Proc. of the 1st International Software Architecture Workshop (ISAW), 1996, pp. 152-154

A Stakeholder-Centric Software Architecture Analysis Approach

Sonia Bot, Chung-Horng Lung, Mark Farrell
Nortel
P.O. Box 3711, Station C
Ottawa, Ontario, Canada K1Y 4H7
{sdbot | lung | mark_farrell}@nortel.ca

1. Position

SAAM (Software Architecture Analysis Method) [4, 5] currently provides a valuable first order analysis of software architecture robustness based on scenarios [2] and the interaction between non-functional quality attributes such as scalability, modifiability, integrability, portability, performance, and reliability. However, its approach towards representing the objectives of the stakeholder[1] and tracing them through the analysis, ensuring representation, balance, coverage, dependencies, and interpretation of the analysis results is somewhat oversimplified for use in large scale telecommunications software systems.

At Nortel’s Software Engineering Analysis Lab, a modelling technique was developed, when applied with SAAM ensures the rigor required for ensuring that stakeholder objectives are explicitly addressed and traced. Our model is practical in representing and validating software architectures across multiple stakeholders. The model serves as our engine for analysis and provides continuity for the assets generated and evolved throughout the software lifecycle**.**

2. Working Definition of Software Architecture

Software architecture has been defined in various ways. These definitions focus on architectural representation, but the definition does not clearly address the full range of evaluation issues associated with a software architecture, and in particular, the needs of the stakeholders.

At Nortel’s Software Engineering Analysis Lab, the working definition of software architecture is adopted from [3]. A software system architecture comprises:

- A collection of software and system components, connections, and constraints
- A collection of system stakeholders' need statements
- A rationale which demonstrates that the components, connections, and constraints define a system that, if implemented, would satisfy the collection of system stakeholders' need statements

3. Stakeholder-Centric Modelling

The stakeholder-centric model used to drive and focus software architecture analysis is simple and practical. From the onset, the context of the analysis is set by identifying the various stakeholders and their objectives. Each stakeholder represents a context.

Quality Function Deployment (QFD). Software QFD [1] relational matrices are used. QFD forms a structure that separates concepts on a scale from abstract to concrete, defines their overlapping relationships, prioritizes decisions, and defines measures for success. Stakeholders and their objectives have traceable priorities throughout the analysis of an architecture.[2] (Refer to figures 1 and 2.) If a stakeholder objective changes (for example, in terms of priority), the changes automatically adjust throughout the chain of related matrices, say all the way down to an artifact such as a particular scenario.

Forming the Relationships. The stakeholder objectives are translated into their appropriate architectural or quality objectives. For example, in one analysis of a very large telecommunications system currently being designed, the stakeholder objectives were mapped directly onto architectural objectives. (Refer to figure 1). In this case, the first pass results of the analysis identified the non-functional quality attributes (and hence the resulting quality objectives) that were of highest risk to the system. Subsequent analysis passes were driven by stakeholder objectives in the context of the quality objectives. (Refer to figure 2.) In another case, the quality objectives were known upfront, and hence the stakeholder objectives were mapped to quality objectives and then to the architectural objectives. (Refer to figure 2.) Depending on the stakeholders and the concreteness of their objectives, different matrix linking models may be adopted.

1. Stakeholders include product choosers, end users, architects,

tions system, a basic call service (that is, a basic need) must exist. Typically stakeholders do not explicitly express basic needs in their objectives. Basic needs are necessary to do an architectural analysis as they provide a baseline for the analysis. Basic needs are domain and product specific, and are reusable from analysis to analysis.

Modelling Breadth. The mapping and linking of the matrices continues until the artifacts for the analysis are exhausted. For example, objectives and scenarios are the artifacts used throughout the initial studies. There is nothing impeding the linking of matrices to other artifacts such as architectural components and various architectural views.

Modelling Depth. The size of the matrices relate to modelling depth. That is, the greater the number of rows and columns in the matrices, the greater the detail. This can also be envisioned as trees. (Refer to figure 3.) In this example, the objectives and scenarios can be decomposed into more detailed objectives and scenarios. Slices across the trees represent various levels of abstraction, and can be used as a guide for future similar analyses.

Traceability. The linking of the QFD relational matrices and their relationships is the mechanism for forward and backward traceability. This provides cohesion amongst the artifacts. As well, the objectives and scenarios function as engines throughout the analysis. This can be extended to apply to the rest of the development lifecycle.

4. Summary

Using stakeholder-centric modelling with SAAM is bringing significant benefits to the analysis[1] of large scale telecommunications software systems. The analyses are focussed to the original objectives and contexts, thus reducing risk of analysis drift and imbalance.

The use of QFD relational matrices provided quantification, ease of viewing, and reduction of complexity of information which was typically treated in an ad hoc manner, or was totally ignored. In one example, the advantage of coverage issues was highlighted where for a particular objective addressing performance, the QFD results indicated that twice as many scenarios of particular classes were still required in order to achieve credible analysis results for the context of a particular stakeholder.

At Nortel's Software Engineering Analysis Lab, there other teams that are working on quantitative analysis methods for high-level design and code. It is our intent to extend the stakeholder-centric model to cover lifecycle end-to-end analysis for Nortel's various software products.

Acknowledgments

We would like to express our gratitude to Rick Kazman (University of Waterloo, Waterloo, Ontario, Canada) for his graciously providing insight, answering questions about specific issues, and providing comments on the treatment of his special interests.

1. These benefits also apply to the re-engineering of systems.

References

[1] M. Boushi. *Towards Better Object-Oriented Software with Quality Function Deployment*. (Trans. 6th Symp. on Quality Function Deployment).

[2] J.M. Carroll (editor). *Scenario-Based Design: Envisioning Work and Technology in System Development*. (John Wiley & Sons, Inc., New York, 1995).

[3] C. Gacek, A. Abd-Allah, B. Clark, B. Boehm. *On the Definition of Software System Architecture*. (ICSE 17 Software Architecture Workshop, April 1995).

[4] R. Kazman, L. Bass, G. Abowd, M. Webb. *SAAM: A Method for Analyzing the Properties Software Architectures.* (Proc. of ICSE 16, May 1994, 81-90).

[5] R. Kazman, G. Abowd, L. Bass, P. Clements. *Scenario-Based Analysis of Software Architecture*. (to appear in IEEE Software, 1996).

Figure 1. Interlinked relational matrices (example #1).

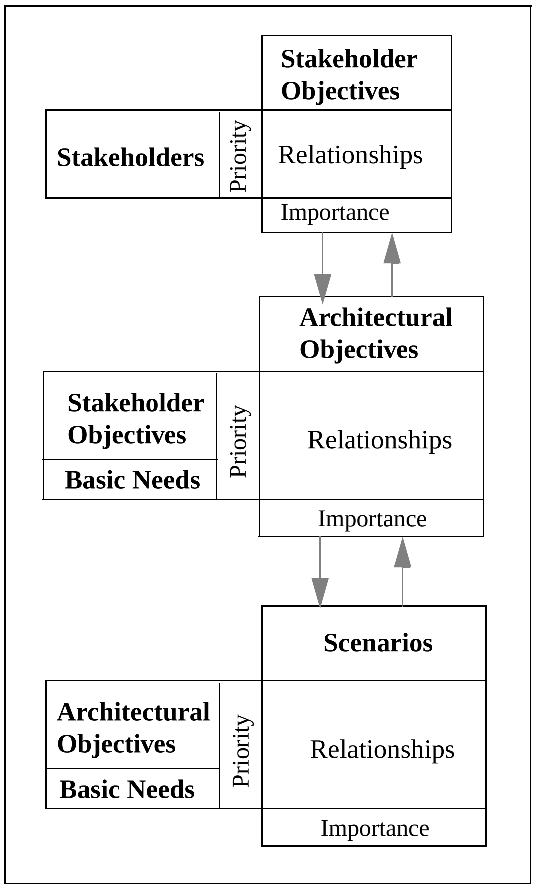


Figure 2. Interlinked relational matrices (example #2).

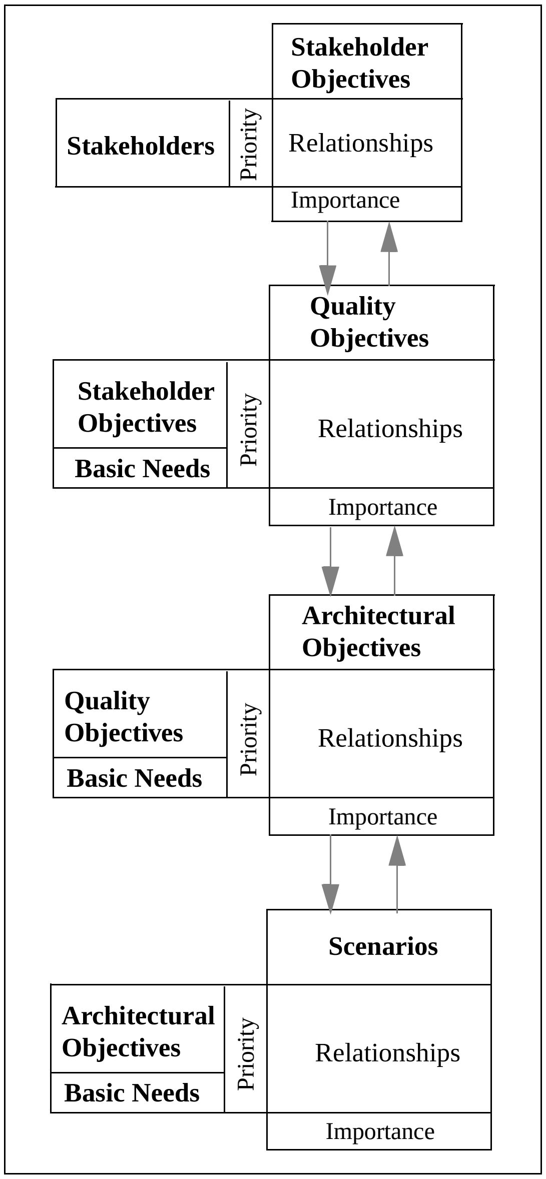
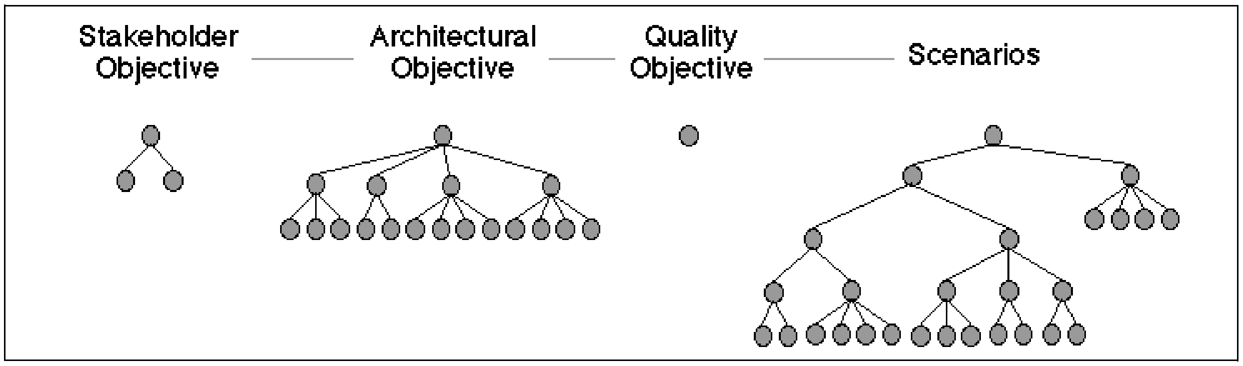


Figure 3. A simple example of objectives and scenarios depicting depth via tree structures


Stakeholder Objective
Architectural Objective
Quality Objective
Scenarios