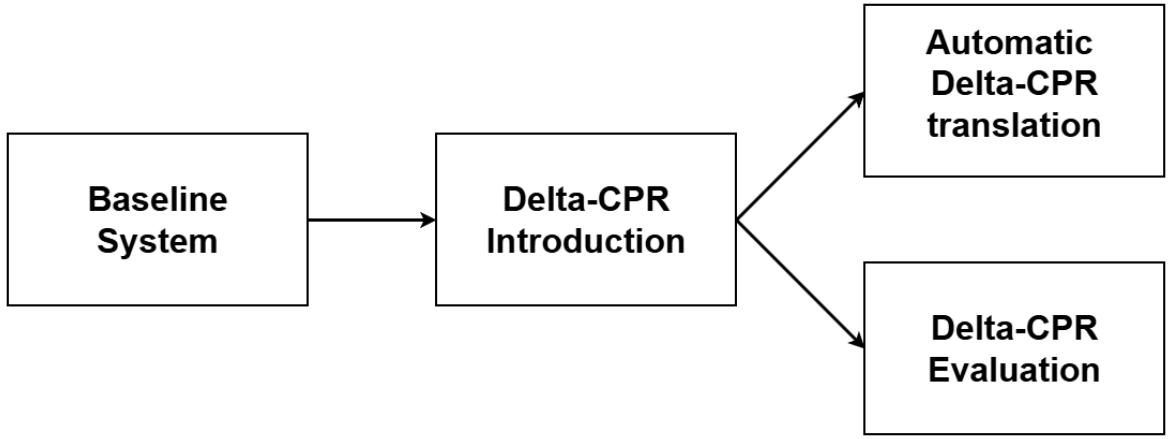
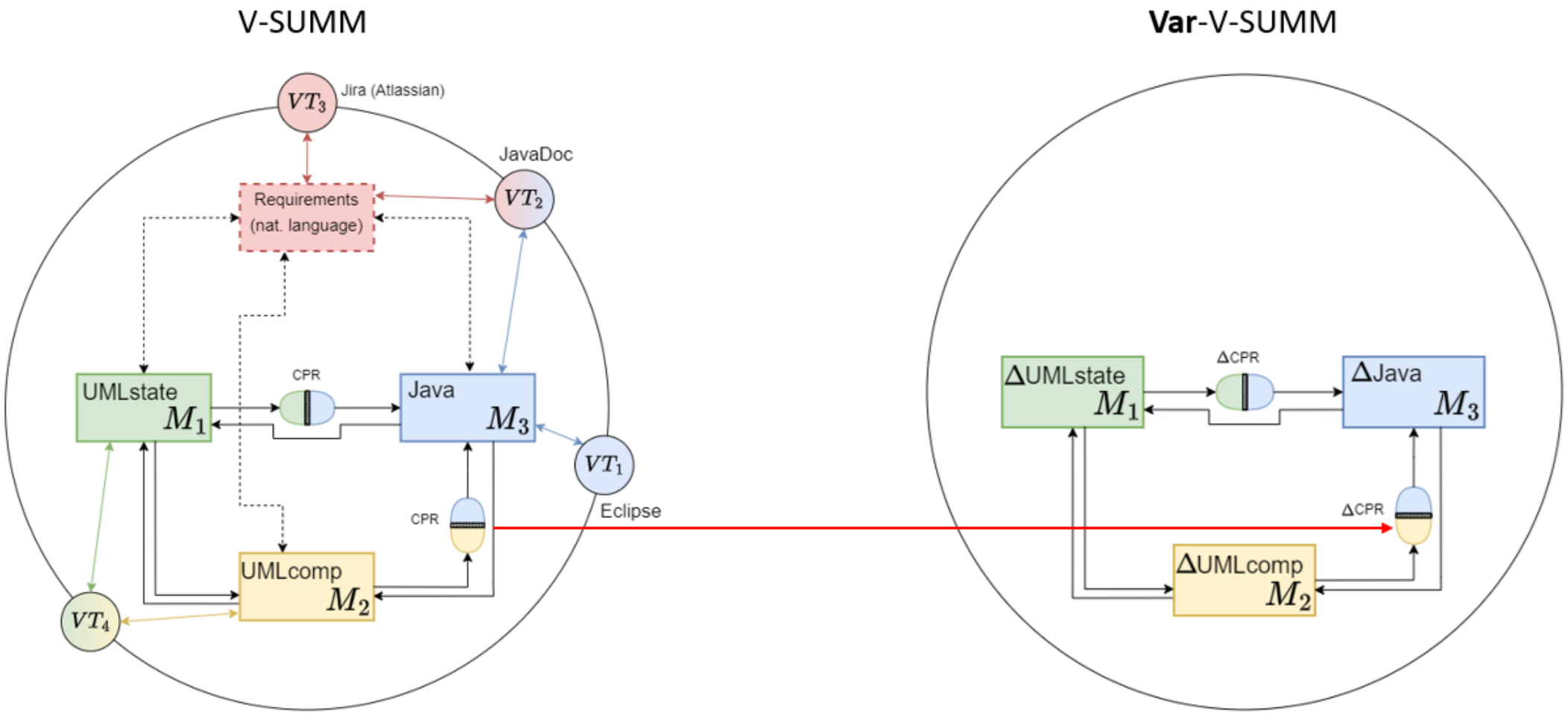
Cross-Domain Consistency Preservation for Delta-oriented Product Lines

Simon Bothe, TVA (KASTEL) KIT

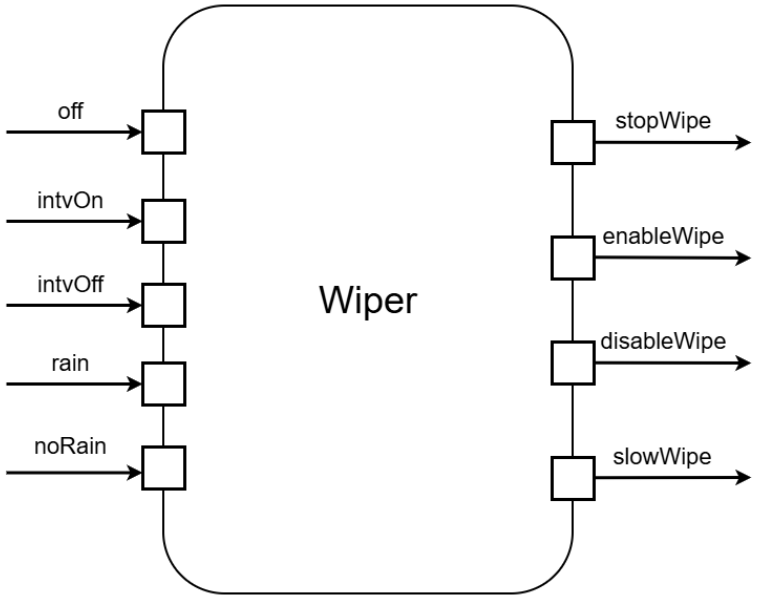


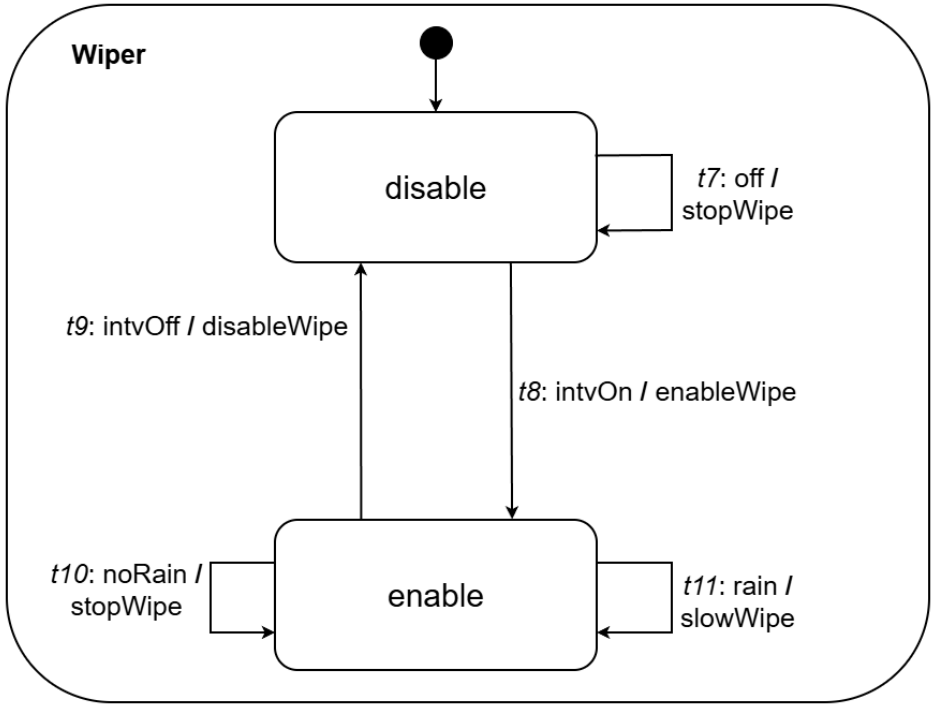
VSUMM and Var-VSUMM[4]



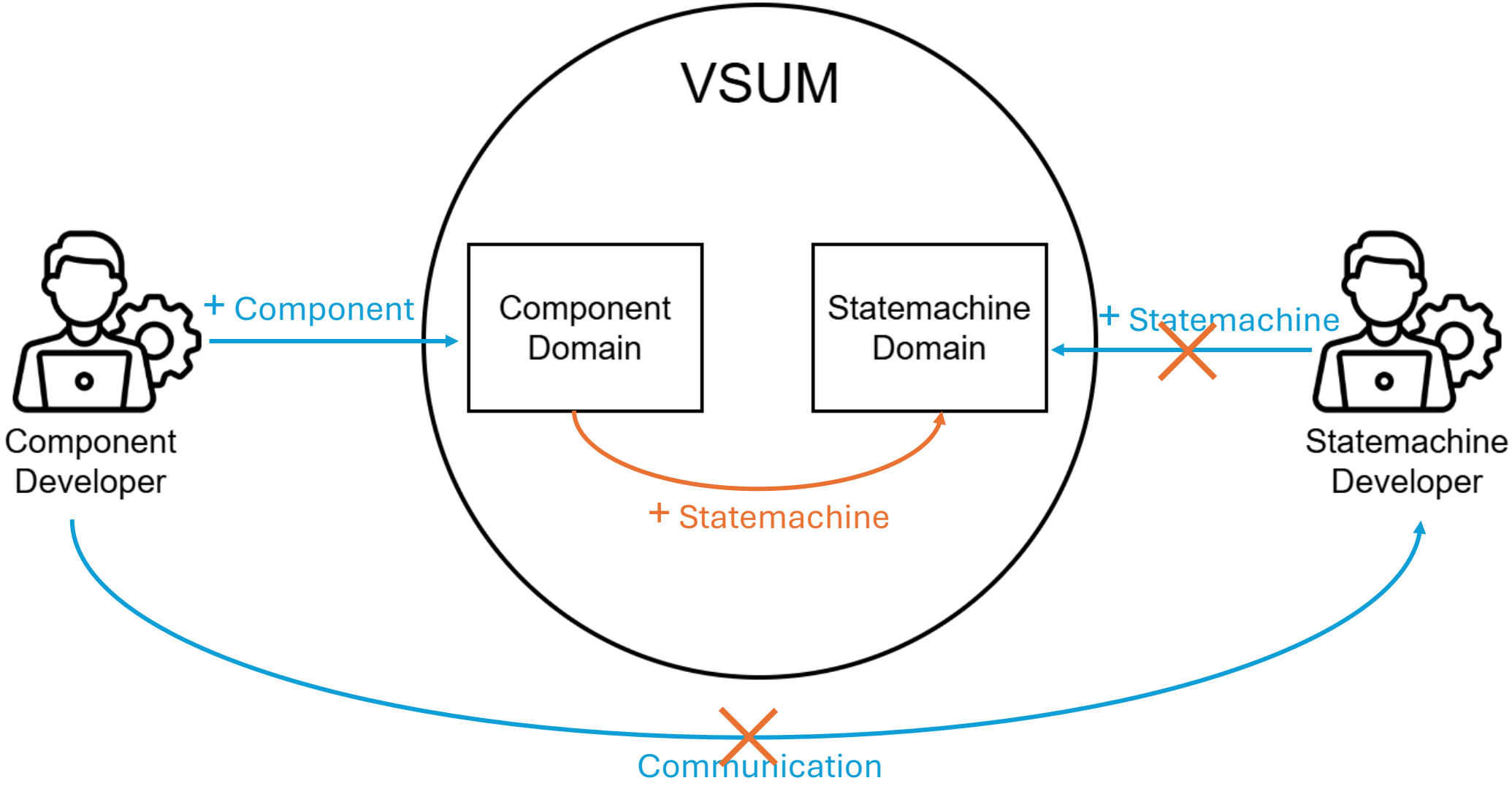
Multi-Domain example - The Body Comfort System[1]

- **Embedded software system** consisting of **Electronic Control Units (ECUs)**
- Two modeling domains in solution space: **Component Domain** and **Statemachine Domain**

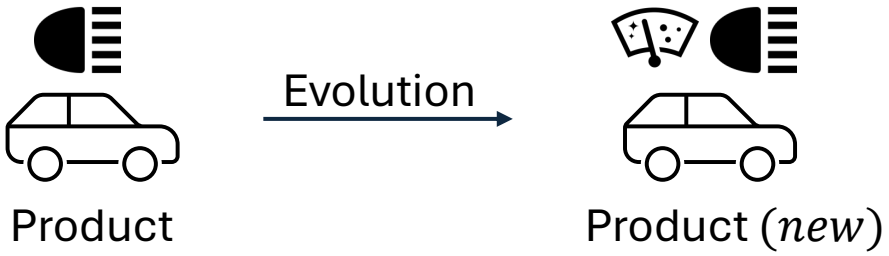




Why use Consistency Preservation?



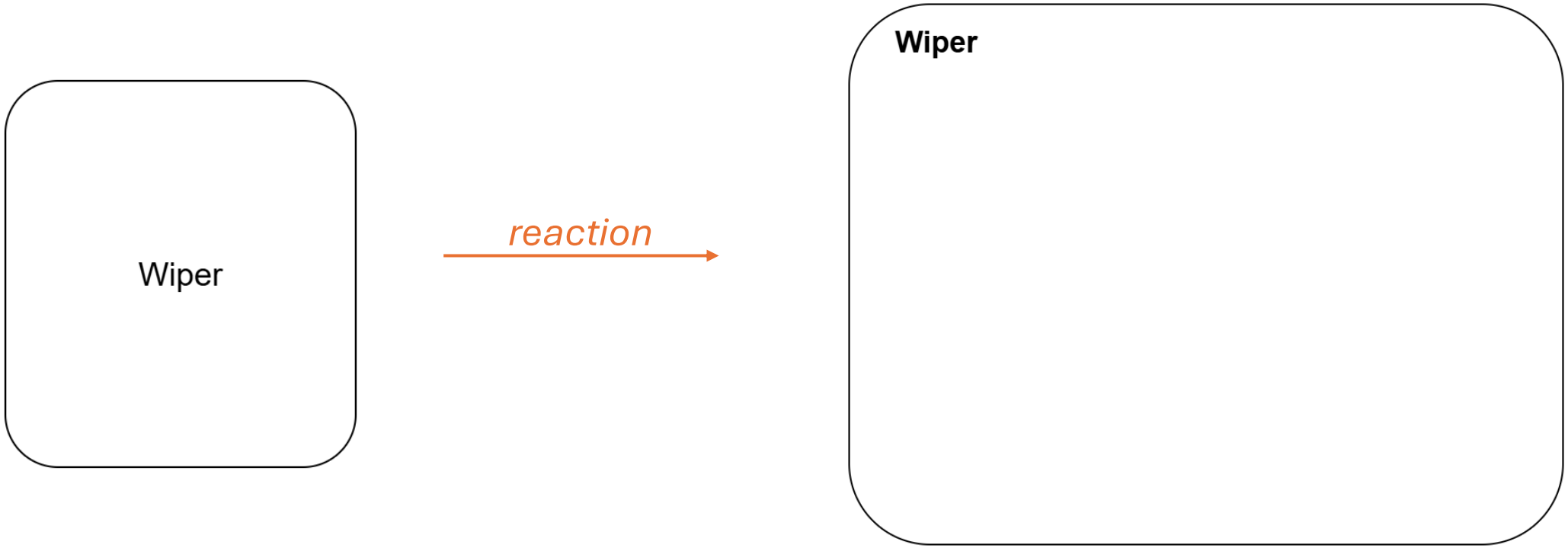
Consistency Preservation Rules (CPRs)



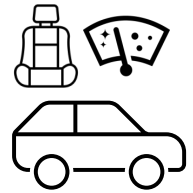
- Changes in one domain potentially require changes in another domain
- How to handle these dependencies **between domains**?
- **Vitruvius Approach**[3] using **Consistency Preservation Rules (CPRs)**
- CPRs trigger for changes in a *source domain* and react with consequential changes in a *target domain*
- Implemented by imperative rules: **Reactions**

Consistency Preservation Rules (CPRs) - Example

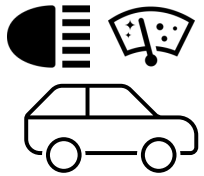
reaction: if component added into system {
    add new region into statemachine
}



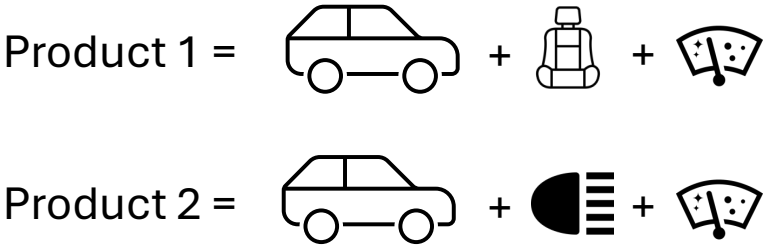
Software Product Lines - Delta-oriented Modeling [2]



Product 1



Product 2

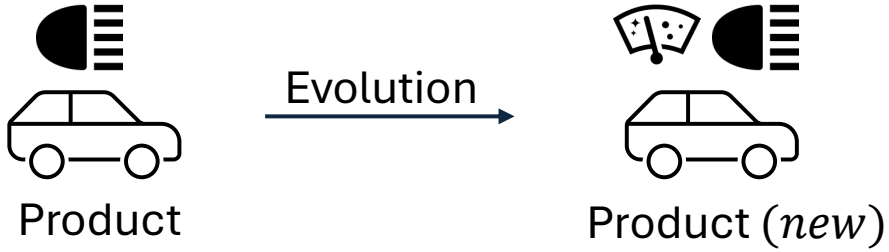


- Construct product by **gradually applying** change operations to a common core **(delta operations)**
- **Encapsulate** delta operations in **deltas**
- Assign feature combination to deltas (**application condition**)
- Partially order deltas (**application order**)

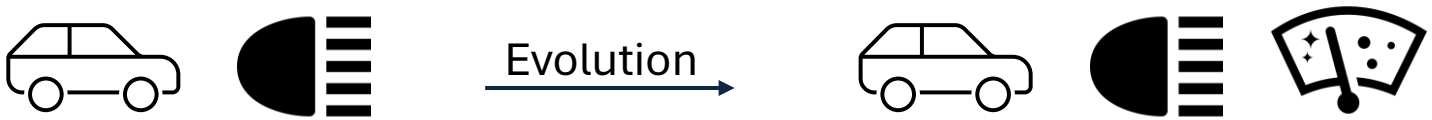
- **RQ1:** Can **Consistency Preservation Rules** be meaningfully applied to **delta domains**?
- **RQ2:** To which degree can **Product-CPRs** be automatically translated to **Delta-CPRs**?

Consistency Preservation on Deltas

Product-based evolution:

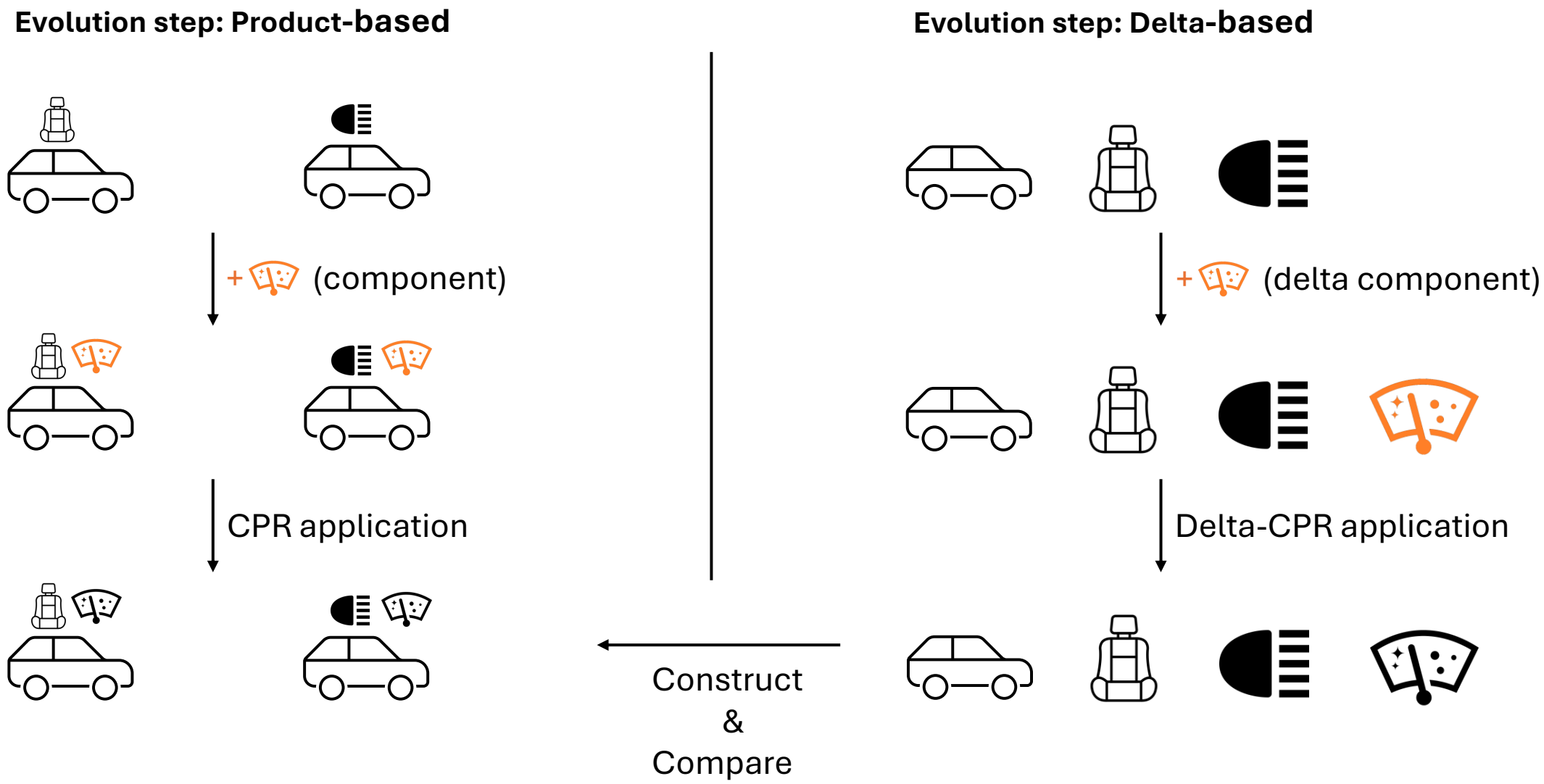


Delta-based evolution:

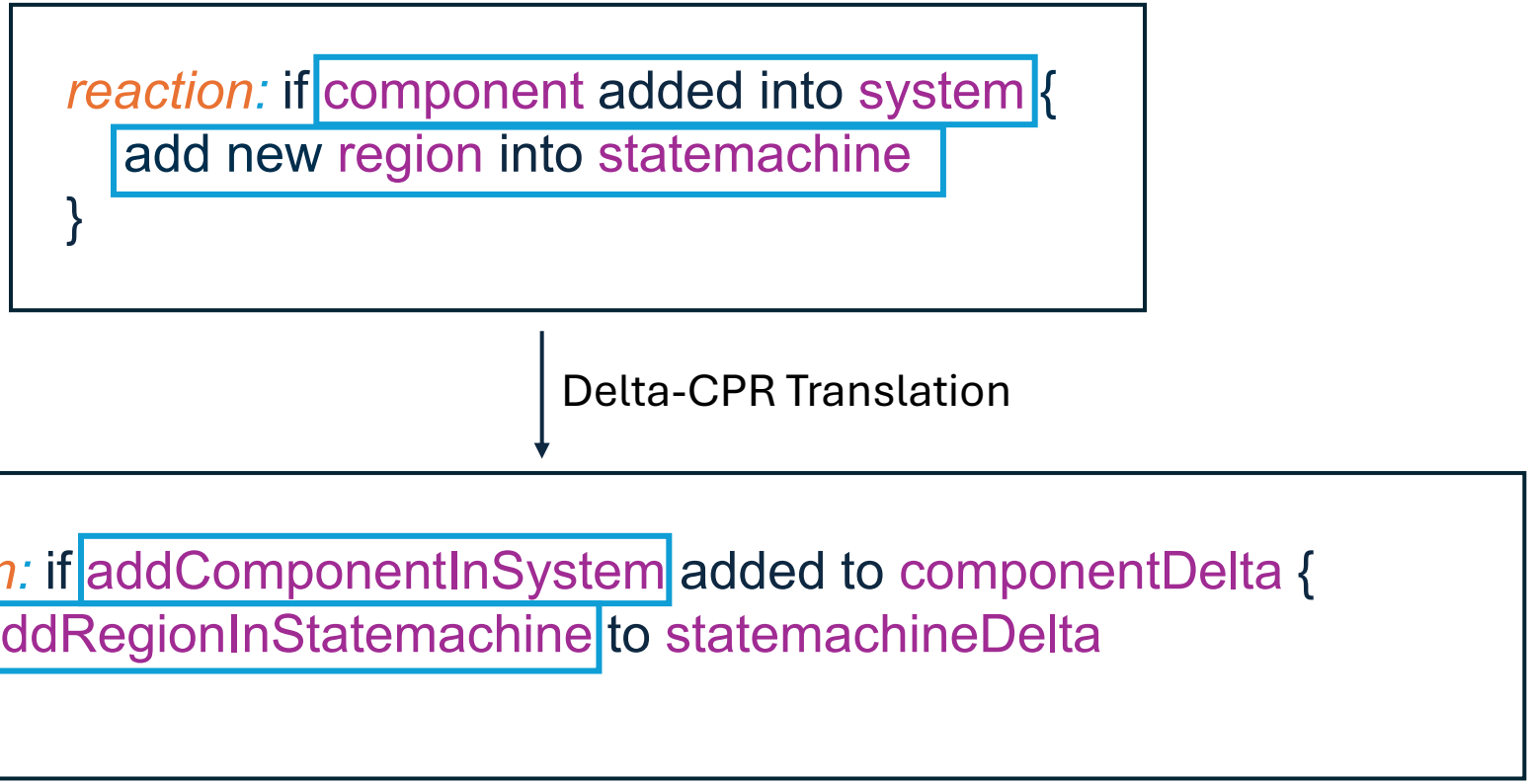


Delta-CPRs (in regard to Product-CPRs) are **correct**, iff **all deltas** after the Delta-CPR application **construct consistent products** for any valid feature selection

Correctness Evaluation – Testing Kit



CPR to Delta-CPR Translation



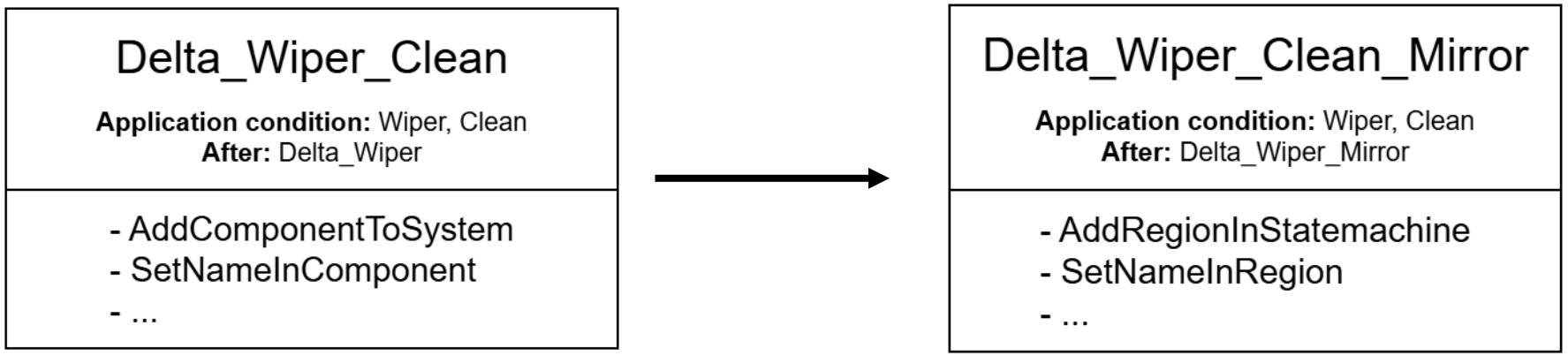
- The CPR to Delta-CPR translation can be partially automated!

Localization of Delta Operations

- In which delta should the operations introduced by CPR changes be placed?

  → **Mirror** existing deltas through all domains

  - Set same application condition in mirror delta
  - Set application order to ensure all dependencies to other operations are satisfied

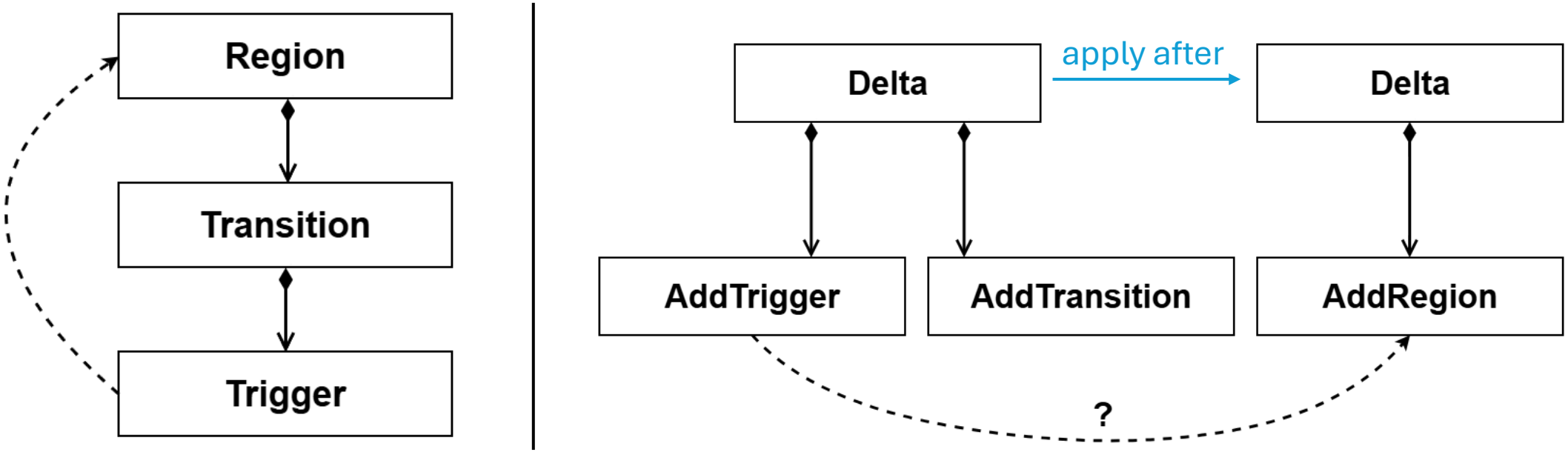


Resolving intra-domain dependencies

- How can intra-domain dependencies (such as compositions) be resolved for CPR application?

  → **Utilize the application order** of deltas to find model elements with dependencies

  - Application order of deltas respect the dependencies between delta operations inside them
  - If existing element are required, *add-operation* can always be found **upwards in the application order**



Problems with Automatic Delta-CPR translation

CPR includes external calls with an Object as an argument:

- External call expects specific object type
- Control flow can depend on state of that object

```
…
checkRegionNamingScheme(region)
…
```

```
…
checkRegionNamingScheme(region.name)
…
```

```
…
checkRegionNamingScheme(addRegion) ??
…
```

```
…
checkRegionNamingScheme(setNameInRegion.newValue)
…
```

Conclusion/Questions

- **Consistency Preservation Rules** can be applied to **delta domains,** and their **correctness** can be tested
- The translation from **Product-CPRs** to **Delta-CPRs** can be **partially automated**.

Other ways to **check/preserve cross-domain consistency** in product lines?

- Declarative approaches?
- 150% model approaches?
- Feature oriented programming?

Sources

- **[1] BCS Case Study**: https://github.com/TUBS-ISF/BCS-Case-Study-Full (as of 03.11.2025)
- **[2] Delta-oriented Modelling**: Dave Clarke, Michiel Helvensteijn, and Ina Schaefer. Abstract delta modelling. Mathematical Structures in Computer Science, 25(3):482–527, March 2015.
- **[3] Vitruvius Approach**: Heiko Klare, Max E. Kramer, Michael Langhammer, Dominik Werle, Erik Burger, and Ralf Reussner. Enabling consistency in view-based system development — The Vitruvius approach. Journal of Systems and Software, 171:110815, January 2021.

Pictograms:

- https://www.flaticon.com/free-icon/car-seat_5102957
- https://www.flaticon.com/free-icon/headlight_2316769
- https://www.flaticon.com/free-icon/administrator_12724606
- https://uxwing.com/car-windshield-wiper-icon/

Graphics:

- **[4]** VSUMM/Var-VSUMM Overview Graphic - Dirk Neumann