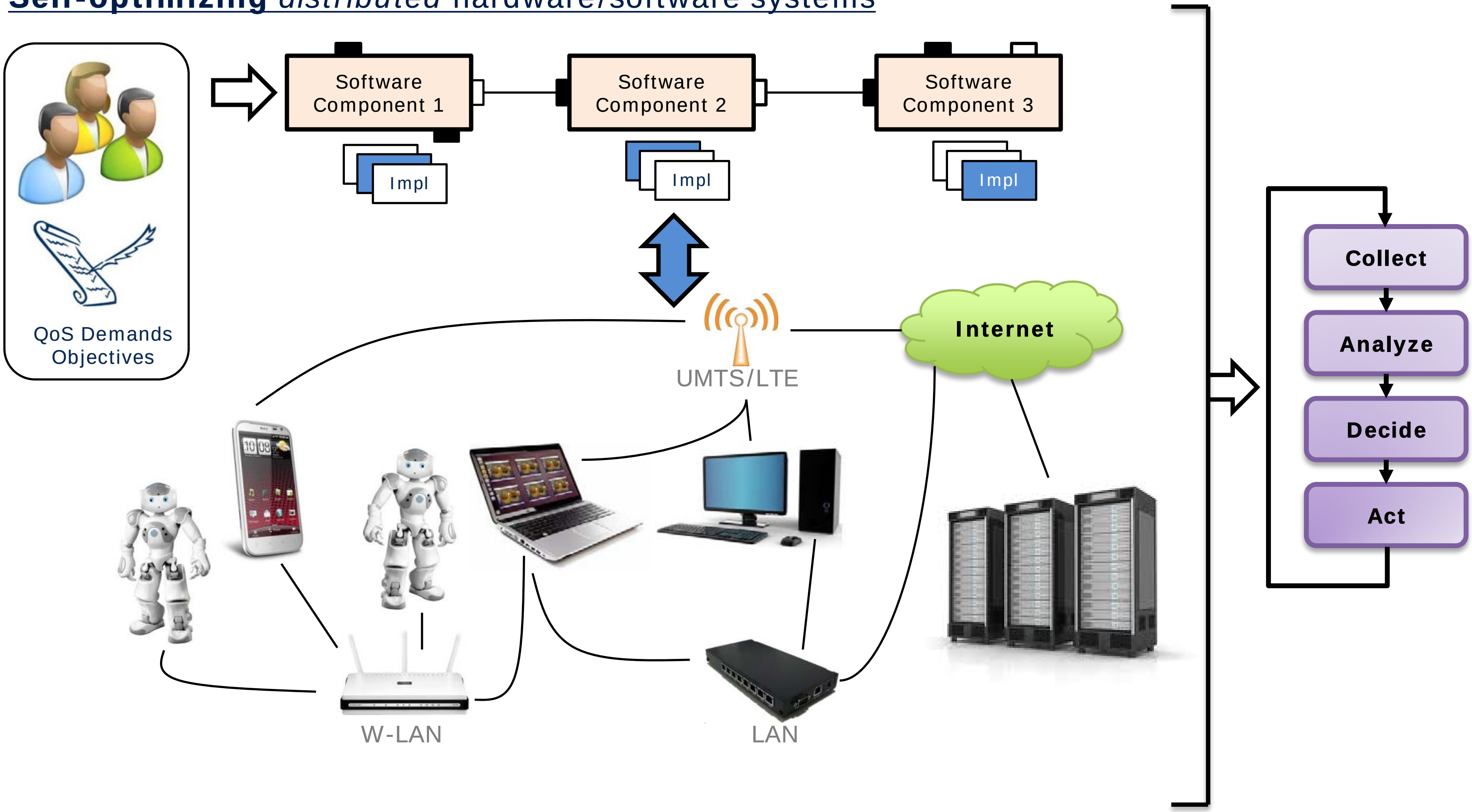
Model-driven Self-Optimization

by Integer Linear Programming and Pseudo-Boolean Optimization

Sebastian Götz, Claas Wilke, Sebastian Richly, Christian Piechnick, Georg Püschel and Uwe Aßmann

ADAPTIVE'13, Valencia, Spain, 28.05.2013

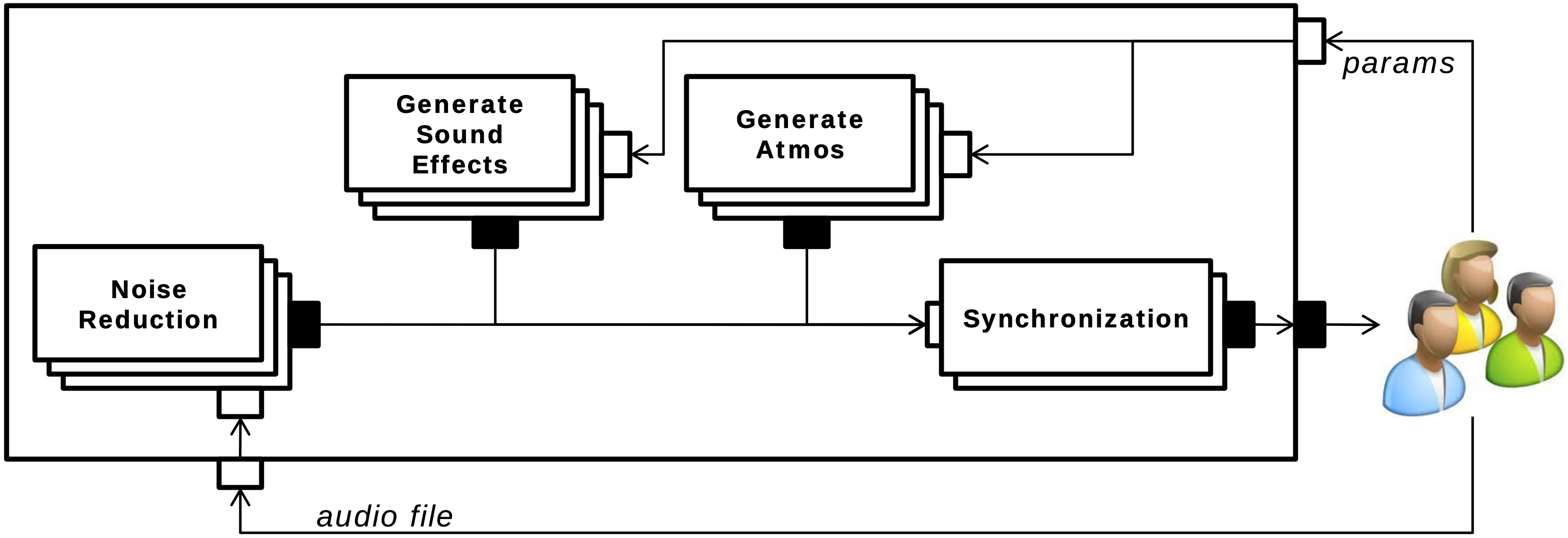
Self-optimizing *distributed* hardware/software systems



Motivation

Example: Audio-Processing (https://auphonic.com/)

- Customers send audio files for grafting
  - Noise reduction
  - Sound design (e.g., adding synthesized sounds)
  - Synchronization of multiple audio streams
  - Etc.



TECHNISCHE UNIVERSITÄT DRESDEN

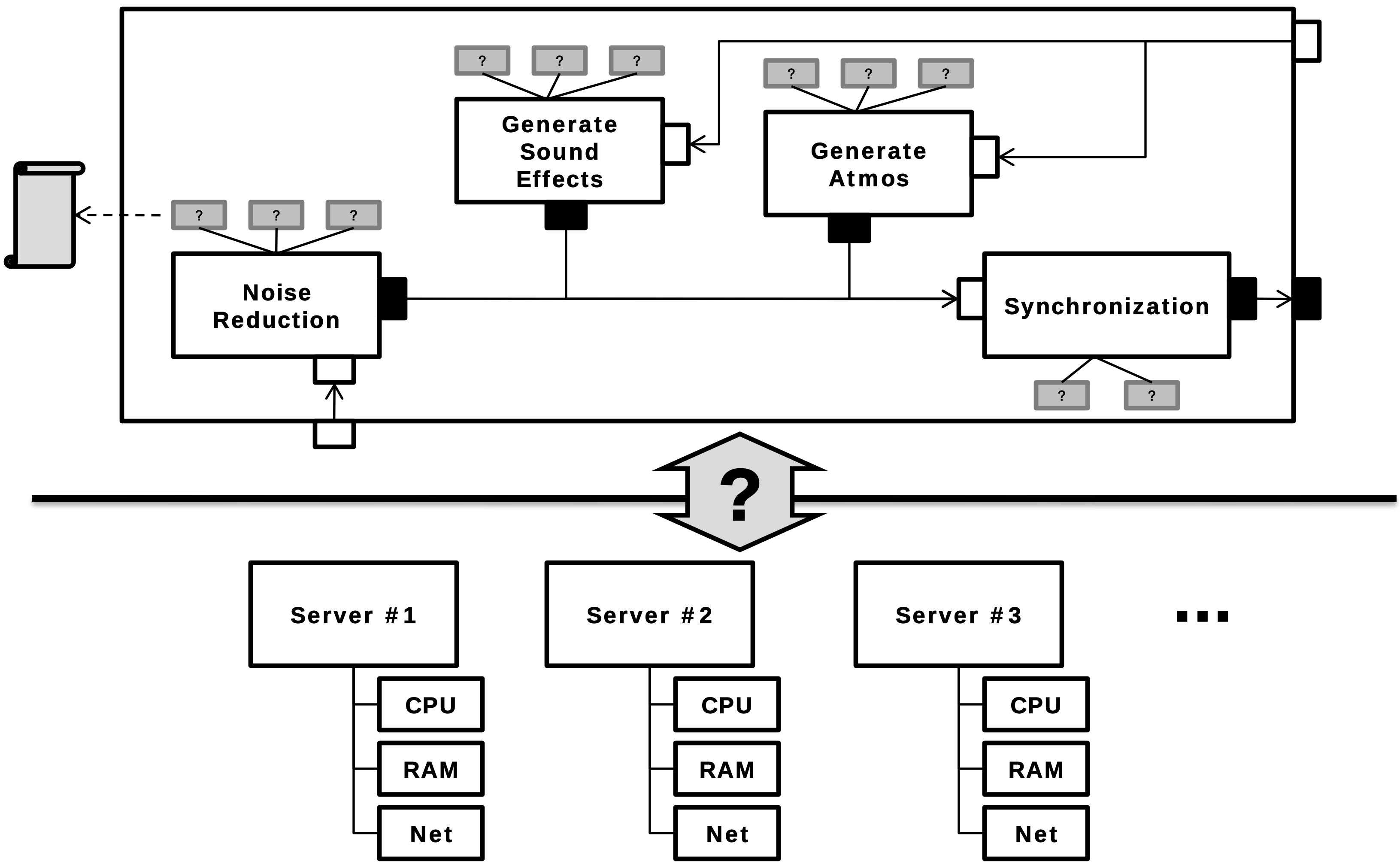
Problems

- **Problem #1:** Formulation of optimization problem

  - Developers **reinvent solutions** to almost equal problems
    - NFPs of interest change
    - Resources of interest change
  - But, the general optimization problem remains the same

TECHNISCHE UNIVERSITÄT DRESDEN

Optimization Problem Description



Problems

- **Problem #1:** Formulation of optimization problem

  - Developers **reinvent solutions** to almost equal problems
    - NFPs of interest change
    - Resources of interest change
  - But, the general optimization problem remains the same

  - **Solution**:
    - Model-driven development of the system
    - Use runtime and design-time models of the system to **generate the optimization problem**

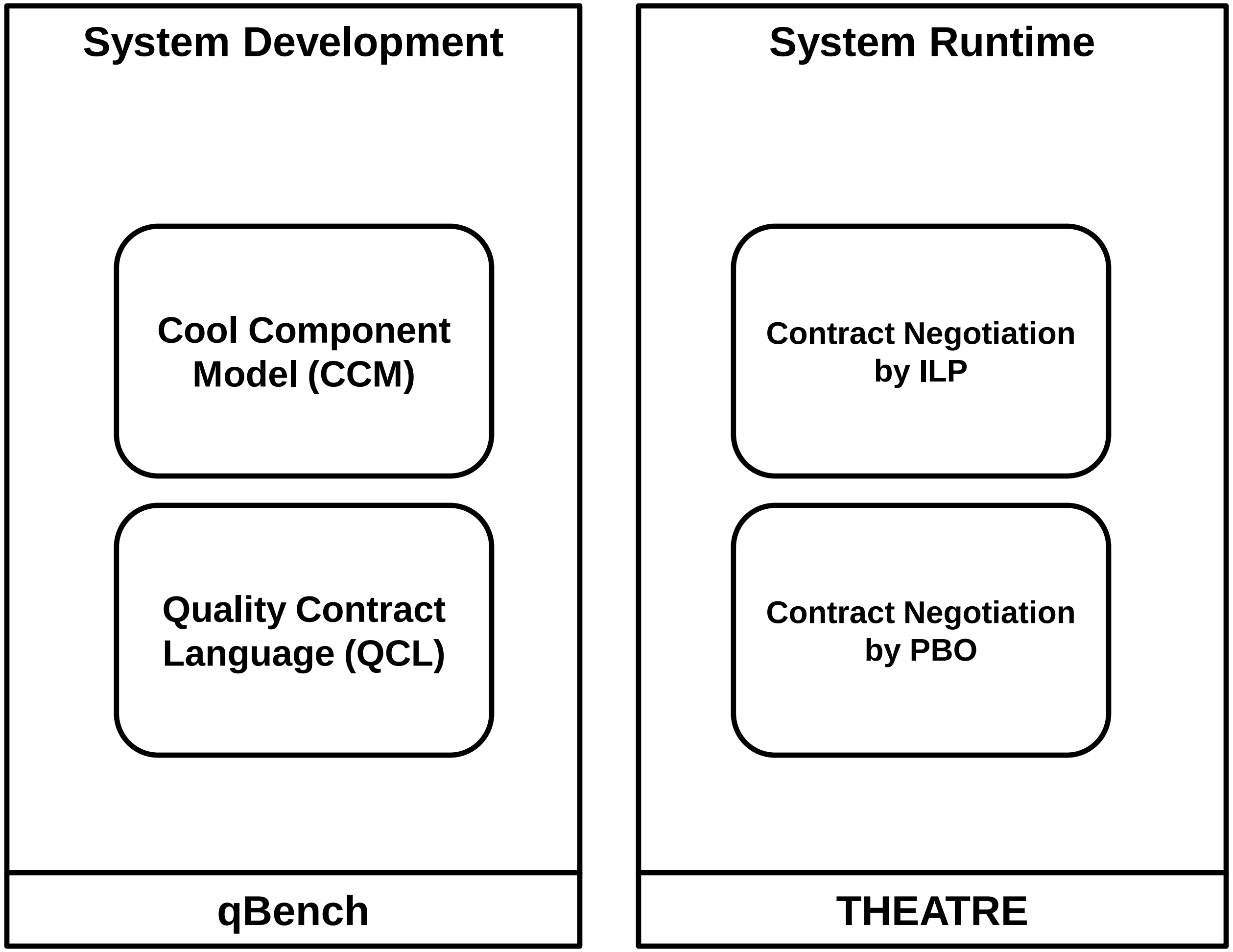
Problems

- **Problem #2:** Complex dependencies between NFPs have to be considered
  - Optimization problem relies on these dependencies
    (e.g., trade-off between response time and noise level)
  - **Solution:**
    - **QoS contracts** covering the non-functional behavior of implementations

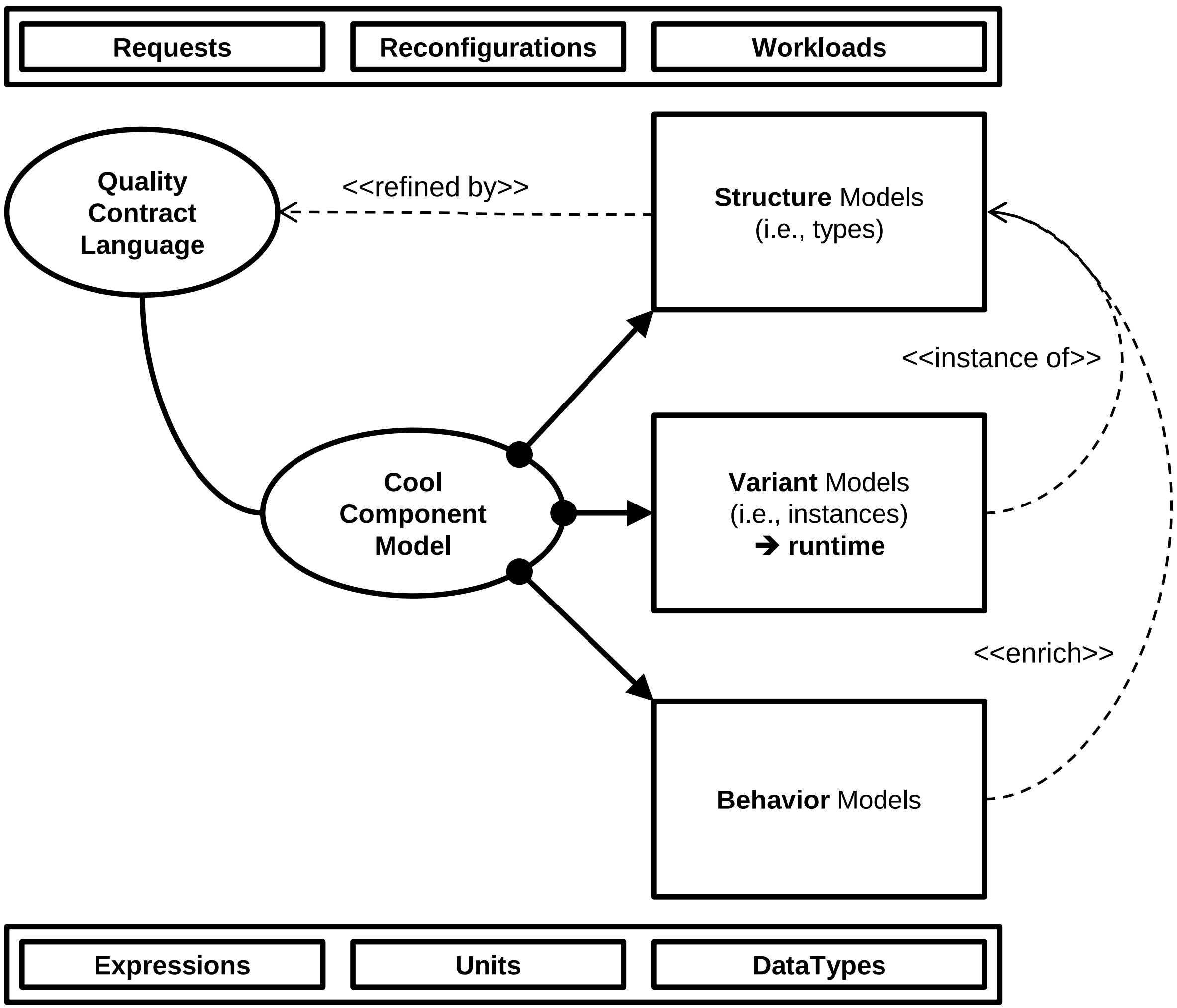
Problems

- **Problem #3:** High computational complexity of optimization
  - Can optimization be performed in budget?
  - **Solution:**
    - **Scalability analysis** of the approach

Multi-Quality Auto-Tuning



Cool Component Model



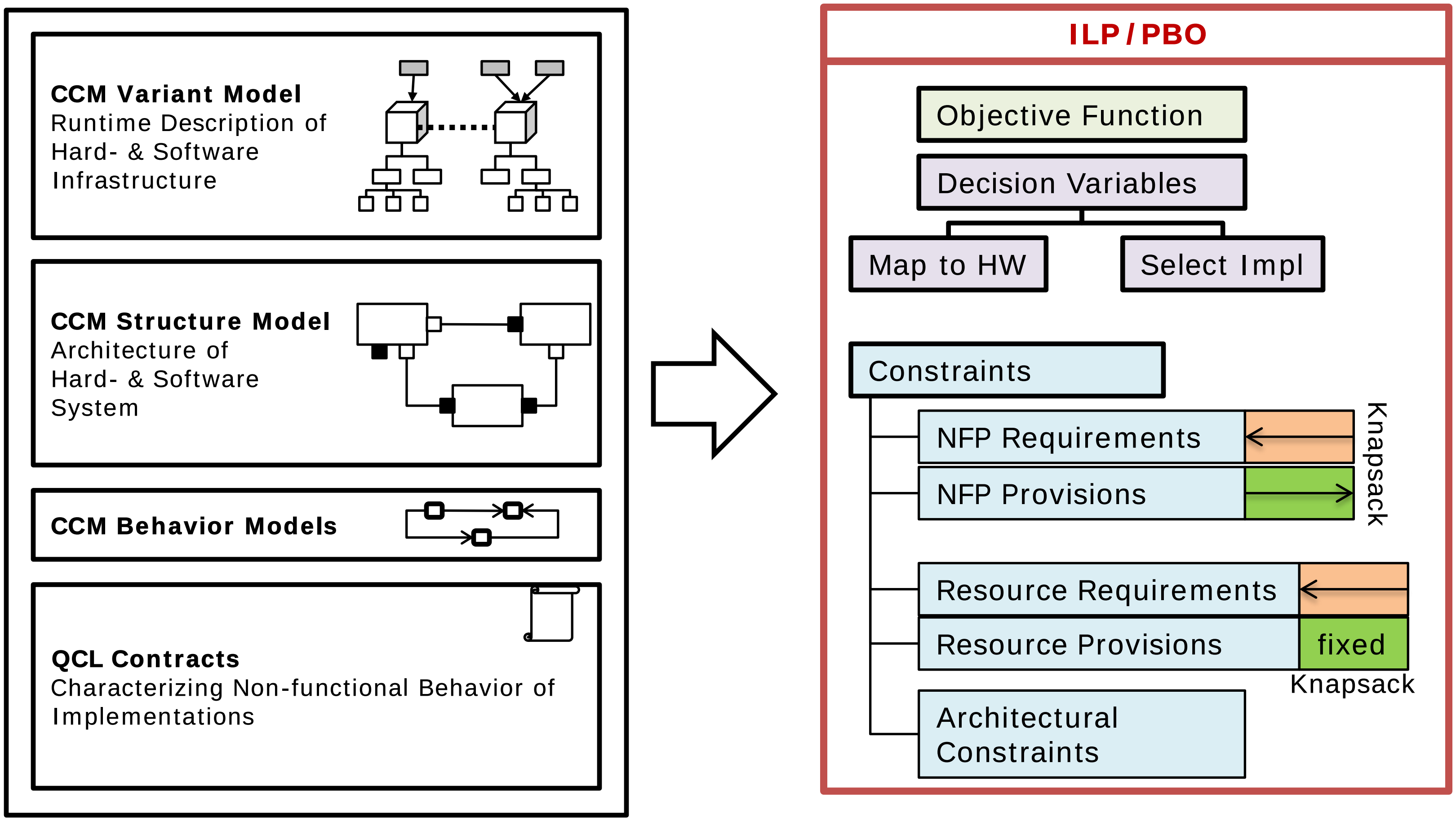
Quality Contract Language

```
contract VLC implements VideoPlayer.play {

  mode fluent {
    requires component Decoder {
      min dataRate: 9 MB/s
    }
    requires resource Net {
      min bandwidth: 10 MB/s
    }

    provides min frameRate: 25 FPS
  }
  mode lowQuality {
    /* More requirements and provisions here ... */
  }
}
```

Contract Negotiation by ILP/PBO

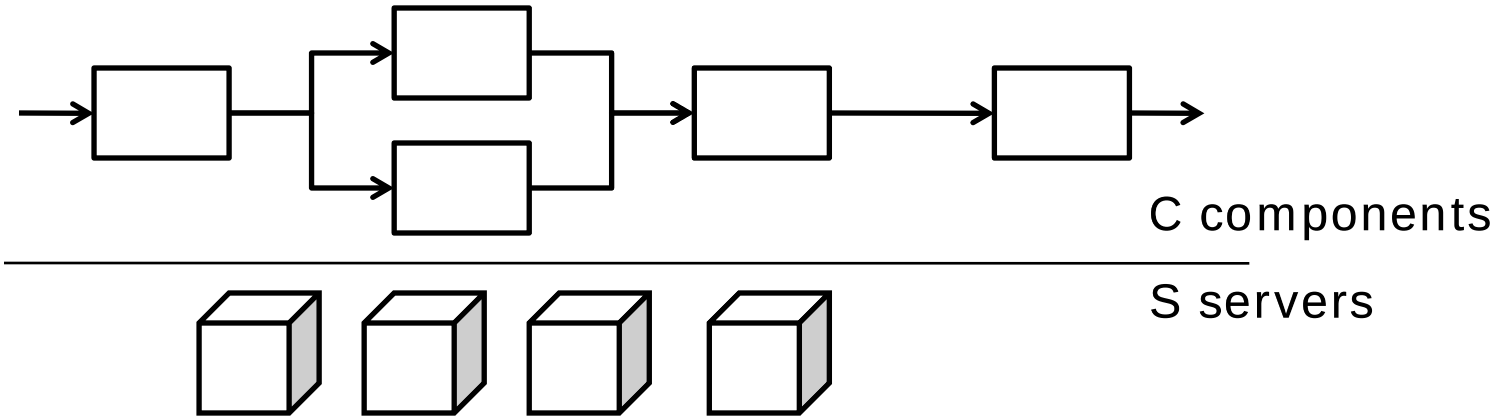


Contract Negotiation by PBO

- Pseudo-Boolean Optimization (PBO) = 0-1 Integer Linear Programming (ILP)
  - i.e., only boolean decision variables
- Allows for application of SAT-solving (e.g., DPLL)
- Could be faster than general ILP solving

Scalability Analysis

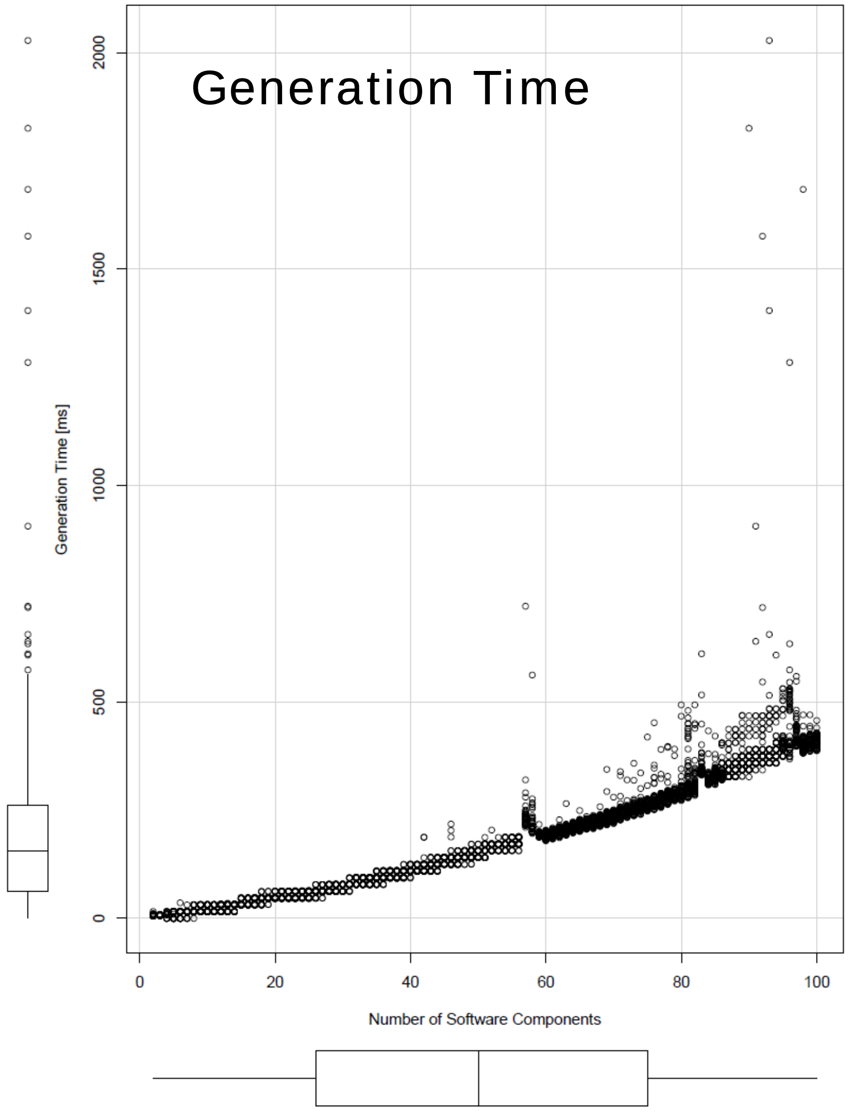
- Performed on typical class of systems: **pipe-and-filter style**

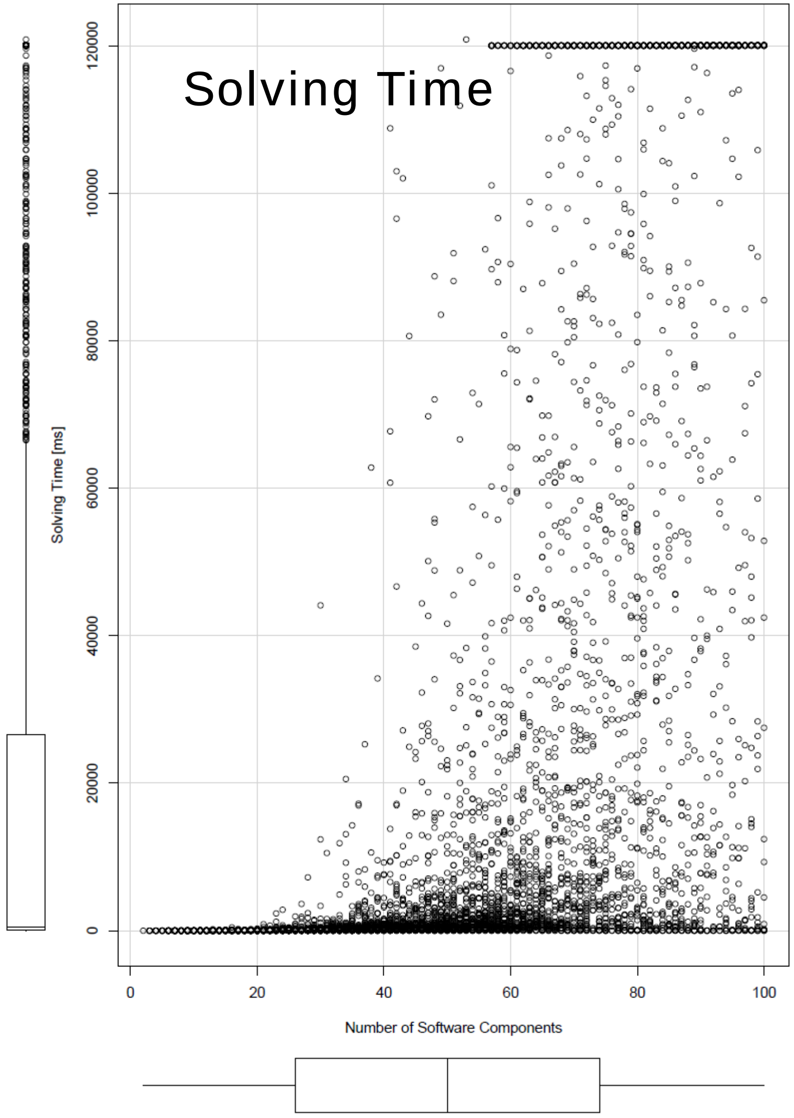


- Each component type has 2 implementations
- 2 NFPs per implementation

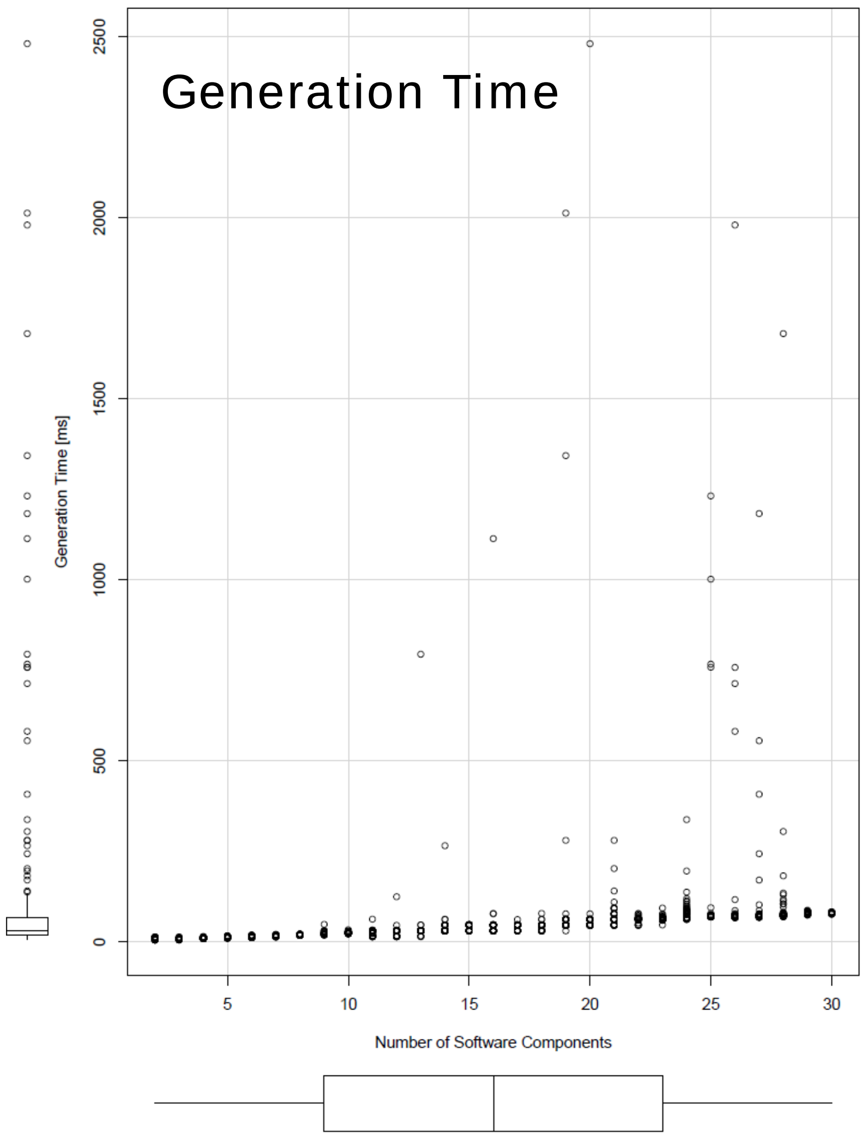
- Measurements taken for C x S systems from **C = [2..100] and S = [2..100]**

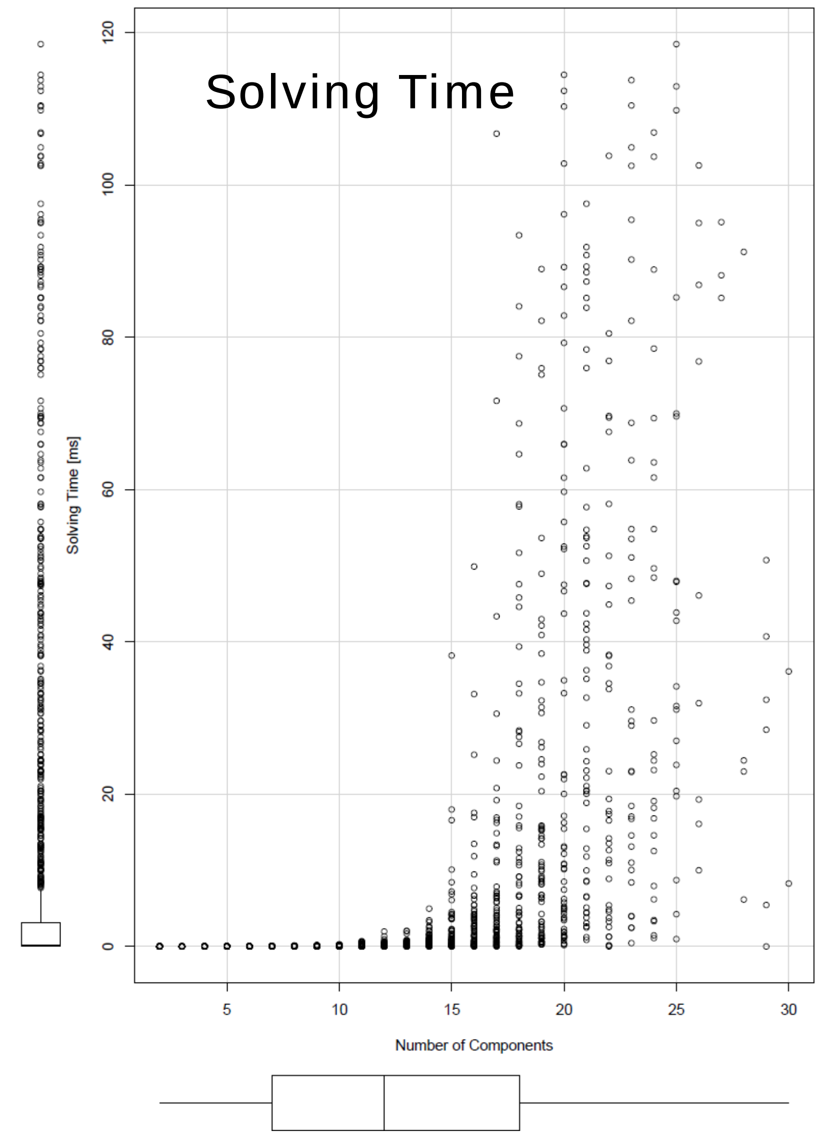
Scalability Analysis: ILP





Scalability Analysis: PBO





TECHNISCHE UNIVERSITÄT DRESDEN

Summary

- **Problem #1**: Developer's reinvent solutions to optimization problems
  - Application of **runtime models** to **generate** the optimization problem

- **Problem #2:** Complex dependencies between NFPs have to be considered
  - Application of **QoS contracts** covering non-functional behavior of implementations

- **Problem #3:** High computational complexity of optimization techniques
  - **Scalability Analysis**
    - ILP solving is predictable up to 25 component types
    - **ILP solving is feasible up to 100 component types**, if typical processing time is »30s
    - **ILP performs much better than PBO**
    - PBO solving is feasible up to 20 component types
    - PBO solving is predictable up to 10 component types

Thank You!

Contact

http://www.inf.tu-dresden.de/~sebgoetz

sebastian.goetz@acm.org