09432 Report: Quantitative Software Design

Astrid Kreissig,[1] Iman Poernomo,[2] and Ralf Reussner[3]

[1] IBM Entwicklungslabor, Böblingen, Germany
[2] Department of Computer Science, King's College London
Strand, London WC2R 2LS UK
[3] Universität Karlsruhe (TH), Germany

Abstract. Between 20.10.09 and 23.10.09, the Dagstuhl Seminar 09432, Quantitative Software Design, was held at the International Conference and Research Center (IBFI), Schloss Dagstuhl.
Quantitative software design is a field of research that is not yet firmly established. A number of challenging open research issues are only recently being addressed by the academic research community (see below). The topic is also gaining increasing emphasis in industrial research, as any progress towards a more systematic and goal-driven software design promises the reduction of costs and risks of software projects, by avoiding current trial-and-error approaches to design. The whole field is therefore of high industrial relevance, though it is far from providing ready-to-use solutions.
The seminar was structured around ten discussion groups that considered the question of how quantitative software engineering is, could and should be done. A number of interesting insights were gained, particularly due to the interdisciplinary nature of the seminar group and the mix of industrialists and academics. This collection summarizes these group discussions.
Keywords: software design, software architecture, software components, quality of service, software quality, machine learning, optimisation and software quality attributes (including dependability, maintainability and security metrics).

Quantitative software design is a field of research that is not yet firmly established. A number of challenging open research issues are only recently being addressed by the academic research community (see below). The topic is also gaining increasing emphasis in industrial research, as any progress towards a more systematic and goal-driven software design promises the reduction of costs and risks of software projects, by avoiding current trial-and-error approaches to design. The whole field is therefore of high industrial relevance, though it is far from providing ready-to-use solutions.

The research area of quantitative software design is not yet firmly established. Its subject is the investigation of the relationship of the design of a software system on quantitatively measurable quality attributes. Such quality attributes include internal quality attributes (such as maintainability), but also externally measurable attributes (such as performance metrics, reliability or availability).

This also includes quality attributes where quantitative metrics are under current investigation, such as security. While there is no debate on the fact that the software design (mainly its architecture) is the main influencing factor on the quality of the resulting software system, an understanding of how an architectures influence on the quality is currently primarily anecdotal. Much progress was made on recent years in the area of model-based and model-driven quality prediction where software architectures are used as an input for the prediction of the quality of the system, namely various performance metrics, such as throughput, response time or reaction time. However, several important scientific questions remain unanswered:

- trade-off decisions between antagonistic quality attributes
- quantitative metrics for relevant quality attributes such as security
- software design as an optimisation problem
- lifting classical maintainability metrics to the architectural level

The aim of the seminar was to bring together industrial and academic experts from relevant areas to establish the field of quantitative software design. We were fortunate enough to have a group whose expertise cut across the relevant domains:software architecture, component-based software engineering, model-based software, quality of service and business informatics.

The seminar was organized into smaller discussion groups who attempted to define and problematise the relevant sub areas of the field. This report summarizes the outcomes of these groups.

Discussion group 1: Dependencies between QoS attributes: towards a taxonomy

Participants: Steffan Becker, Raffaela Mirandola, Assel Akzhalova, Anne Martens

The group developed a tentative taxonomy of the relationships between QoS attributes. Relations were found to range over inclusion, functional dependency, antagonistic and complementarily aspects.

Discussion group 2: Dependencies between QoS attributes: two perspectives on non-functional runtime attributes

Bara Buhnova, Carlo Ghezzi, Florian Matthes, Jens Happe, Lucia Kapova

The quality of software systems used to be understood in terms of various quality attributes that can hardly be studied independently. The aim of this break-out group was to deeper understand the interdependencies among non-functional run-time attributes of software systems and their components. A taxonomy was developed that attempted to localize these quality attributes within software architectures. The taxonomy distinguished between user and system administrator perspective. For illustration, an end user considers performance in terms of response time and throughput of the system as a black box, while the administration view in addition considers performance in terms of the resource utilization aspects of the internal subcomponents of the system. For each of the two perspectives, we have structured the identified run-time quality attributes into hierarchies, and presented the roots of the hierarchies in terms of a dependency map.

The focus of this group was on non-functional runtime attributes. A taxonomy was developed that attempted to localize these quality attributes within software architectures. Interesting discussion about non-functional properties. The taxonomy distinguished between user perspectives: specifically end user/blackbox and and administration perspectives. For example, an end-user will consider performance in terms of response time and throughput of the system as a blackbox. In contrast, an administration view will consider performance in terms of the resource utilization aspects of the internal subcomponents of the system. One of the novel, and contentious, opinion of this group was that reliability and availability are more closely related from an end user perspective, but diverge at an administration view.

Discussion group 3&4: Maintainability

Participants: Astrid Kreissig, Frantisek Plasil, Iman Poernomo, Judith Stafford, Johannes Stammel, Ralf Reussner, Eitan Farchi, Heiko Koziolek

This group considered the problem of the impact architectural design decisions can have on the maintainability attribute. It considered correlations between intuitions of maintainability and currently proposed metrics. The group discussed also how to validate the impact and how to then predict the impact of the architectural design on the maintainability.

Discussion group 5: Software Design as an Optimisation problem

Participants: Anne Martens, Assel Akzhalova, Raffaela Mirandola, Lucia Kapova, Bara Buhnova, Heiko Koziolek, Eitan Farchi, Carlo Ghezzi

The optimization design decision requires choosing between different alternatives optimizing for multiple objectives. For example, optimize performance and cost of development and configuration of a composition of software components mapped to some hardware configuration. In addition, each stakeholder involved in the design decision has different preferences over the design alternatives. Another observation is that the stakeholder preference is typically not linear. When the preference is linear it may be mapped to a utility or measure. Sometimes an external measure – for example, throughput – can be measured and/or optimized but the mapping of these external measures to stakeholders utility is non trivial. As a consequence partial order is the basic model that represents a stakeholder preference. When partial order applies, as it typically does, appropriate optimization techniques from economy and game theory could be used.

One of the greatest research challenges in the context of quantitative analysis of software is validation of metrics. It seems that an axiomatic approach could server to provide some initial necessary conditions for the quality of diffrent metrics. Correlation with cost functions based on historical data can serve as a validation mechanism for suggested metrics.

A range of different optimization design problems and optimal control techniques were considered. One example was the tradeoff analysis for response time versus reliability and search-based techniques. Another example was of conflicting stakeholder requirements and the possibility of employing economic optimal decision making theory.

Discussion group 6: Agility and architecture

Johannes Stammel, Christine Hofmeister, Frantisek Plasil, Florian Matthias, Ralf Reussner

This group considered possible compatibilities and incompatibilities between agile approaches to software development and software architecture-oriented design and implementation.

It was observed that, while agile processes work well for projects in the small, larger modern forms of development, such as service oriented architectures, are not readily described in code as per agile principles.

On the other hand, it was felt that there is good motivation for introducing architecture into agile processes, to facilitate cross project reuse and to better deal with extra-functional analysis.

It was suggested that one means of reconciliation might be through the use of Domain Specific Languages (DSLs). If a development team has a DSL, then it can be used in an agile process. The development of the DSL itself would not be susceptible to agile development, but the DSL could then be used to support agility. A case in point that is already ubiquitous would be rapid application development frameworks such as Smalltalk, Delphi and, more recently, Eclipse. These can be considered as DSLs that support agile development. If such DSLs were developed to encapsulate architectural issues, the two realms might be reconciled.

The group concluded that such research will demand support for a range of difficult issues, such as co-evolution, refactoring at both code and architectural levels, permanent traceability and new forms of syntax (possibly visual).

Discussion group 7: Quantitative perspective of systems of sytems

Astrid Kreissig, Iman Poernomo, Jens Happe, Judith Stafford, Steffen Becker, Florian Matthes

This group had a promising interdisciplinary nature, attempting to seek common ground between business informatics and software engineering. On the business side, it is understood that Key Performance Indicators (KPIs) are forms of metrics over an organizations work, and are increasingly important as a means of doing business management at the top level of an organization. They are different from technical software metrics. However, as organizations become increasingly dependent on integrated automation, the group considered how technical software and resource metrics at the lower levels of an organization might be linked to these higher level KPIs.

It was understood that simply combining existing metrics of multiple systems is not sufficient to arrive at useful recommendations for key performance indicators (KPIs) in organizations. This is for two reasons: 1) a mismatch in terminology and 2) complexity in the nature of the socio-technical system itself. The group investigated the kind of problems that need to be addressed in a solution, taking the form of interaction between the KPI view and systems quantity attributes.

Discussion group 8: Design-time versus run-time

Florian Matthes, Assel Akzhalova, Bara Buhnova, Carlo Ghezzi, Christine Hofmeister, Florian Matthes, Frantisek Plasil, Jens Happe

This group considered possible inconsistencies between design-time models and runtime and considered the reason for their occurrence. Such deviations occur because of changes in environment e.g., usage patters, behaviour of called services. The group developed a goal-oriented control cycle model: moving from a quantitative goal, via planning to a design-time model, whose implementation leads to an interaction trace and analyzable run-time model that can be checked for deviation from the goal and some appropriate control action. It was felt that there is a need to develop the science behind this loop and to investigate the suitability for runtime usage of existing models.

Discussion group 10: MDD

Steffen Becker, Jens Happe

While model driven engineering is now emerging as a viable industrial approach to developing systems, the question of *quality* of transformations appears to be very much an open question. The focus of this group was on what the metrics for model transformations might be, particularly in rule-based transformation languages.

Abstract: Quantitative Software Engineering

Eitan Farchi, IBM, Haifa

Optimization design decision requires choosing between different alternatives optimizing for multiple objectives For example, optimize performance and cost of development and configuration of a composition of software components mapped to some hardware configuration. In additin, each stakeholder involved in the design decision has different preferences over the design alternatives. Another observation is that the stakeholder preference is typically not linear. When the preference is linear it may be mapped to a utility or measure. Sometimes an external measure, e.g., throughput, can be measured and/or optimized but the mapping of these external measures to stakeholders utility is non trivial. As a consequence partial order is the basic model that represents a stakeholder preference. When partial order applies, as it typically does, appropriate optimization techniques from economy and game theory could be used.

One of the greatest research challenges in the context of quantitative analysis of software is validation of metrics. It seems that an axiomatic approach could server to provide some initial necessary conditions for the quality of diffrent metrics. Correlation with cost functions based on historical data can serve as a validation mechanism for suggested metrics.