Multi-Quality Auto-Tuning by Contract Negotiation

Verteidigung der Dissertation
von Dipl.-Inf. Sebastian Götz

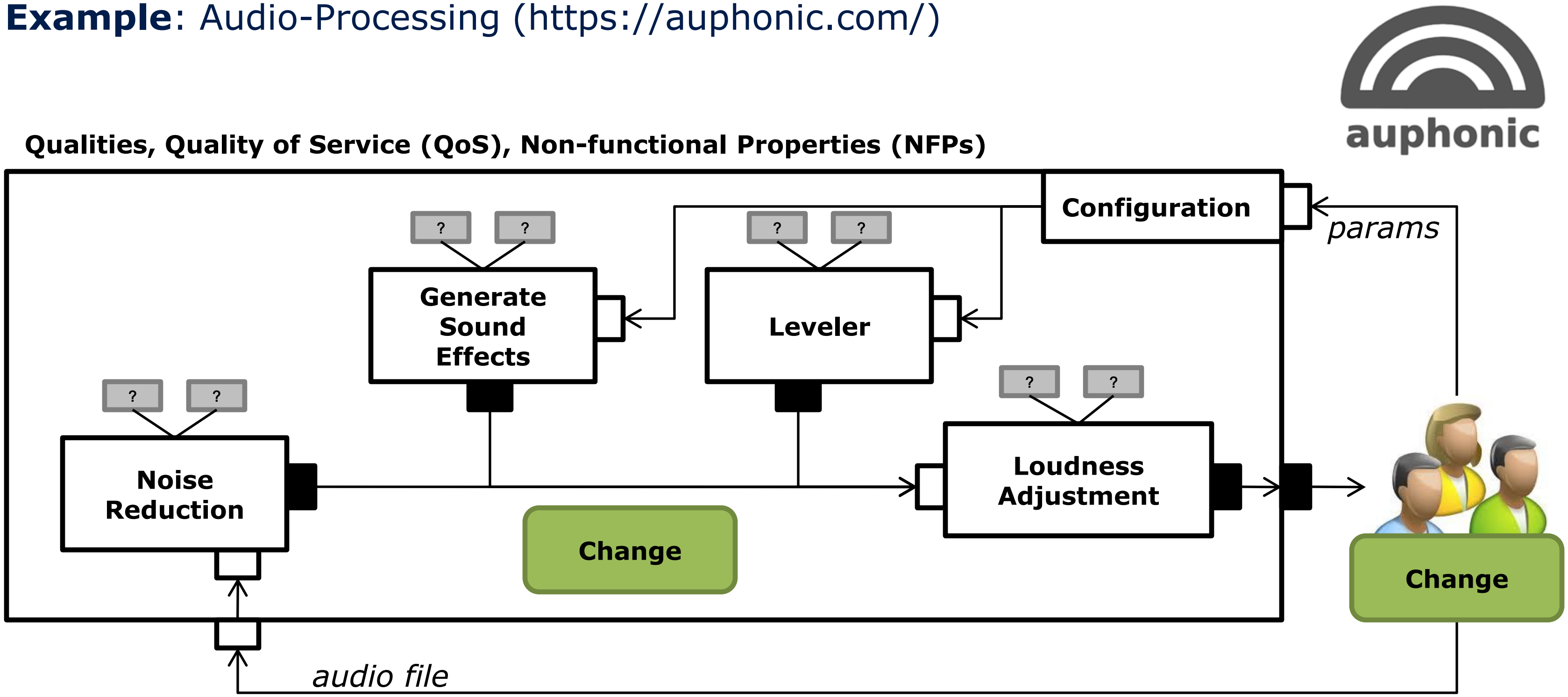
Betreuer: Prof. Dr. rer. nat. habil. Uwe Aßmann
Zweitgutachter: Prof. Dr. rer. nat. habil. Heinrich Hußmann
Fachreferent: Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

17.07.2013

Europa fördert Sachsen. SACHSEN
ESF Europäischer Sozialfonds
Bundesministerium für Bildung und Forschung
DFG Deutsche Forschungsgemeinschaft
DRESDEN concept Exzellenz aus Wissenschaft und Kultur

Motivation

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

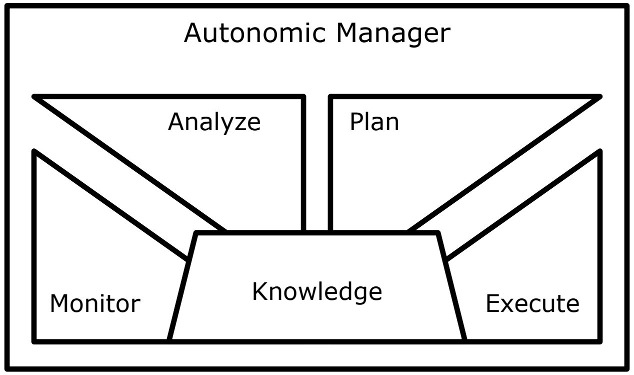


Context: Self-adaptive Systems

Goal: Self-adaptive Systems (SAS)

Robert Laddaga 1997:

> *"Self Adaptive Software **evaluates** its own behavior and **changes behavior** when the evaluation indicates that it is not accomplishing what the software is intended to do, or **when better functionality or performance is possible**."* [L97]



MAPE-K Loop [KC03]

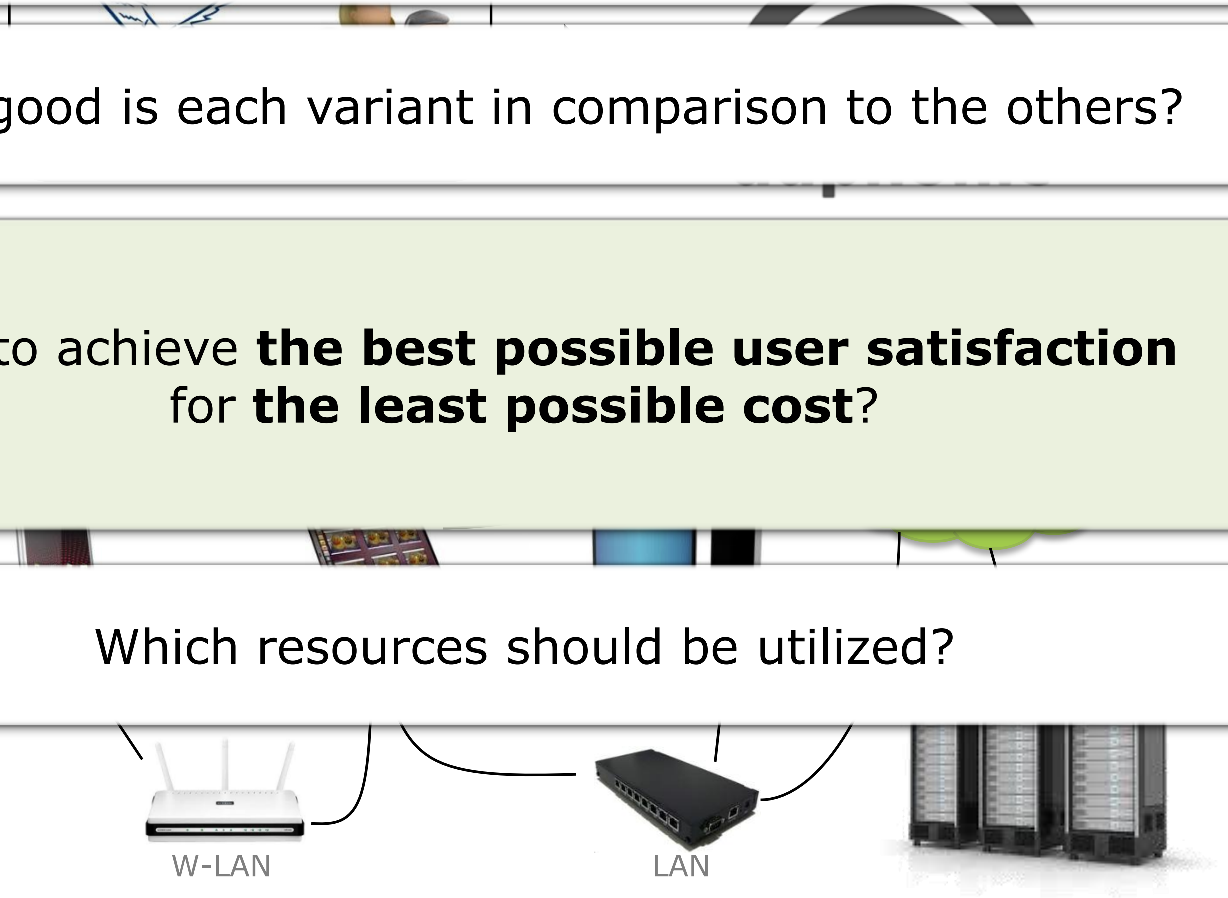
Context

Which variant of which software should be used?

How good is each variant in comparison to the others?

How to achieve **the best possible user satisfaction** for **the least possible cost**?

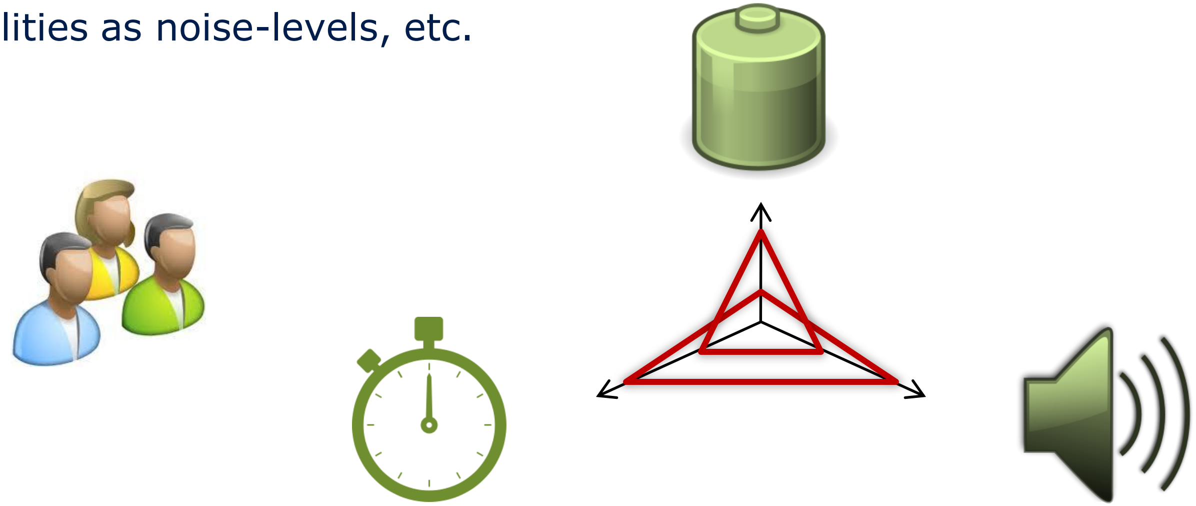
Which resources should be utilized?



TECHNISCHE UNIVERSITÄT DRESDEN

Motivation

- **User objectives relate to qualities**: energy, performance, domain-specific qualities as noise-levels, etc.



- Often **multiple, competing qualities** are to be considered **in combination** [ST09]

A novel approach to **design** & **operate self-optimizing systems** covering **multiple objectives**.

Multi-Quality Auto-Tuning (MQuAT)

Problems / Related Work

Problem 1: Developers **cannot reuse solutions** to build self-optimizing systems although many specific approaches exist.

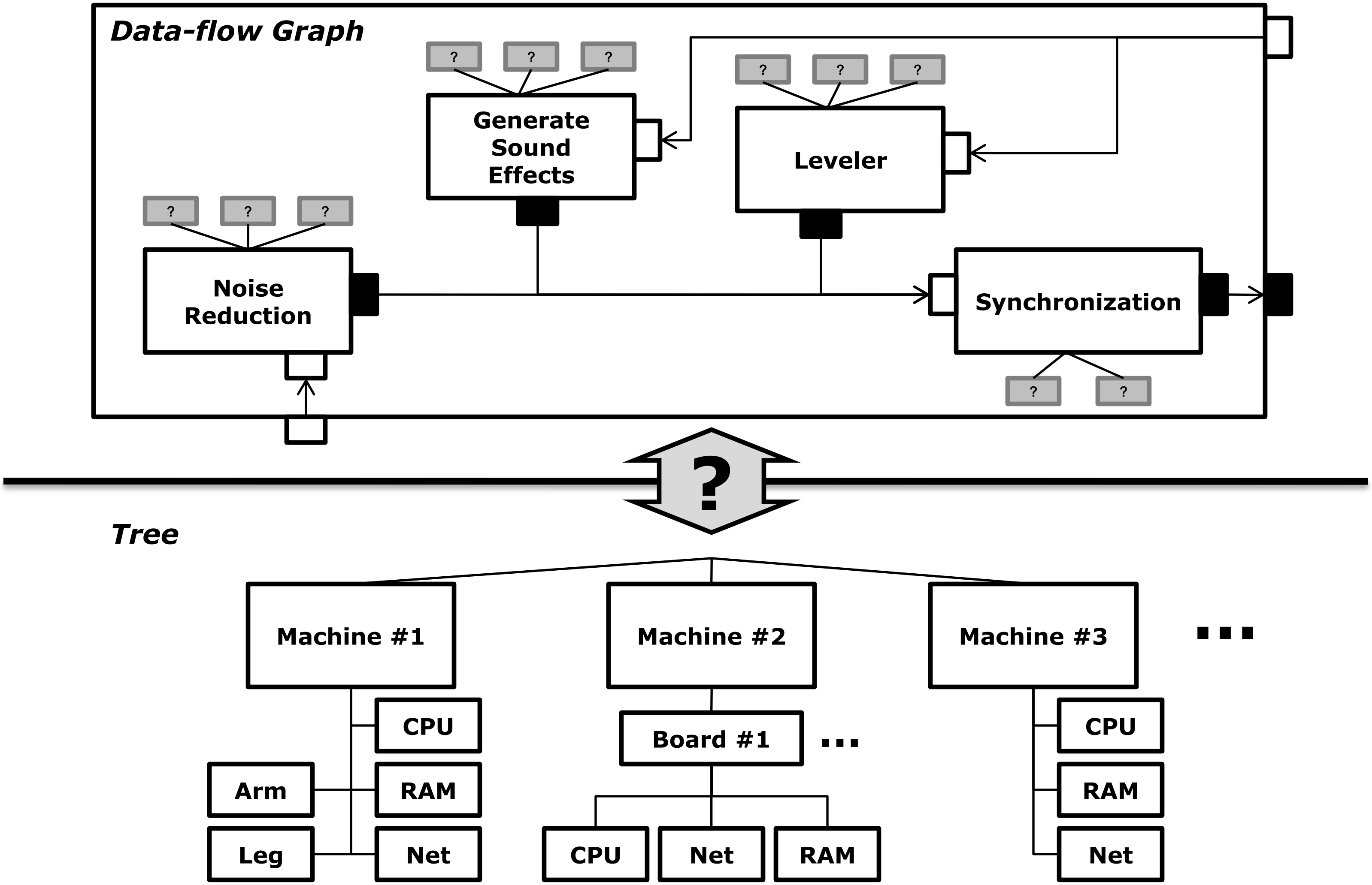
- Fixed set of considered properties (e.g., bandwidth, response time)
- Fixed architecture (e.g., specific to servers, mobile phones or cars)
- Fixed optimization technique (e.g., integer linear programming)

Goal: A generic approach to self-optimizing systems.

Solution: A **model-driven development approach** to self-optimization

- A **component-based metamodel** enabling the developer to specify the properties of interest and the system‘s architecture.
- **Technology bridges** to utilize multiple optimization techniques (generation of optimization problems).

Optimization Problem Description



Problems / Related Work

Problem 2: Existing (specific) approaches **do not cover dependencies between qualities.**

- Quality-contract-based approaches
  - COMQUAD → QoS characteristics (e.g., `response_time < 5ms`) [RZ03]
  - THESEUS → SLAs; QoS intervals (e.g., `2ms < response_time < 5ms`) [S10]
  - No context-dependent QoS statements (e.g., `response_time(size) = f(size)`)
  - Both projects identified the need to cover QoS dependencies [ZM03, S10]

Goal: Explicit coverage of (context-dependent) interaction between qualities.

Solution:

- An extended notion of quality contracts and
- A process for quality contract refinement.

Problems / Related Work

Problem 3: Competing qualities demand for **multi-objective optimization** having a **high computational complexity** (NP-hard) [NW99]

- Multi-objective approaches (e.g., OCTOPUS)
  - **„a priori":** aggregation of objectives prior to optimization
  - **„a posteriori":** optimization delivers set of multi-dimensional solutions (Pareto front)
- **Optimization at runtime** requires feasible, assessable time requirements

Goal: A generic, *assessable* runtime multi-objective optimization approach.

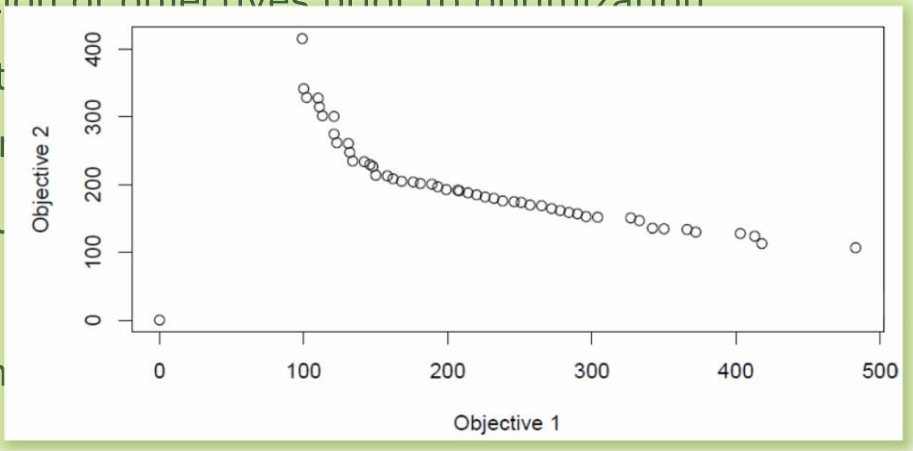
Solution:

- **4 runtime technology bridges** to multi-objective optimization techiques.
- **Scalability analysis** of supported techniques.

Problems / Related Work

Problem 3: Competing qualities demand for **multi-objective optimization** having a **high computational complexity** (NP-hard) [NW99]

- Multi-objective approaches (e.g., OCTOPUS)
  - **„a priori“:** aggregation of objectives prior to optimization
  - **„a posteriori“:** optimizat
    (Pareto fr
- **Optimization at runtime** requ

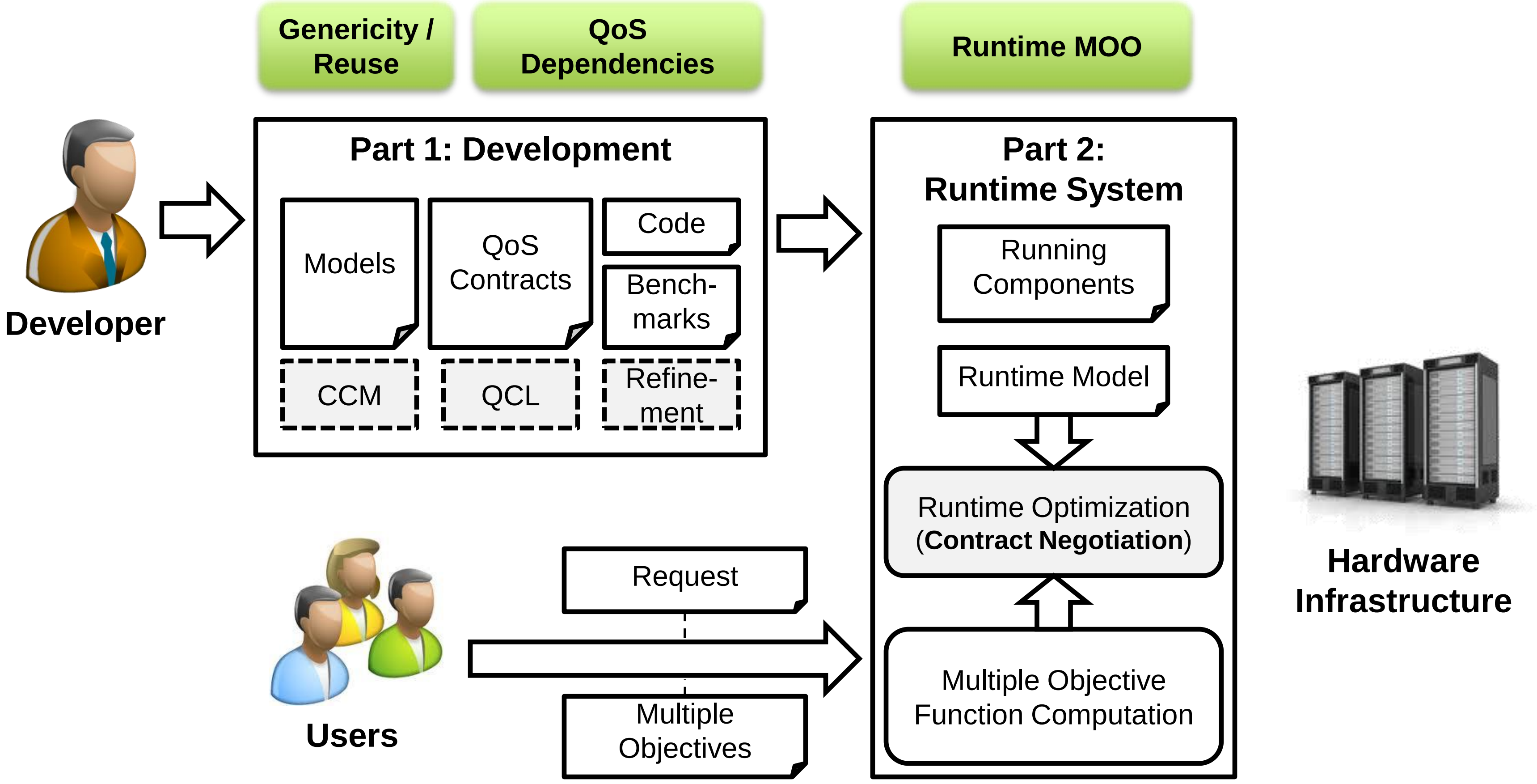


Goal: A generic, *assessable* runtim

Solution:

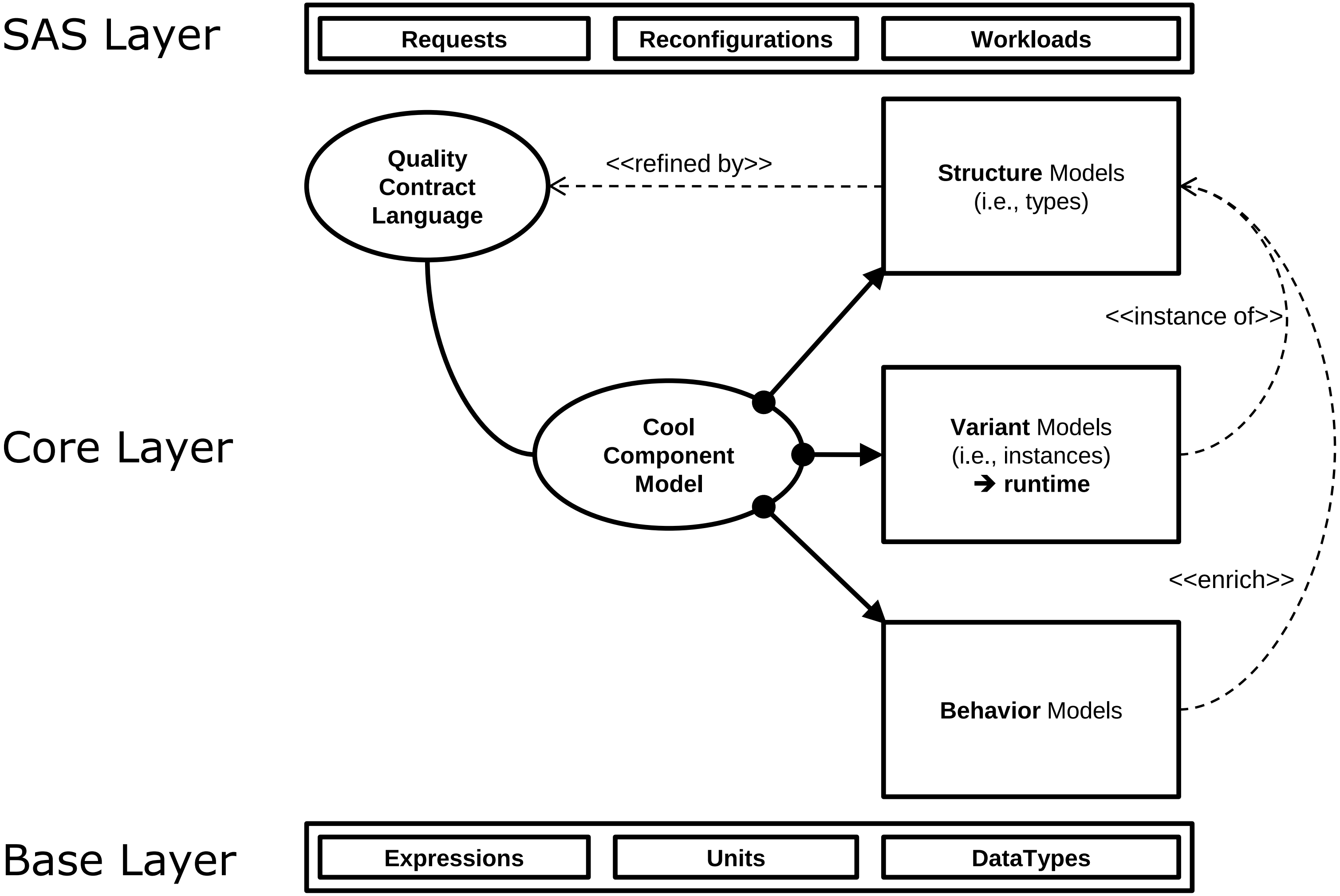
- **4 runtime technology bridges** to multi-objective optimization techiques.
- **Scalability analysis** of supported techniques.

Overview


Genericity / Reuse
QoS Dependencies
Runtime MOO
Developer
Part 1: Development
Models
QoS Contracts
Code
Bench-marks
CCM
QCL
Refine-ment
Part 2: Runtime System
Running Components
Runtime Model
Runtime Optimization (Contract Negotiation)
Multiple Objective Function Computation
Hardware Infrastructure
Users
Request
Multiple Objectives

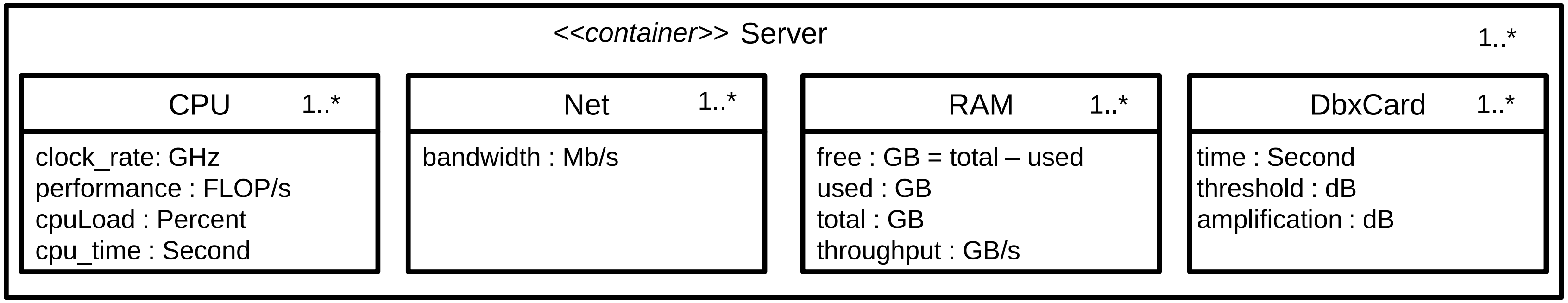
PART 1: DEVELOPMENT

Cool Component Model [GWS+10]



Cool Component Model [GWS+10]

- Example CCM Structure Model for Servers:



<<*container*>> Server 1..*

CPU 1..*
clock_rate: GHz
performance : FLOP/s
cpuLoad : Percent
cpu_time : Second

Net 1..*
bandwidth : Mb/s

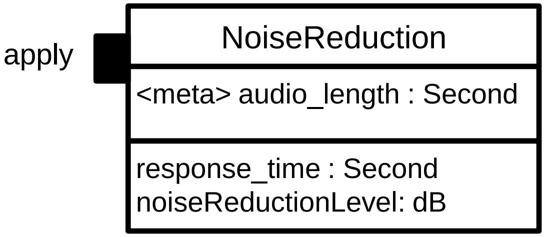
RAM 1..*
free : GB = total – used
used : GB
total : GB
throughput : GB/s

DbxCard 1..*
time : Second
threshold : dB
amplification : dB

- Example Unit Library

```
library {
  simple unit Watt : Integer
  simple unit Second : Integer;
  simple unit dB : Real;

  complex unit Joule = Watt Second;

  factor KW = 1000 Watt;
}
```

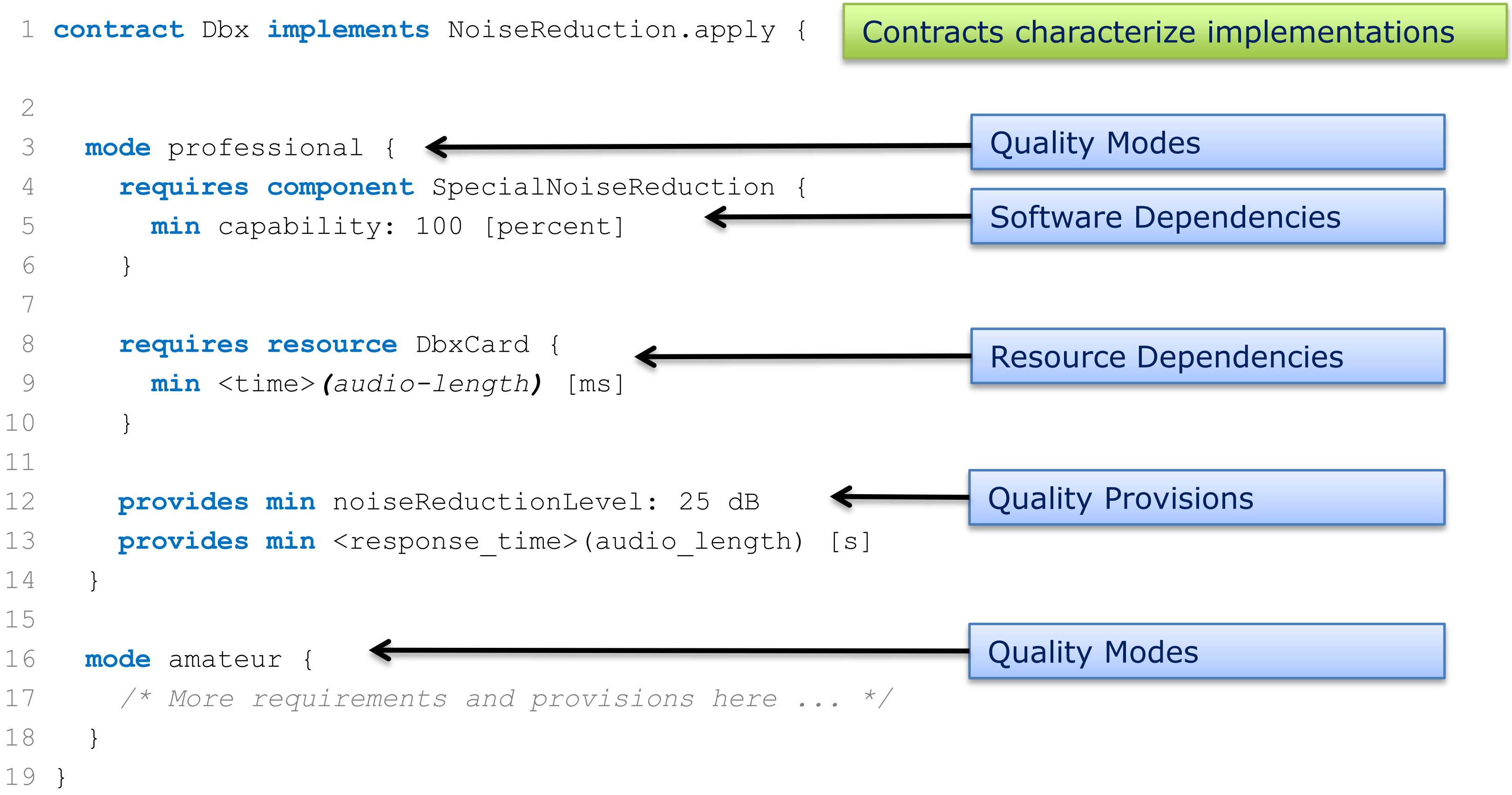
- Example CCM Structure Model for Sort:



apply

NoiseReduction
<meta> audio_length : Second
response_time : Second
noiseReductionLevel: dB

Quality Contract Language [GWC+12a]



```
contract Dbx implements NoiseReduction.apply {

  mode professional {
    requires component SpecialNoiseReduction {
      min capability: 100 [percent]
    }

    requires resource DbxCard {
      min <time>(audio-length) [ms]
    }

    provides min noiseReductionLevel: 25 dB
    provides min <response_time>(audio_length) [s]
  }

  mode amateur {
    /* More requirements and provisions here ... */
  }
}
```

Contracts characterize implementations

Quality Modes

Software Dependencies

Resource Dependencies

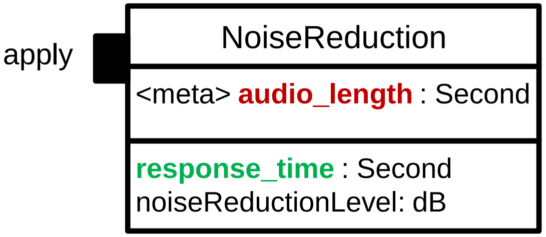
Quality Provisions

Quality Modes

Contract Refinement [GWC+12b]

- Target systems and user input are unknown to developer.
- Developer creates **contract templates**:

```
contract Dbx implements NoiseReduction.apply {
  mode professional {
     ...
     provides min response_time:
        <response_time>(audio_length)  [s];
  }
  ...
}
```



apply

NoiseReduction

<meta> **audio_length** : Second

response_time : Second
noiseReductionLevel: dB

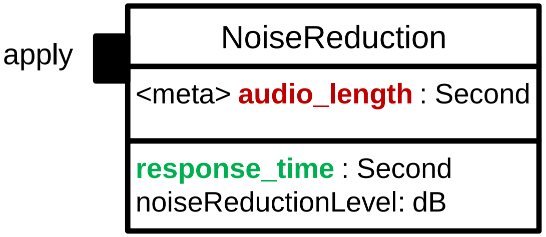
- Developer creates **Benchmark Suite** using **Profiler Framework** [WGR13]

```
for(i = 0; i <= N; i++) {
  Profiler.getProfiler(„response_time“).start();
  dbx.apply(sample_files[i]);
  Profiler.getProfiler(„response_time“).stop();
}
```

Contract Refinement [GWC+12b]

- Target systems and user input are unknown to developer.
- Developer creates **contract templates**:

```
contract Dbx implements NoiseReduction.apply {
  mode professional {
     ...
     provides min response_time:
       <response_time>(audio_length) [s];
  }
  ...
}
```



- Benchmarks executed at **deployment** time on **each target machine**:

audio_length	response_time
1s	945ms
2s	1823ms
...	...
120s	110215ms

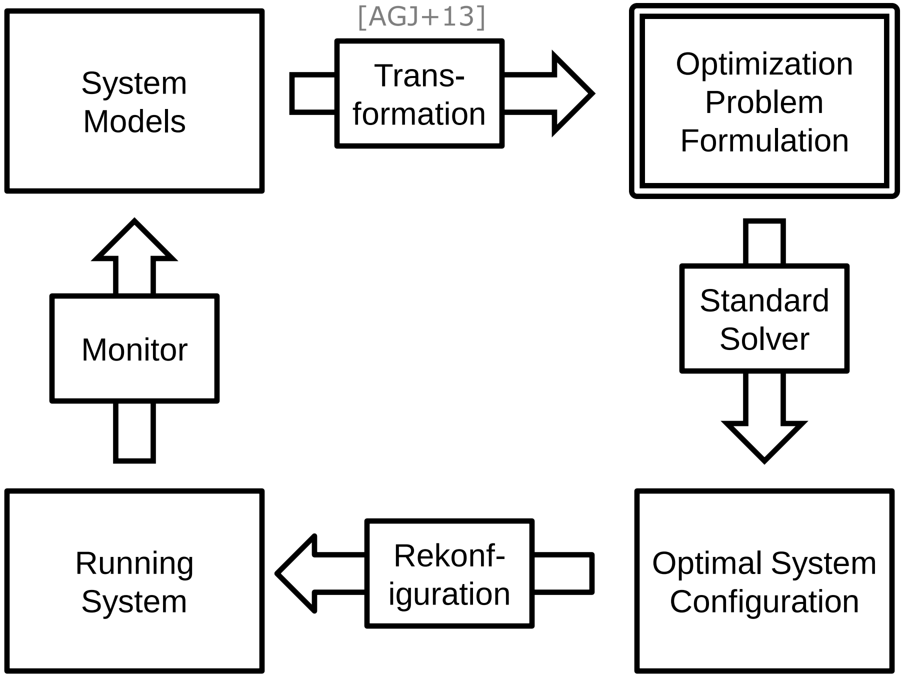
```
<response_time>(audio_length) [s]
⇩
1.147*10^(-6)*audio_length^2-1922 [s];
```

One contract per machine and implementation.

PART 2: RUNTIME

Contract Negotiation

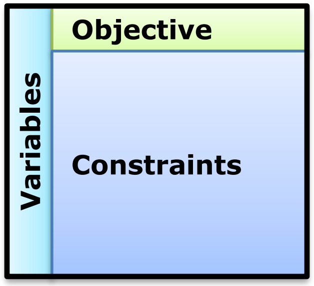
… denotes a **global optimization problem** of a system of **components**, which are **known and controllable** by **central coordinators** as known from the self-adaptive system‵s community.



		Objectives	
		Single	Multiple
Accuracy	Exact	ILP PBO	MOILP
	Approx.	ACO	

Contract Negotiation by ILP [GWC+11]

- Base: Integer Linear Programming (ILP)

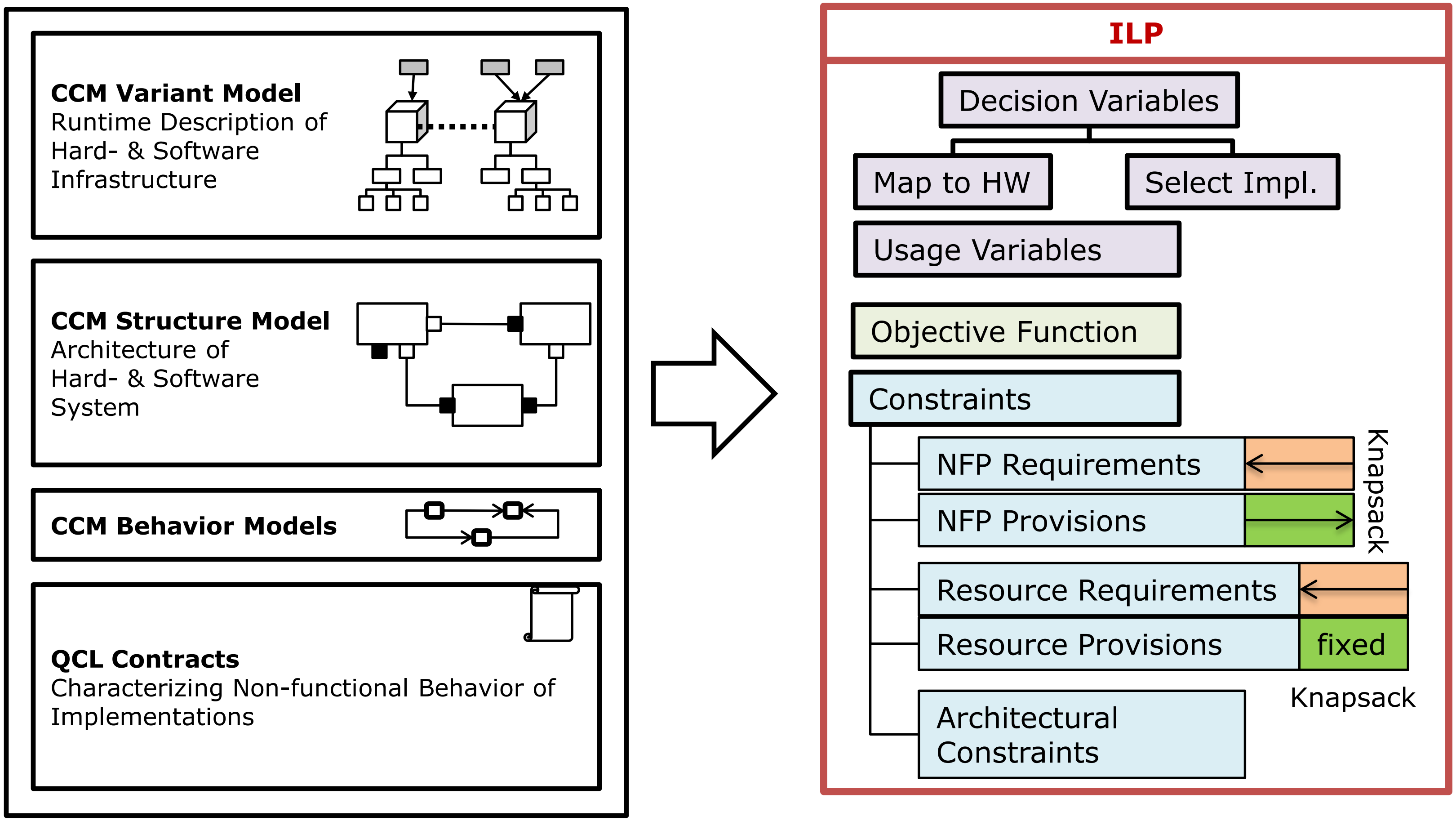


- **Goal**: determine the variable assignment, which
  - Maximizes objective function and
  - Adheres to the constraints.

- Avoids pruning of whole search space (worst case)

Contract Negotiation by ILP [GWC+11]

- Integer Linear Programming (ILP)



CCM Variant Model
Runtime Description of Hard- & Software Infrastructure

CCM Structure Model
Architecture of Hard- & Software System

CCM Behavior Models

QCL Contracts
Characterizing Non-functional Behavior of Implementations

ILP

Decision Variables

Map to HW

Select Impl.

Usage Variables

Objective Function

Constraints

NFP Requirements

NFP Provisions

Knapsack

Resource Requirements

Resource Provisions

fixed

Knapsack

Architectural Constraints

TECHNISCHE
UNIVERSITÄT
DRESDEN

ILP by Example

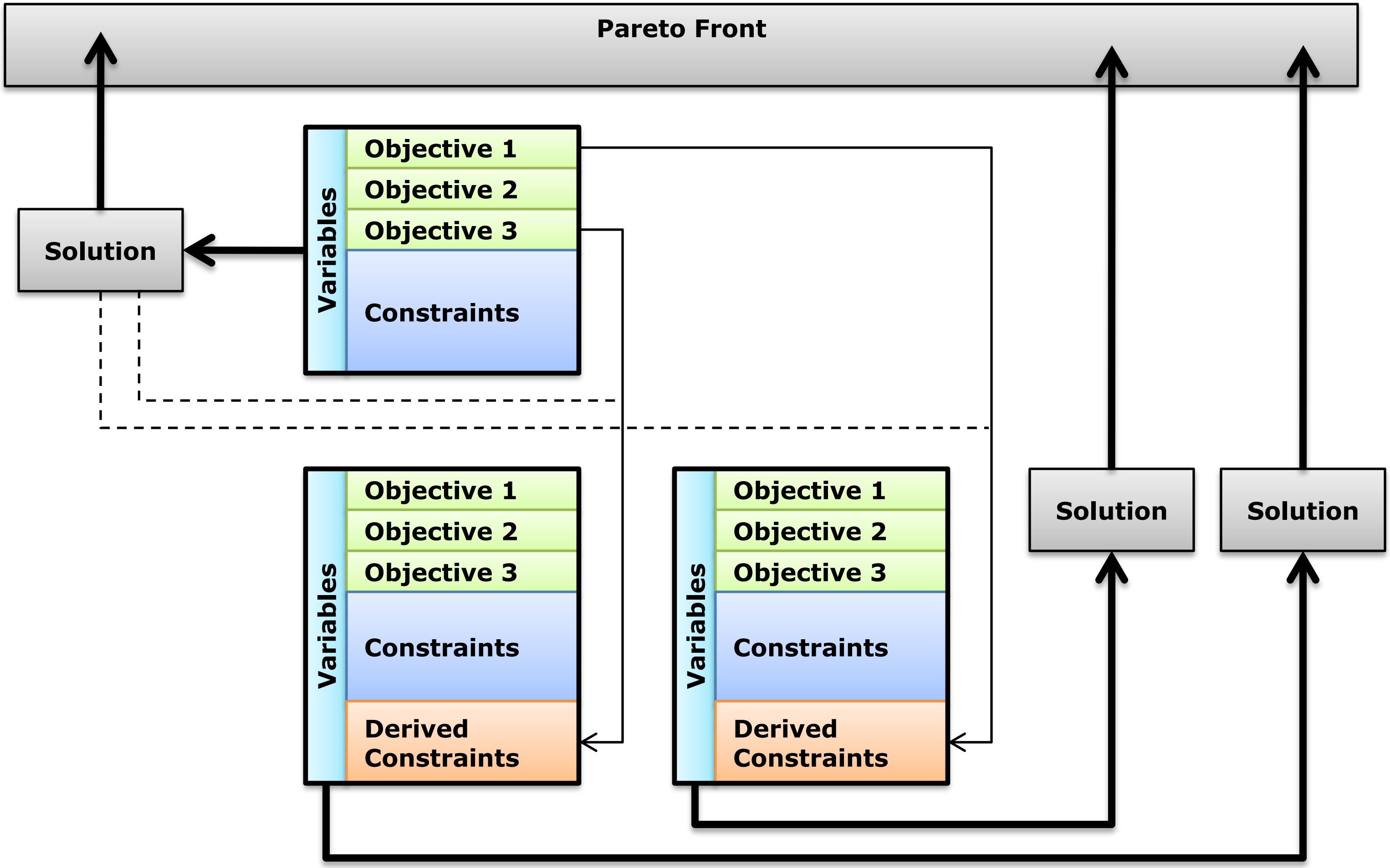
```
/* objective function: minimize energy consumption (based on cpu_time) */
min: 5700.0 b#Quicksort#delayed#R1 + 495.0 b#UnsortedFilter#slow#R1
+ 10285.0 b#Quicksort#immediate#R1 + 6160.0 b#Javasort#immediate#R1
+ 385.0 b#UnsortedFilter#fast#R1 + 2250.0 b#Random#slow#R1
+ 5940.0 b#Javasort#delayed#R1 + 2695.0 b#Random#fast#R1;

/* architectural constraints */
b#Random#fast#R1 + b#Random#slow#R1 = b#Quicksort#delayed#R1 + b#Quicksort#immediate#R1
                                      + b#Javasort#immediate#R1 + b#Javasort#delayed#R1;
b#UnsortedFilter#fast#R1 + b#UnsortedFilter#slow#R1 = 1;
b#Quicksort#immediate#R1 + b#Quicksort#delayed#R1
+ b#Javasort#immediate#R1 + b#Javasort#delayed#R1 = b#UnsortedFilter#slow#R1
                                                  + b#UnsortedFilter#fast#R1;

/* resource negotiation */
usage#R1#Core[TM]_i7_CPU_Q_720_@_1.60GHz#frequency <= 1596.0;
usage#R1#Core[TM]_i7_CPU_Q_720_@_1.60GHz#frequency >= 0;
usage#R1#Core[TM]_i7_CPU_Q_720_@_1.60GHz#frequency =
  100 b#Javasort#delayed#R1 + 100 b#UnsortedFilter#slow#R1 + 100 b#Quicksort#delayed#R1
+ 300 b#Random#fast#R1 + 300 b#Quicksort#immediate#R1 + 100 b#Random#slow#R1
+ 300 b#Javasort#immediate#R1 + 300 b#UnsortedFilter#fast#R1;

...
```

ILP by Example

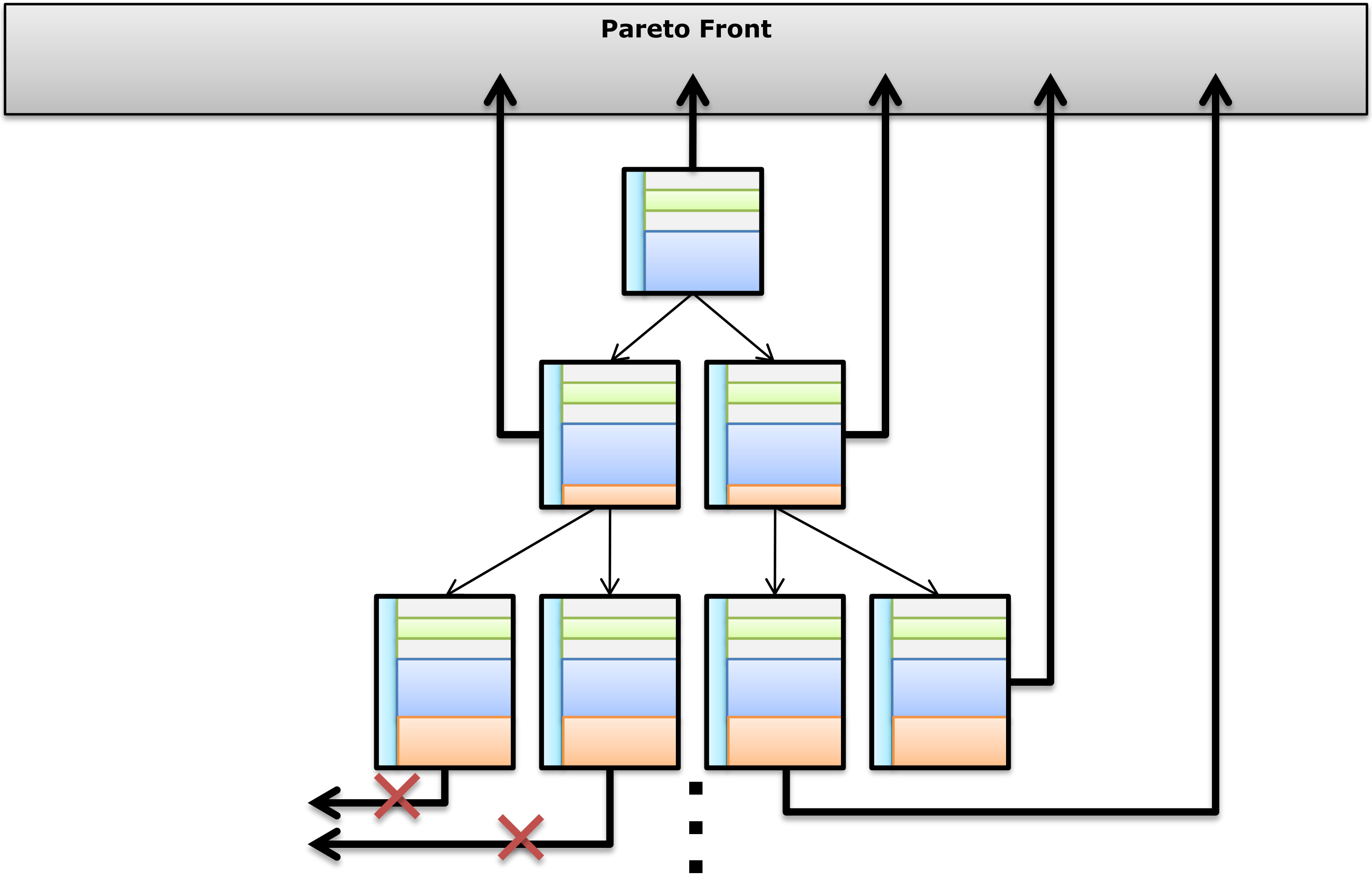
```
...

/* software NFP negotiation */
Sort#response_time = 382.05714282441 b#Quicksort#delayed#R1
                   + 377.31428570997804 b#Quicksort#immediate#R1
                   + 399.771428570494 b#Javasort#immediate#R1
                   + 416.34285718949195 b#Javasort#delayed#R1;
Filter#response_time = 23.921216866850248 b#UnsortedFilter#slow#R1
                     + 28.407017552658598 b#UnsortedFilter#fast#R1;
ListGen#response_time = 107.6078431285458 b#Random#slow#R1
                      + 106.7843137012918 b#Random#fast#R1;
Sort#response_time >= 50 b#UnsortedFilter#fast#R1;
ListGen#response_time >= 50 b#Quicksort#delayed#R1 + 50 b#Javasort#immediate#R1
                                                   + 50 b#Javasort#delayed#R1;
/* user request */
Filter#response_time <= 200.0;

/* boolean restriction */
binary b#Quicksort#delayed#R1, b#UnsortedFilter#slow#R1, b#Quicksort#immediate#R1,
b#Javasort#immediate#R1, b#UnsortedFilter#fast#R1, b#Random#slow#R1,
b#Javasort#delayed#R1, b#Random#fast#R1;
```

Contract Negotiation by MOILP

Klein und Hannan ´82



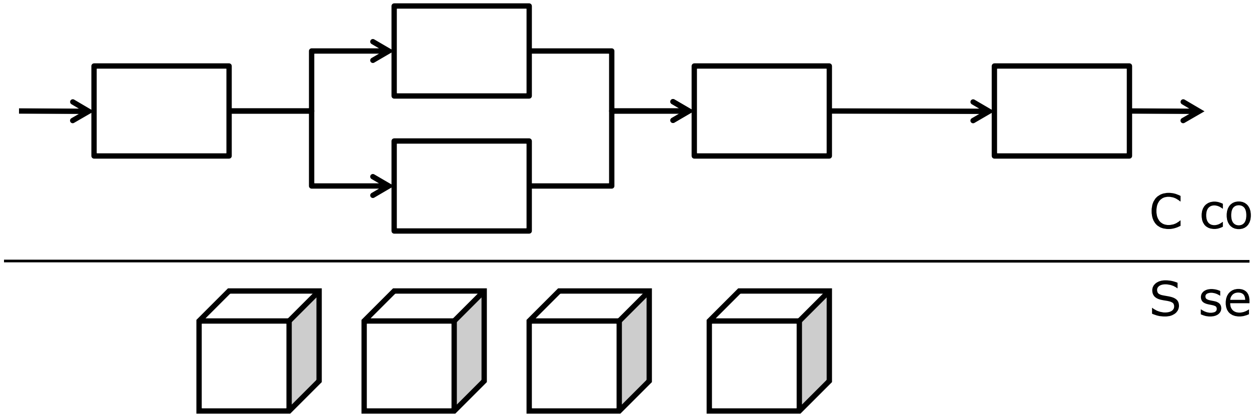
Contract Negotiation by MOILP



Quadradic Growth until Termination

Scalability Analysis [GWR+13]

- Performed on **data-flow graphs (pipe-and-filter style)**

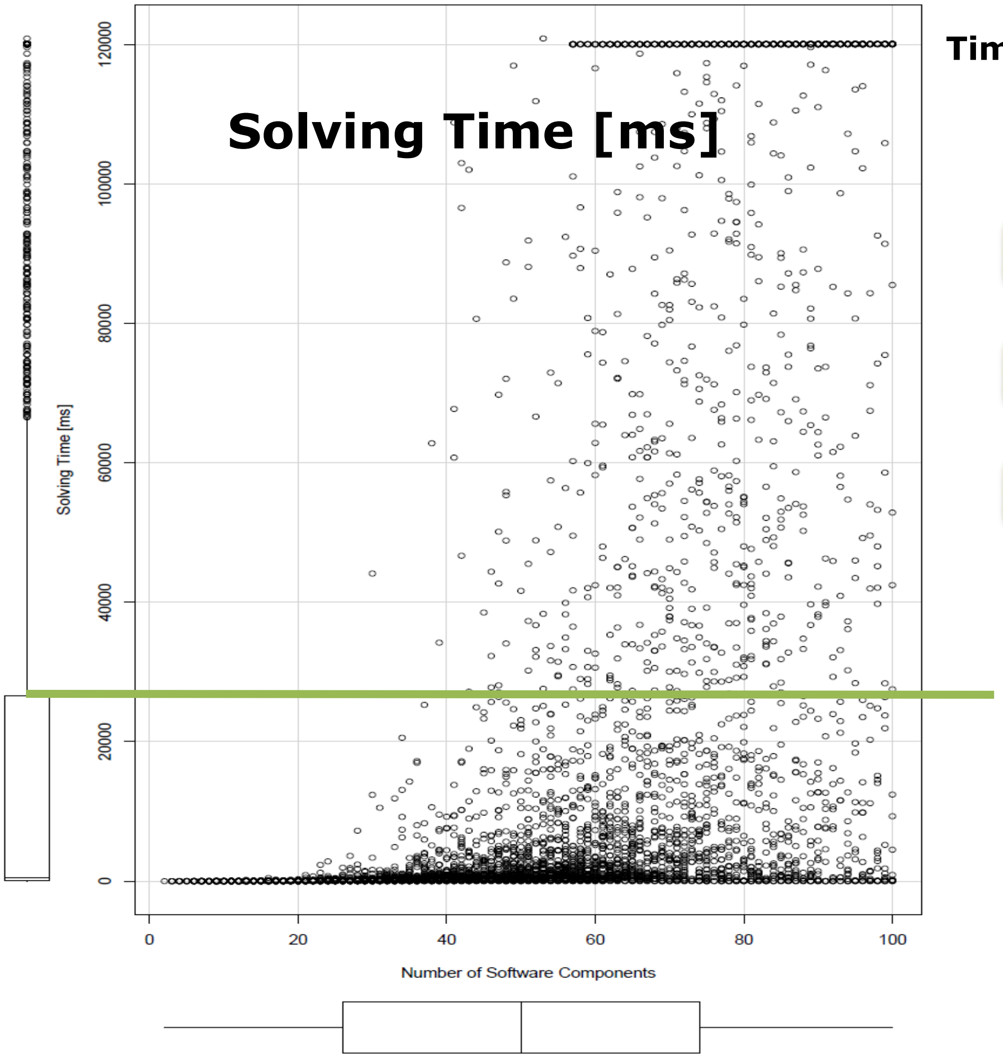


C components

S servers

- Measurements taken for C x S systems from **C = [2..100] and S = [2..100]**
- **All measurements made on Alienware X51** (Win7 64bit, SSD HDD, 8GB DDR1600 RAM, Intel Core i7-2600 with 4 physical cores at 3.4GHz)
- Concrete numbers will differ on other machines, solvers, etc.
- Focus on **principle findings**.

Scalability Analysis: ILP [GWR+13]



Timeout: 2min

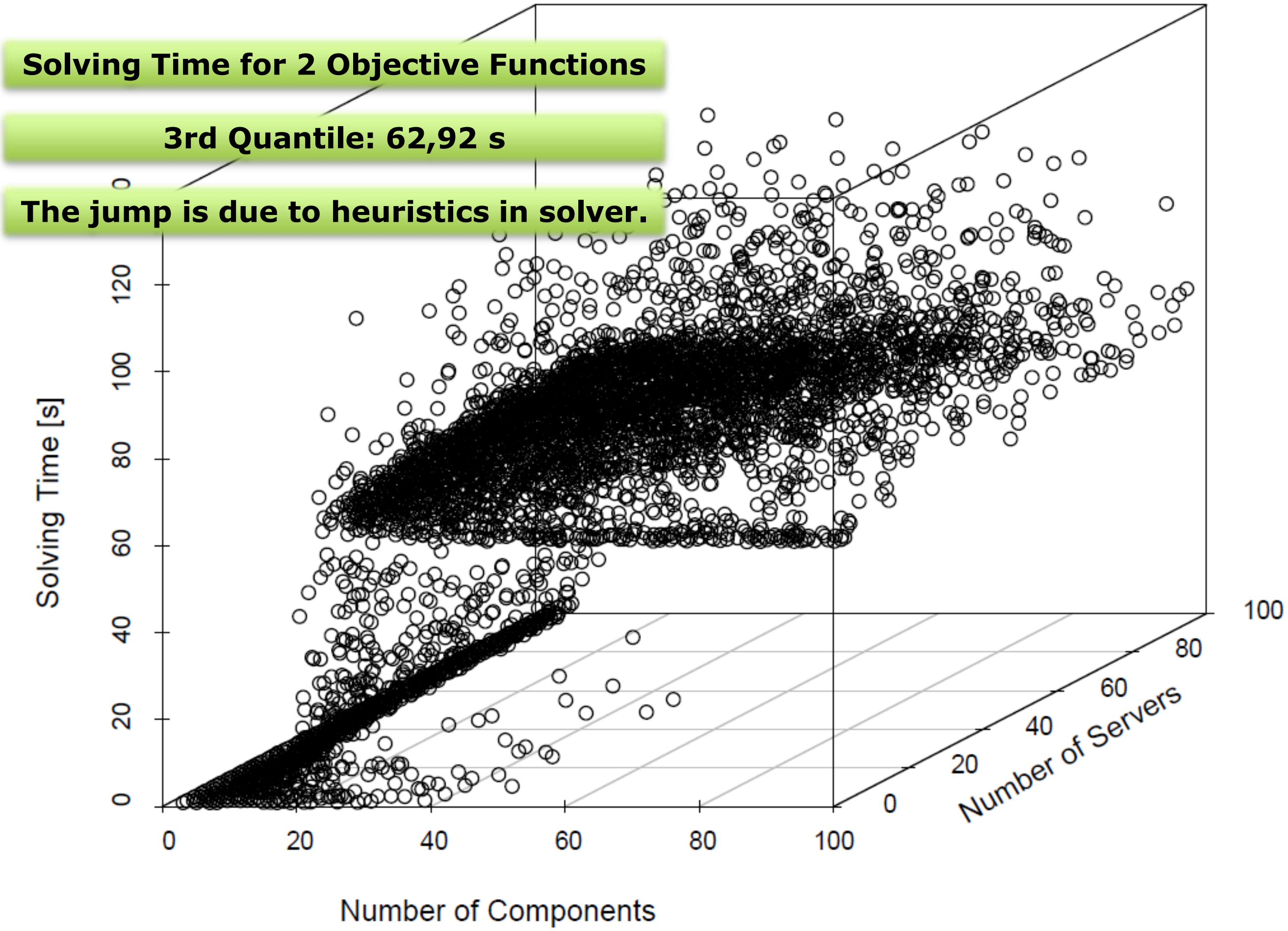
Feasible up to 100x100

Predictable up to 25 Components

Reason: Worst-case situations

3rd Quartile: 26,58s

Scalability Analysis: MOILP

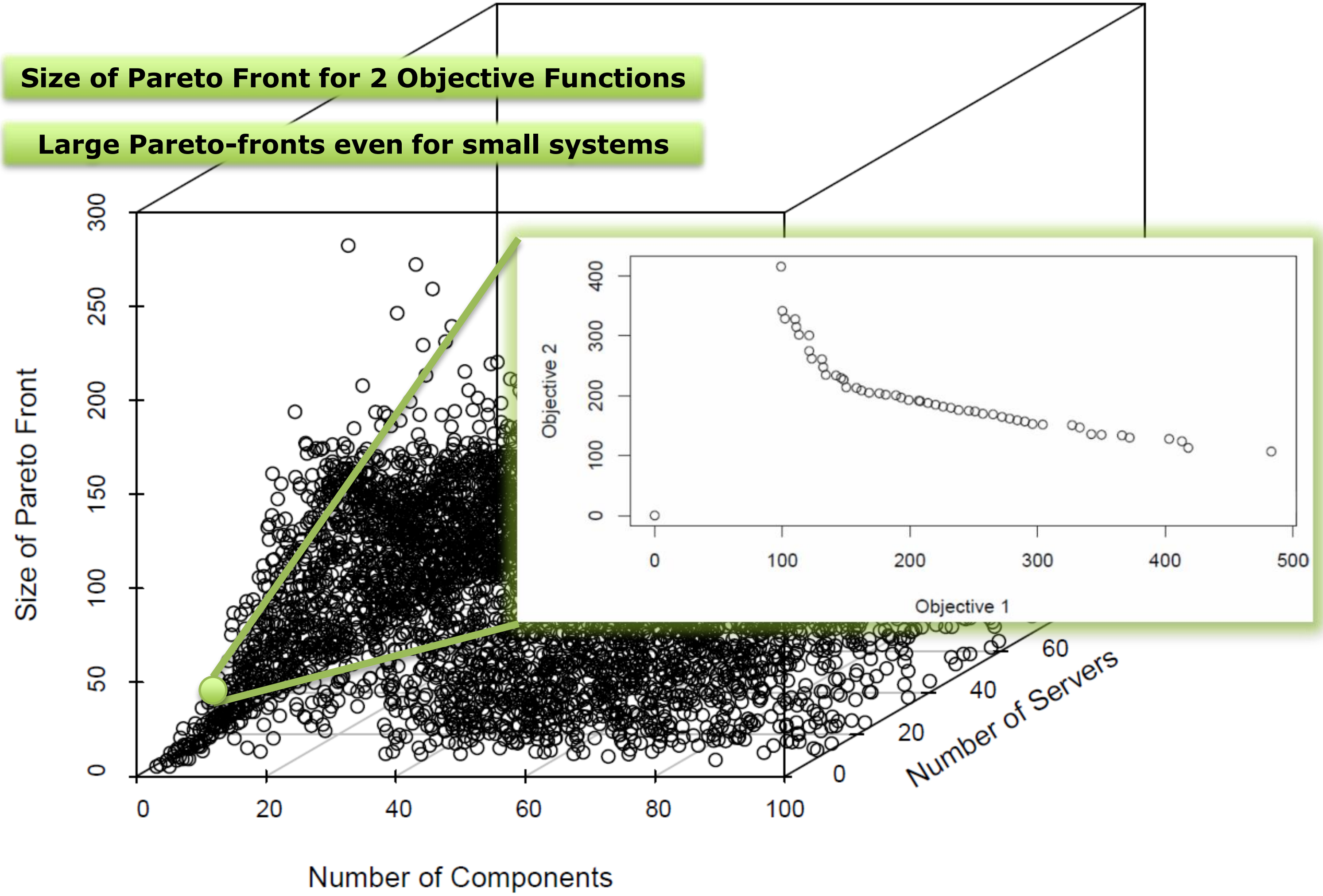


Solving Time for 2 Objective Functions

3rd Quantile: 62,92 s

The jump is due to heuristics in solver.

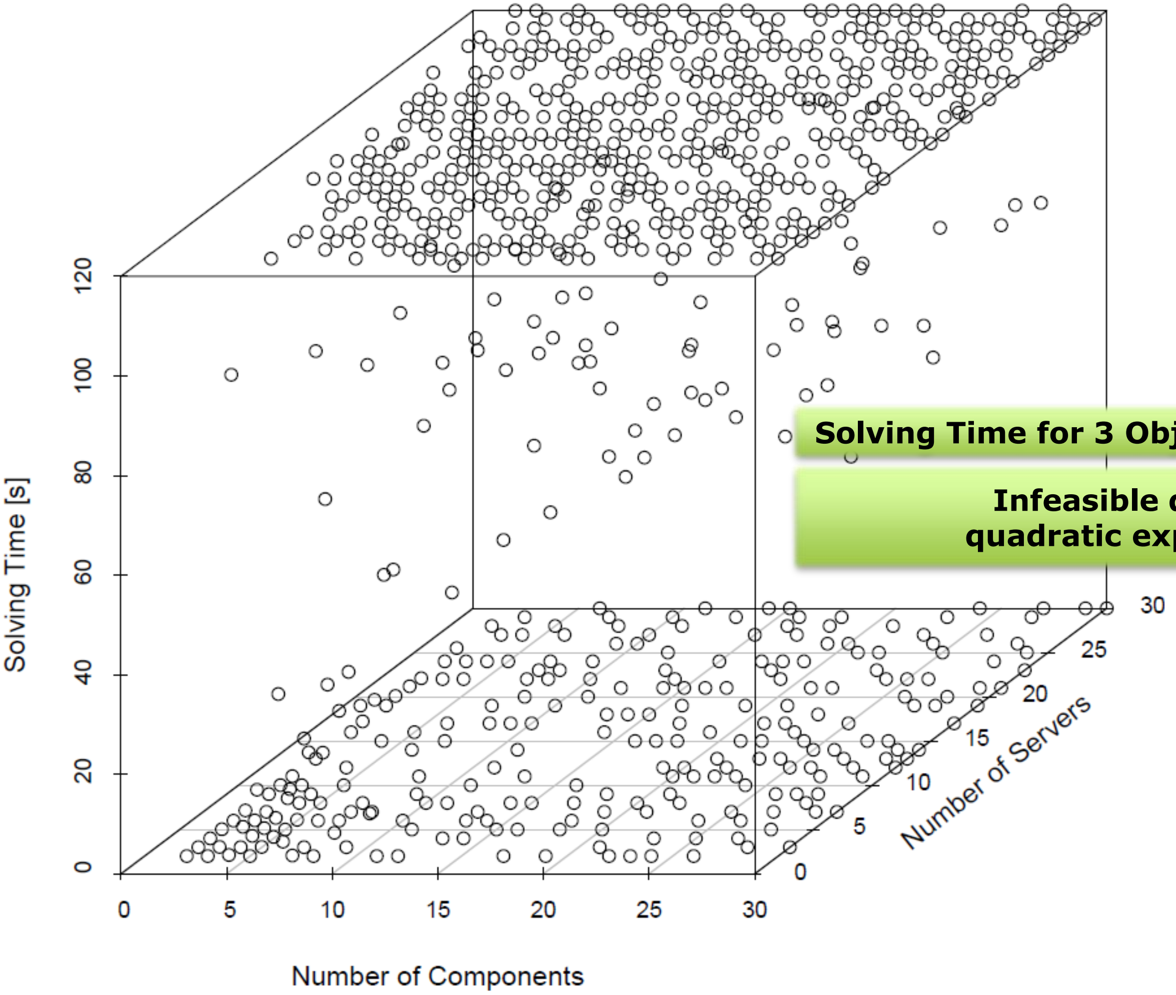
Scalability Analysis: MOILP



Size of Pareto Front for 2 Objective Functions

Large Pareto-fronts even for small systems

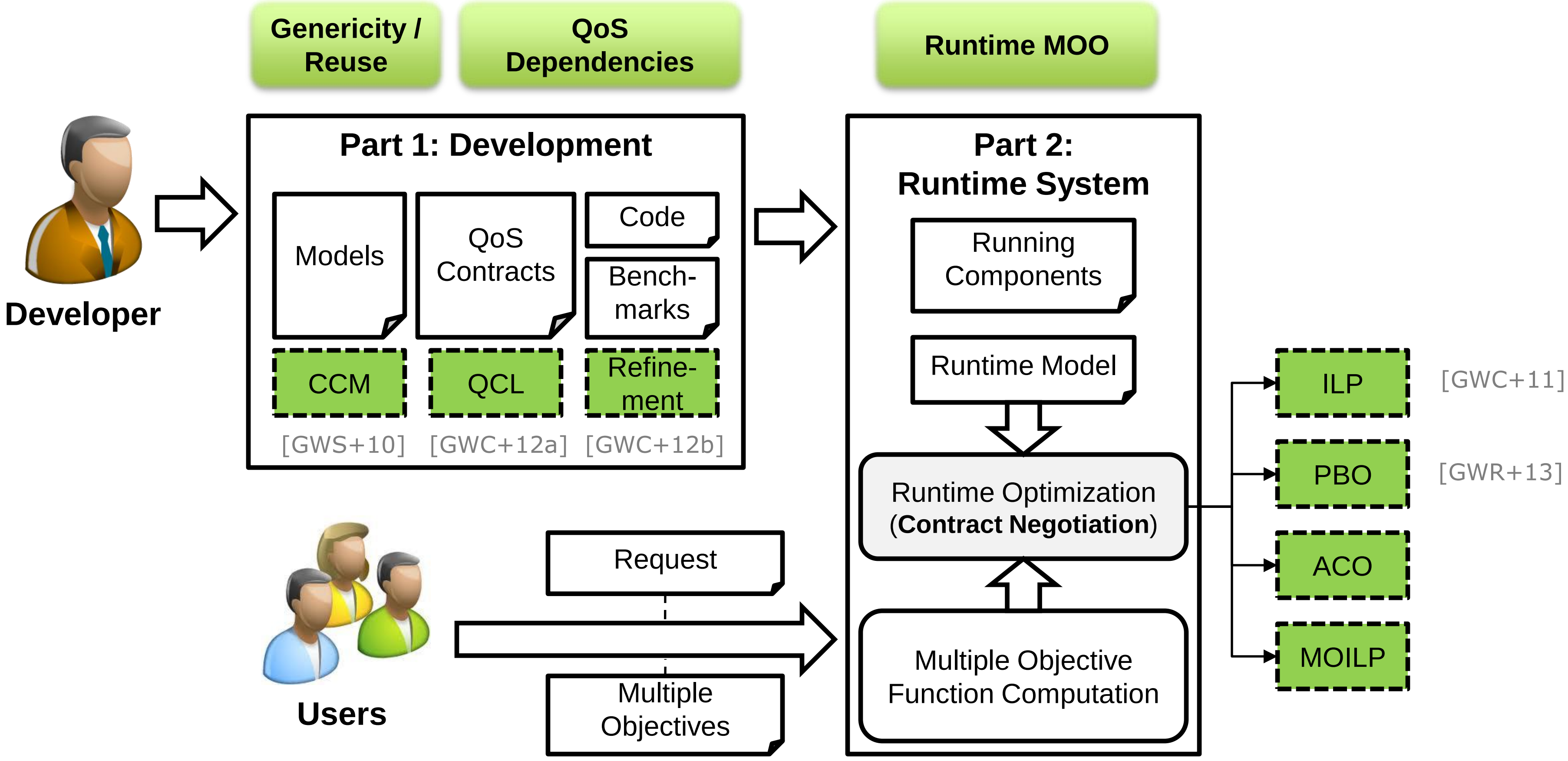
Scalability Analysis: MOILP



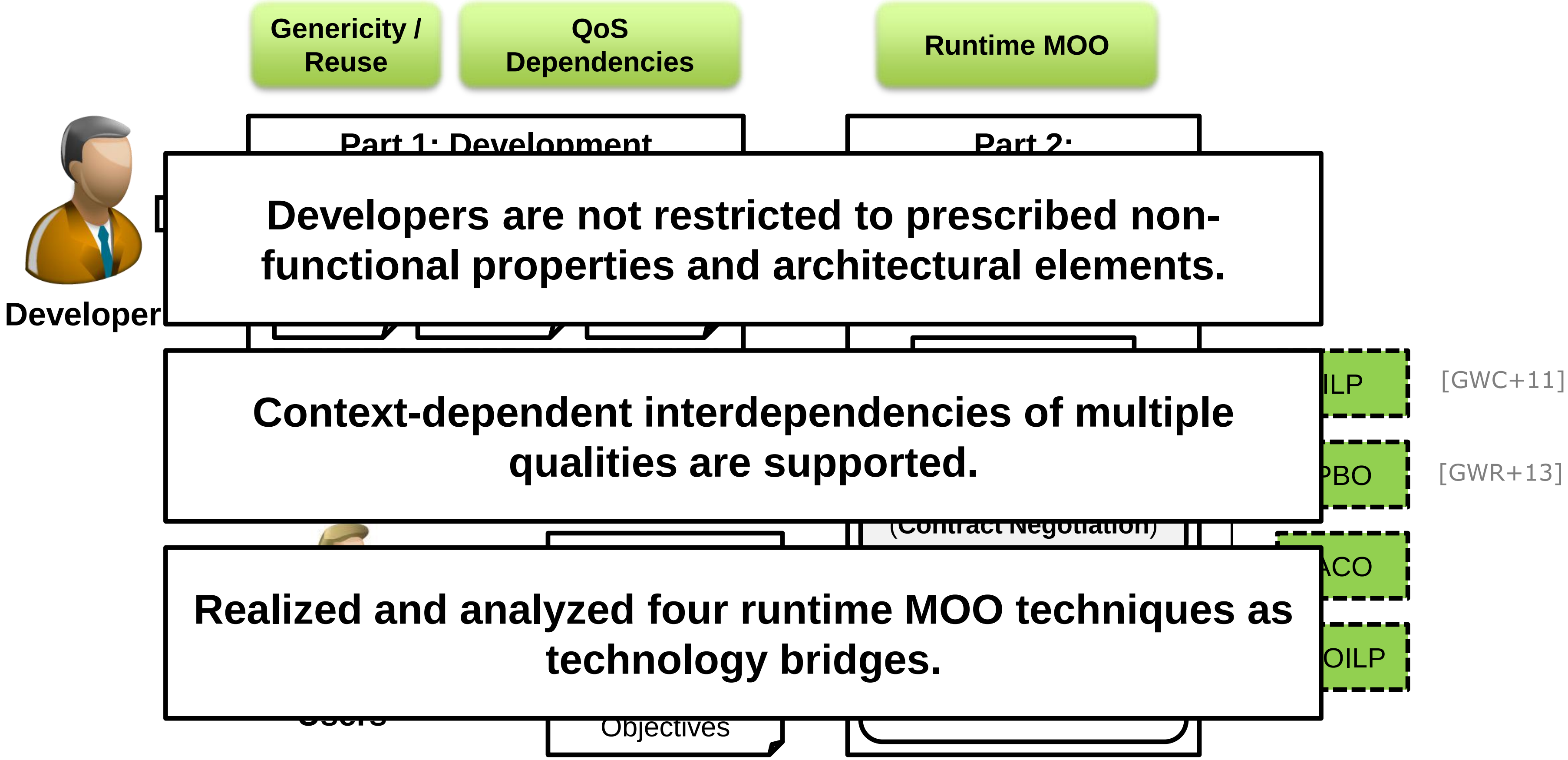
Solving Time for 3 Objective Functions

Infeasible due to quadratic explosion.

Contributions



Contributions



Developers are not restricted to prescribed non-functional properties and architectural elements.

Context-dependent interdependencies of multiple qualities are supported.

Realized and analyzed four runtime MOO techniques as technology bridges.

Future Work

- **Bootstrapping:** MQuAT for Monitoring, Optimization and Reconfiguration
  - Both are components with different implementations, too.
  - Scalability analysis is a first step for the optimization component
  - Collaboration planned with Prof. Fischer (Numerical Optimization Group)

- **Green Software Engineering** (CRC 912: HAEC, NFG ZESSY)
  [WGR+11, WRP+12, WRP+13, WGR13, GMT+13, WRG+13a, WRG+13b]
  - *Open Challenges*: Sustainability, Negotiation of Energy-Sources (Solar, Battery, Provider, etc.)

- Software Engineering for **Robotic** and **Cyber-Physical Systems**
  [GLR+11, GLP+12, PRG+12]
  - *Open Challenge*: Optimization across discrete and continuous system parts

Own Related Publications

[GWS+10] **S. Götz**, C. Wilke, M. Schmidt, S. Cech and U. Assmann. *Towards Energy Auto Tuning.* In: Proceedings of First Annual International Conference on Green Information Technology, GREEN IT 2010, GSTF (2010) p. 122-129.

[GWC+11] **S. Götz**, C. Wilke, S. Cech and U. Assmann. *Runtime Variability Management for Energy-efficient Software by Contract Negotiation.* In Proceedings of the 6th International Workshop on Models@run.time, ACM/IEEE (2011) p. 61-72.

[GWC+12a] **S. Götz**, C. Wilke, S. Cech and U. Assmann. *Architecture and Mechanisms of Energy Auto Tuning.* In Sustainable ICTs and Management Systems for Green Computing. IGI Global (2012) p. 45-73.

[GWC+12b] **S. Götz**, C. Wilke, S. Richly and U. Assmann. *Approximating Quality Contracts for Energy Auto-Tuning Software.* In Proceedings of First International Workshop on Green and Sustainable Software (GREENS'12), IEEE (2012) p. 8-14.

[GWR+13] **S. Götz**, C. Wilke, S. Richly and U. Aßmann. *Model-driven Self-Optimization using Integer Linear Programming and Pseudo-Boolean Optimization.* In Proceedings of the Fifth International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE), XPS Press (2013) p. 55-64.

[AGJ+13] U. Assmann, **S. Götz**, J.-M. Jezequel, B. Morin and M. Trapp. *Uses and Purposes of M@RT Systems.* To appear in State-of-the-Art Survey Volume on Models@run.time. Springer LNCS, 2013.

Own Future Work Publications

[WRG+13b] C. Wilke, S. Richly, **S. Götz**, C. Piechnick and U. Aßmann. *Energy Consumption and Efficiency in Mobile Applications: A User Feedback Study.* To appear in Proceedings of the IEEE International Conference on Green Computing and Communications (GreenCom), 2013.

[WRG+13a] C. Wilke, S. Richly, **S. Götz**, and U. Aßmann. *Energy Profiling as a Service.* To appear in GI Proceedings of Workshop "Umweltinformatik zwischen Nachhaltigkeit und Wandel" (UINW), 2013.

[GMT+13] **S. Götz**, J. Mendez, V. Thost and A.-Y. Turhan. *OWL 2 Reasoning To Detect Energy-Efficient Software Variants From Context.* To appear in Proceedings of the 10th OWL: Experiences and Directions Workshop (OWLED), 2013.

[PRG+13] G. Püschel, **S. Götz**, C. Wilke and U. Aßmann. *Towards Systematic Model-based Testing of Self-adaptive Systems.* In Proceedings of The Fifth International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE), XPS Press (2013), p. 65-70.

[WGR13] C. Wilke, **S. Götz** and S. Richly. *JouleUnit – A Generic Framework for Software Energy Profiling and Testing.* In Proceedings of the 1st Workshop "Green In Software Engineering Green By Software Engineering" (GIBSE), ACM/IEEE (2013) p. 9-13.

[WRP+13] C. Wilke, S. Richly, C. Piechnick, **S. Götz**, G. Püschel and U. Aßmann. *Comparing Mobile Applications' Energy Consumption.* In Proceedings of The 28th Annual ACM Symposium on Applied Computing (SAC 2013), ACM (2013) p. 1177-1179.

Own Future Work Publications

[WRP+12] C. Wilke, S. Richly, G. Püschel, C. Piechnick, **S. Götz** and Uwe Assmann. *Energy Labels for Mobile Applications.* To appear in Proceedings of 1. Workshop zur Entwicklung energiebewusster Software / First Workshop for the Development of Energy-aware Software (EEbS 2012), 2012.

[WGR+11] C. Wilke, **S. Götz**, J. Reimann and U. Assmann. *Vision Paper: Towards Model-Based Energy Testing.* In Proceedings of 14th International Conference on Model Driven Engineering Languages and Systems (MODELS 2011), Springer (2011) p. 480-489

[PRG+12] C. Piechnick, S. Richly, **S. Götz**, C. Wilke and U. Aßmann. *Using Role-Based Composition to Support Unanticipated, Dynamic Adaptation - Smart Application Grids.* **(Best Paper Award)** In Proceedings of The Fourth International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE), XPS Press (2012) pp. 93-102

[GLP+12] **S. Götz**, M. Leuthäuser, C. Piechnick, J. Reimann, S. Richly, J. Schroeter, C. Wilke und U. Aßmann. Entwicklung cyber-physikalischer Systeme am Beispiel des NAO Roboters. In Proceedings of Chemnitz Linux-Days, Universitätsverlag Chemnitz (2012) p. 42-52

[GLR+11] **S. Götz**, M. Leuthäuser, J. Reimann, J. Schroeter, C. Wende, C. Wilke and U. Assmann. *A Role-based Language for Collaborative Robot Applications.* In Proceedings of 1st International ISoLA Workshop on Software Aspects of Robotic Systems (ISOLA SARS 2011), Springer (2011) p. 1-15

TECHNISCHE
UNIVERSITÄT
DRESDEN

Literature

[KC03] J. O. Kephart and D. M. Chess. *The vision of autonomic computing*. In: IEEE Computer, 36:41-50, January 2003.

[L97] R. Laddaga. *DARPA self adaptive software broad agency announcement (baa) 98-12 proposer information pamphlet - excerpt*. http://people.csail.mit.edu/rladdaga/BAA98-12excerpt.html, December 1998.

[NW99] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley Interscience, 1999.

[RZ03] S. Röttger and S. Zschaler. *CQML+: Enhancements to CQML*. In Proceedings of the 1st International Workshop on Quality of Service in Component-Based Software Engineering, pages 43-56. Cepadues-Editions, 2003.

[ST09] M. Salehie and L. Tahvildari. *Self-adaptive software: Landscape and research challenges*. ACM Trans. Auton. Adapt. Syst., 4:14:1-14:42, May 2009.

[S10] Josef Spillner: *Methodik und Referenzarchitektur zur inkrementellen Verbesserung der Metaqualität einer vertragsgebundenen, heterogenen und verteilten Dienstausführung*. Dissertation. TU Dresden. 2010.

[ZM03] S. Zschaler and M. Meyerhöfer. *Explicit Modelling of QoS-Dependencies*. In Proceedings of the 1st International Workshop on Quality of Service in Component-Based Software Engineering, p. 57-66. Cepadues-Editions, 2003.