Quality-Aware Design of Software Systems

Barbora Bühnová

Faculty of Informatics, Masaryk University
Brno, Czech Republic

LaSArIS Seminar

October 10, 2013

1. Introduction
   - Motivation
   - Focus of the Talk
2. Performance and Reliability Techniques
   - Foundations
   - Industrial Techniques
   - Research Techniques
3. Conclusion
   - Challenges

1. Introduction
   - Motivation
   - Focus of the Talk
2. Performance and Reliability Techniques
   - Foundations
   - Industrial Techniques
   - Research Techniques
3. Conclusion
   - Challenges

Motivation

Large-scale software systems with complex architecture

- support of critical business processes in enterprise inf. systems
- quality = customer trust & satisfaction = money

Different ways of understanding the quality

- not only system correctness!

Other quality attributes

- performance
- reliability
- security
- energy consumption
- maintainability
- ... and many others

Focus of the talk

Focus

- Information systems with complex architectures
- Quality in terms of performance and reliability

Goal

- Formal techniques assisting software architects in the development of high-quality systems

1. Introduction
   - Motivation
   - Focus of the Talk

2. Performance and Reliability Techniques
   - Foundations
   - Industrial Techniques
   - Research Techniques

3. Conclusion
   - Challenges

Performance

Performance reflects the ability of a software system to fulfil the requirements on fast response time and high throughput of the system while minimizing the usage of computational resources.

Performance attributes

- response time
- throughput
- resource utilization

Reliability

Reliability is the probability that a software system will perform the required functionality according to the design restrictions without faults and failures in a given time span.

Reliability attributes

- probability of failure on demand
- mean time to failure

Performance vs. reliability

Differences

- Conflicting objectives
- Tuning techniques
- Prediction questions

Similarities

- Quantitative quality attributes
- Both influenced by very similar architectural elements
- Architectural models and prediction techniques

Industrial techniques for performance/reliability assessment

After implementation (measurement-based)

- profiling and measurement of an implemented and deployed system
- pro – low effort (no additional model needed)
- cons – too late to revert initial design decisions

Before implementation (prototype-based)

- implement a prototype and measure its characteristics when deployed on the target platform
- pro – supports early decisions
- cons – very expensive, time consuming, hardware can be hardly changed, imprecise (many measurements needed for statistical validity)

Industrial techniques for performance/reliability tuning

After implementation

- faster/more reliable hardware (execution environment in general)
- redundancy (reliability), component derating (reliability)
- multi-threading (performance)
- code and architecture refactoring

During implementation

- fine-tuning of micro-level issues (performance)
- optimizing compilers (performance)
- error detection (reliability), fault tolerance (reliability)

Donald Knuth: "We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil" [1974]

Goals of ongoing research

Develop techniques with the following properties

- integrate both quality assessment and tuning
- design-time techniques (model-based)
- integrated into the development process
- easy evaluation of different configurations (changing/updating both software and hardware)
- automated quality assessment
- model-based prototype generation
- combination of formal models with UML

Additionally

- cost-effective (comparing to industrial techniques)
- time-effective (scalability of formal analysis)

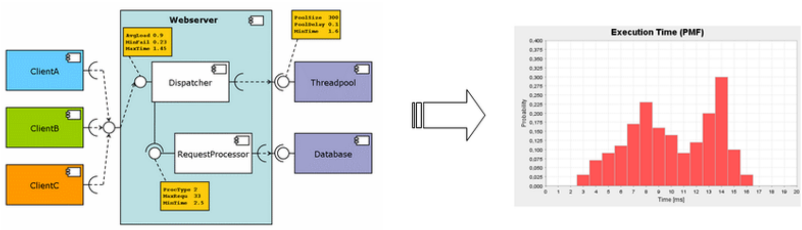
Quality engineering techniques

Focus

- Information systems with complex architectures

Implications

- Complex systems → formal methods may fail due to system size
- Defined architecture → compositional reasoning

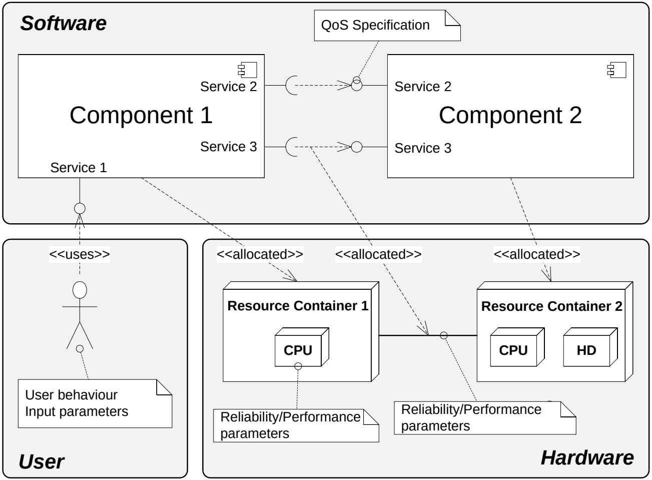


Techniques for systems with complex architectures

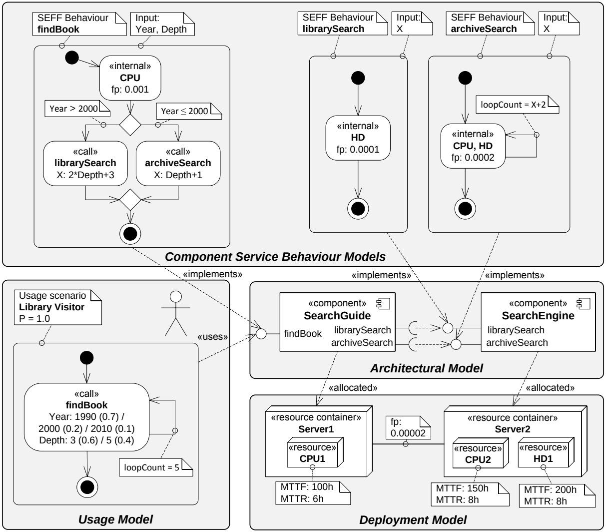
Architecture-driven analysis

- defined in a modular way
- each architectural entity seen as independent
- each element assigned with a (certified) quality information
  – i.e. software component → service: QoS as response time or probability of failure-free operation
  – i.e. hardware component → CPU: processing rate, mean time to failure/repair
- parameterized specification needed (due to independence) → easy element reuse and update

Architecture-based models



Reliability example in Palladio



Architecture-based techniques

The techniques support architecture design in:

- prediction of the expected values of performance and reliability attributes
- evaluation of alternative design decisions
- sensitivity analysis (as an effect of parameterization
  – identification of crucial components (both software and hardware)
  – relaxing uncertainties (in input parameters, system usage)
- suggestions for design improvement (architecture optimization)
- trade-off analyses (performance and reliability as conflicting objectives)

Challenges of design-time quality assessment

Performance

- high dependence on low-level details (platform dependent, e.g. scheduling strategies)

Reliability

- accuracy of the input data (failure probabilities and hardware availability)

Both

- knowledge gap between software engineers/architects and quality experts
- minimization of the modelling effort

Thank you

Thank you for your attention**!**

Any **questions?**