

Integrating Architecture-based Trade-off Analysis into the design process through tool-assisted modelling

David Colquitt¹, John Leaney^{1,2}, Tim O'Neill¹

1. Faculty of Engineering

2. Faculty of Information Technology

University of Technology, Sydney

[David.Colquitt | John.Leaney | Tim.ONeill]@uts.edu.au

Abstract

The engineering of complex computer-based systems is commonly underpinned by the development of various models attempting to structure both the problem at hand and the various solution alternatives. Typically this results in a combinatorial explosion of informational and design elements that are both conceptual and resource intensive.

The following position paper presents a model-based approach to the design of complex computer-based systems, incorporating an effective way of structuring and architecturally trading-off the requirements, both functional and non-functional.

This model-based approach is currently being applied as the foundation of several real-world model-based development, design, evolution and engineering projects.

1. Introduction

Architecture-based engineering has evolved from the principals that the properties of the end-system artefact can be predicted by carefully formulating and analysing models of specific aspects of the system. Theoretically there are at least as many models of a system as there are logical ways of partitioning and reasoning about the system [1] consequently numerous approaches exist for deriving models of system function, structure, quality attributes and behaviour.

The difficulty lies in combining the various models to satisfy the inherent design need for a recursive method of:

- (i) identifying system use and constraints,
- (ii) specifying function and structure to facilitate the use,
- (iii) associating the design choice and the constraints it affects, and,
- (iv) testing the function does not violate the constraints.

The following presents the author's model-based approach to engineering complex computer based systems including a method by which architecture-based

trade-off analysis can be tightly integrated into the general design process, adding rigour to what has previously been a largely heuristic exercise.

2. Model-based Architecture Engineering

Empirical and theoretical studies both support the notion that the further along the system design and development path, the more costly it becomes to change design decisions and the more committed stakeholders are to the solution [2, 3]. The tangible saving in terms of both effort and money has created a decisive need to reason about the finished systems properties using a baseline of only the earliest design artefacts. Systems architecture is one such discipline that seeks to model the abstract form of the system and reason about the relationship between the qualities of the end system artefact and the structure of the design representation.

Naturally the need to design for specific properties also requires a method or process by which these same properties can be tested for in architectural representations, giving rise to the field of architecture-based analysis. Architecture trade-off analysis (ATA) evolved from earlier practices oriented towards candidate design selection to methods designed to be integrated into regular design processes.

The authors believe that a rigorous Architecture-based Engineering process should involve the following steps (as illustrated in Figure 1):

1. Model Business Drivers and Quality Requirements
2. Develop Goal-based Requirements
3. Model "best-practice" Architectural Styles
4. Perform Architecture-based Trade-off Analysis
5. Combine the Requirements models and the Style models
6. Exploit the Evolutionary advantages

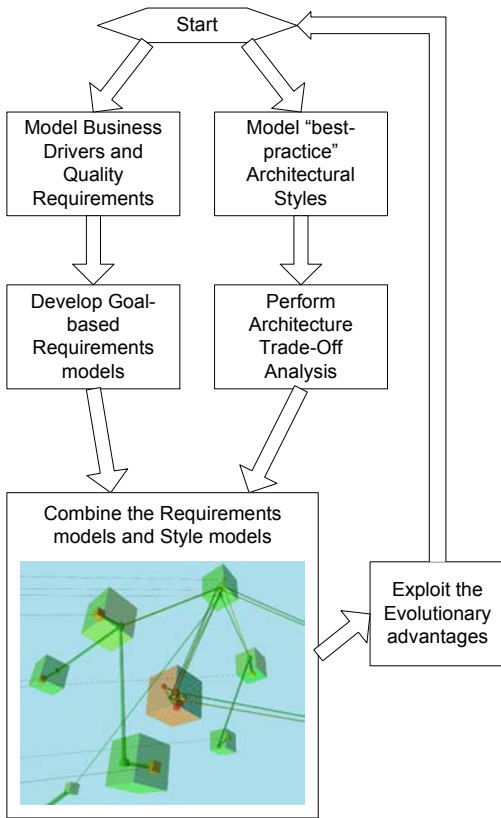


Figure 1 – Architecture-based Engineering Process

2.1 Model Business Drivers and Quality Requirements

Given the pervasive nature of computer-based systems in most organisations there is a clear need to understand the business pressures brought to bear in creating the system need. It has been the author’s experience that these drivers are extremely useful in deriving the constraints for the system. Examples from the telecommunications domain are the need to counter eroding margins on existing service sets constraining the dynamic performance required of the system and the need to converge technology sets constraining platforms and available resources [4]. In capturing these business constraints the business analysis techniques such as PESTLE and SWOT are well accepted and can be used alongside interviewing.

Quality attributes (or non-functional requirements) are another essential facet of system constraints. Firstly, they need to be elicited within the system context because of the lexical diversity that is likely to exist amongst the stakeholder group. Secondly, important quality attributes need to be modelled in order to understand the interactions and trade-offs that exist, such as the relationship between security and performance, and between scalability and availability [5]. Streamlined

applications of the processes from the IEEE [6] and SEI [7] have proven effective in developing testable quality attribute models.

2.2 Develop Goal-based Requirements

Goal-based requirements is a discipline born of the need to derive requirements for complex systems in a way that can be related to business drivers, is available to all stakeholder groups, reduces the overheads of the requirements exercise and importantly establishes relationships between design goals through the constraint-based interactions [8]. The goal model essentially defines the functional capability space of the system and can be used as the primary functional “specification” of a system without the need for a cumbersome SRS and it’s requisite copious list of “shalls”.

2.3 Model “best-practice” Architectural Styles

Similar to design patterns, architecture styles present loose configurations of components and connections, with specific properties, as being capable of satisfying design quality goals. However, unlike design patterns, architectural styles are perhaps the earliest design decisions made, dealing with the abstract arrangement of components and connections, rather than the more concrete notions of software objects.

By examining the reported quality attributes of other systems and the subsequent system structure and properties credited with providing these qualities, some degree of direction can be gained as to how to imbue these properties into the system at hand.

Unfortunately, the existing difficulty with styles as presented by the SEI [9] is the generally loose descriptions of the structure required to adhere to the style. Similarly it is a fairly coarse grained method of examining whether a system is likely to have a set of required quality attributes. The properties of the style are likely to change once it is instantiated in the structure of the target system and additionally there is no body of knowledge surrounding how styles interact and what the transitive effects of their quality attributes are. It is naïve to think that a designer can just cross-reference a list of desired system qualities and choose the architectural styles that will fulfil them, a la ABAS [10].

Our experience has shown that using a flexible architecture-based modelling tool can aid in developing models of the system to add quantitative evidence behind the style argument [11]. Existing style-based design methods are at best heuristic, qualitative and vague in nature. Would you really trust their output?

2.4 Perform Architecture-based Trade-off Analysis

Architecture trade-off analysis (ATA) is designed to assess the extent to which quality concerns have been addressed in the system architecture. There are two different techniques; one is to develop quantitative simulation models = labelled measuring techniques, the other to develop ways of qualitatively examining the system through the use of scenarios, checklists and questionnaires = labelled questioning techniques.

The amount of effort required to develop measurement-based models of the system, compounded by their general focus on only one quality attribute has led to the proliferation of approaches applying questioning techniques. Despite aspiring to quickly evaluating early system architectures, existing ATA methods are still human-resource intensive and oriented towards architectures that have a significant amount of maturity; indicating most important design decisions have already been made.

It is contended that an effective ATA method is one that uses a flexible architecture-based modelling tool that provides the capability to quickly and inexpensively develop quantitative models of system behaviour from reasonably abstract representations. The amount of detail obtained should be commensurate with the level or maturity of the design and knowledge of its critical properties. The method should be used to evaluate other systems with similar requirements in order to identify clear architectural styles that can be applied to satisfy the requirements of the functional capability space.

In this way it is believed that an appropriate architecture-based analysis method can be underpinned by the assurance of quantitative methods without placing inordinate resource demands on the design team. It would also overcome the lack of attribute interactions commonly cited of specialised measurement-based models like time-coloured Petri nets for performance [12].

The decreased effort in producing the analysis would allow the trade-off to take place in concert with design steps rather than as some exit criteria from a specific design stage. This would provide a much needed link between discrete design decisions and the consequences these have for the system to fulfil its constraint obligations. The authors are trialling one such method and toolset, ABACUS [13], and already it is showing promising results at closing the feedback loop [14].

2.5 Combine the Requirements models and the Style models

The final stage of the process is to combine the information in the requirements models (business, quality and goal-based), which are problem and formulation

based and the information in the style models (instances and trade-off) which are essentially solution based.

The requirements models offer functions and associated constraints which can then be associated with the constraints each style satisfies, the result is a new system model that includes both functional and non-functional concerns and can be verified in both a quantitative and qualitative manner.

Having populated a new architectural model of the system, the same techniques of assessment applied to the best practice models in the earlier ATA's can be used on the target system. The normalised way of modelling the architectures provides the architects with the capability to evaluate the system comparatively against the "best practice" systems as well as draw absolute, quantitative conclusions about the systems capability.

2.6 Exploit the Evolutionary advantages

Within complex systems the most common methodological approaches are iterative – incremental, suggesting the need for techniques that can be applied continuously throughout the design process. The advantages of tool-supported ATA cited in section 2.4, such as fast model development, strong association between design decisions and consequent system capability and measurable results, all support such a life-cycle approach. Further to this it also provides the capability to evolve the model of the system throughout the system life-time, not just development lifecycle.

Architectural erosion is a commonplace characteristic of systems that receive significant maintenance and expansion, post deployment. Given the investment involved in developing them, this is the case for most complex systems. Having the capability to not only model the structure of the system, but also the behaviour provides a powerful way of evaluating future system enhancements.

3. Conclusion

We have presented a model-based approach to engineering complex computer based systems including a method by which architecture-based trade-off analysis can be tightly integrated into the general design process. We believe this method adds rigour to what has previously been a largely heuristic exercise and is practical for complex system projects where resourcing and time is limited.

4. Acknowledgements

The discussions with, patience of, and funding from our colleagues at UTS, Alcatel Australia Ltd and Avolution Pty Ltd has contributed to this position.

5. References

1. O'Neill, T., J. Leaney, and M. Denford. *Current Developments in the Definition and Description of System Architectures*. in *Engineering of Computer-Based Systems (ECBS)*. 2004. Brno.
2. Boehm, B.W., *A spiral model of software development and enhancement*. Computer, 1988. **21**(0018-9162): p. 61-72.
3. Schön, D.A., *The reflective practitioner : how professionals think in action*. 1991, Aldershot, England: Arena. x, 374.
4. Colquitt, D., *Next Generation Networks, industry and technical drivers*. 2003, University of Technology, Sydney: Sydney, Australia.
5. Soliman, J., *Quality Requirements*. 2003, University of Technology, Sydney: Sydney, Australia.
6. *IEEE standard for a software quality metrics methodology*, in *IEEE Std 1061-1998*. 1998.
7. Kazman, R., et al. *The architecture tradeoff analysis method*. in *Engineering of Complex Computer Systems, 1998. ICECCS '98. Proceedings. Fourth IEEE International Conference on*. 1998.
8. van Lamsweerde, A. *Goal-oriented requirements engineering: a guided tour*. in *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*. 2001.
9. Bass, L., P. Clements, and R. Kazman, *Software architecture in practice*. 2nd ed. ed. 2003: Addison-Wesley.
10. Klein, M. and R. Kazman, *Attribute-Based Architecture Styles*. 1999, Software Engineering Institute: Pittsburgh.
11. O'Neill, T., *System Evolution using the ABACUS Methodology*. 2003, Avolution: Sydney.
12. Abowd, G., et al., *Recommended Best Industrial Practice for Software Architecture Evaluation*. 1997, Software Engineering Institute (SEI), Carnegie-Mellon University (CMU): Pittsburgh, Pennsylvania.
13. <http://www.avolution.com.au>.
14. Leaney, J., M. Denford, and T. O'Neill. *Enabling Optimisation in the design of Complex Computer based systems*. in *17th IEEE International Conference on Engineering of Computer Based Systems*. 2004. Brno Czech Republic.