Research Abstract: Semantic Concepts for the Specification of Non-functional Properties of Component-Based Software

Steffen Zschaler
Dresden University of Technology

E-mail: Steffen.Zschaler@inf.tu-dresden.de

1. Research area

Non-functional Properties, Component-Based Software, Specification, Language Definition, Semantics

2. Technical problem to be solved

The area of semantics for *functional* specifications has been well understood for quite a long time already. In contrast, for *non-functional* specifications there exists—to the best of my knowledge—no commonly accepted understanding of what constitutes a definition of semantics. Various specialized approaches can be found in the literature (e.g., [10, 7, 8]), but they are either incomplete or very domain specific.

Secondly, component-based software development [11] is an important trend in software engineering. Its main promises are reusability and exchangeability of components in a system. Both are achieved through a principle of locality: Every component needs to explicitly specify what services it provides and what it needs in order to provide these services. The *global* behaviour of a component-based software system is derived from the *local* specifications of its components and the network of interconnected components. In this context, the composability of non-functional properties is an interesting research topic. Although there is some literature [1, 2, 12], the problem has not yet been solved completely.

In my research I try to define a framework which can be used to provide semantics for non-functional specifications of component-based systems. Some of the key questions driving my work are:

- What is the fundamental difference between functional and non-functional specifications? What formal apparatus is required to provide for this difference?
- What effects need to be taken into consideration when composing components with non-functional properties?
- What is the dependency between functional and non-functional specifications?
- What evaluation or analysis algorithms will be applied to a non-functional specification? What information must be extractable from the semantics of a non-functional specification?

3. Related work and justification that previous work has not solved the problem

In his thesis [1], Aagedal defines CQML, a specification language for non-functional properties of component-based systems. The definition remains largely at the syntactic level, semantic concepts are mainly explained in plain English without formal foundations. Staehli [10] describes a formal technique for specifying non-functional properties of multimedia presentations. As an extension and combination of these efforts, the two authors recently and independently of my research published a short paper on "QoS Semantics for Component-Based Systems" [9]. Their work is restricted to timeliness and data quality properties and does not cover resource demand at all. In contrast, I use more abstract definitions which cover any kind of measurement, including but not limited to timeliness and data quality. Also, resource demand and resource allocation is a central element of my semantic domain. My approach allows to define measurements independently from a specific application whereas the error functions of [9] are always defined for a specific application only.

There are various publications proposing the use of value functions for the specification of non-functional properties [7, 8], using a task-based computational model. Resources are considered in [7], however only where resource demand can be adjusted during execution. The model is completely oriented towards adaptation, admission control is not considered in this model. The authors of [8] combine the concepts of measurement and constraints over measurements into the notion of a "metric", although they do not use distance measurements only.

4. Research hypothesis

The main research hypothesis is that semantics of non-functional specifications differ substantially from semantics of functional specifications, so that a dedicated formal apparatus is required to treat non-functional specifications. Specifically, non-functional specifications have an *optimization semantics*, extending the correctness-based semantics of functional specifications. With this I mean that any two correct implementations of a functional specification cannot be distinguished from the point of view of the specification: They *fulfill the specification equally well*. In contrast, of two correct implementations of a non-functional specification one may *fulfill the specification better than* the other.

A secondary hypothesis is that there is a minimal set of architectural/functional elements that are required to write meaningful non-functional specifications. It is an important question to be answered in the thesis, which elements constitute this minimal set. For example the constraint: "No request must take longer than 50 milliseconds to be handled", is a non-functional specification which only requires the functional element `request` to be defined.

5. Details of the proposed solution

In my thesis, I propose a model of the semantic domain of non-functional specifications for component-based systems. The thesis proceeds by defining an initial domain and refining it to fit three scenarios, each one a bit more complex than the previous one: a) a single, monolithic application system which can use all the resources of its platform, b) multiple application systems competing for shared resources, and c) component-based application systems in which a global non-functional requirement needs to be met by instantiating and using locally specified components.

I try to model only a minimal set of artifacts, which are indispensable for defining the semantic domain of non-functional specifications. For monolithic application systems I have so far identified the following artifacts (see also Fig. 1):

- An application system S whose non-functional properties are being specified.
- A QoS-Management-System S_M which manages the application system to ensure that the non-functional requirements will be fulfilled. QoS-Management-System and application system form a closed-loop control system.
- A set (R_n) of available resources. A resource is anything which is needed for the application system to provide its service. Examples for resources are CPU, memory, but also files, locks etc. Each resource has a maximum capacity. The QoS-Management-System allocates resources to the application system. The amount of resources allocated to an application system determines to a large extent the quality of service delivered by the application system. This relationship is expressed by the application specification F_S explained below.

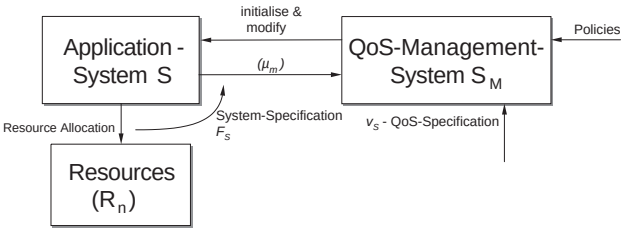

Figure 1. Monolithic Application System Model.

- A set (μ_m) of measurements. Measurements are defined as in measurement theory [5]. Each measurement assigns a quality value to the current state (and possibly history) of the application system. The QoS-Management-System uses these measurement to obtain measurement values at different points in time and to observe the quality of the services delivered by the application system. Examples for measurements are response time, data precision, confidentiality, or synchronisation of output streams. I have so far identified various possibilities to specify such measurements, e.g., [1, 3, 6, 10].
- An application specification: A function F_S from a tuple of relevant resource allocations to a set of tuples of measurement values. F_S describes the measurement values that can be expected when the QoS-Manager allocates a certain amount of resources for the application system.
- A function v_S from a tuple of measurement values to a real number. This is the specification of the required quality of service of the application system. It is provided as an objective function (or value function [7, 8]). The intended meaning is that the QoS-Management-System should control the application system such that v_S applied to the values of the relevant measurements becomes maximal.
- Finally, a set of policies, which control the behaviour of the QoS-Management-System. Policies include in-

formation on how often measurement values are obtained, and whether gauging occurs at regular intervals.

As I am still at the beginning of my work, the above model is very rough. Having completed initial models for scenarios a) and b), I am currently working on extending the model to component-based systems, after which I will return to refine the initial models. The refinement will be driven by the needs of analysis techniques which are to be applied to non-functional specifications, such as resource demand analysis (i.e., determination of the minimum required resources if F_S and v_S are known), feasibility analysis (i.e., determination whether a given v_S can ever reach its maximum, or whether dependencies between measurements prevent this), or compositional analysis (i.e., determination of non-functional properties of a system from the non-functional properties of individual components).

6. Methods used to carry out this research

As I have already indicated in Section 5, a major part of my research is defining a model of the semantic domain of non-functional specifications. Various sources form the basis for such a model: literature on operations research and linear optimization, control theory, decision theory [4, 13], and measurement theory [5]; literature on specific approaches to non-functional specifications [1, 3, 6, 10]; existing research on real-time and QoS-aware systems; literature on component-based software [11] and composability [2, 12]; but also my personal experience from the COMQUAD project (www.comquad.org).

My present plan is to complete a first set of rough models for each of the three scenarios presented in Section 5. After this, I plan to refine these models, based on the needs for analysis algorithms. In order to be able to do so, I first need to identify and select the algorithms I will investigate.

There are several criteria for evaluating my research:

1. I need to make plausible that there is indeed a structural difference in semantics between functional and non-functional specifications. As I have explained in Section 4, I believe this difference to be the optimization semantics of non-functional specifications. I need to show, that non-functional specifications can indeed be modelled using concepts from operations research and optimization.

2. Any definition of semantics is only useful together with a semantic mapping from some language's syntactical elements on elements of the semantic domain. I plan to provide an example of such a mapping for the component quality modelling language (CQML) [1].

3. In the opposite direction, it is important to show that the semantic domain contains no superfluous concepts. Where I include concepts which cannot be mapped from concepts in CQML, I will need to either provide mappings from other languages or present an argumentation, why I consider this concept indispensable to non-functional specifications (and consequently missing in any of the languages investigated). Such an argumentation could be supported by findings from the research area of real-time systems, or more general QoS literature.

References

[1] J. Ø. Aagedal. *Quality of Service Support in Development of Distributed Systems*. PhD thesis, University of Oslo, 2001.

[2] M. Abadi and L. Lamport. Composing specifications. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Stepwise Refinement of Distributed Systems - Models, Formalisms, Correctness*, volume 430 of *LNCS*, pages 1–41, Berlin, Germany, 1989. Springer-Verlag.

[3] M. Abadi and L. Lamport. An old-fashioned recipe for real time. *ACM ToPLaS*, 16(5):1543–1571, Sept. 1994.

[4] P. Fishburn. Preference structures and their numerical representations. *Theoretical Computer Science*, 217(2):359–383, Apr. 1999.

[5] G. Ford. *Lecture Notes on Engineering Measurement for Software Engineers*. Number CMU/SEI-93-EM-9. Carnegie Mellon University, 1993.

[6] S. Leue. QoS specification based on SDL/MSC and temporal logic. In G. v. Bochmann, J. de Meer, and A. Vogel, editors, *Workshop on Multimedia Applications and Quality of Service Verification*, Montreal, 1994.

[7] J. W. S. Liu, K. Nahrstedt, D. Hull, S. Chen, and B. Li. EPIQ QoS characterization. ARPA Report, Quorum Meeting, July 1997.

[8] B. Sabata, S. Chatterjee, M. Davis, J. J. Sydir, and T. F. Lawrence. Taxonomy for QoS specifications. In *Proc. 3rd Int'l Workshop on Object-oriented Real-Time Dependable Systems (WORDS'97)*, Newport Beach, California, Feb. 1997.

[9] R. Staehli, F. Eliassen, J. Ø. Aagedal, and G. Blair. Quality of service semantics for component-based systems. In *Middleware 2003 Companion, 2nd Int'l Workshop on Reflective and Adaptive Middleware Systems*, 2003.

[10] R. Staehli, J. Walpole, and D. Maier. Quality of service specification for multimedia presentations. *Multimedia Systems*, 3(5/6), Nov. 1995.

[11] C. Szyperski. *Component Software : Beyond Object-Oriented Programming*. Addison-Wesley Publishing Company, Nov. 1997.

[12] M. Werner and J. Richling. Komponierbarkeit nichtfunktionaler Eigenschaften – Versuch einer Definition. In *GI Fachtagung Betriebssysteme 2002*, Gesellschaft für Informatik, Berlin, 2002. In German.

[13] P. L. Yu. Multiple criteria decision making: Five basic concepts. In G. L. Nemhauser, A. H. G. R. Kan, and M. J. Todd, editors, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, chapter X, pages 663–699. Elsevier Science Publishers B.V., 1989.