Furthermore, it defines two methodologies for describing NFRs, an integration strategy, a set of refinement rules and a mapping strategy[29-33].

2.4 Functional Requirements, Non-Functional Requirements and Architecture Should not be Separated. The position which is put forward by the Barbara et al [4] in this paper is that FRs, NFRs and architecture must be focused together, because they constraint each other. In addition, the architecture addresses the NFRs in the early stage of design. It is also well known that both FRs and NFRs must be realized through the architecture. In this paper authors argue that FRs, NFRs and architectural design must be developed in a tightly integrated approach. The authors motivate their approach with an example in this paper.

2.5 From Requirements To Architectural Design Using Goals And Scenarios. Eric Yu [5] proposed GRL (Goal Oriented Requirement Language) in this paper. GRL is a language for supporting goal and agent oriented modeling and reasoning of requirements, especially for dealing with NFRs. A UCM (a scenario oriented architectural notation) is also presented in this paper. Goals are used in the refinement of non-functional (NFRs) and functional requirements (FRs), exploring alternatives, their operationalization into architectural constructs.

2.6 Reconciling Software Requirements And Architecture: The CBSP Approach. Paul et al [6] propose an approach known as CBSP (Component, Bus, System and Property. The aim is to reconcile software requirement and architecture. Furthermore, CBSP minimize the gap between high level requirement and architectural descriptions. CBSP also allows identifying and isolating 'ilities' for the purpose of improving nun-functional properties and allows capturing and maintaining arbitrarily complex relationship between requirement and architectural artifacts.

2.7 A Framework for the integration of functional and non-functional analyses of software architectures. Cortellessa et al [7] have proposed a framework to support the integration of functional and non-functional analyses of software system at architectural level. The proposed framework lies on an XML-based integration core and semantics relation between the models are represented. The scope of the paper is limited to the integration of two methodologies, the XML model and semantic rules.

2.8 An Experience-Based Approach For Integrating Architecture and Requirements Engineering. Barbara et al [8] argued that the process of integrating FRs, NFRs and architectural options (AO) should be fundamentally based on experience. The authors have presented a comprehensive approach to convert the major issues related to the FRs and NFRs and AOs. The proposed approach supports the elicitation, specification and design activity.

2.9 Towards Improved Traceabilityof Non-Functional Requirements. Cleland-Huang put forward the position of tracing the NFRs. He also explains the challenge of traceability is due to the interdependencies between NFRs and architecture. Three critical areas of traceability are identified by the author first, then existing techniques are evaluated and finally a holistic approach is proposed through different ways.

2.10 From Requirements To Architecture: The State Of Art In Software Architecture Design. In this paper Lin Liao presents a thorough overview of the methodologies that are using currently on software architecture design. It also focuses on the involvement of non-functional requirements with architecture.

3. Evaluation Criteria For Existing Techniques. In order to analyze and evaluate the existing trend of the integration of NFRs and architecture, we need some evaluation criteria. We have defined the evaluation criteria presented in table 1 for comparing the techniques. We suggest the evaluation criteria described in table 1 on the basis of thoroughly analyzing the existing approaches. The evaluation criteria selected relate almost all the approaches up to some extent. The evaluation criteria and their possible values are mentioned below in detail. The possible values that are possessed by the evaluation criteria are assigned to the corresponding techniques after the thorough analyses of the existing techniques. Moreover, the analyses evaluate the

strengths and weaknesses of various techniques and provide a concise review of the related literature based on the criteria presented in table 1. We propose the reported techniques below.

3.1 Quality Attributes. Quality attributes conflict each other. Therefore, all the quality attributes cannot be focused at the same time. For example, we improve performance, the maintainability is decreased. It is important to know the quality attributes are focused by the techniques. It has possible two values Yes or No.

3.2 Framework. Framework is nothing than a number a blocks and organizing them in order to depict a specific scenario and idea to cater for the problem. The organization of blocks is to tackle the issue of integrating the NFRs and architecture. It describes the name of framework.

3.3 Approach. A way of dealing with non-functional requirements is proposed by techniques that describe the approach of the techniques to tackle the issue of integrating the NFRs and architecture. It describes the name of framework.

3.4 Case Study. The proposed approach is either verified through an application or not is focused by this evaluation criterion. Here we mention the case study or 'No', if not implemented.

3.5 Notation. In this evaluation criterion we focus on the approach proposed by technique, using language or some notation to argue their approach. It either names the language (or notation) or states No.

3.6 Methodology. It describes the method, which has been adapted by the technique to illustrate the proposed approach. The values either name the method or No.

3.7 Tools. This criterion tells the use of tool described in the techniques or not. It has possible two values, Yes if used, No if not used. This criterion is very important because through this we can find out the use of tools in the description of the non-functional requirements [34-36].

The evaluation criteria discussed above are listed in a table 1 to illustrate thoroughly in order to comprehend the discussion based on these criteria. We have focused seven parameters to evaluate, because these are the factors that are discussed almost in all the techniques that are surveyed.

4. Analysis and Discussion. Table 2 presents the comparison of the above techniques based on the evaluation criteria described in table 1. In the first row of the table we have given the parameters used as evaluation criteria.

Table-I: Criteria To Evaluate Integration Of NFRs and Architecture.

NO	PARAMETER	DESCRIPTION	POSSIBLE VALUES
01	Quality Attribute(S)	Specific Quality Attributes, are Focused	Yes, No
02	Framework	The Proposed Framework	Name of Framework. N/A (If Not Used)
03	Approach	The Proposed Approach	Name of Approach N/A (If Not Used)
04	Case Study	A Practical Example, Which Is Taken Into Account To Describe The Proposed Approach	System/Application Name Which is Implemented, N/A (If not Implemented)
05	Notation	The Proposed Approach Used Notation Or Not	Name Of Notation, N/A (If Not Used)
06	Methodology	The Method Of Presenting The Proposed Approach	Name Of Method, N/A (If Argued)
07	Tool	The Tool Used In Proposed Idea	Yes, No

From the above comparison we conclude that some of the quality attributes could be focused at architectural level because the quality attributes compete and conflict each other. Nelson et al [1] [3] focus the runtime quality attributes to achieve the applicability. The applicability of the integration of software architecture and nfrs is evident from the comparison. Similarly, only nelson et al [1] [3] [37] propose a

specific framework.. After analyzing the reported techniques we also conclude that, almost all the techniques practically implemented their proposed approach through a case study. It means the integration of nfrs and architecture is applicable to practical application. This comparison of the existing approaches shows that nfrs have very less support of notations. There is no specific notation for non-functional requirements. Therefore, only block diagram and use case diagram could describe this issue up to some extent [23] [38]. It is also evident from the comparison that none of the techniques used tool to illustrate the nfrs. Therefore, we can say that nfrs have very rare support of the tools as well. It is also concluded that almost all the techniques propose a specific way to cater the issue of the integration of nfrs and architecture. It is also evident that the focused issue is not being discussed through a practical approach, and shows that this issue can only be argued. That is why the techniques focus scenarios and experience-based approaches.

Table-II: Analyses of The Existing Trend of Integrating Architecture and NFRs

S#	Technique	NF Attributes	Framework	Approach	Case- study	Notation	Methodology	Tool
1	Nelson et al (2000)	Yes	ZCL Framework	N/A	An appointment system	N/A	Configuration model (CL)	No
2	Lopez and Hernan	No	N/A	Integrating NFRs with use-case	Stock behavior system	Use case diagram	Scenario-based Approach	No
3	Nelson et al (2001)	Yes	Parmenides framework	N/A	N/A	Block diagram	Mapping strategy	No
4	Barbara et al (2002)	No	N/A	Experience-based	Navigation system of rocket	Block diagram	Experience-based	No
5	Lin Liu Eric Yu	No	N/A	Scenario-based Approach	Mobile telecom system	N/A	Use Case Maps (UCM)	No
6	Paul et al	No	N/A	CBSP approach	Natural disaster	Block diagram	CBSP taxonomy	No
7	Cortellesa et al	No	N/A	XML-based Approach	Set & counter application	N/A	Integrating NFRs & FRs	No
8	Barbara et al (2003)	No	N/A	Experience-based	N/A	Class diagram	N/A	No
9	Jane Cleland-Huang	Yes	N/A	Goal-oriented	A telephony system	Block diagram	GCT model	No
10	Lin Liao	No	N/A	Survey-based	N/A	Goal graph	N/A	No

Conclusion. Software architecture is widely used to address the functional requirements of the application. A number of techniques, methodologies, languages, tools, and processes are introduced to cater the functional requirements of the software. An important consideration is that, the software architecture addresses the non-functional requirements as well. Specially, runtime qualities of the software system (such as performance, security, availability and fault tolerance) are thoroughly addressed by the architecture.

In this paper, we have surveyed the integrating techniques of the software architecture and NFRs in order to check their applicability in the domain of real life. We have analyzed the reported techniques on the basis of our evaluation criteria and have presented a comparison. We conclude that nun-functional requirements should also take into account at architectural level. Functional requirements and non-functional requirements must be deal together in order to achieve better interconnection between design and requirements.

REFERENCES

[1] Rosa, N. S., Justo, G. R., & Cunha, P. R. (2000). Incorporating non-functional requirements into software architectures. In Parallel and Distributed Processing (pp. 1009-1018). Springer Berlin Heidelberg.

[2] López, C., & Astudillo, H. (2005). Use case-and scenario-based approach to represent nfrs and architectural policies. In Proceedings of 2nd International Workshop on Use Case Modeling (WUsCaM-2005), Use Cases in Model-Driven Software Engineering Held in conjunction with Models.

[3] Rosa, N. S., Justo, G. R., & Cunha, P. R. (2001, March). A framework for building non-functional software architectures. In Proceedings of the 2001 ACM symposium on Applied computing (pp. 141-147). ACM.

[4] B. Paech, et,al., (2002), Functional requirements, non-functional requirements and architecture specification cannot be separated – A position paper".

[5] Liu, L., & Yu, E. (2001, May). From requirements to architectural design-using goals and scenarios. In ICSE-2001 Workshop: From Software Requirements to Architectures (STRAW 2001) May (pp. 22-30). Workshop (STRAW 2001), Toronto, Canada.

[6] Khan, F., Khan, Farman, Jabeen, Q., Jan, S.R., and Khan, S., "Applications, Limitations, and Improvements in Visible Light Communication Systems" in VAWKUM Transactions on Computer Sciences.

[7] Cortellessa, V., Di Marco, A., Inverardi, P., Mancinelli, F., & Pelliccione, P. (2005). A framework for the integration of functional and non-functional analysis of software architectures. Electronic Notes in Theoretical Computer Science, 116, 31-44.

[8] Jan, M. A., Nanda, P., He, X., & Liu, R. P. (2013). Enhancing lifetime and quality of data in cluster-based hierarchical routing protocol for wireless sensor network. High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on (pp. 1400-1407).

[9] Paech, B., von Knethen, A., Dörr, J., Bayer, J., Kerkow, D., Kolb, R., & München, T. U. (2003, May). An Experience-Based Approach for Integrating Architecture and Requirements Engineering. In STRAW (pp. 142-149).

[10] Jan, M.A., Nanda, P., & He, X. (2013). Energy Evaluation Model for an Improved Centralized Clustering Hierarchical Algorithm in WSN in Wired/Wireless Internet Communication, Lecture Notes in Computer Science. (pp. 154–167), Springer, Berlin, Germany.

[11] Kasser, J. E., & Massie, A. (2001, July). A framework for a systems engineering body of knowledge. In 11th International Symposium of the INCOSE, INCOSE, Melbourne, Australia.

[12] Jan, M.A., Nanda, P., He, X., & Liu, R. P. (2014)., "A robust authentication scheme for observing resources in the internet of things environment" in 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 205-211, IEEE

[13] Liao, L. (2002). From Requirements to Architecture: The State of the Art in Software Architecture Design. Department of Computer Science and Engineering, University of Washington, 1-13.

[14] Burge, J., & Brown, D. (2002). NFRs: Fact or fiction.

[15] Jan, M.A., & Khan, M. (2013). A Survey of Cluster-based Hierarchical Routing Protocols. IRACST–International Journal of Computer Networks and Wireless Communications (IJCNWC). Vol.3, 138-143.

[16] Cleland-Huang, J. (2005, November). Toward improved traceability of non-functional requirements. In Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering (pp. 14-19). ACM.

[17] Kaur, H., & Sharma, A. (2014). NFR" s: Definition. International Journal, 4(7).

[18] Pohl, K., & Rupp, C. (2011). Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant. Rocky Nook, Inc.

[19] Jan, M.A., & Khan, M. (2013). "Denial of Service Attacks and Their Countermeasures in WSN". IRACST–International Journal of Computer Networks and Wireless Communications (IJCNWC). Vol.3.

[20] Zhu, L., & Gorton, I. (2007, May). Uml profiles for design decisions and non-functional requirements. In Proceedings of the Second Workshop on Sharing and Reusing Architectural Knowledge Architecture, Rationale, and Design intent (p. 8). IEEE Computer Society.

[21] Khan. F. (2014). Throughput & Fairness Improvement in Mobile Ad hoc Networks. the 27th Annual Canadian Conference on Electrical and Computer Engineering (p. 6). Toronto, Canada: IEEE Toronto.

[22] Shahzad Khan, F. K. (2015). Delay and Throughput Improvemenet in Wireless Sensor and Actor Networks. 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW) (pp. 1-8). Riyadh: IEEE Riyad Chapter.

[23] Chung, L., & do Prado Leite, J. C. S. (2009). On non-functional requirements in software engineering. In Conceptual modeling: Foundations and applications (pp. 363-379). Springer Berlin Heidelberg.

[24] Ozkaya, I., Bass, L., Sangwan, R. S., & Nord, R. L. (2008). Making practical use of quality attribute information. IEEE Software, 25(2), 25.

[25] Khan, F., Nakagwa, K.. (2013). Comparative Study of Spectrum Sensing Techniques in Cognitive Radio Networks. in IEEE World Congress on Communication and Information Technologies (p. 8). Tunisia: IEEE Tunisia.

[26] Tarvainen, P. (2008). Adaptability evaluation at software architecture level. The Open Software Engineering Journal, 2(1).

[27] Jan, M.A., Nanda, P., He, X., & Liu, R. P. (2014). "PASCCC: Priority-based application-specific congestion control clustering protocol". Computer Networks, vol. 74, 92-102

[28] Chung, L., Nixon, B. A., Yu, E., & Mylopoulos, J. (2012). Non-functional requirements in software engineering (Vol. 5). Springer Science & Business Media.

[29] Chung, L., Nixon, B. A., Yu, E., & Mylopoulos, J. (2012). Non-functional requirements in software engineering (Vol. 5). Springer Science & Business Media.

[30] Khan. F. (2014). Secure Communication and Routing Architecture in Wireless Sensor Networks. the 3rd Global Conference on Consumer Electronics (GCCE) (p. 4). Tokyo, Japan: IEEE Tokyo.

[31] Topic "Non-functional Requirements in Architectural Decision Making" aVILABLE AT ; http://www.infoq.com/articles/non-functional-requirements-in-architectural-decision-making Accessed on 10/04/2016

[32] Hasan, M. M., Loucopoulos, P., & Nikolaidou, M. (2014). Classification and qualitative analysis of non-functional requirements approaches. In Enterprise, Business-Process and Information Systems Modeling (pp. 348-362). Springer Berlin Heidelberg.

[33] Khan, F., Bashir, F. (2012). Dual Head Clustering Scheme in Wireless Sensor Networks. in the IEEE International Conference on Emerging Technologies (pp. 1-8). Islamabad: IEEE Islamabad.

[34] Jan, M.A., Nanda, P., He, X., & Liu, R. P. (2015). "A Sybil Attack Detection Scheme for a Centralized Clustering-based Hierarchical Network," in Trustcom/BigDataSE/ISPA, vol.1, PP-318-325, IEEE.

[35] Losavio, F., Matteo, A., & Pacilli Camejo, I. (2014). Unified Process for Domain Analysis integrating Quality, Aspects and Goals. CLEI Electronic Journal, 17(2), 2-2.

[36] Yrjönen, A., & Merilinna, J. (2009, October). Extending the NFR Framework with Measurable Non-Functional Requirements. In NFPinDSML@ MoDELS.

[37] Khan, F., Kamal, S. A. (2013). Fairness Improvement in long-chain Multi-hop Wireless Adhoc Networks. International Conference on Connected Vehicles & Expo (pp. 1-8). Las Vegus: IEEE Las Vegus, USA.

[38] Grunbacher, P., Egyed, A., & Medvidovic, N. (2001). Reconciling software requirements and architectures: the CBSP approach. In Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on (pp. 202-211). IEEE.