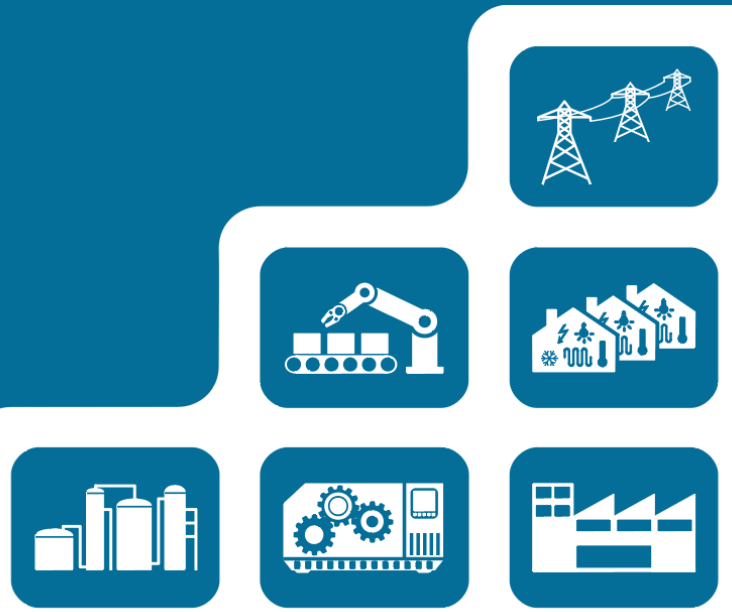
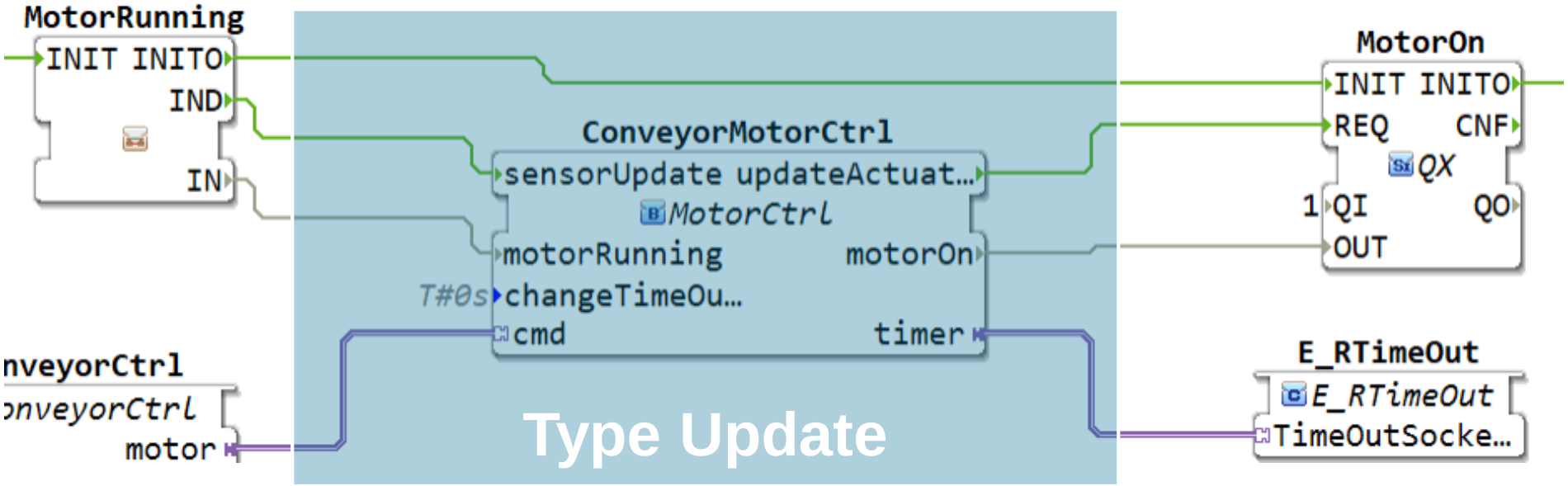
JKU
LINZ INSTITUTE
OF TECHNOLOGY

Improving Developer Experience with Refactoring Mechanisms for IEC 61499 Applications



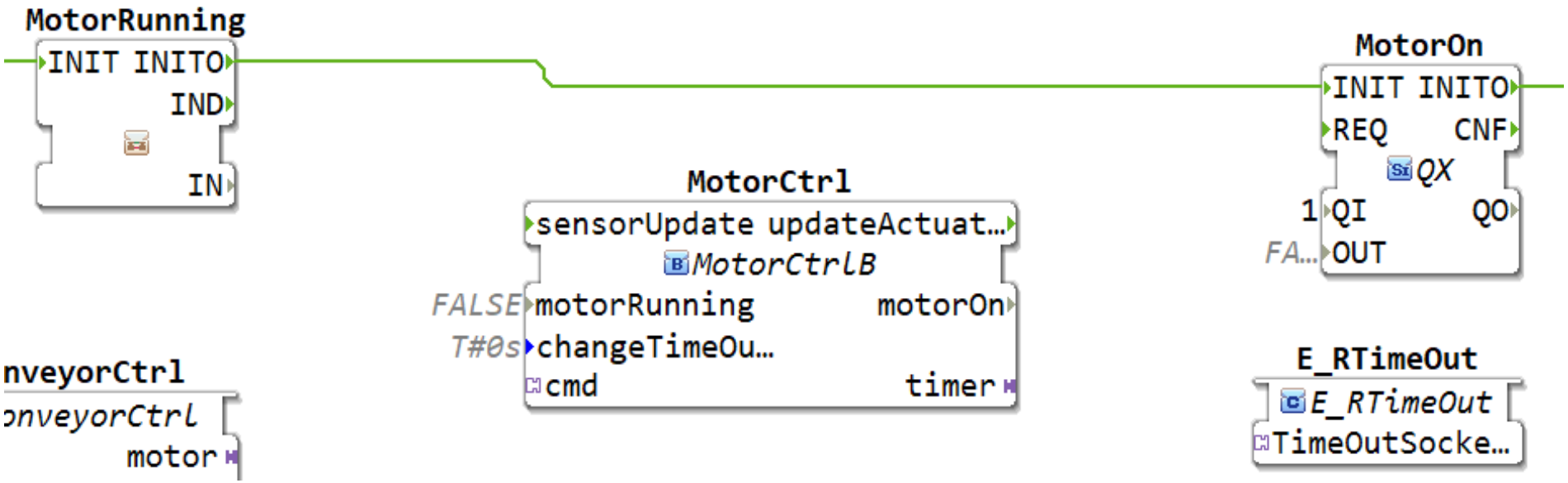
DI Dr. Bianca Wiesmayr, DI Michael Oberlehner
LIT | Cyber-Physical Systems Lab
Johannes Kepler University Linz

Motivation: Reduce Development Effort



Change of block / interface also requires changes to connections!

Option A: Fully Manual Process

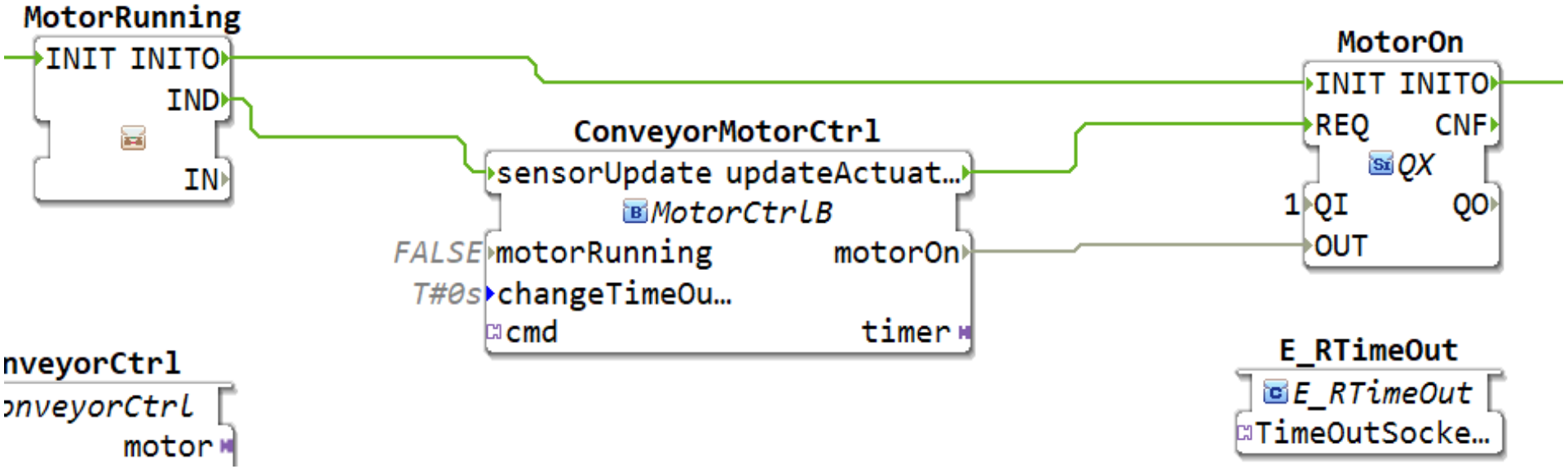


High viscosity

High error proneness

Hard mental operation

Option B: Update and Attempt Reconnect

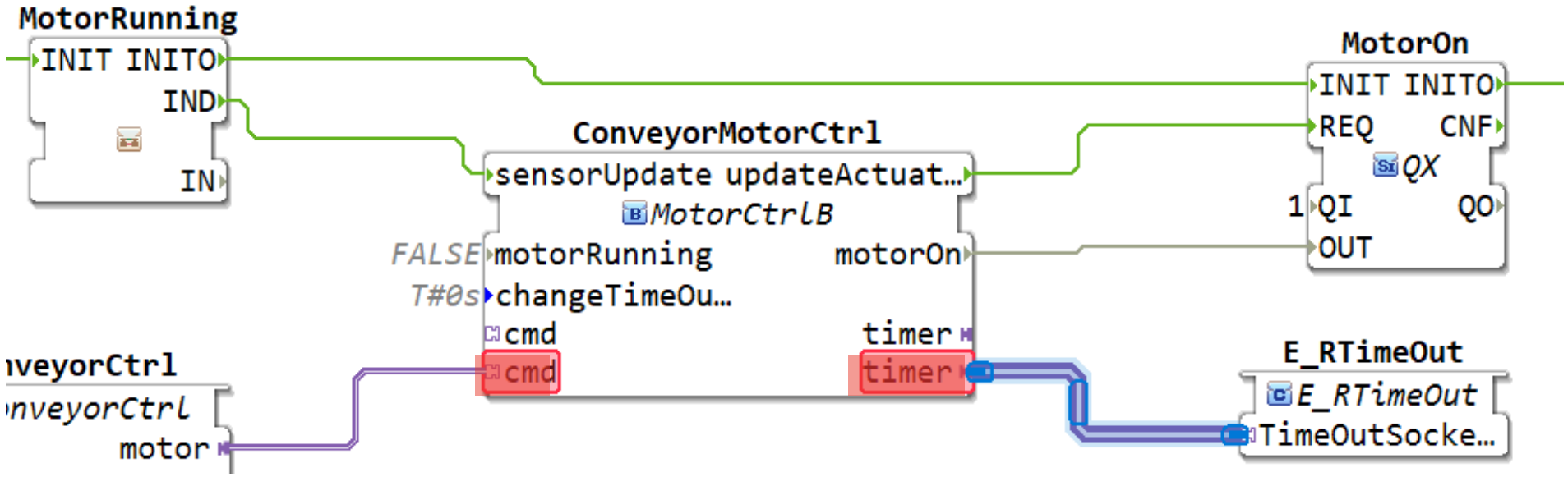


Reduced viscosity

Even higher error proneness

Hard mental operation

Option C: Update and Retain Connections

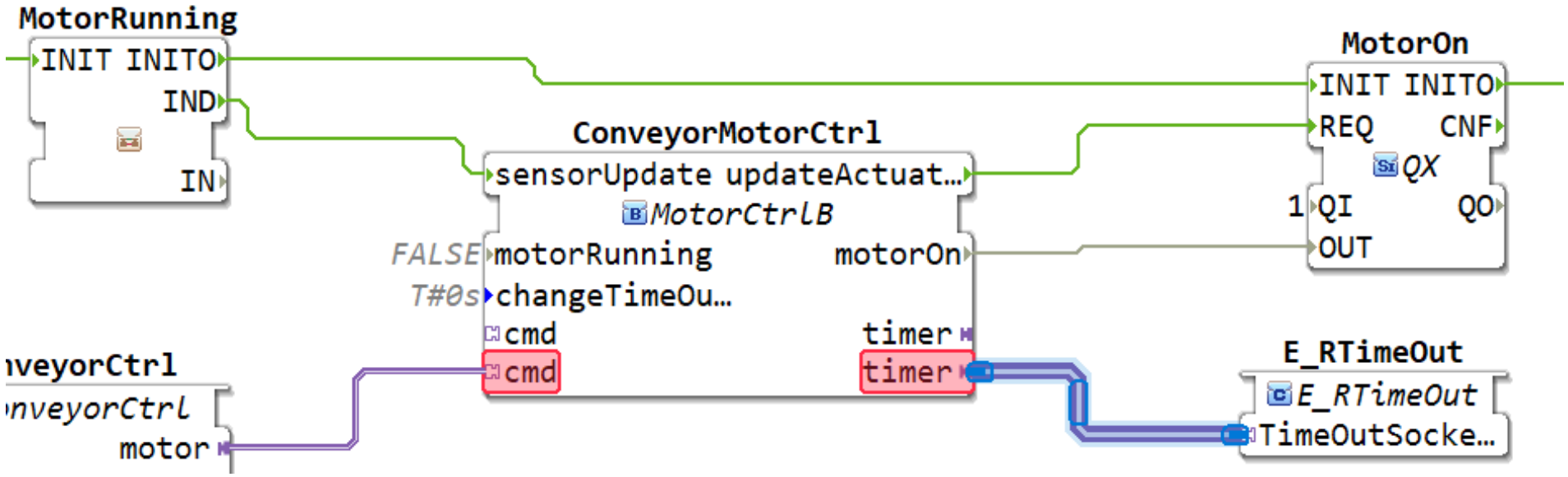


Reduced viscosity

Lower error proneness

Not so hard mental operation, but low role expressiveness

Option C: Update and Retain Connections



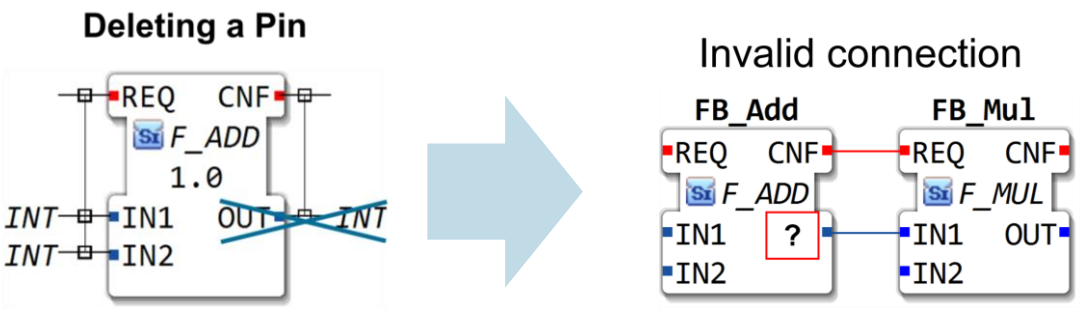
Reduced viscosity

Lower error proneness

Not so hard mental operation, but low role expressiveness

Problem: Inconsistent Application after Type Update

RQ
How to reduce inconsistencies after type update triggers?



Strategies Against Inconsistencies

- Refactoring Operations
- Error Visualization
- Safe Type Update
- Repair Operations

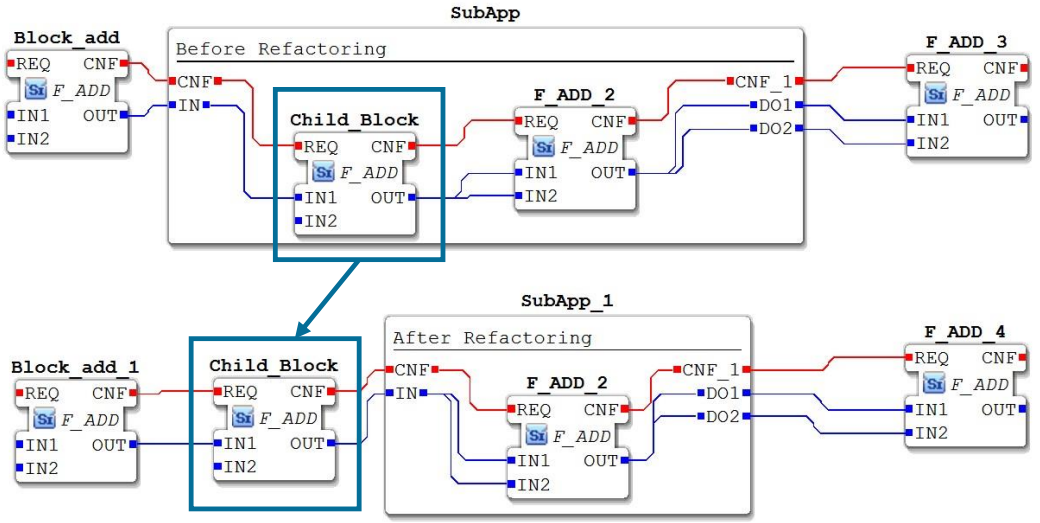
Introduction Refactoring & IEC 61499

Definition Refactoring

„A change made to the **internal structure** of software to make it **easier to understand** and **cheaper to modify** without changing ist oberservable behaviour“ (Fowler M., 1999)

Refactoring in IEC 61499

Restructuring a graphical function block diagram without changing its functionality



Refactoring Operations

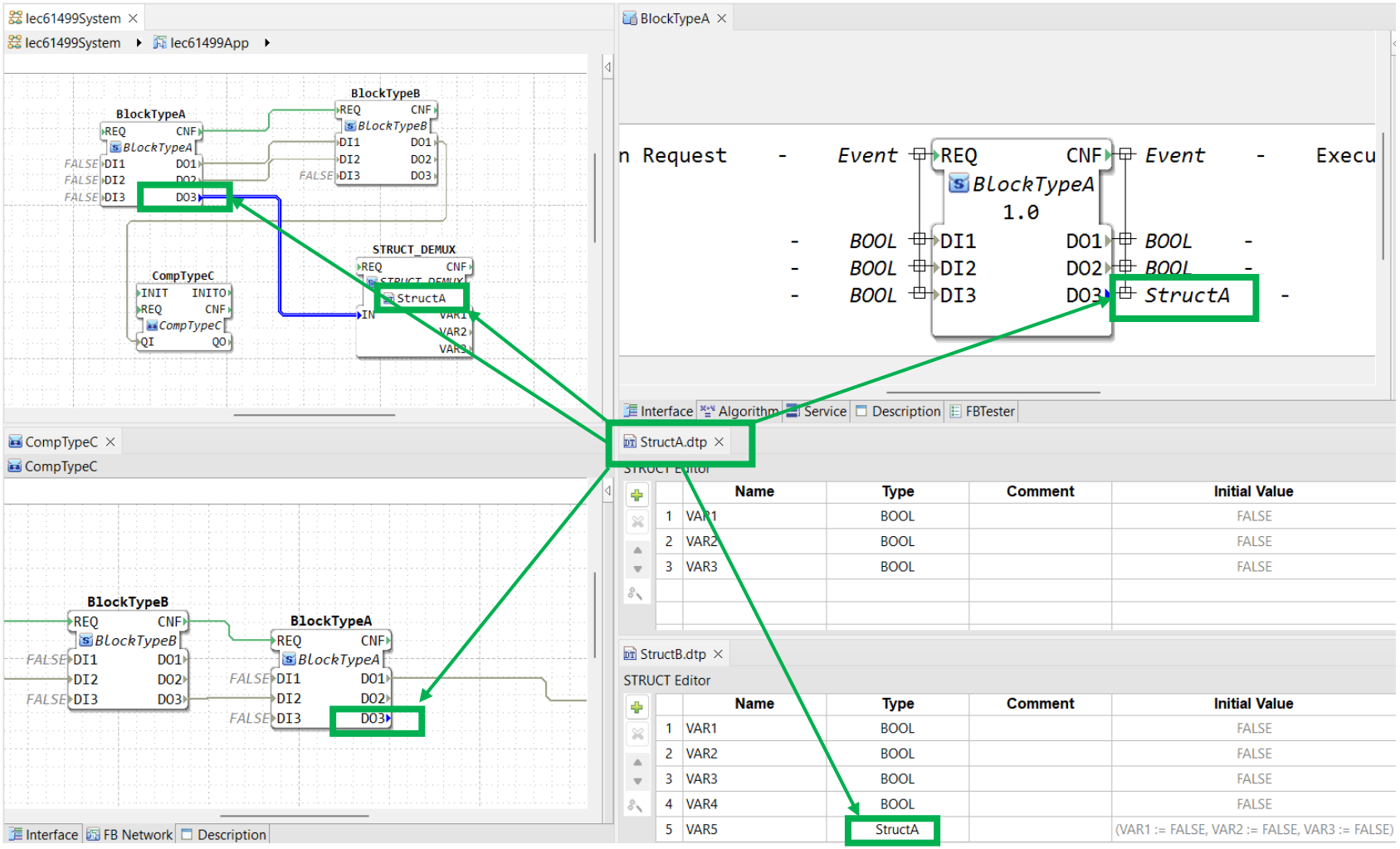
RQ
Can we establish a refactoring catalog for IEC 61499 by searching for existing Refactoring Operations in other languages using a generic meta-model as an orientation?

- 29 new IEC 61499 Refactoring Operations
  - 5 from BPMN
  - 16 from UML
  - 8 from Simulink
- 12 new Refactoring Operations for IEC 61131-3
- 1 new Refactoring Operations for BPMN
- 1 new Refactoring Operation for Simulink

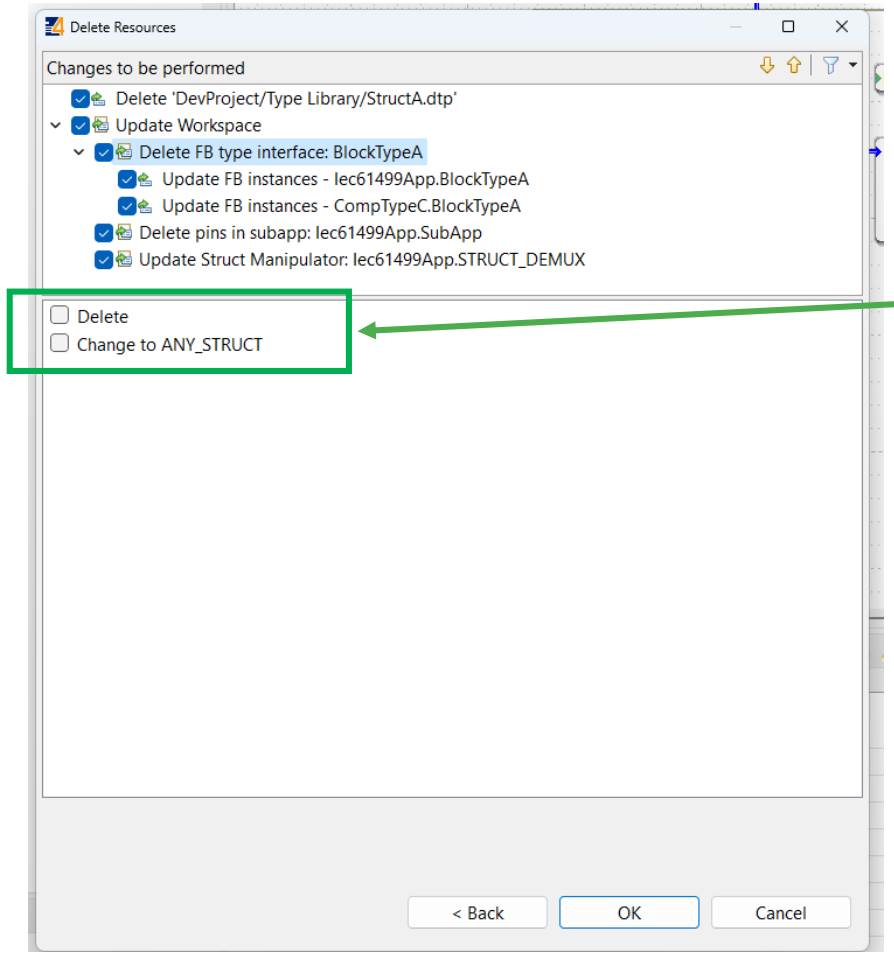
- Refatoring Operations are implemented in Eclipse 4diac [2]

[2] https://eclipse.dev/4diac/

Example Refactoring – Delete Structured Type

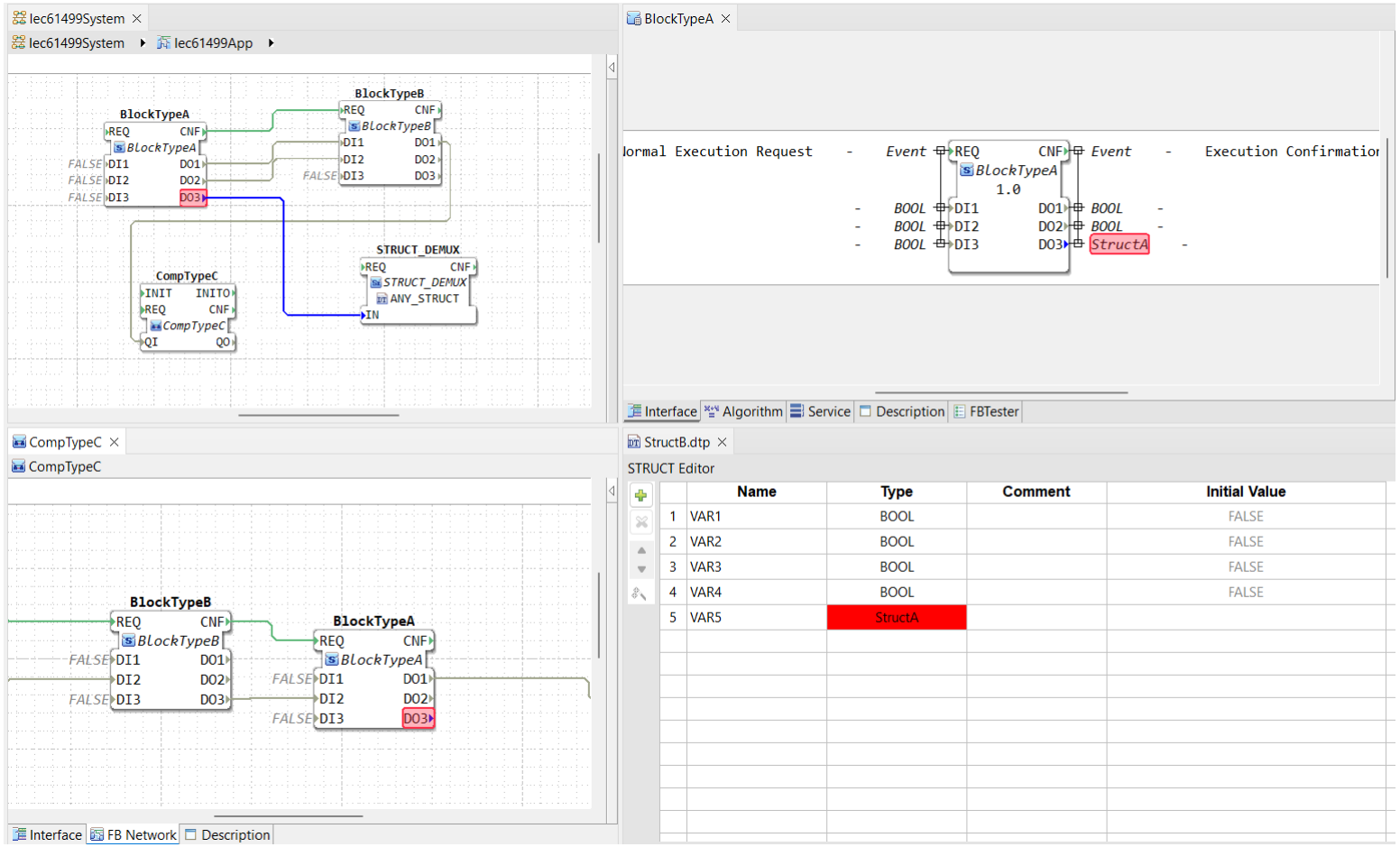


Delete StructA - Refactoring Preview



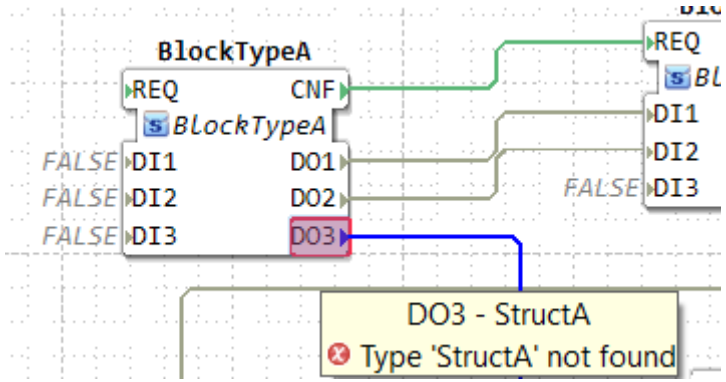
- The preview shows all affected instances of the regarding type
- It offers the user the possibility to apply repair operations during refactoring

Delete StructA - After Refactoring

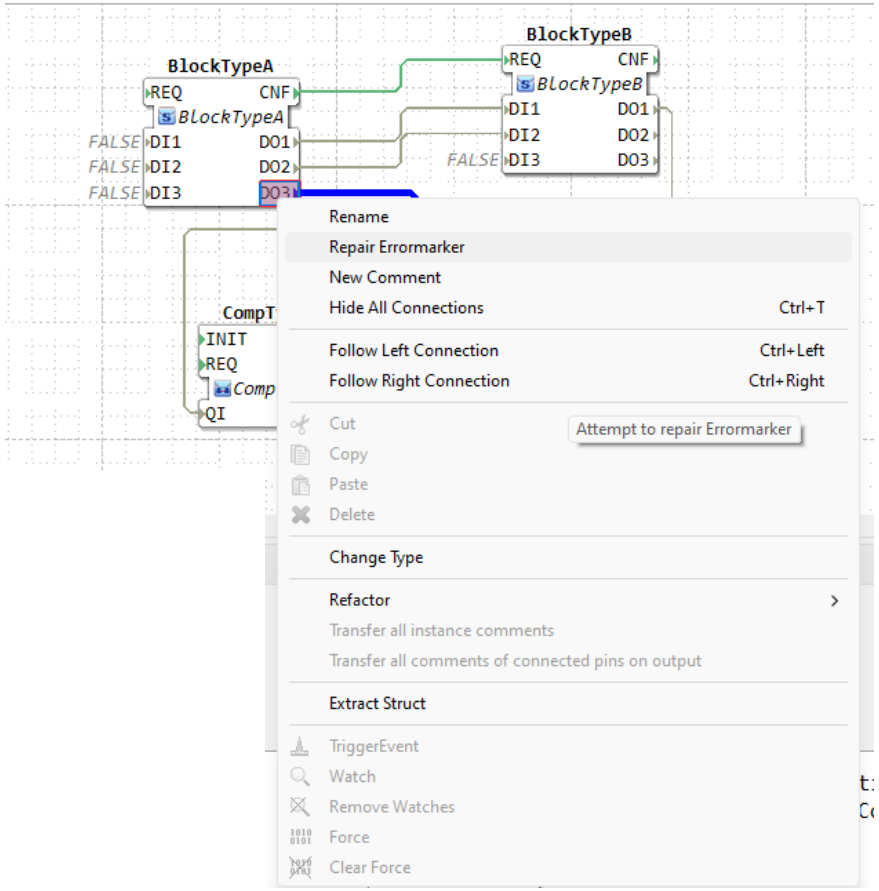


- The editor gives a visual hint to the user which elements are inconsistent after the refactoring
- It is also possible to show them in a collected view

Example – Repair Again by using Leftover Information



4diac IDE shows a dedicated error message and offers repair mechanisms



Next Steps

- Evaluation of Usability of Refactoring Operations
  - Is it easy and intuitive for the user to operate?
- Compare Maturity with other Tools
  - Can we compete with the Refactoring features of other tools?
- Implement Recommender System
  - Additional support for the user (e.g. refactoring hints for code smell)

LINZ INSTITUTE
OF TECHNOLOGY

Thank you!

Michael.oberlehner@jku.at
LIT | Cyber-Physical Systems Lab
Johannes Kepler University Linz