TTC 2018 CASE PRESENTATION

Quality-based Software-Selection and Hardware-Mapping as a Model Transformation Problem

Sebastian Götz, Johannes Mey, Rene Schöne and Uwe Aßmann

The TTC Case

Optimally combine **heterogeneous hardware** and **adaptive software**

by deriving a

solution model from a **problem model**.

Our History of the Case

In the beginning, there was a PhD in 2013:

- [Götz 2013] *Multi-Quality Auto-Tuning by Contract Negotiation*

Our History of the Case

In the beginning, there was a PhD in 2013:

- [Götz 2013] *Multi-Quality Auto-Tuning by Contract Negotiation*

which was improved by faster intermediate model generation in 2016:

- [Schöne et al. 2016] *Incremental Runtime-Generation of Optimisation Problems Using RAG-Controlled Rewriting*

Our History of the Case

In the beginning, there was a PhD in 2013:

- [Götz 2013] *Multi-Quality Auto-Tuning by Contract Negotiation*

which was improved by faster intermediate model generation in 2016:

- [Schöne et al. 2016] *Incremental Runtime-Generation of Optimisation Problems Using RAG-Controlled Rewriting*

which was still a bit slow, so now there is

- TTC 2018

The Problem

Problem 1: *"Software Selection"*

- **Software model**:
  - Software *component* specifications:
    - functionality
  - *Implementations* of component specs:
    - provide non-functional properties
    - require other components

Selection Task

- Fulfill requests
  - chose implementations
  - ensure non-functional requirements
- **Solution Part 1**: Trees of assignments

Problem 2: *“Hardware Mapping”*

- **Hardware model**
  - *Resources* with sub-resources and properties
- **Contracts**
  - Implementations specify resource requirements

Resource Allocation Task

- Map assignments to hardware
  - ensure resource requirements
- **Solution Part 2**: Resource mapping

Problem 3: *"Quality-Based"*

- **Contracts**
  - Implementations provide non-functional properties depending on hardware

Optimization task

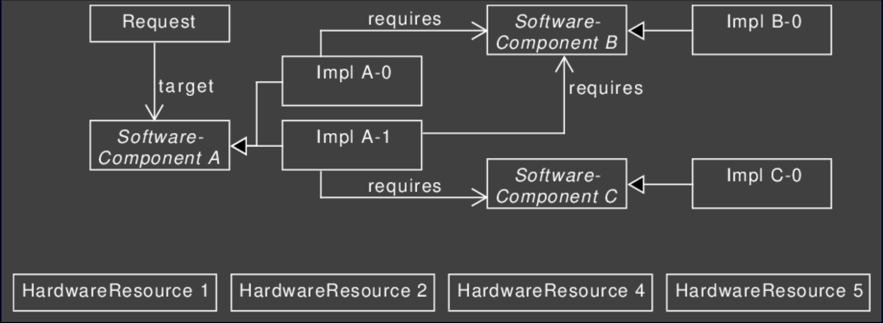
- Optimize aggregated non-functional property of system
  - *Here*: minimize energy
- **Solution Part 3**: Assignments + mapping with minimal energy

The Models in Detail

- Model: two *grammars* with *overlay edges* and *connecting references*
  - Problem model:
    - software and hardware part
  - Solution model:
    - tree of dependent assignments
- Grammar?
  - Reference Attribute Grammar: efficient analysis
  - Parser available
  - Simple solution within model

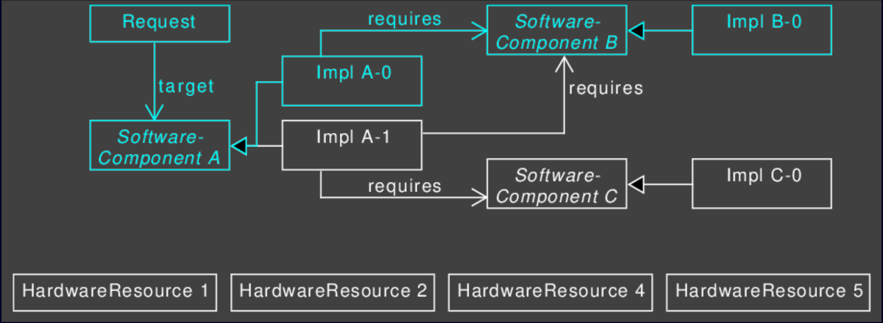
The Models in Detail

Components



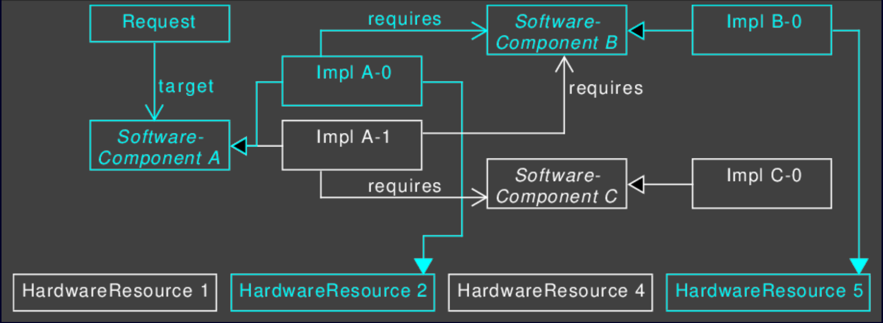
The Models in Detail

Solution Part 1: Implementation Selection



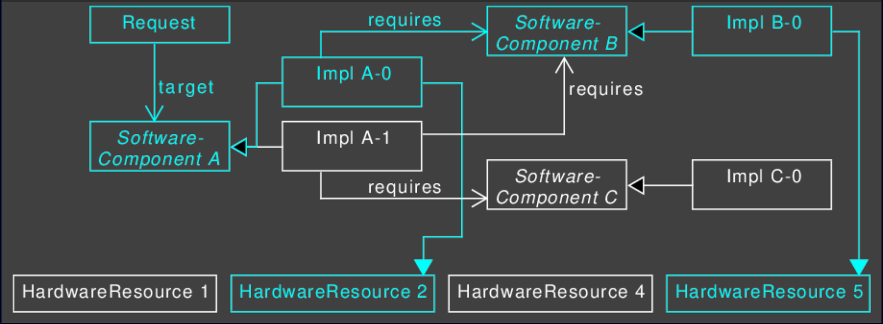
The Models in Detail

Solution Part 2: Hardware Mapping



The Models in Detail

Solution Part 3: Optimization

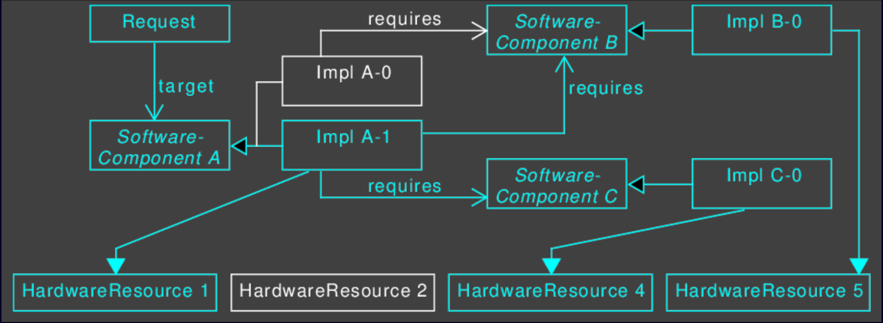


Valid: ✓

Optimal: ✗

The Models in Detail

Solution Part 3: Optimization



Valid: ✓
Optimal: 👌

Task and Solutions

Case Scenarios

- Five **sizes**:
  - minimal, small, medium, large, huge
- Three **types**:
  - standard
  - more hardware components
  - more (complex) software components
- Flexible scenario **generator**:
  - 10 parameters for software/hardware config
  - Fixed hardware types, and software properties
  - Flexible shape of sotware model and solution tree

Case Scenarios

ID	Requests	Impl’s	Resources	Scenario
0	1	1	1	minimal
1	1	6	5	small
2	1	6	15	small-hw
3	1	62	47	small-sw
4	15	30	68	medium
5	15	30	225	medium-hw
6	10	155	465	medium-sw
7	20	60	90	large
8	20	60	300	large-hw
9	20	310	930	large-sw
10	50	150	225	huge
11	50	150	750	huge-hw
12	50	620	2325	huge-sw

A Simple Attribute Grammar Reference Solution

- Simple reference implementation
  - Based on reference attribute grammar
  - Iterator over model
  - Some pruning
- Performance:
  - Almost full state space exploration
  - Encouraging for TTC partitipants
  - Always finds optimal solution . . . eventually

Evaluation criteria

Solution time

Time to compute a valid solution

Solution quality:

Validity of solution + Quality of found objective value

Scalability:

Largest scenario for which a valid solution can be found

Measurement results

✅= valid and in time 🏁= valid, but timeout ❌= invalid

👌= optimal (if known from ILP solver)

Scenario	ACO	EMFeR	ILP (direct/ext)	Simple
0 trivial	6 👌	194 👌	24 / 21 👌	1 👌
1 small	8 👌/❌	212 👌	37 / 40 👌	6 👌
2 small-hw	11 👌	240 👌	44 / 61 👌	8 👌
3 small-sw	451 ✅	7min52s ❌	377 / 572 👌	15min❌
4 medium	1min33 ✅/❌	8min22s ❌	8min28s 👌/ 🏁	15min❌
5 medium-hw	4min48s ✅	11min15s ❌	15min 🏁/ ❌	15min❌
6 medium-sw	15min ❌	11min15s ❌	15min ❌	15min❌

Some Observations

- *ACO* sometimes returns invalid solutions
- *ILP direct* much better than *ILP external*
- *EMFeR* for scenarios 3-6 aborts search before timeout
- *Simple* either is fastest and optimal, or runs into timeout

References

[Götz 2013] Götz, Sebastian. “Multi-Quality Auto-Tuning by Contract Negotiation.” PhD Thesis, Technische Universität Dresden, 2013. http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-119938.

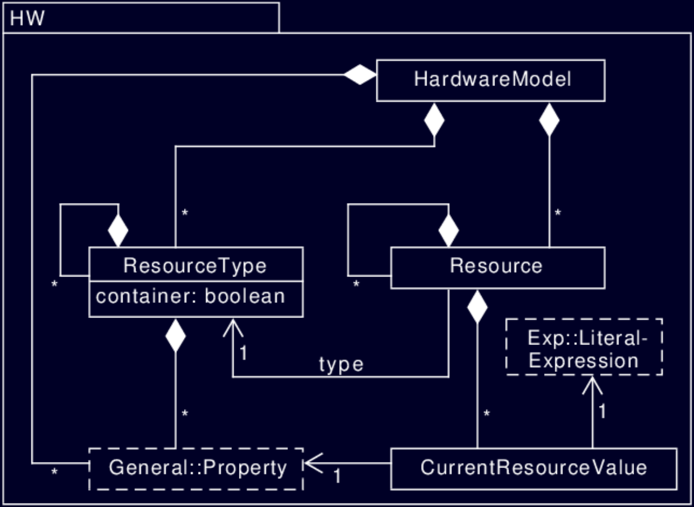
[Schöne et al. 2016] Schöne, René, Sebastian Götz, Uwe Aßmann, and Christoff Bürger. “Incremental Runtime-Generation of Optimisation Problems Using RAG-Controlled Rewriting.” In Proceedings of the 11th International Workshop on Models@run.Time. Saint-Malo: ceur, 2016. http://ceur-ws.org/Vol-1742/.

Backup

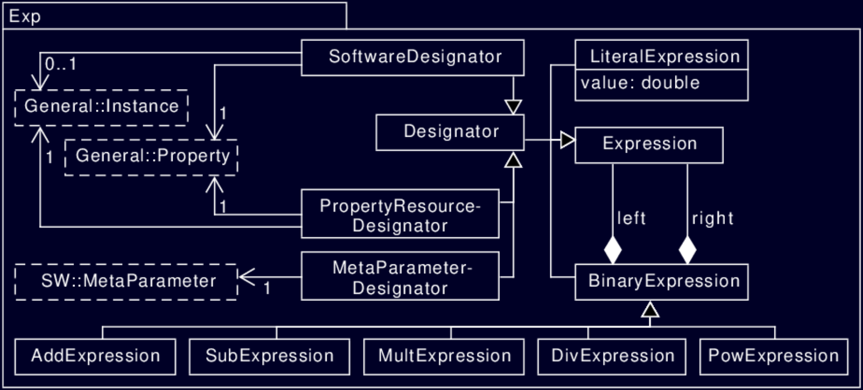
Questions to the Audience

- Accessibility of the benchmark?
- Explanation of the case clear enough?
- How complex was the problem (compared to previous years)?
- Anything missing or improvable in the benchmark framework?

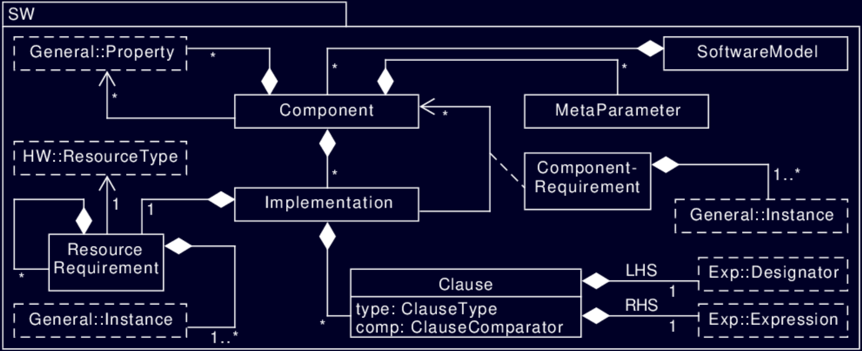
Grammar Hardware



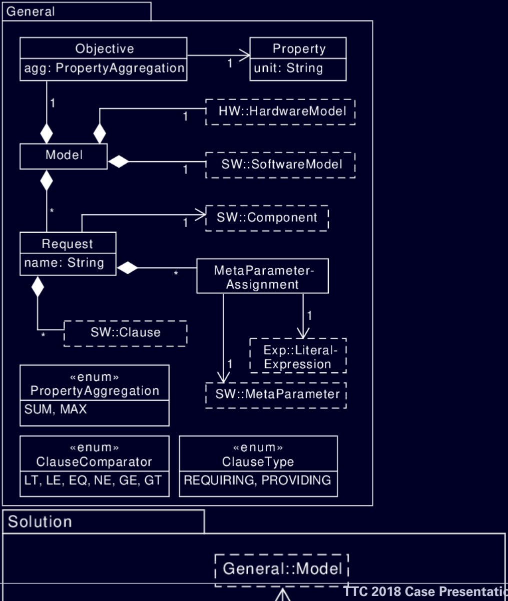
Grammar Expression



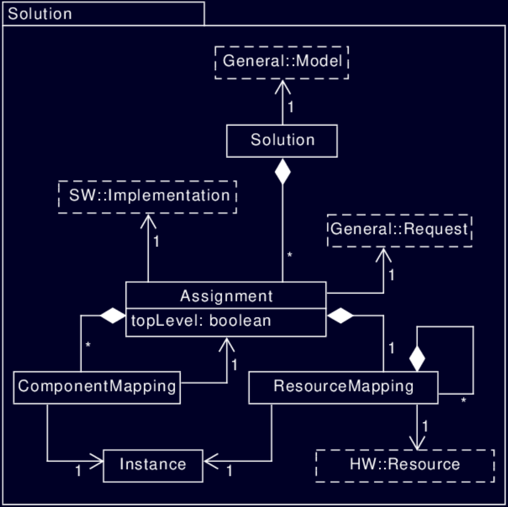
Grammar Software



Grammar General



Grammar Solution


Solution
General::Model
1
Solution
SW::Implementation
1
*
General::Request
1
Assignment
topLevel: boolean
*
1
1
ComponentMapping
ResourceMapping
*
1
1
Instance
1
HW::Resource