Towards a Family-based Analysis of Applicability Conditions in Architectural Delta Models

Arne Haber[1], Thomas Kutz[1], Holger Rendel[1],
Bernhard Rumpe[1], Ina Schaefer[2]

VARY: VARiability for You, Wellington, New Zealand, Oct 16, 2011

[1] Software Engineering
RWTH Aachen

http://www.se-rwth.de/

[2] Software Systems Engineering
TU Braunschweig

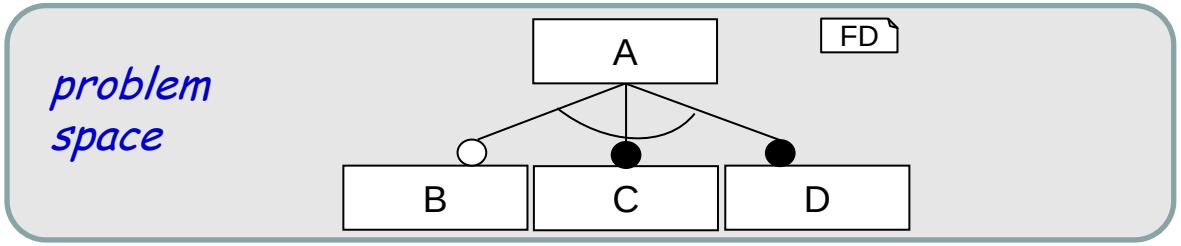
http://www.tu-bs.de/sse

Architectural Variability

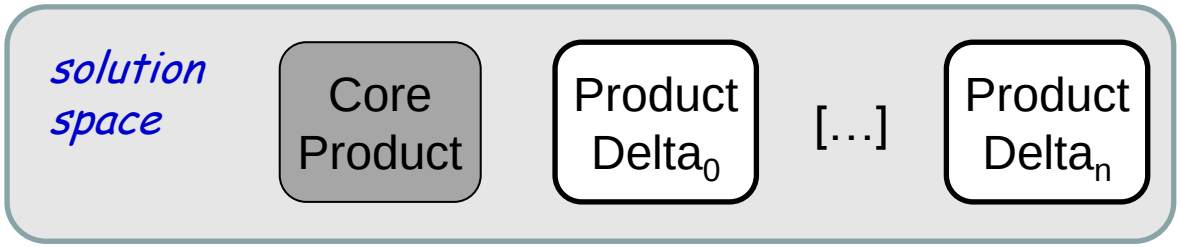
- Current complexity of software systems
- Developing variable software systems requires defining variability for each development phase
- Goal of our contribution:
  - Representation of variability on architectural level
- In this talk:
  - Modelling of delta oriented product lines
  - + family-based analysis of such a product line

Delta Modeling [Schaefer, VaMoS'10]

- Traditional Feature Modelling:

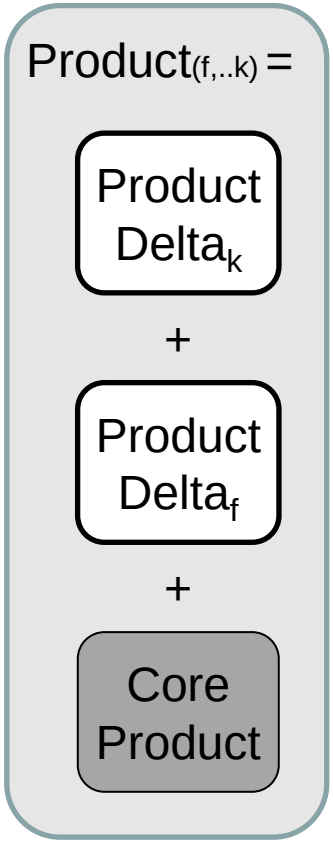


- Vs. Delta Modeling:



- Transformational approach
- Product for valid feature configuration

- Modification / Extension of Core Products
- Application order constraints over deltas
- Determine partial ordering for conflict resolution



MontiArc - Modeling Architectural Elements

- ADL for information-flow architectures with message-based asynchronous communication

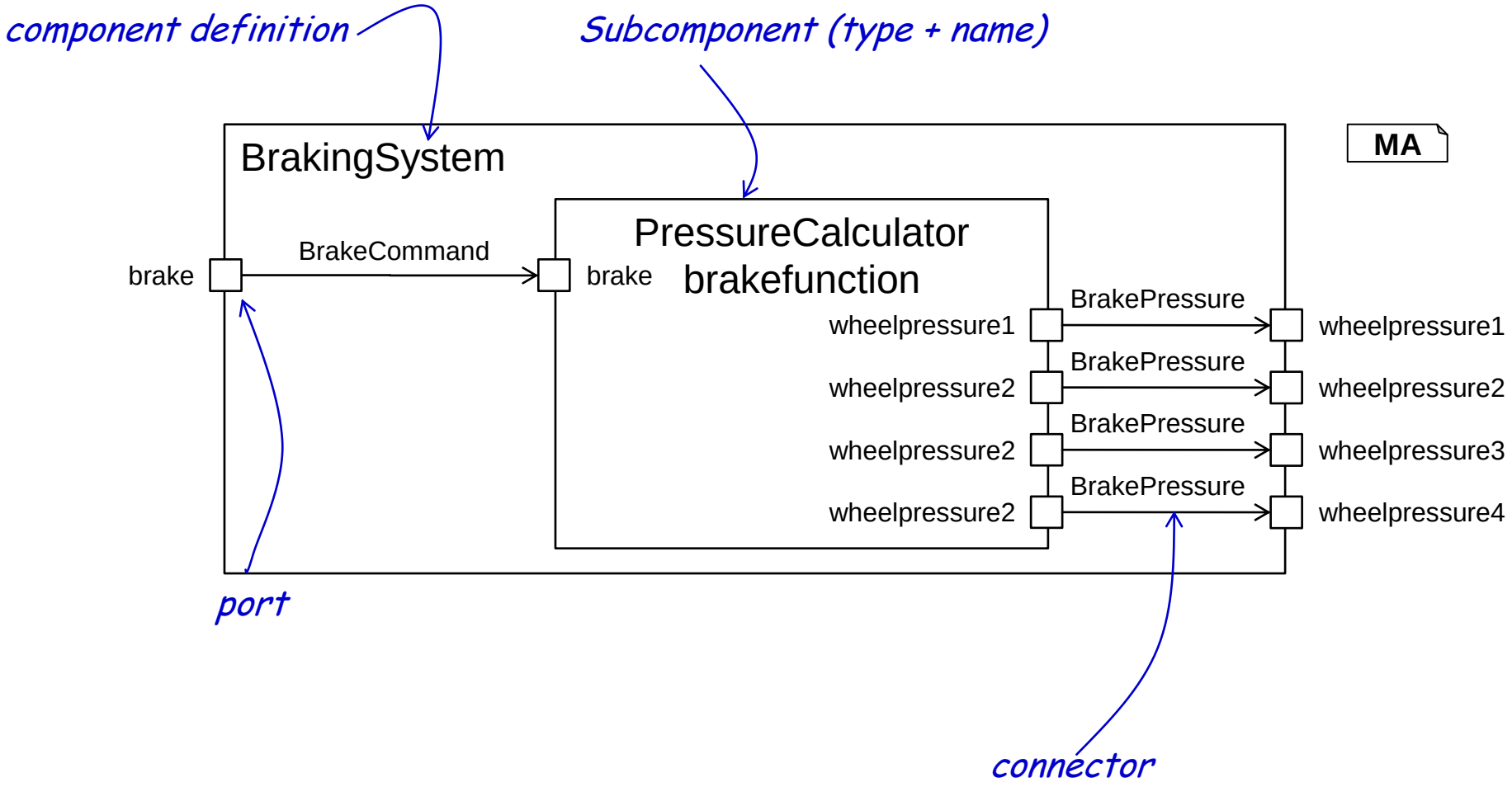
- MontiArc basic concepts:
  - Components
  - Ports – component interfaces
  - Connections – communication links

  - Subcomponents – references to component definitions
  - Inner components – locally defined subcomponents

- Comfort functionality:
  - Autoconnect (for interfaces with same types/names)
  - Implicit naming

- Event-based timed simulation (several variants)

Example: Core Architecture

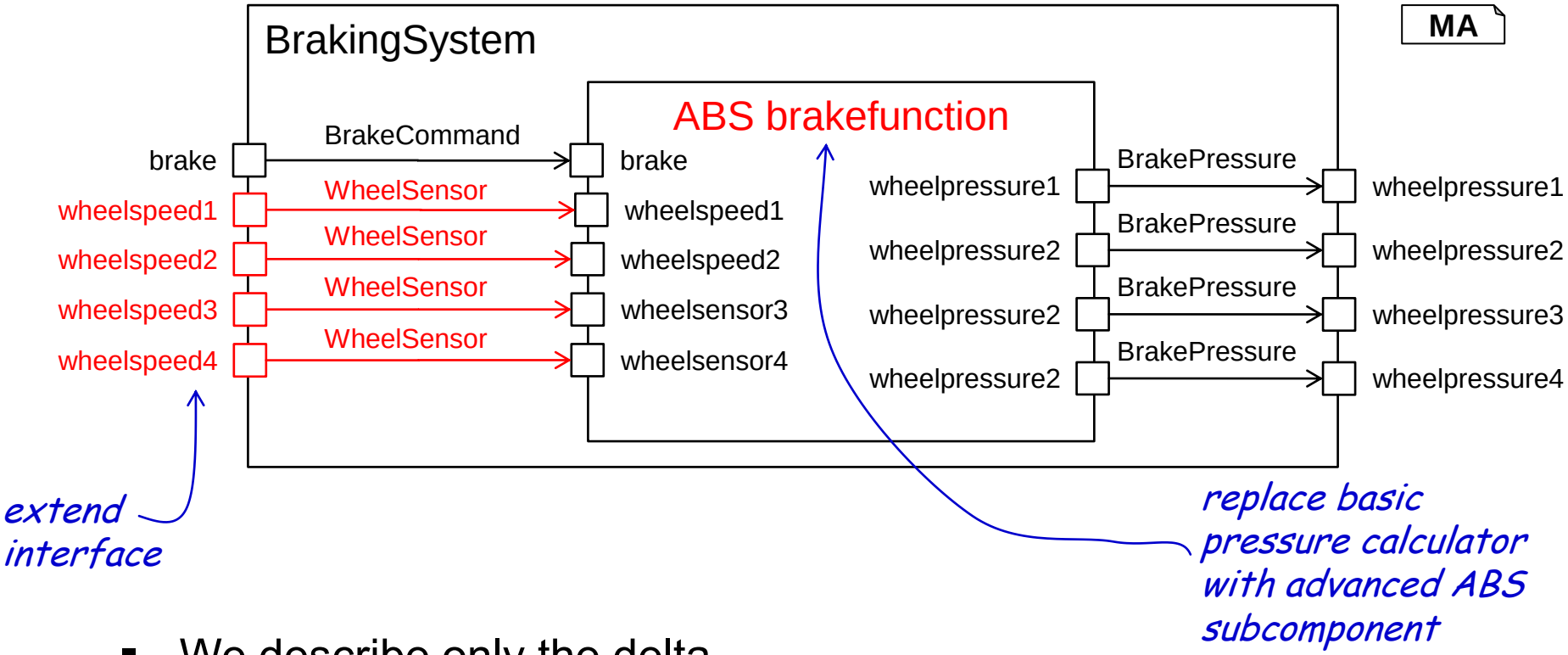


Δ-MontiArc - Overview

- Extension of MontiArc with delta modeling concepts
- Core architecture defined by MontiArc models
- Delta models defined with Δ-MontiArc
  - Operations for modification of core architecture: add, remove, replace, rename, modify, …
  - Application order constraints for conflict resolution
    - determine partial order
    - logical expression over delta models
- Product generation
  - Product configuration = subset of delta models to be applied
  - Application order compliant to order constraints
  - Sequently modify ASTs and symbol table entries for each applied delta

Example: Delta AntiLockBrakingSystem

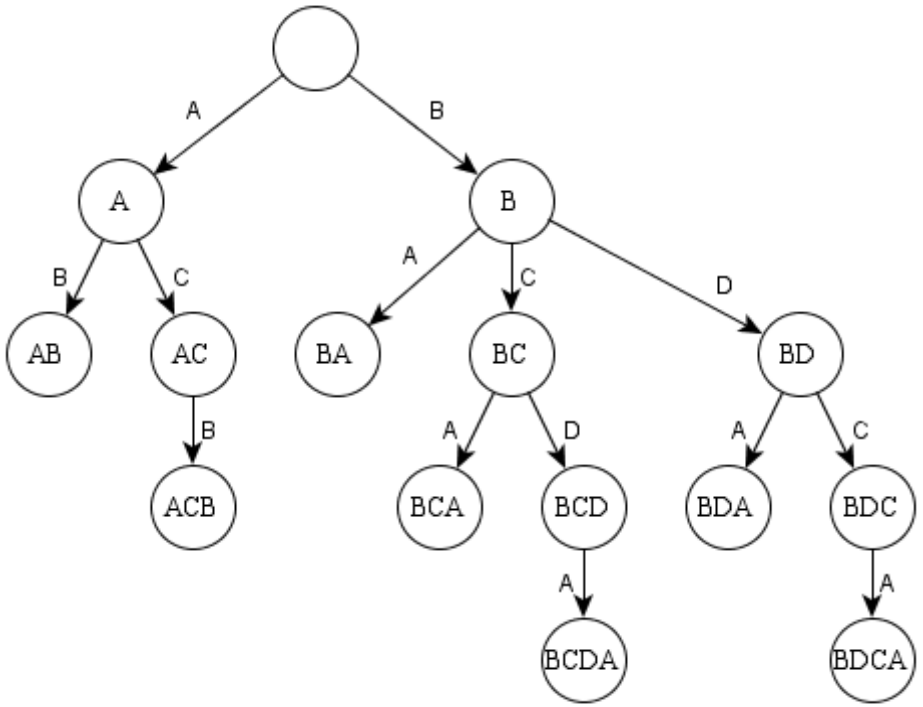
delta AntiLockBrakingSystem



- We describe only the delta …
  - although it is applied here
- we use a textual description for deltas

Verifying Application Order Constrains

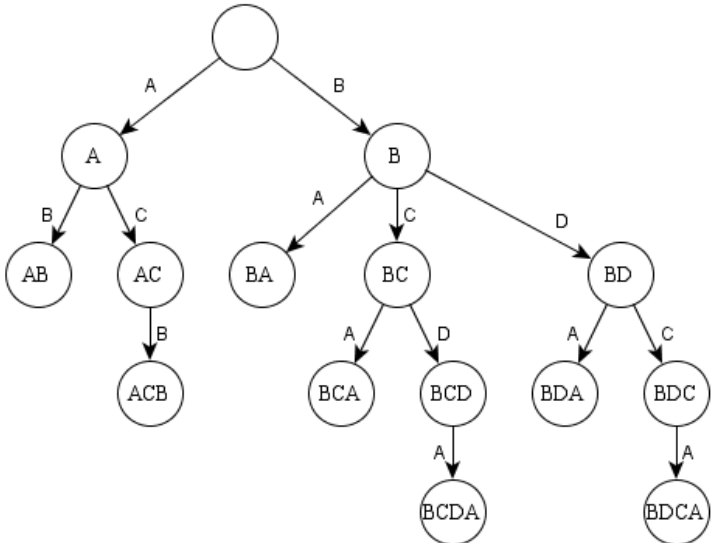
- Analysis of applicability conditions, e.g.
  - elements to remove/replace/rename must exist
  - connections created only if not existing; typing correct
- Requires analyzing all product variants
- Family-based analysis on family application order tree (FAOT)



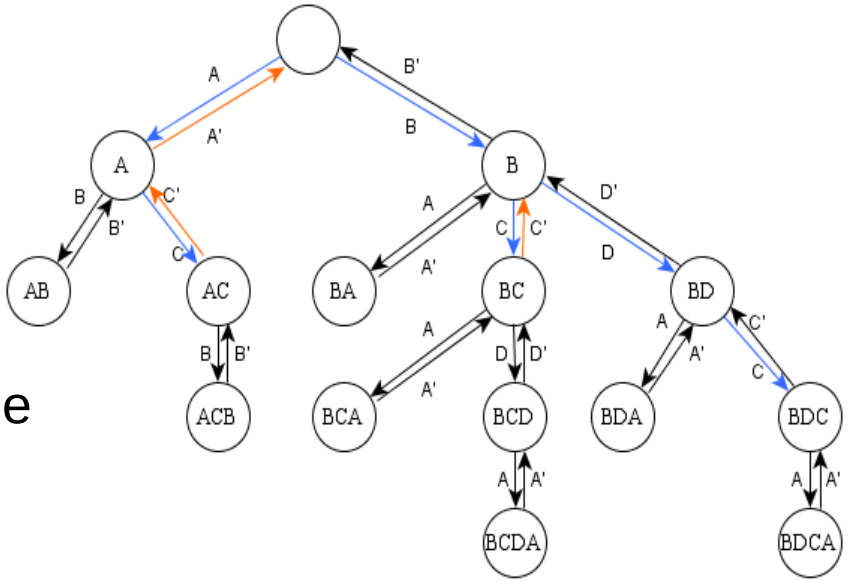

```
Given: Deltas A,B,C,D

A (no constraint)
B after !D
C after A ^ B
D after (B || C) && !A
```

Verifying Application Order Constrains

- Different Approaches:
  - Simple Approach: Generate all possible delta application orders
  - Needs to store all intermediate products at decision nodes



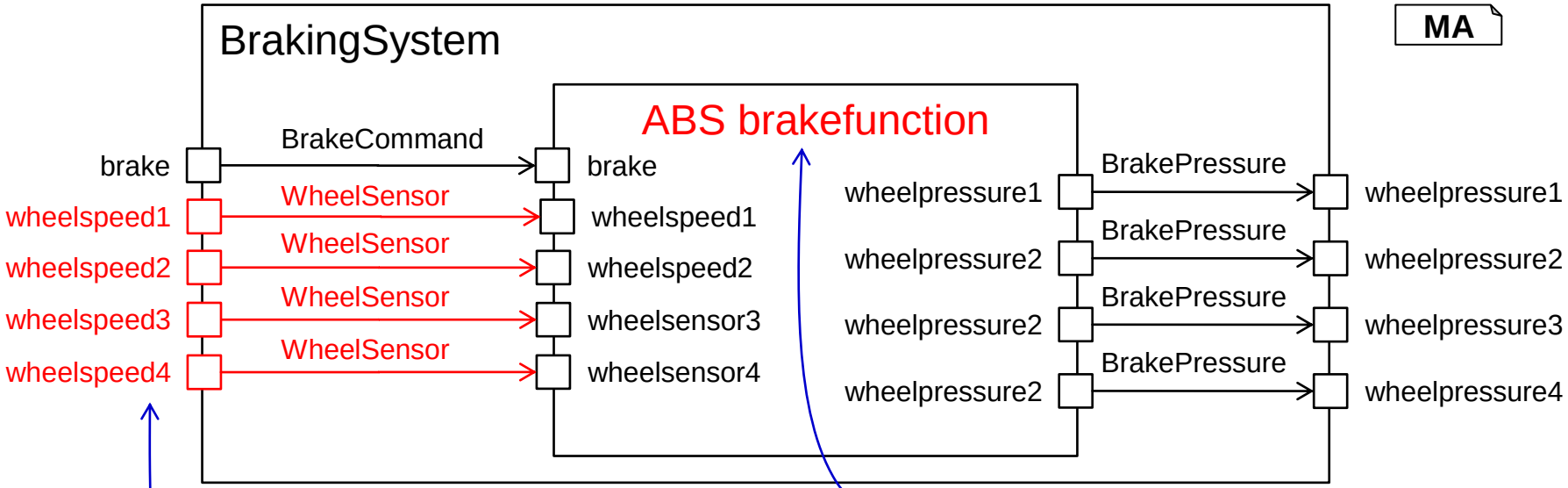
- We improve this by using inverse deltas to backtrack in FAOT
- Inverse deltas D‘ undo delta operations
  - apply(apply(P,D),D‘(P)) = P for Product P,
- considerable speed up gained,
- but further enhancements possible



Conclusion

- Δ-MontiArc
  - powerful mechanism for variability in architectures
- Family application order tree (FAOT)
  analysis extends delta oriented architectural PLs
- Introduced inverse deltas
  - allow efficiency improvement in FAOT
  - may also be used for evolution/refactoring of delta oriented PLs
- More work to come …

Example: Delta AntiLockBrakingSystem



extend interface

replace basic pressure calculator with advanced ABS subcomponent

```
delta AntiLockBrakingSystem {
  modify component BrakingSystem {
    add port in WheelSensor wheelspeed1,
               in WheelSensor wheelspeed2,
               in WheelSensor wheelspeed3,
               in WheelSensor wheelspeed4;
    replace component brakefunction with component
      ABS brakefunction;
  }
}
```