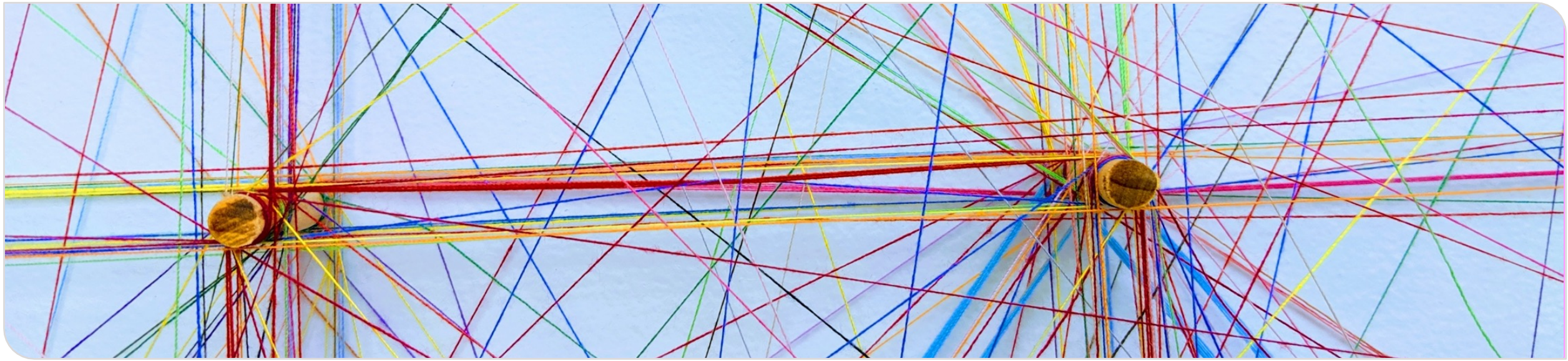
Karlsruhe Institute of Technology

Combined Modeling of Software and Hardware with Versions and Variants

Masters Thesis (Excerpt) – August 2023 – Philip Ochs

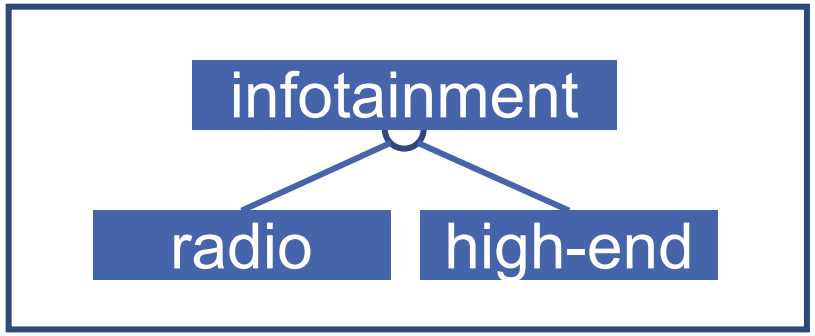


Motivation

- The term **cyber-physical system** describes the

  "[…] tight conjoining of and coordination between computational and physical resources." [1]

- Cyber-physical systems can be managed as **product line** [3]

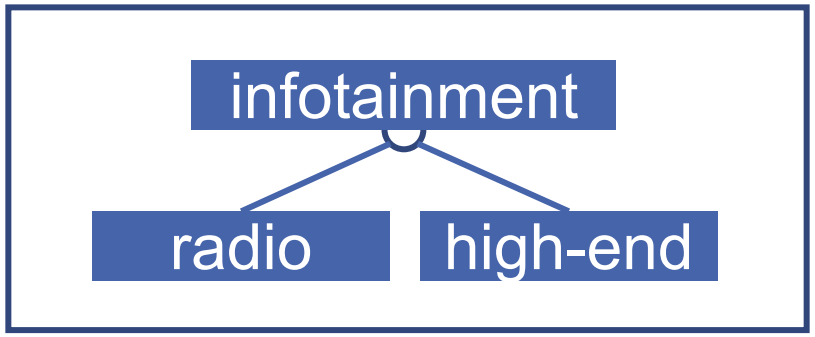


Feature Model

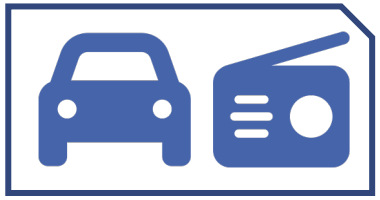
[2]

Motivation: Problem Space & Solution Space

Problem Space



Feature Model

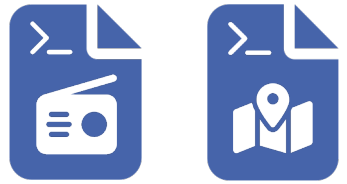


Config A

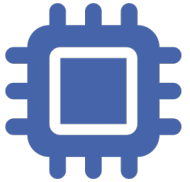
Config B

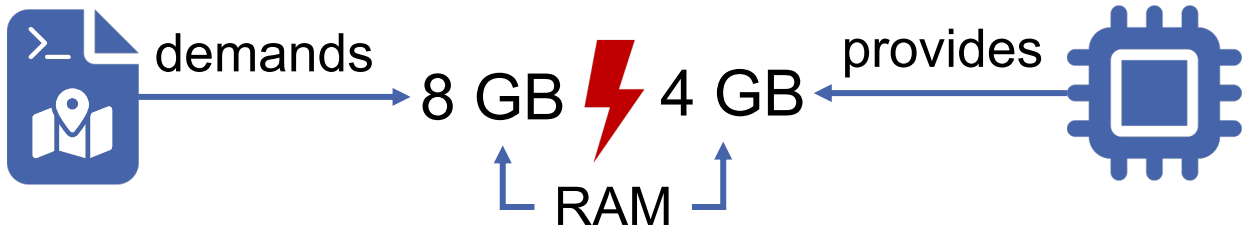
Solution Space

Software Components



Hardware Components

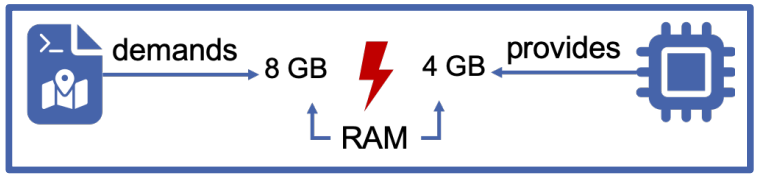




Research Goals

RQ1: How can solution space artefacts be described in a meta-model?

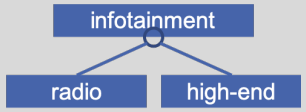
RQ2: How can be checked if a valid configuration is actually realizable?



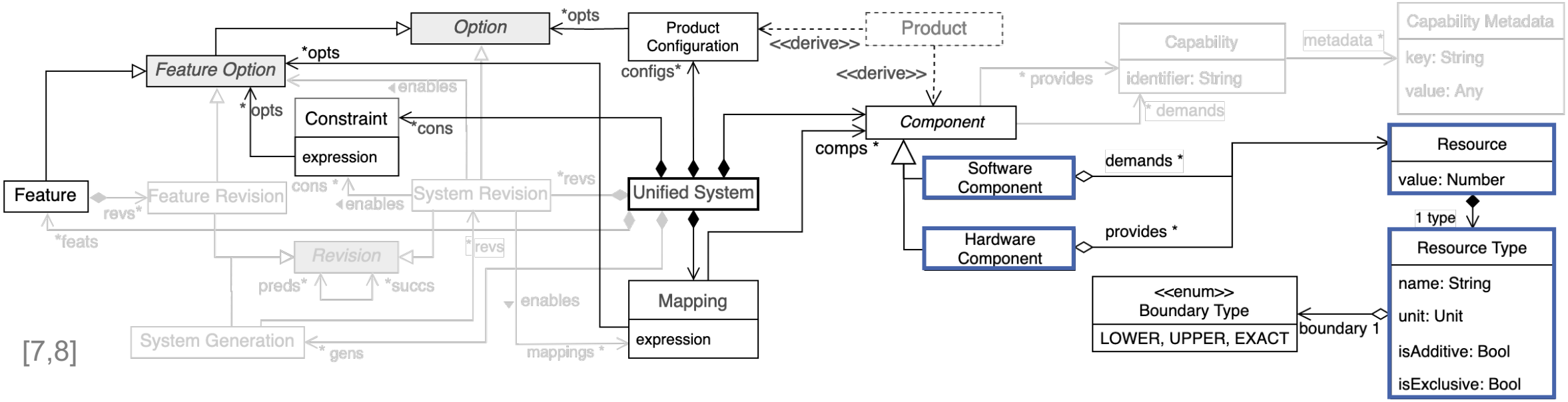
Unified Conceptual Model (UCM)

- Meta-Model to describe Problem and Solution Space

Problem Space



Solution Space

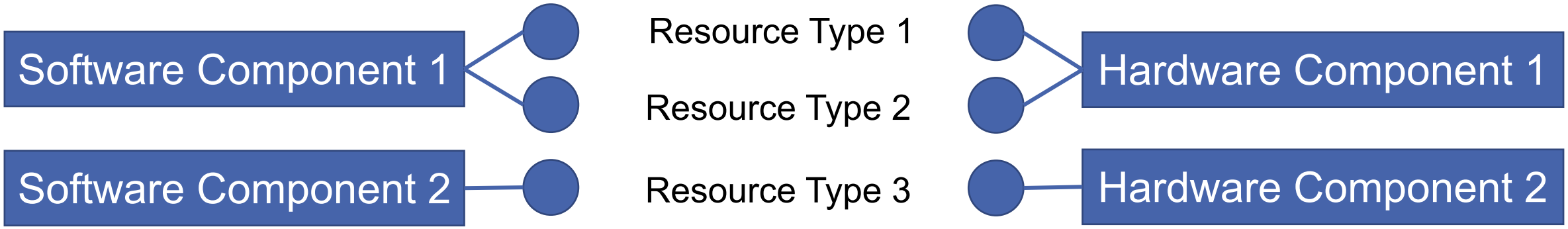


[7,8]

Ressource Allocation Problem

- Software components specify resource demands
- Hardware components specify resource provisionings
- Here: A **decision problem**

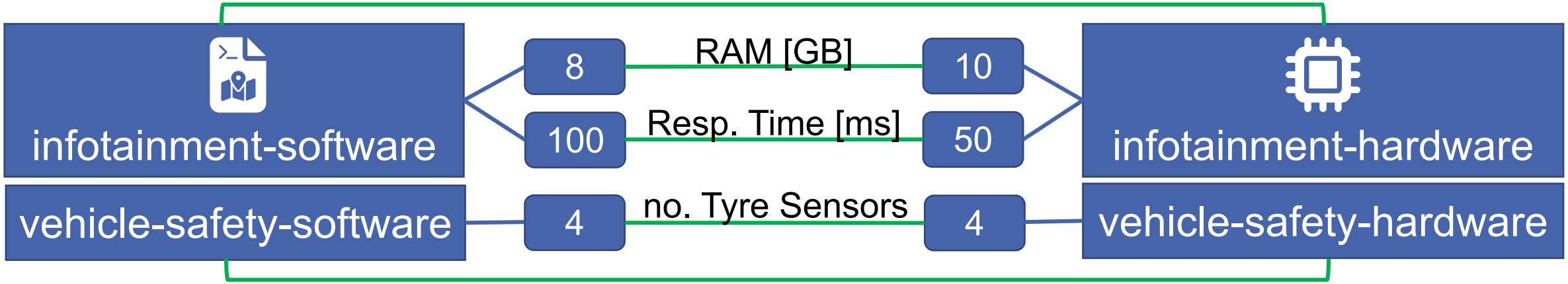
- **Goal: Find a resource assignment**:
  - from each resource demand
  - to a resource provisioning
  - satisfying all constraints
- **Component assignment** follows from resource assignment [4]



Ressource Allocation Problem: Example

- Software components specify resource demands
- Hardware components specify resource provisionings
- Here: A **decision problem**

- **Goal: Find a resource assignment**:
  - from each resource demand
  - to a resource provisioning
  - satisfying all constraints
- **Component assignment** follows from resource assignment [4]



Roadmap: Problem Space → Solution Space

Problem Space

Solution Space

Configuration

Mapping

Resource Demands

from SW Components

Resource Provisionings

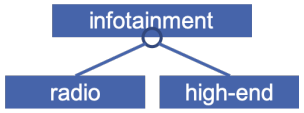
from HW Components

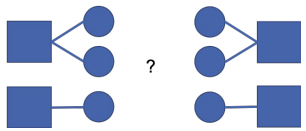
Resource Assignment Problem

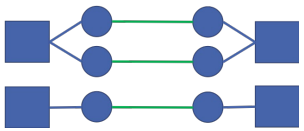
solve

Component Assignments

or unsat

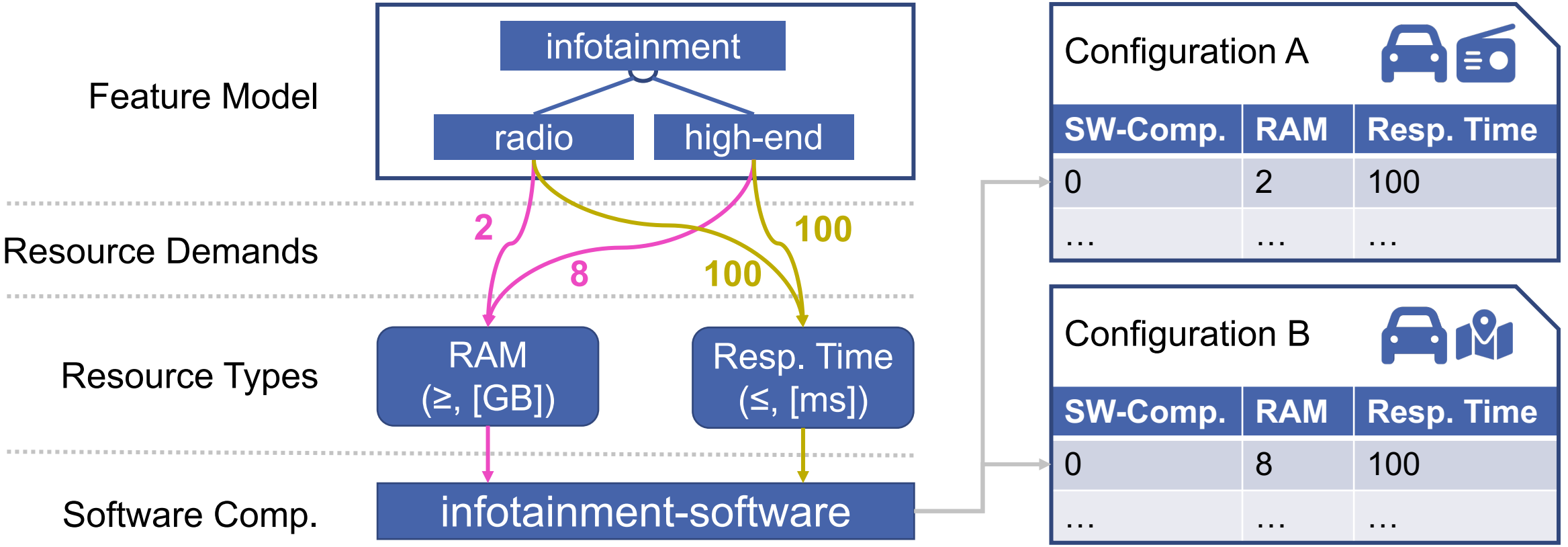






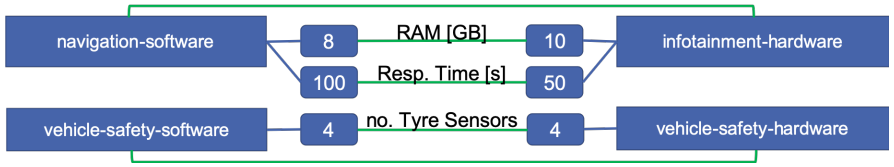
Mapping Variability-Aware Resource Demands

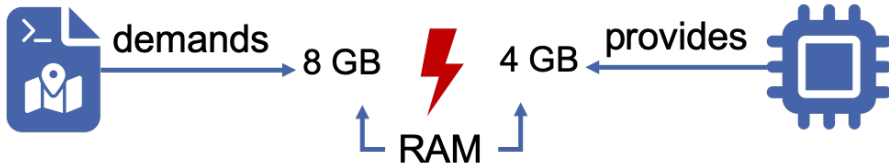
- **Software components** and their **demands** are defined by **features**



Solve the Resource Allocation Problem

- with (C)OTS tools, e.g. *Z3 Theorem Prover* [9]
- **2 possible results**:
  - problem **satisfiable**
    ⇒ software components can be assigned to hardware components
    ⇒ valid configuration also realizable
  - problem **unsatisfiable**
    ⇒ valid configuration not realizable

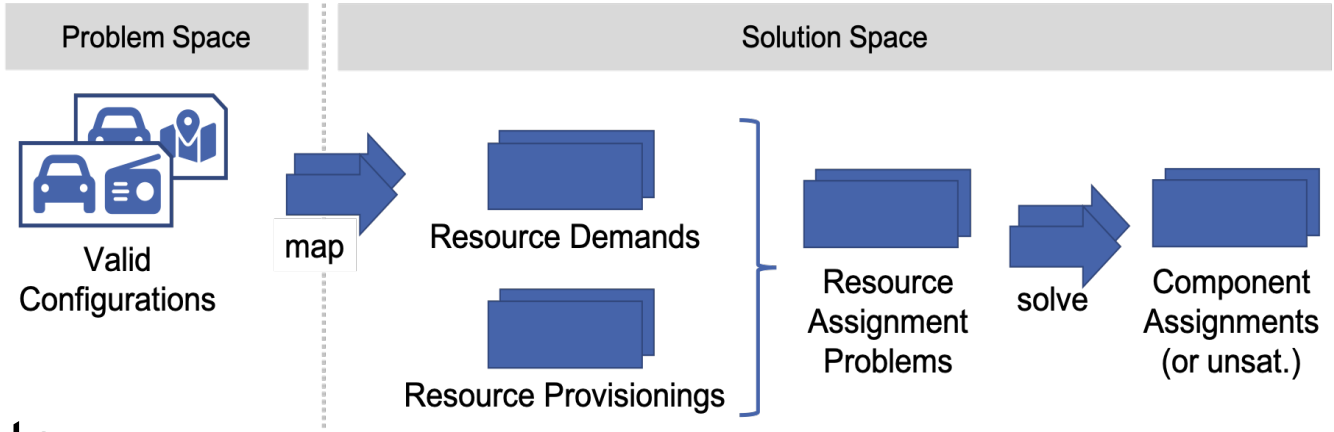




- Implications:
  - ∃ valid but non-realizable configuration ⇒ problem and solution space diverge
  - All valid configurations also realizable ⇒ consistent problem and solution space

Application Details

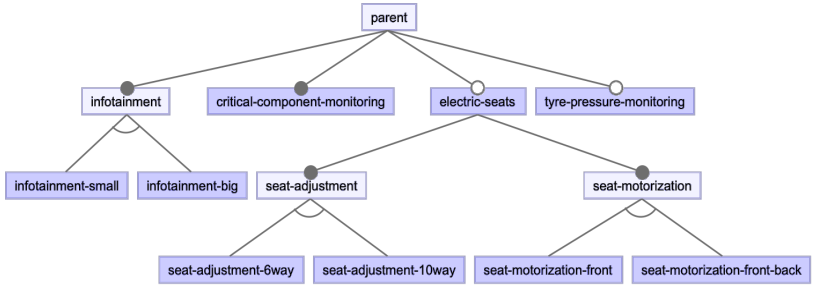
- Build-Time: Realizability
  - Product Line consistent
    ⇒ all configurations offered to a customer are known to be realizable



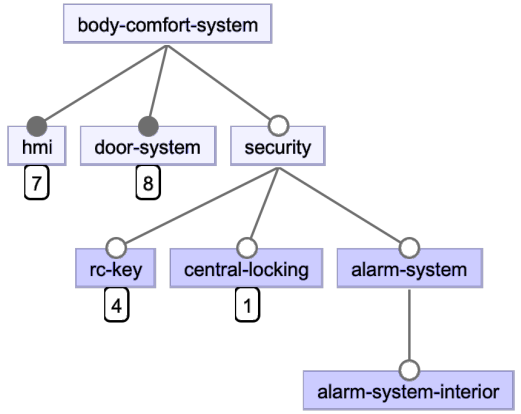
- Run-Time: Update-Ability
  - software component gets update
    → update changes its resource demands
  - Which product variants in field may receive the update so that they're still functioning?

Evaluating the Realizability Analysis Method

- Along **2 case studies**, consisting of problem and solution space
  - “Case Study 1” for verification



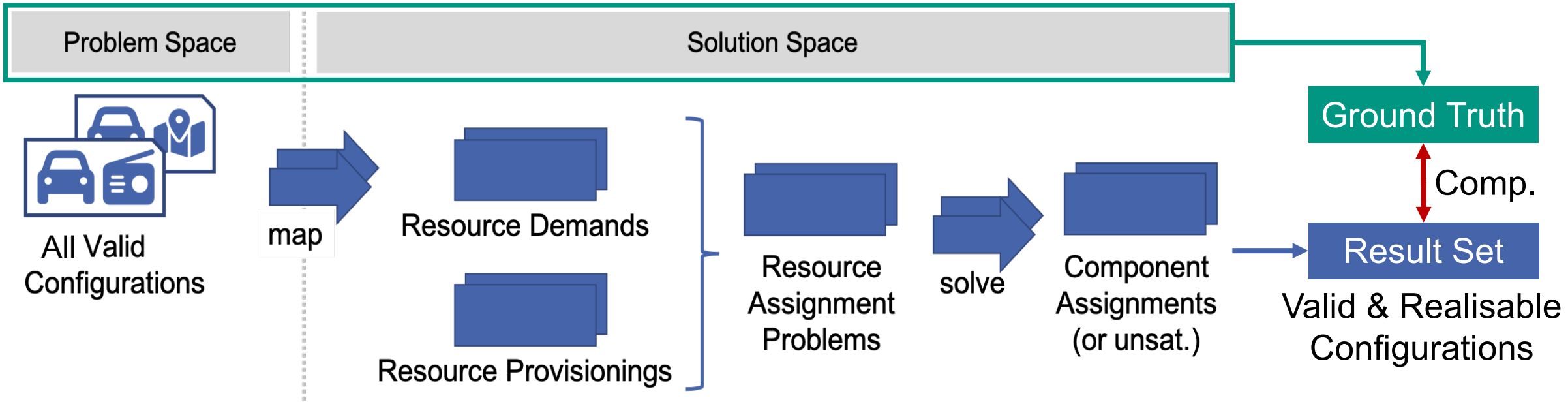
  - “Body Comfort System” (BCS) case study: known case study in research context [5,6]



- **Design**: Selecting a single certain feature yields unsatisfiable resource demands ⇒ configuration not realisable (Ground Truth)

Evaluation Procedure

- Computation of result sets: All configurations both valid and realisable
- Comparison between computed and expected result sets (Ground Truth):
  Equality ⇔ both contain the exactly same elements



Evaluation Results

Case Study		Case Study 1	BCS Case Study
no. Valid Configurations		20	7512
no. Realisable Configurations	Ground Truth	4	4200
	Realizability Analysis Method	4	4200
	Equality	**True**	**True**

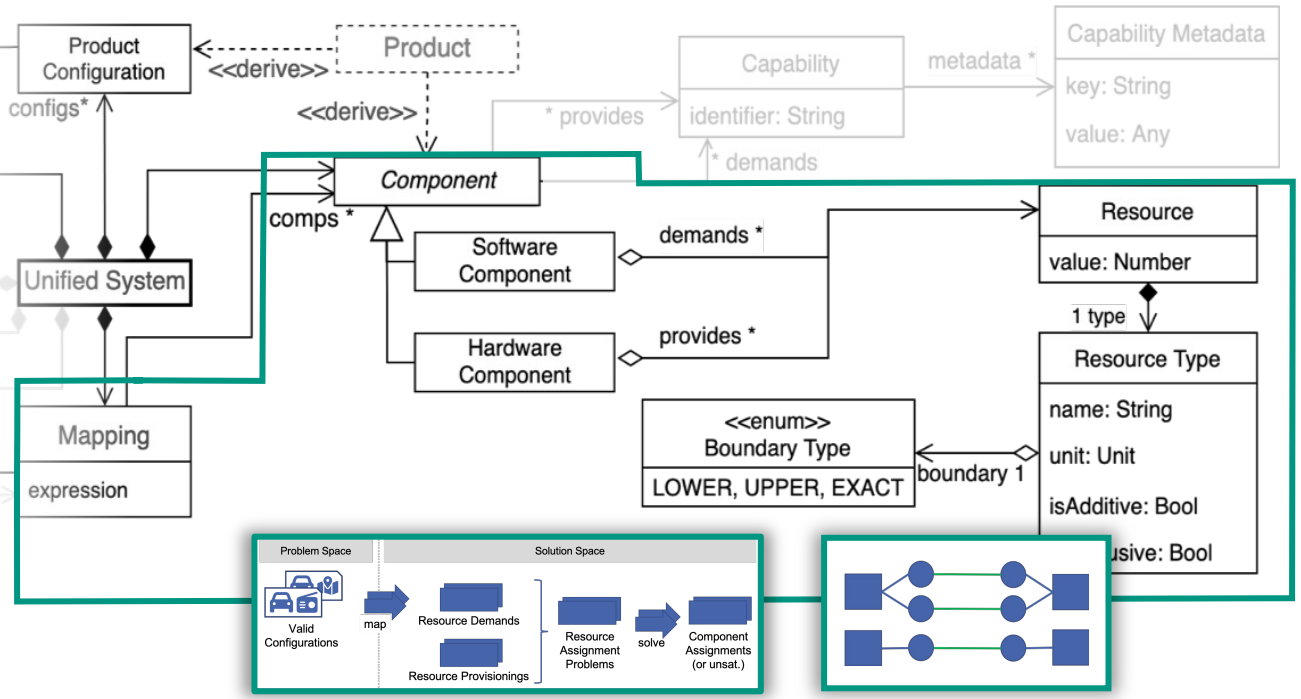
- **result sets** (realisable configurations)
  - for both case studies (Case Study 1 & BCS Case Study)
  - are **equal to the Ground Truth**
- Assuming the Realizability Analysis Method to be **evaluated positively**

Summary

- Research Questions:

RQ1: How can solution space artefacts be described in a meta-model?

RQ2: How can be checked, if a valid configuration is actually realizable?



- Outlook/ WIP
  - evaluating performance of analysis method (w.r.t. computationally expensive steps)
  - implementing/ re-modelling versioning of a product line

Literature

[1] Definition Cyber-Physical Systems: https://www.nsf.gov/pubs/2010/nsf10515/nsf10515.htm (accessed: 10.05.2023)

[2] Image Source Infotainment Systems: https://ppacaraudio.com.au/wp-content/uploads/2022/06/before-after-Kenwood-DDX9020DABS-for-Nissan-Navara-NP300-DX-RX-2015-2020-Stereo-Upgrade-.png.webp (accessed: 10.05.2023)

[3] Krzysztof Czarnecki and Ulrich Eisenecker. 2000. Generative Programming: Methods, Tools, and Applications. ACM/Addison-Wesley.

[4] West, Douglas Brent (1999), Introduction to Graph Theory (2nd ed.), Prentice Hall, Chapter 3, ISBN 0-13-014400-2

[5] Lity, Sascha, et al. "Delta-oriented Software Product Line Test Models - The Body Comfort System Case Study." Technical report, TU Braunschweig (2013).

[6] Case Study “Body Comfort System” (BCS) GitHub Repository: https://github.com/TUBS-ISF/BCS-Case-Study-Full (accessed: 09.05.2023)

[7] Ananieva, Sofia, et al. “A conceptual model for unifying variability in space and time: Rationale, validation, and illustrative applications“. In: Empirical Software Engineering 27.5 (09.2022), S. 101. issn: 1382-3256, 1573-7616. doi: 10.1007/s10664-021-10097-z.

[8] Jan Willem Wittler, Thomas Kühn und Ralf Reussner. „Towards an Integrated Approach for Managing the Variability and Evolution of Both Software and Hardware Components“. In: Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume B. Graz Austria: ACM, Sep. 2022, S. 94–98. doi: 10.1145/3503229.3547059.

[9] Z3 Theorem Prover: Bjørner, Nikolaj, et al. “Programming Z3”. Jan. 2021. url: https://theory.stanford.edu/~nikolaj/programmingz3.html (accessed 19.01.2023).

Appendix

Resource Types in the UCM

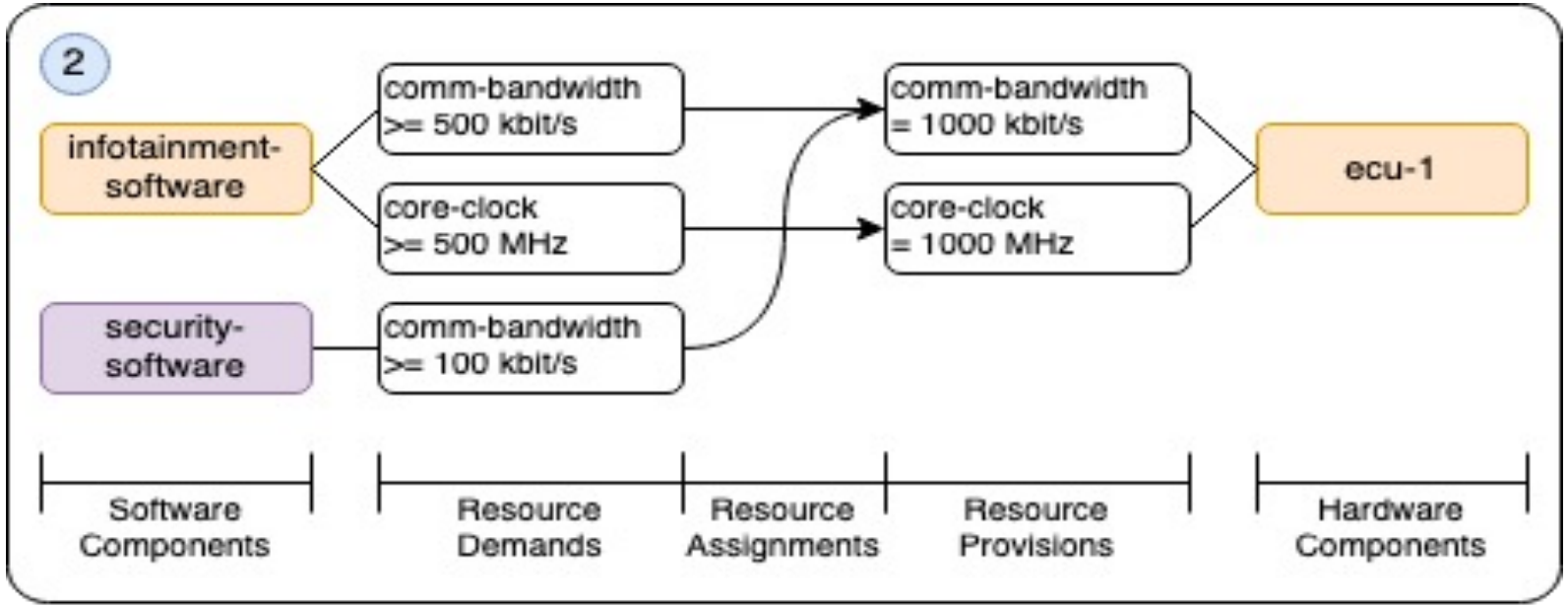
- The resource type defines **3 central properties** of a resource:
  1. $isAdditive$ $(\in \{\text{TRUE}, \text{FALSE}\})$
     If multiple resource demands or resource provisionings sum up
  2. $isExclusive$ $(\in \{\text{TRUE}, \text{FALSE}\})$
     If a resource is provided only for a single demand
  3. $boundaryType$ $(\in \{\text{LOWER}, UPPER, EXACT\})$
     Comparison operator between resource demands and provisionings

- **Example**: Random Access Memory (RAM)
  can be described as additive, non-exclusive and with a lower boundary:

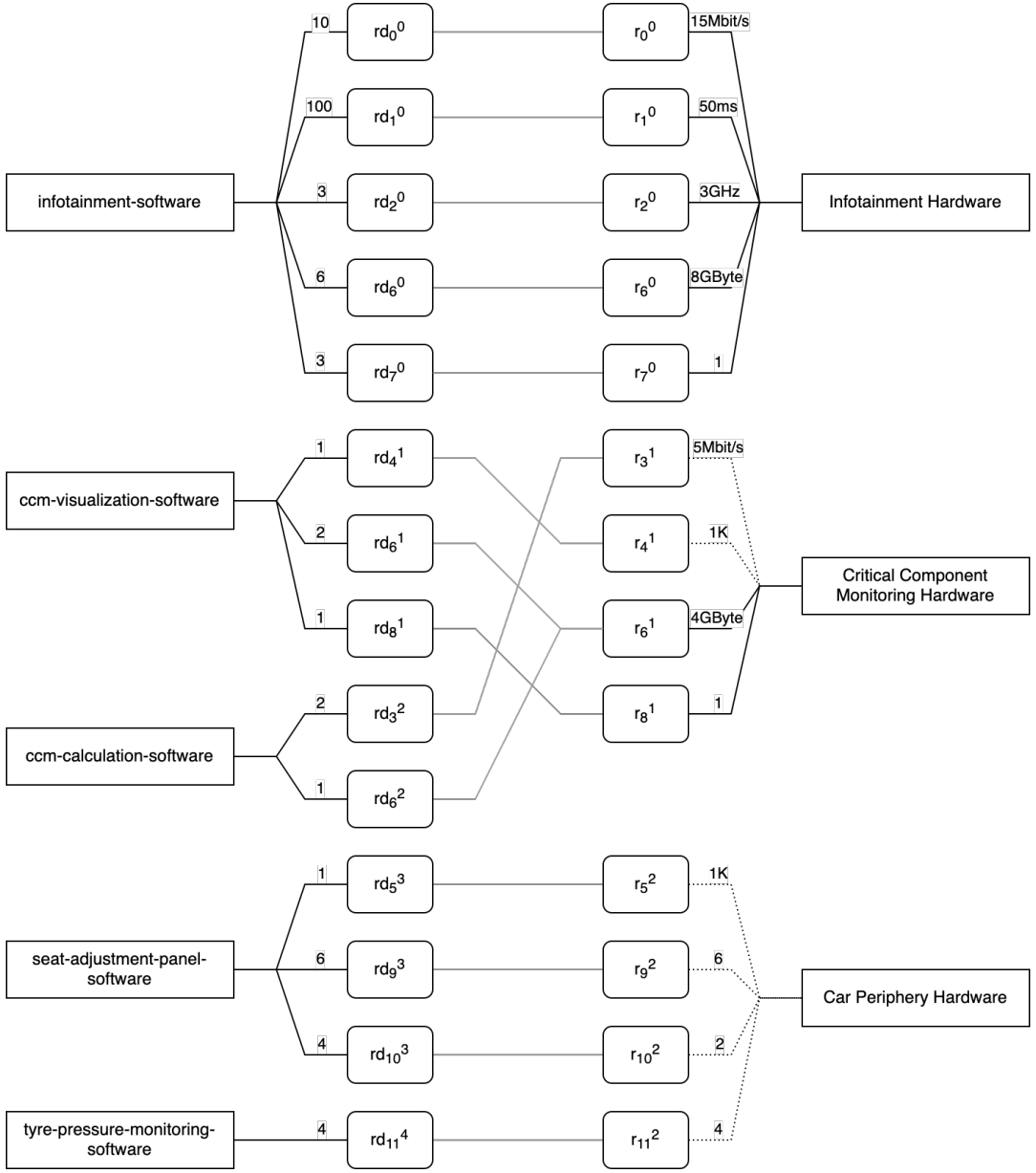
  2 software components demand at least 4 GB of RAM respectively
  ⇒ in total, at least 8 GB of RAM are demanded

Component Assignment

- Software components are assigned to hardware components by assigning all their resource demands to suitable resource provisionings
- Formulate and solve assignment problem

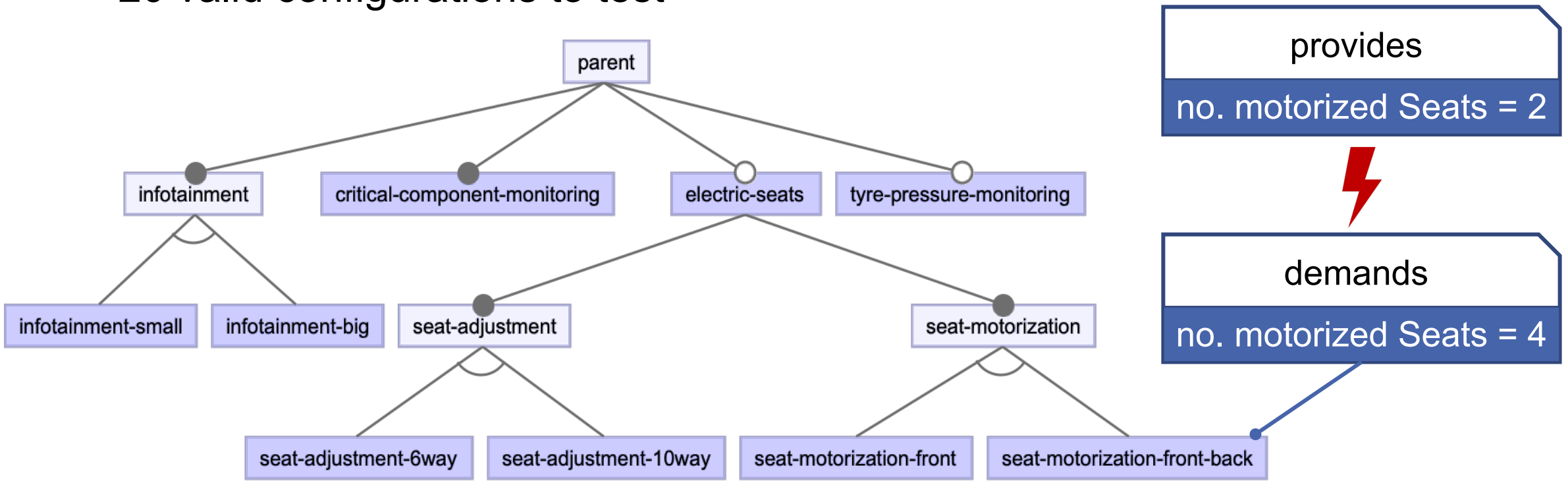


Case Study 1 Assignment Ex.



Case Study 1

- “Toy Example” for **Verification**
  - Covers all possible resource types ⇒ covers all developed constraints
  - 20 valid configurations to test



“Body Comfort System“ (BCS) Case Study

- Known case study in research context [5,6]
  - Feature model part of a real world example (28 features)
  - Extended by solution space artefacts
  - 7512 valid configurations to test