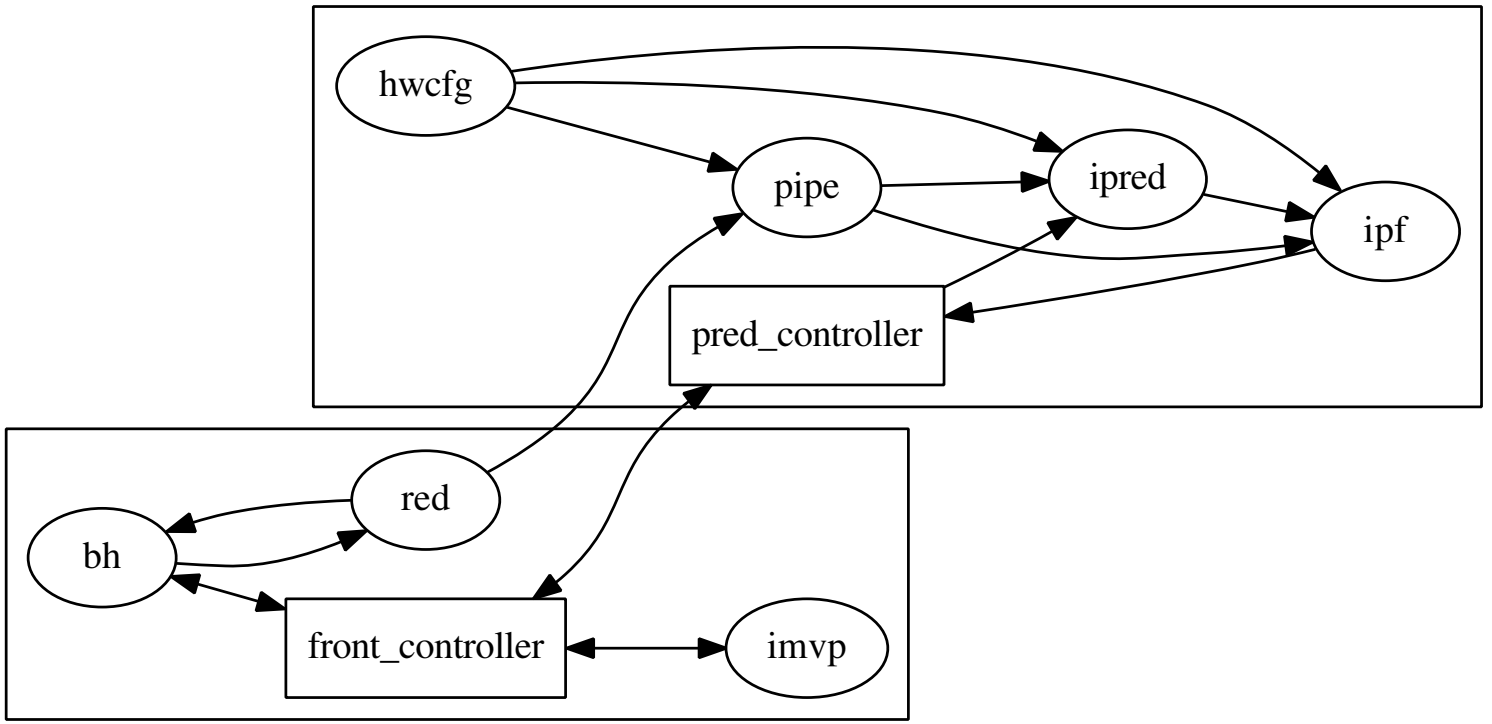
Programming Model Centric Interactive Debugging

Idea: **Integrate programming model concepts in interactive debugging**

Programming Model Centric Interactive Debugging

1 Provide a Structural Representation

- Draw application architecture diagrams
- Represent the relationship between the entities
- Offer catchpoints on architecture-related operations

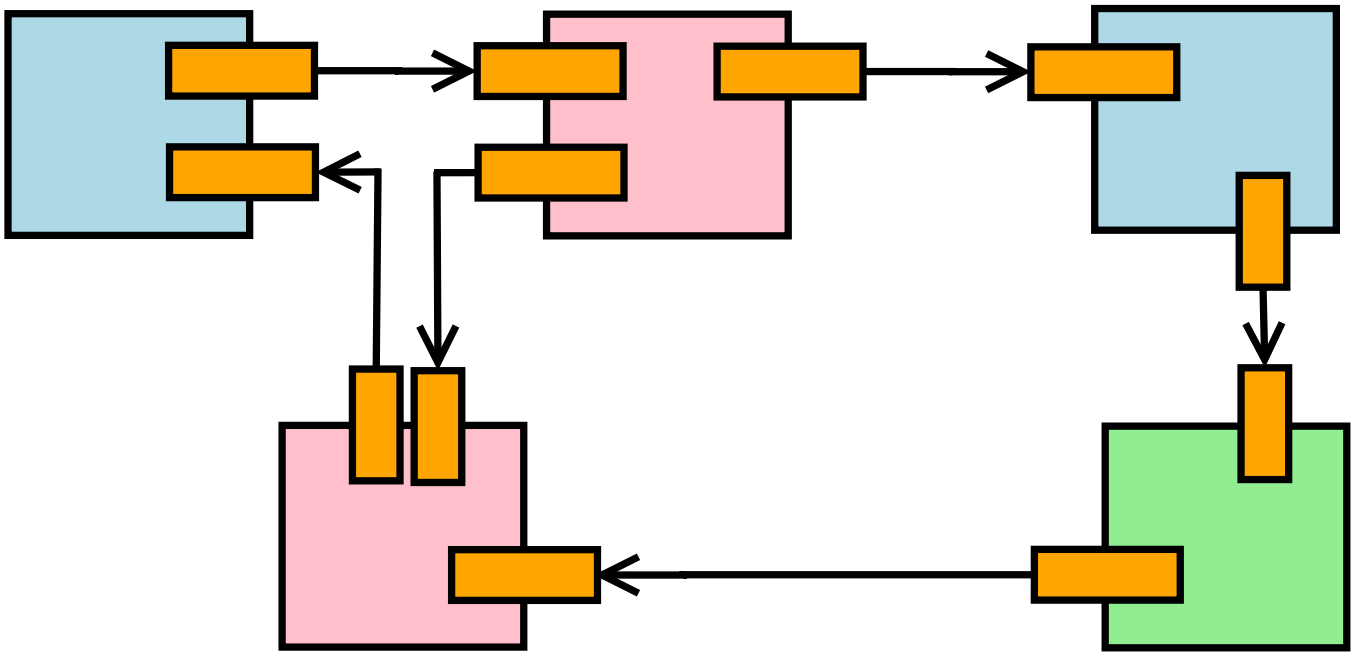


Graph of a dataflow from the case-study

Programming Model Centric Interactive Debugging

1 Provide a Structural Representation

- Draw application architecture diagrams
- Represent the relationship between the entities
- Offer catchpoints on architecture-related operations

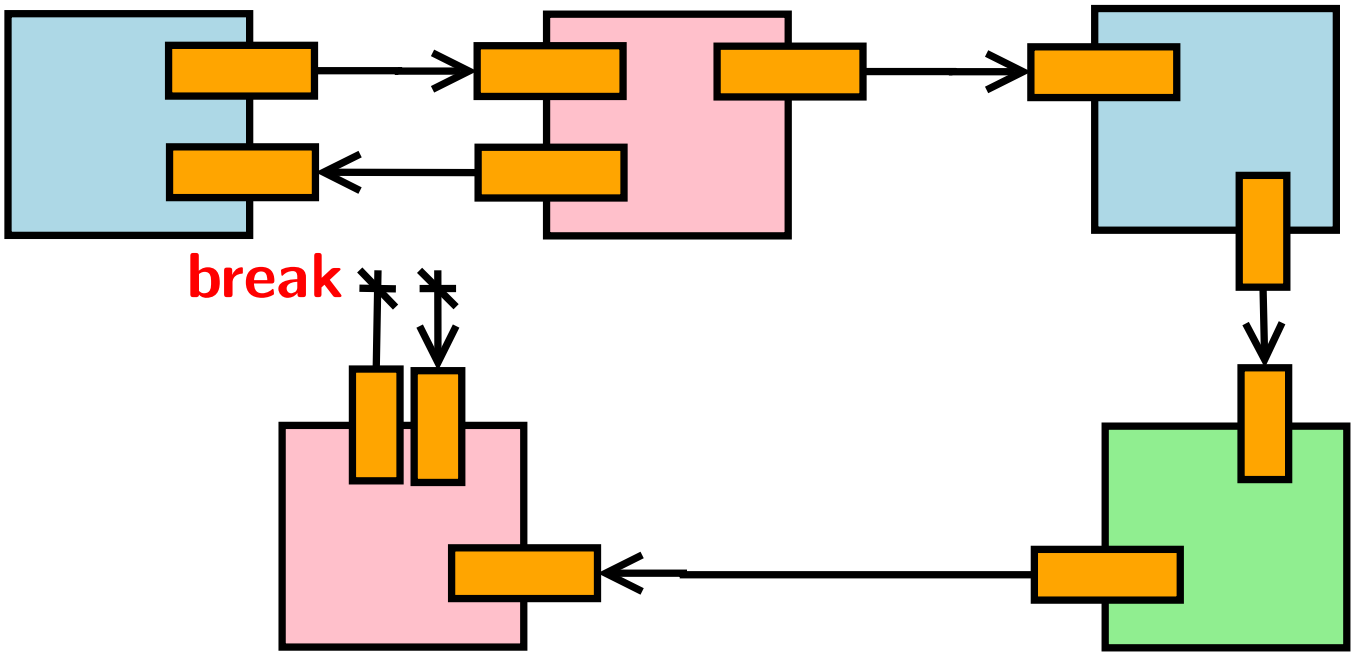


Reconfiguration of an application based on components

Programming Model Centric Interactive Debugging

1 Provide a Structural Representation

- Draw application architecture diagrams
- Represent the relationship between the entities
- Offer catchpoints on architecture-related operations

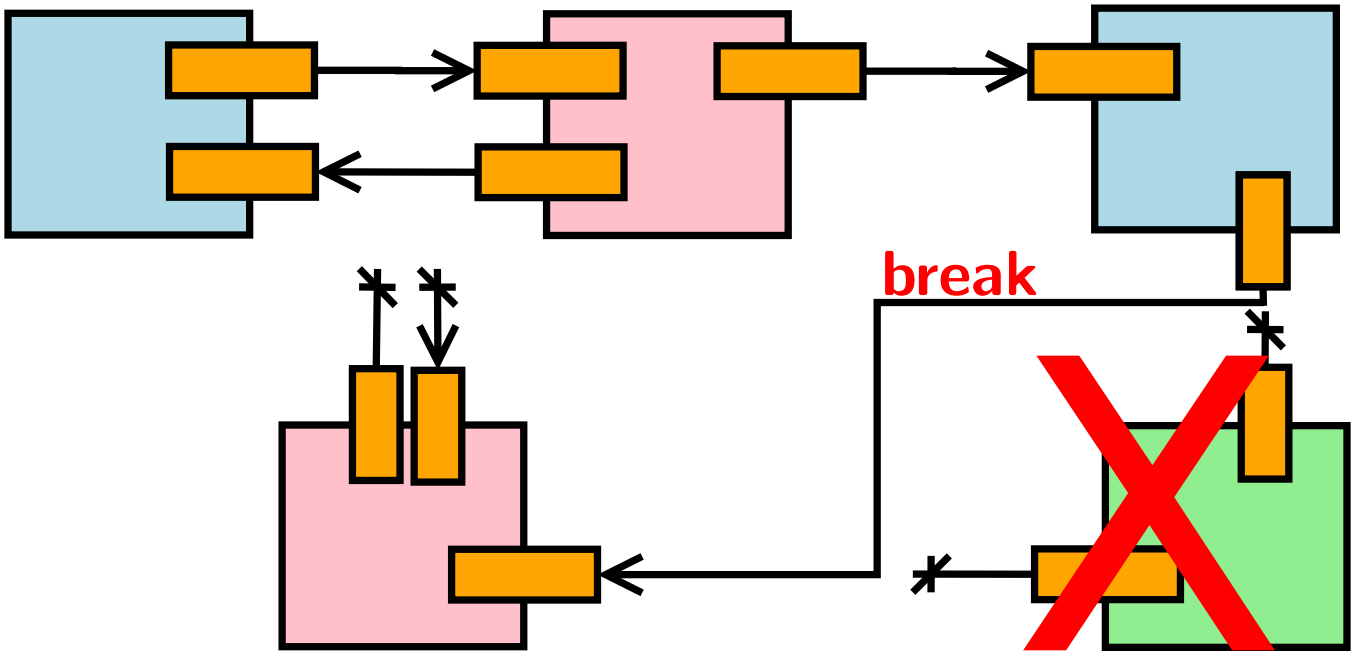


Reconfiguration of an application based on components

Programming Model Centric Interactive Debugging

1 Provide a Structural Representation

- Draw application architecture diagrams
- Represent the relationship between the entities
- Offer catchpoints on architecture-related operations

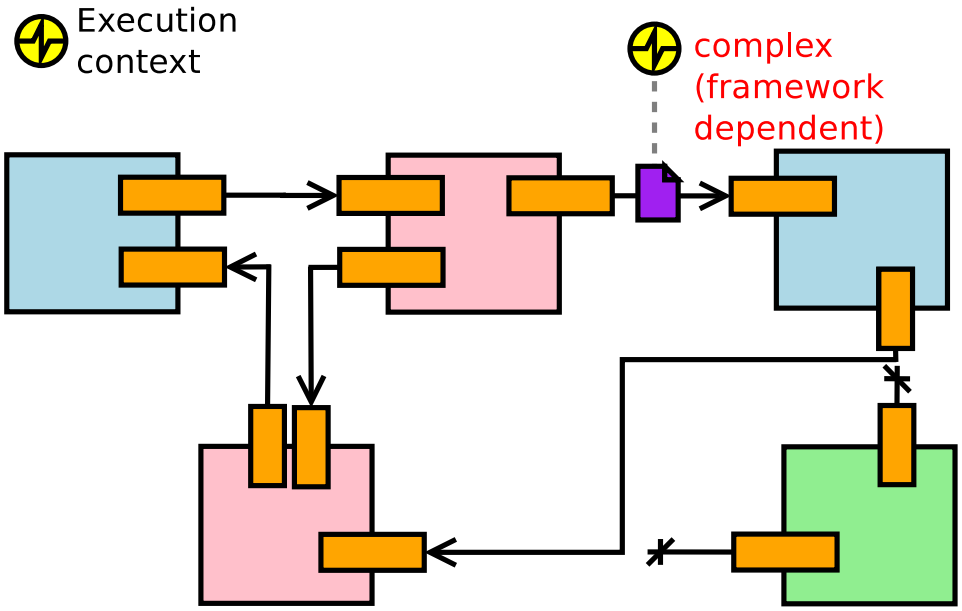


Reconfiguration of an application based on components

Programming Model Centric Interactive Debugging

2 Monitor Dynamic Behaviors

- Monitor the collaboration between the tasks
- Detect communication, synchronization events
  - interpret their pattern and semantics
    (one-to-one, one-to-many, global or local barriers)
- Offer communication-aware catchpoint mechanisms

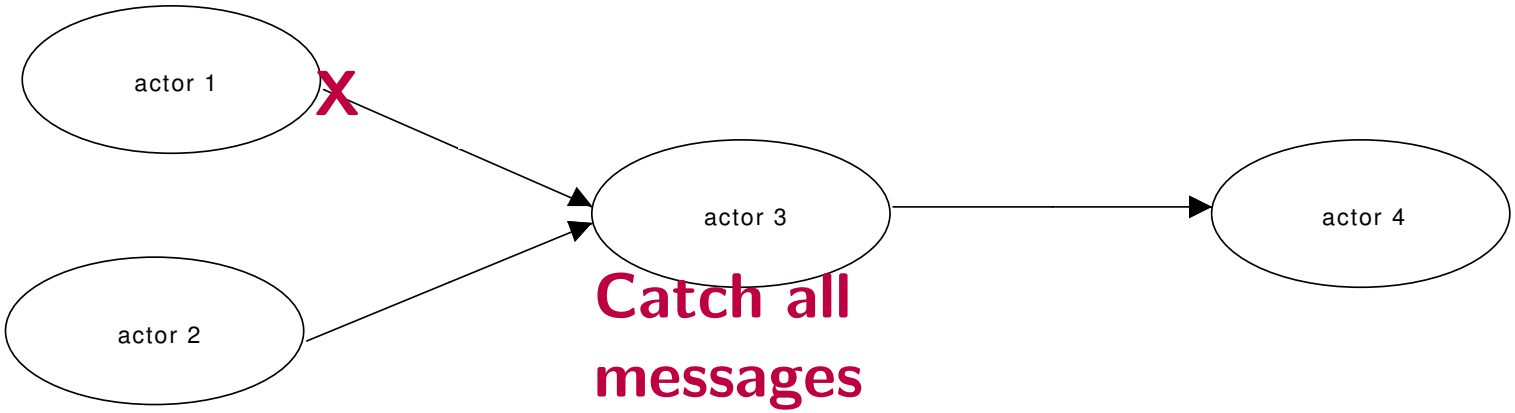



```
(pink) ask_service()
(env.) transmit request...
(blue) exec_service() {...}
```

Programming Model Centric Interactive Debugging

2 Monitor Dynamic Behaviors

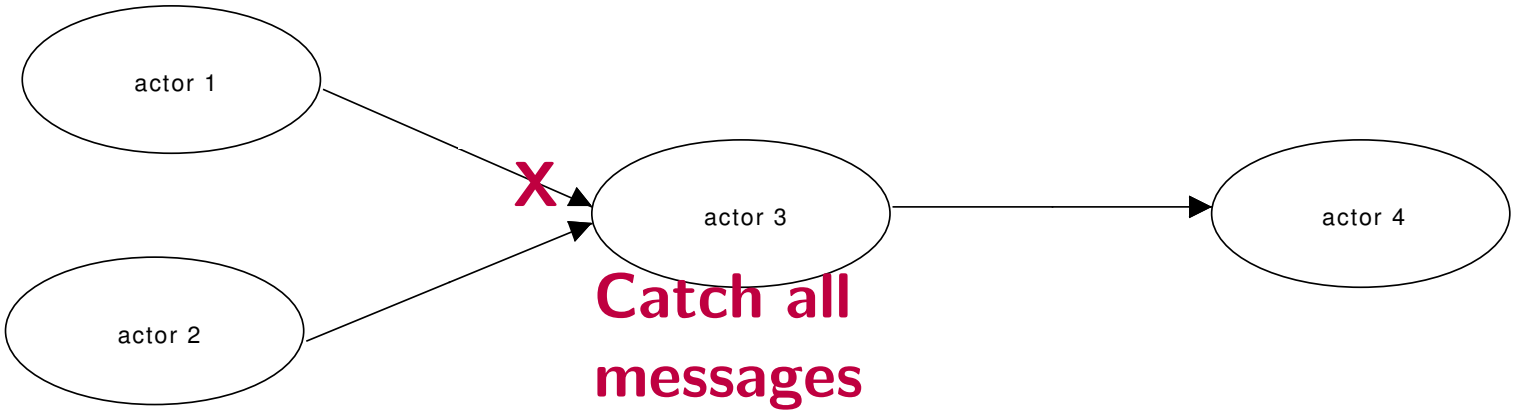
- Monitor the collaboration between the tasks
- Detect communication, synchronization events
  - interpret their pattern and semantics (one-to-one, one-to-many, global or local barriers)
- Offer communication-aware catchpoint mechanisms



Programming Model Centric Interactive Debugging

2 Monitor Dynamic Behaviors

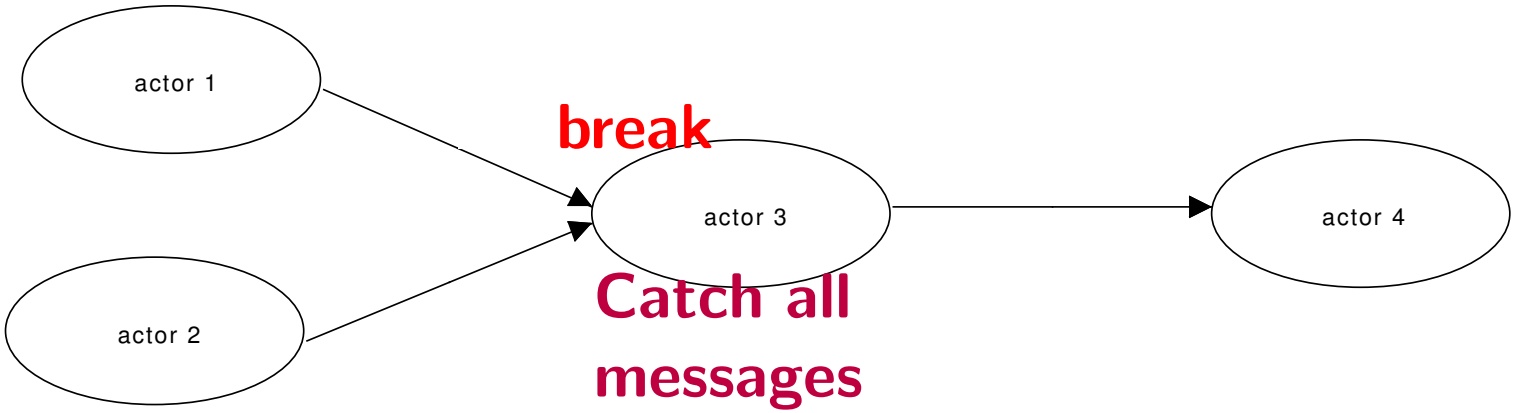
- Monitor the collaboration between the tasks
- Detect communication, synchronization events
  - interpret their pattern and semantics (one-to-one, one-to-many, global or local barriers)
- Offer communication-aware catchpoint mechanisms



Programming Model Centric Interactive Debugging

2 Monitor Dynamic Behaviors

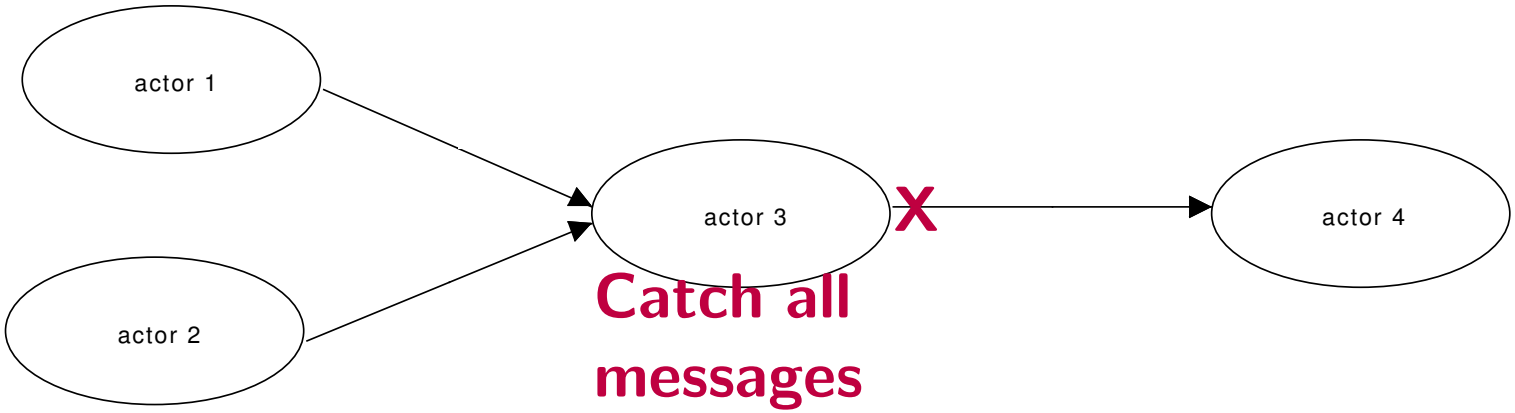
- Monitor the collaboration between the tasks
- Detect communication, synchronization events
  - interpret their pattern and semantics
    (one-to-one, one-to-many, global or local barriers)
- Offer communication-aware catchpoint mechanisms



Programming Model Centric Interactive Debugging

2 Monitor Dynamic Behaviors

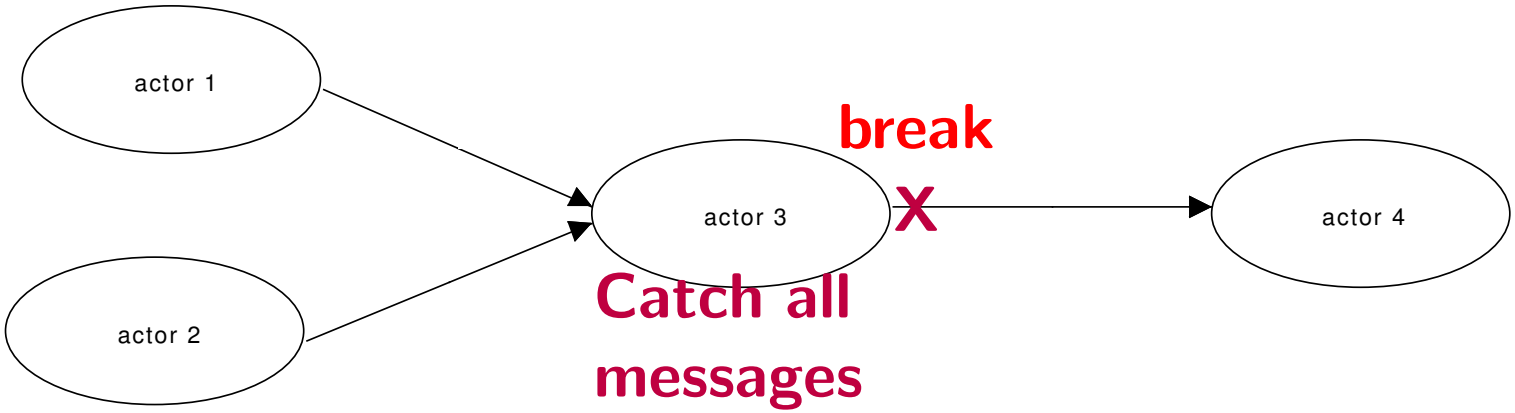
- Monitor the collaboration between the tasks
- Detect communication, synchronization events
  - interpret their pattern and semantics (one-to-one, one-to-many, global or local barriers)
- Offer communication-aware catchpoint mechanisms



Programming Model Centric Interactive Debugging

2 Monitor Dynamic Behaviors

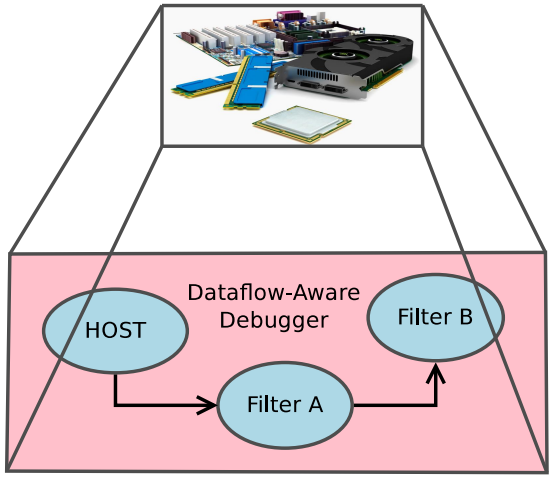
- Monitor the collaboration between the tasks
- Detect communication, synchronization events
  - interpret their pattern and semantics (one-to-one, one-to-many, global or local barriers)
- Offer communication-aware catchpoint mechanisms



Programming Model Centric Interactive Debugging

3 Interact with the Abstract Machine

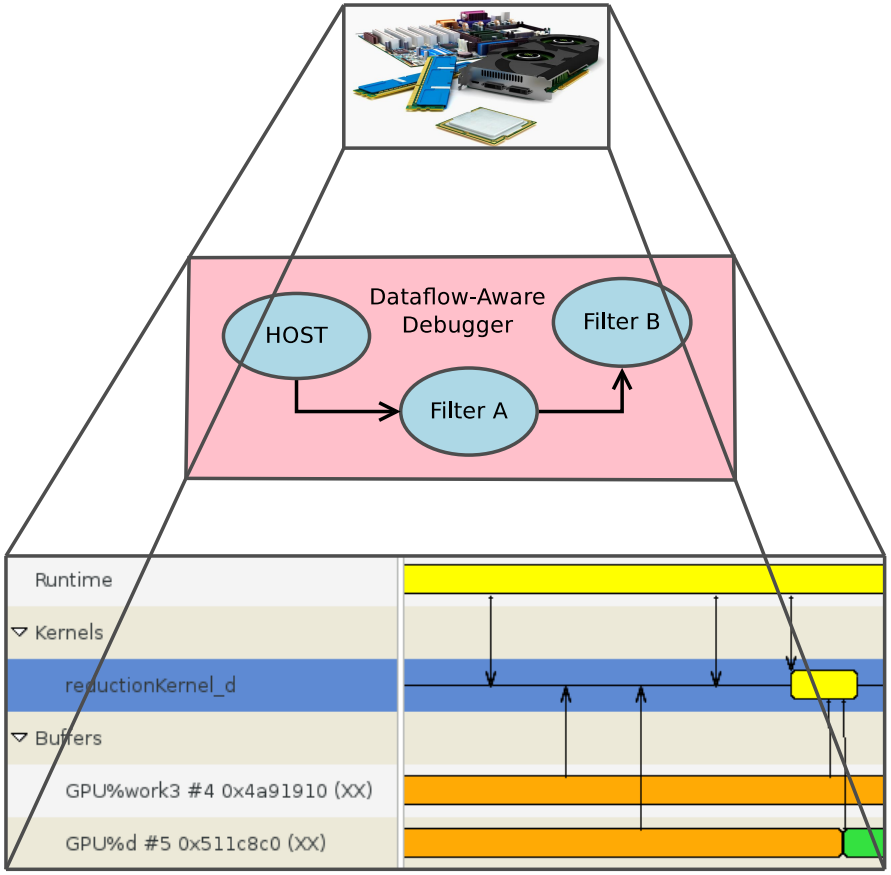
- Recognize the different entities of the model
- Provide details about their state, schedulability, callstack, ...
- Provide support to understand how they reached their current state



Programming Model Centric Interactive Debugging

3 Interact with the Abstract Machine

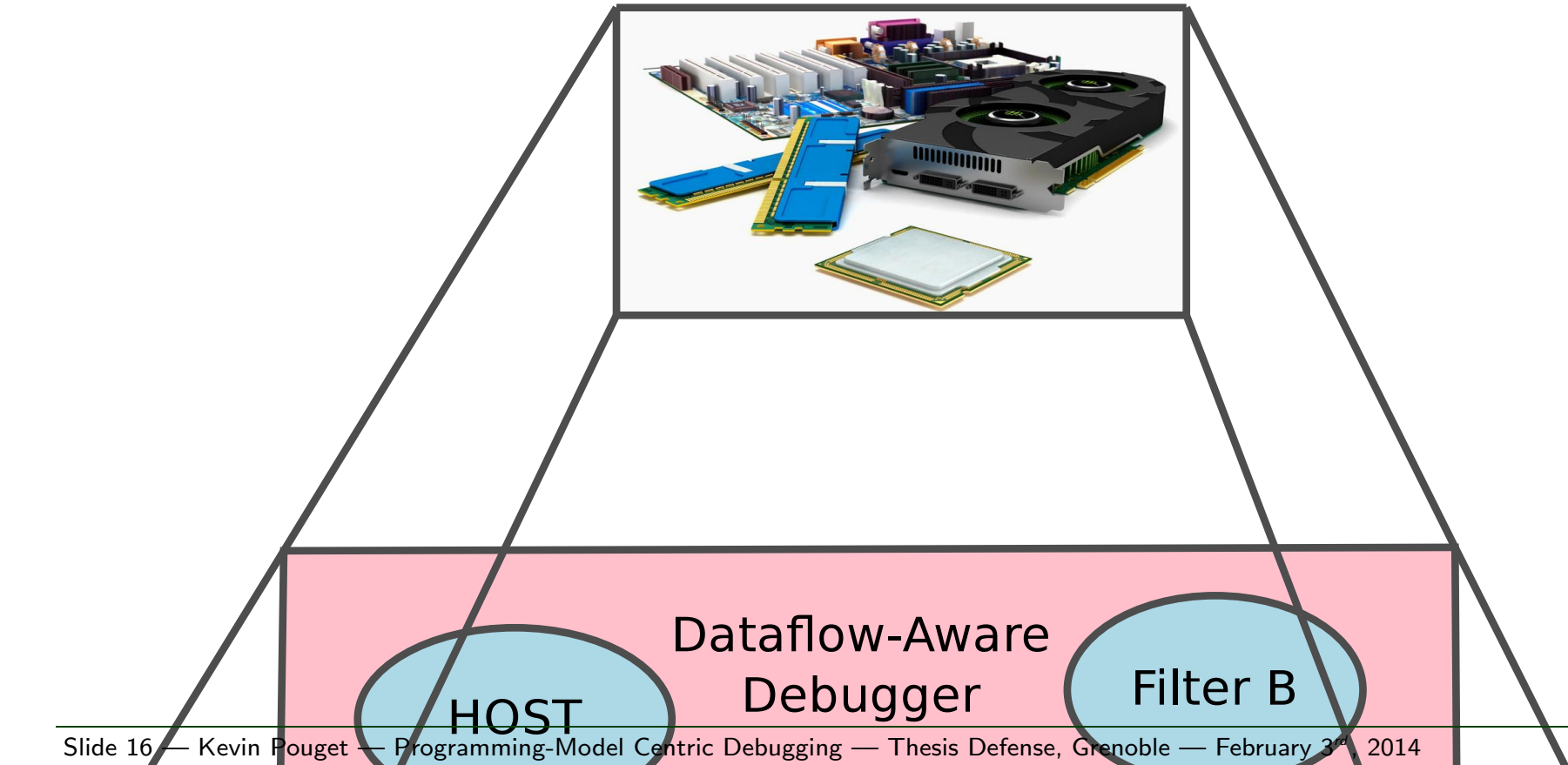
- Recognize the different entities of the model
- Provide details about their state, schedulability, callstack, ...
- Provide support to understand how they reached their current state



Programming Model Centric Interactive Debugging

3 Interact with the (abstract) Machine

- Support interactions with *real* machine
  - memory and processor inspection
  - breakpoints and watchpoints (maybe per entity)
  - step-by-step execution ...



Programming Model Centric Interactive Debugging

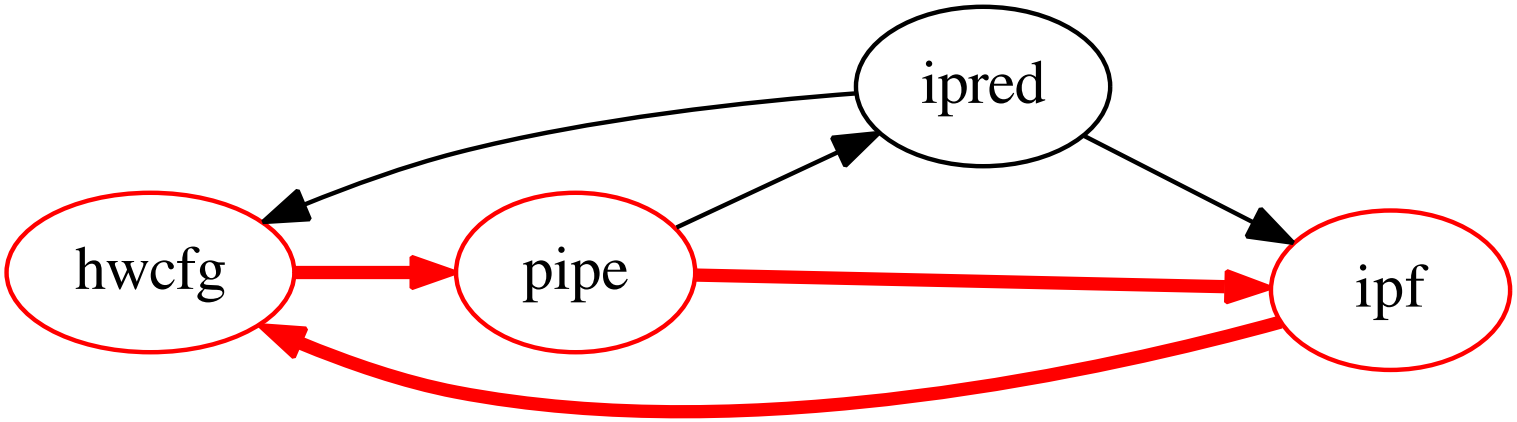
4 Open Up to Model and Environment Specific Features

- Follow messages over *multiple* entities
- User-defined constraints on the graph topology
- Deadlock detection in task-based models

Programming Model Centric Interactive Debugging

4 Open Up to Model and Environment Specific Features

- Follow messages over *multiple* entities
- User-defined constraints on the graph topology
- Deadlock detection in task-based models



cycles in the graph of blocking communications $\Longrightarrow$ deadlock

Programming Model Centric Interactive Debugging

1. Provide a Structural Representation
2. Monitor Dynamic Behaviors
3. Interact with the Abstract Machine
4. Open Up to Model and Environment Specific Features