



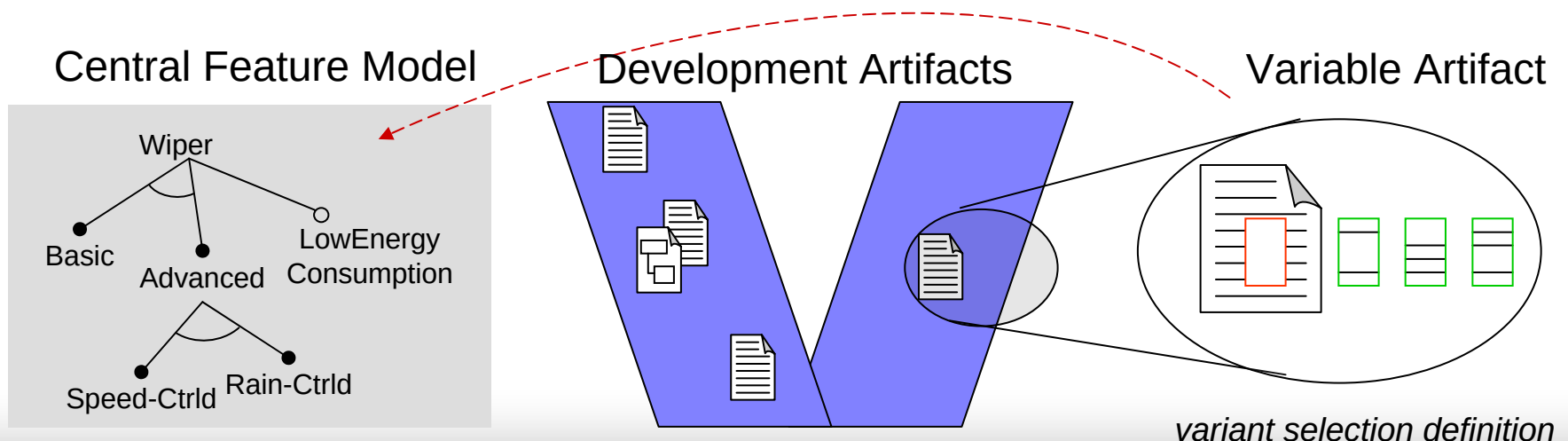
Applying Aspect-oriented Techniques in Automotive Software Product Line Engineering

Katharina Mehner, Technical University of Berlin
Marc-Oliver Reiser, Technical University of Berlin
Matthias Weber, DaimlerChrysler AG

IEEE Workshop on Automotive Requirements Engineering
11 Sep 2006

Motivation

- Challenge in automotive software development is diversity
 - Hardware and software diversity
 - Advantage in competition for manufacturers and suppliers
- Software product lines help conquer variability
 - Derive instances from variable product line infrastructure
 - Definition and management of variability
 - Insufficient alignment of features with software artifacts
 - Lack of traceability induces difficulties for maintenance and evolution

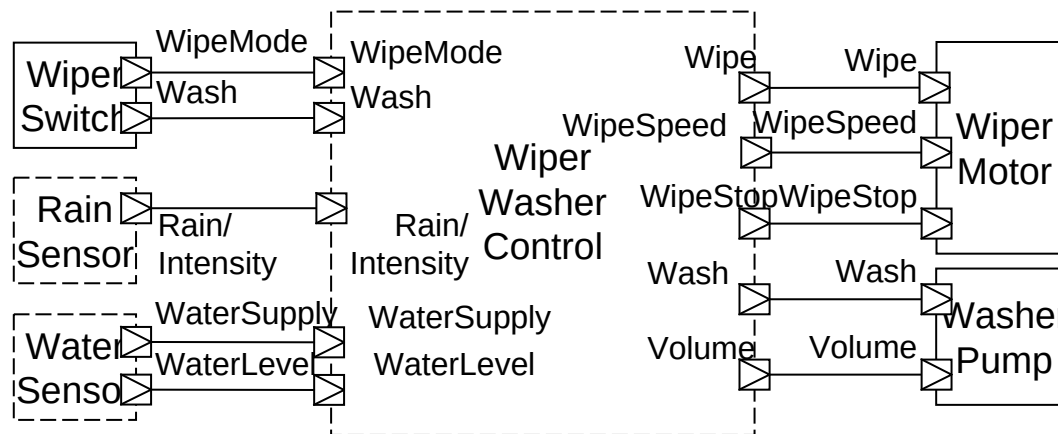


Outline

- Variability modeling in product lines
- Shortcomings of classical variability definition
- Introduction to aspect-orientation
- Modeling variability with aspects
- Comparison and Outlook

Variability Definition in EAST ADL

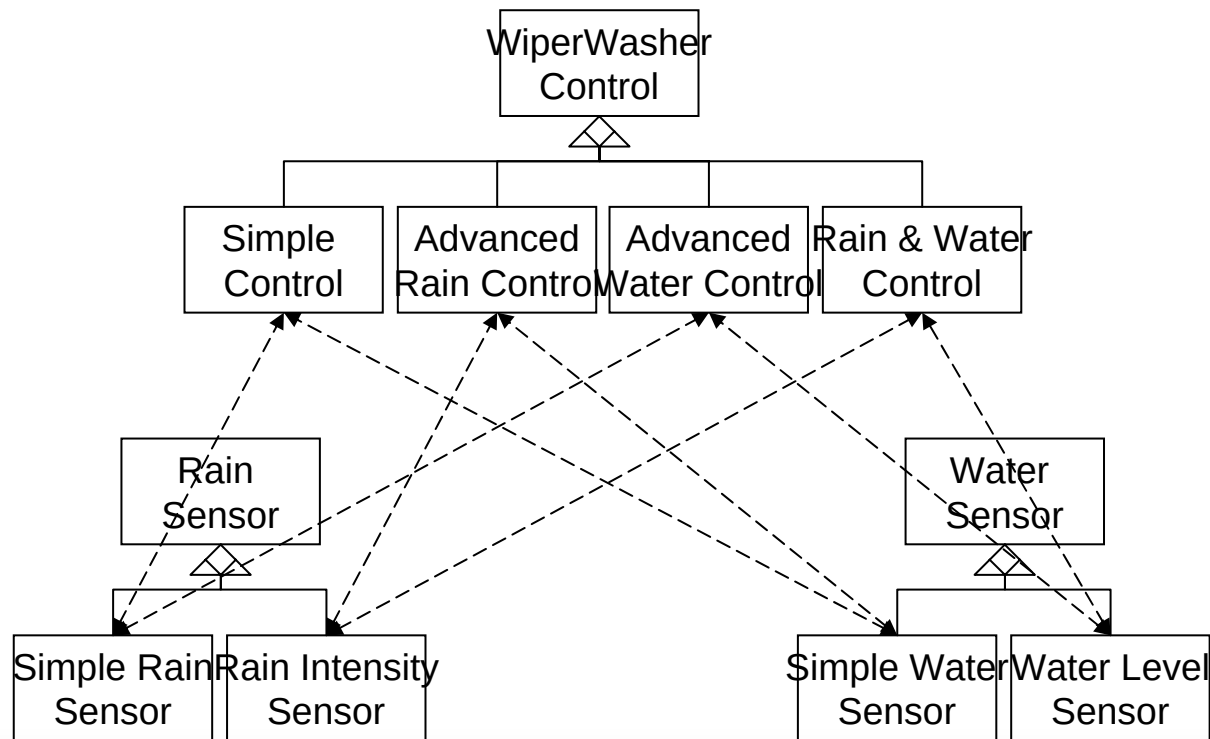
- Modeling the Functional Analysis Architecture
- Hierarchical component notation based on UML2
- Behavior and communication via ports
- Variation points and variants



- *Rain simply- or intensity-controlled: one signal*
- *Water supply- and additionally level-controlled: two signals*
- *Rain-controlled wiping is prioritized over manual wiping*
- *Wash and wipe is prioritized over rain-controlled wiping*

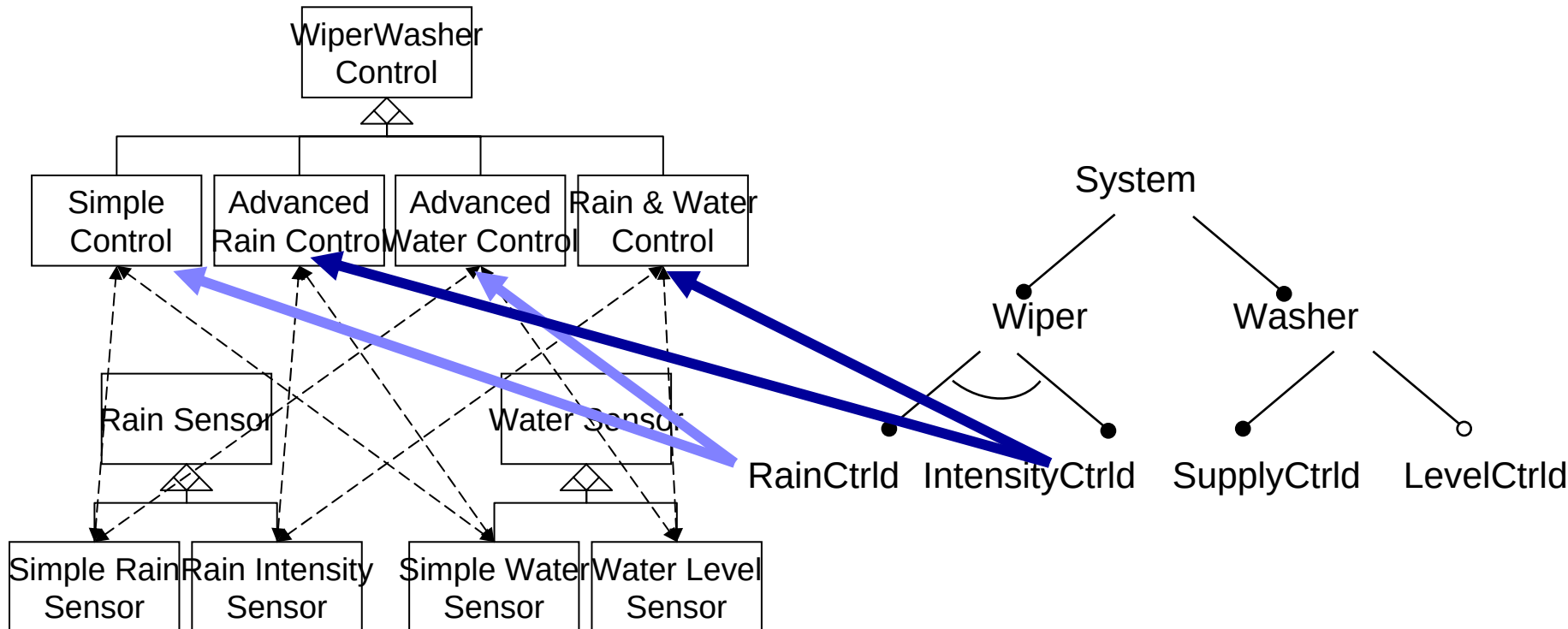
Variant Diagrams

- Predefined variation points
- Trade-off
 - Variability at coarse grained or fine grained level
 - Variability at fine grained level
- Under certain circumstances explosion of the number of variants
- No support for collaborations



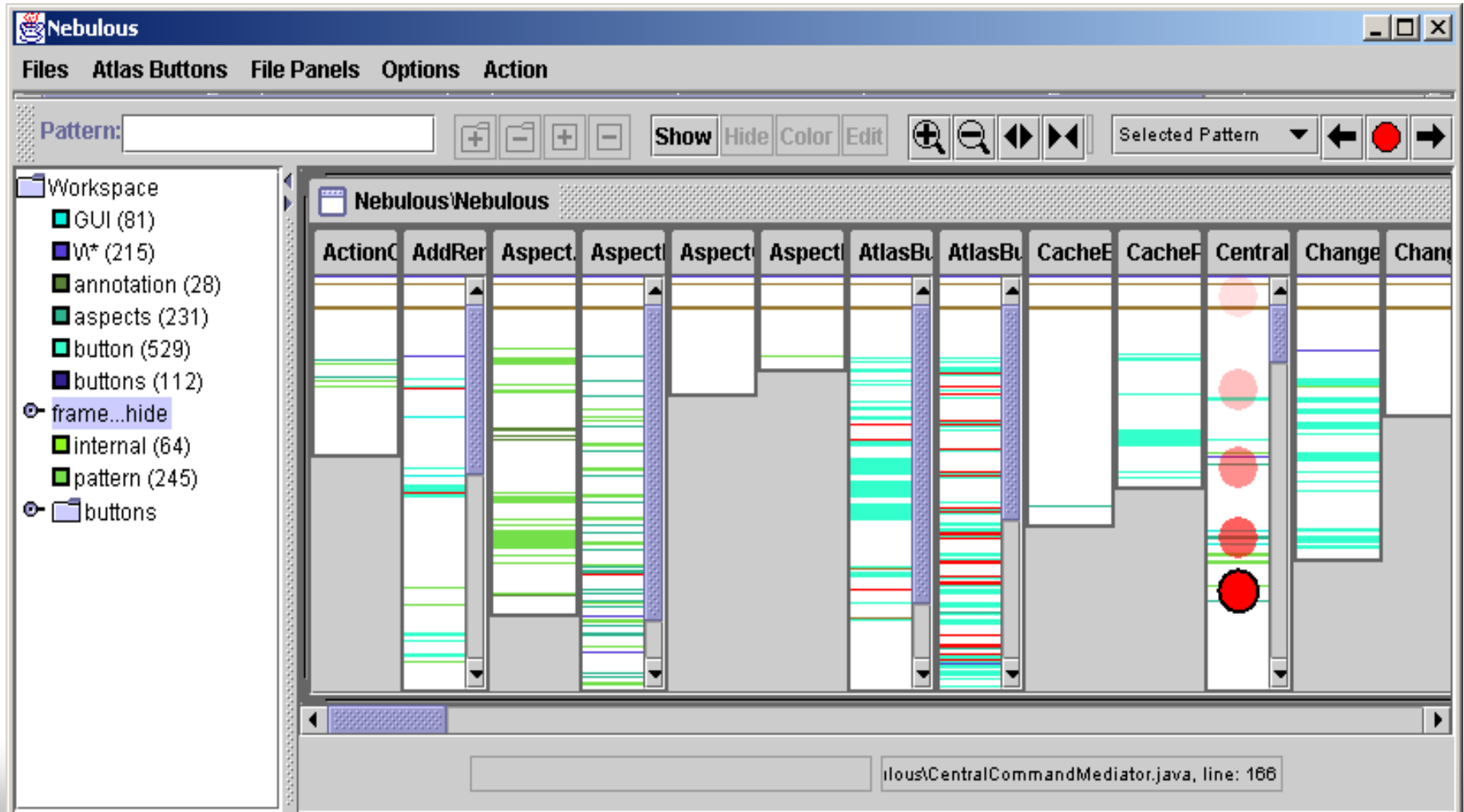
Crosscutting

- Single feature affects many variation points in many artifacts (Scattering)
- Different variations affect a variation point (Tangling)
- Tyranny of dominant decomposition
- Lack of traceability: Difficult to maintain and evolve



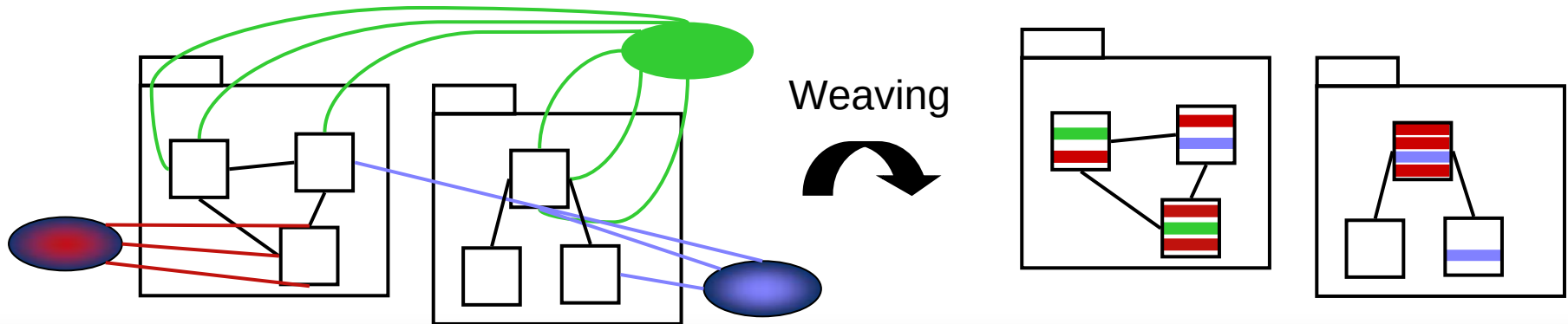
Aspect-Orientation: Identify Crosscutting

- Object-oriented decomposition
- Behavior composition through method calls



Aspect-Orientation: Separation and Integration

- New modules called aspects
- Interception of base behavior
 - Pointcuts
 - Advice
 - Non-invasive
 - Declarative
 - Pattern matching
- Dynamic crosscutting
- Alignment of concerns with code



Aspect-Oriented Programming (AspectJ)

```
public class Figure{
    int x,y;
    //...
    public void setXY(int x, int y) {...}
    public void setX(int x) {...}
    public void setY(int y) {...}
}

public aspect Logger {

    pointcut move(): call(void Figure.set*(..));

    after(): move() {
        System.out.println("A figure element moved.");
    }
}
```

Aspect

Pointcut

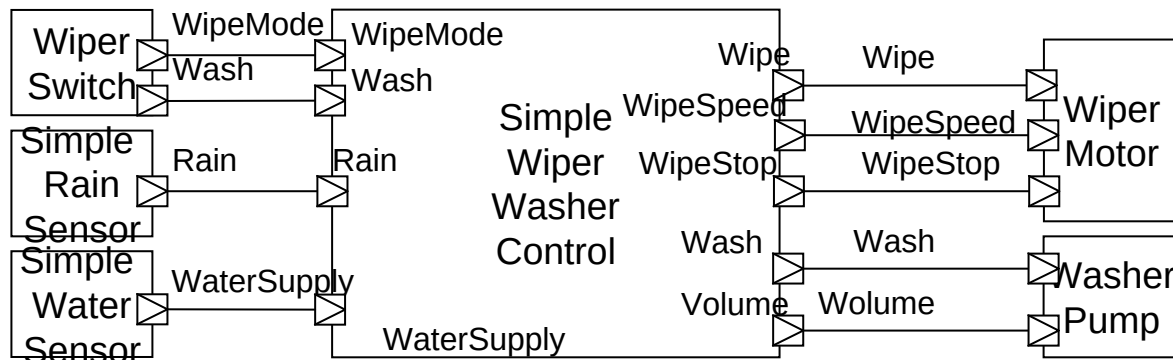
Advice

Our Approach

- Reduce model complexity
- Align features with artifacts
 - Support for collaboration
- Aspects for dimension of variability
 - Structural crosscutting
 - Pointcut for describing variation points
 - Advice contains variant

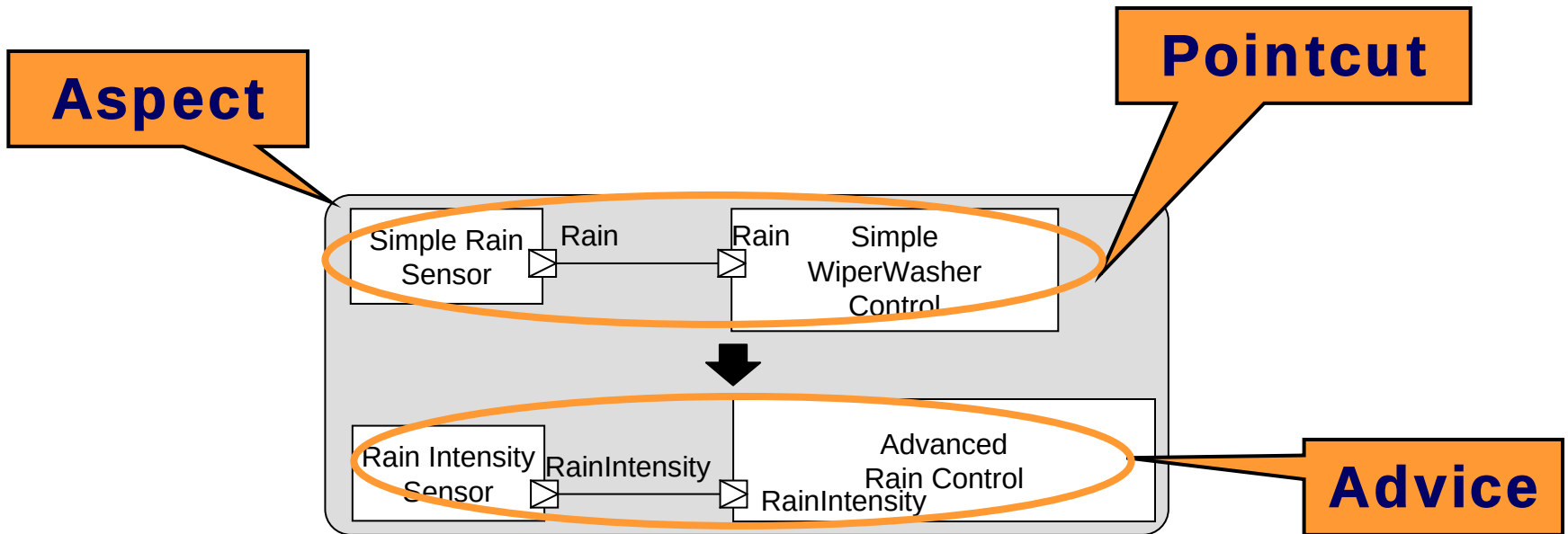
Role of Base

- Default diagram
 - Semantically complete
 - Simplest functionality
- Alternatively
 - Semantically incomplete
 - Functionality that can be most easily adapted
- Corresponds to “base” type in automotive product lines



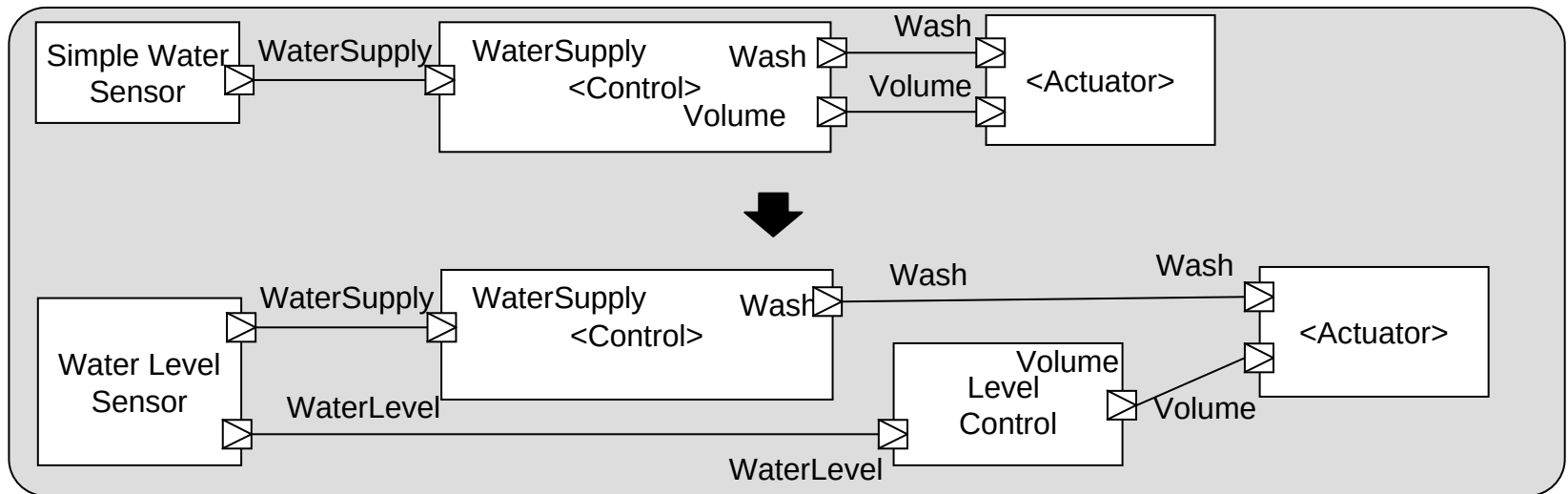
Aspect for Rain-Intensity Control

- Requires change of the control logic
- Transformation rule
- Concrete matching
- Replace a component

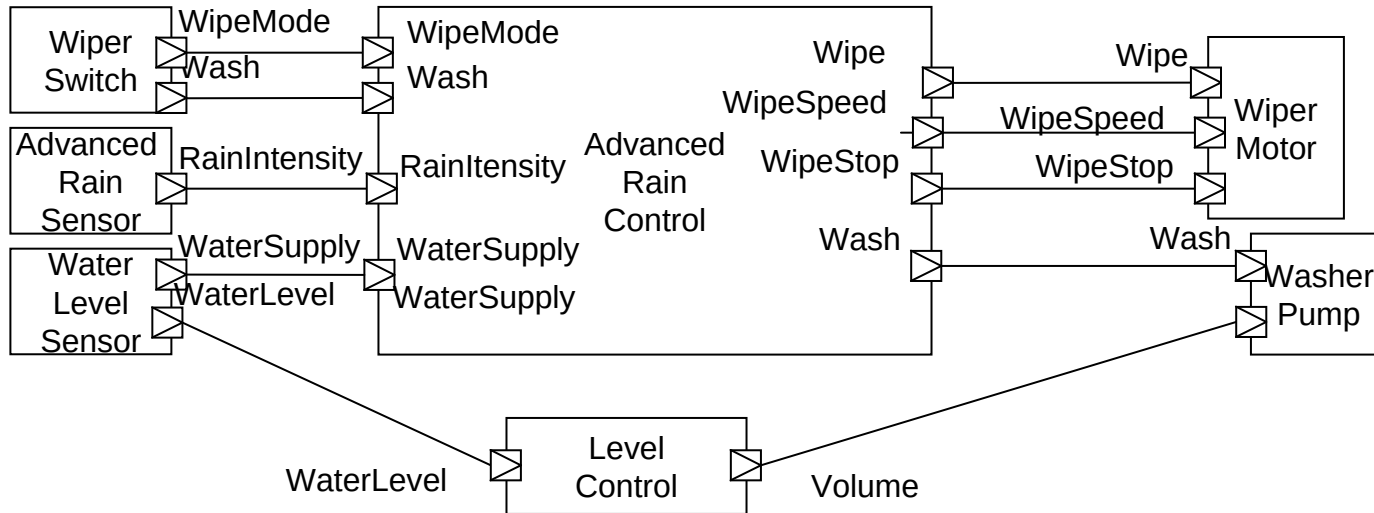


Aspect for Water-Level Control

- Independent of the control logic
- Re-use and adaptation of existing components
- Matching via structural properties

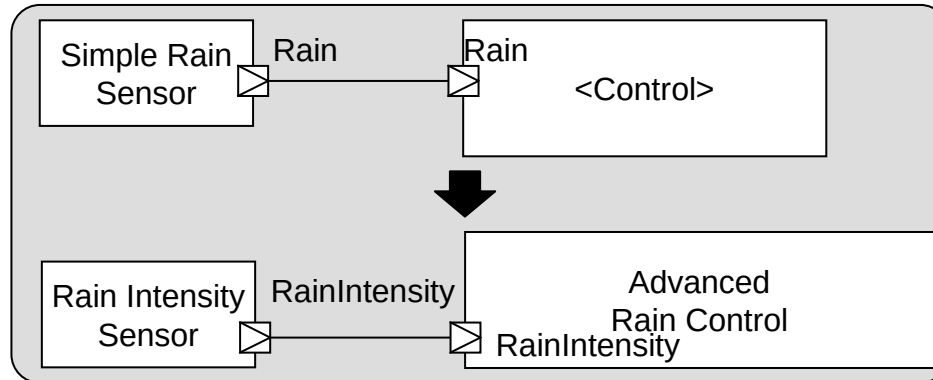


Transformation Result



Independent Aspect Composition

- New aspect for Rain-Intensity Control
- Apply aspects in any order
- No concrete matching
- Confluence



Conceivable Aspect Transformations

- Replacing
- Wrapping
 - Preprocessing
 - Postprocessing
 - Supplying input signals
 - Suppressing output signals
- Insertion
- Applicable to different hierarchy levels
- Abstract from nesting level

Comparison

- Example
 - Classic: 4 variants
 - AO: 1 default component, 2 aspects, 3rd aspect via composition
- Reduction of complexity
 - Not always the case in traditional approach
 - Not all aspects are truly orthogonal
- Correspondence feature with aspect
 - Better encapsulation of collaboration
- Explicit versus implicit variation points
- Supporting crosscutting

Outlook

- Further assessment of aspect potential needed
- Feature interaction traceable to component level
- Conflict management
 - Order of aspects
 - Dependencies between aspects
- Behavioral crosscutting
- Case study in a diploma thesis
- Formalisation using Graph Transformation
- Support with Graph Transformation Tool AGG
- Considerations for implementation