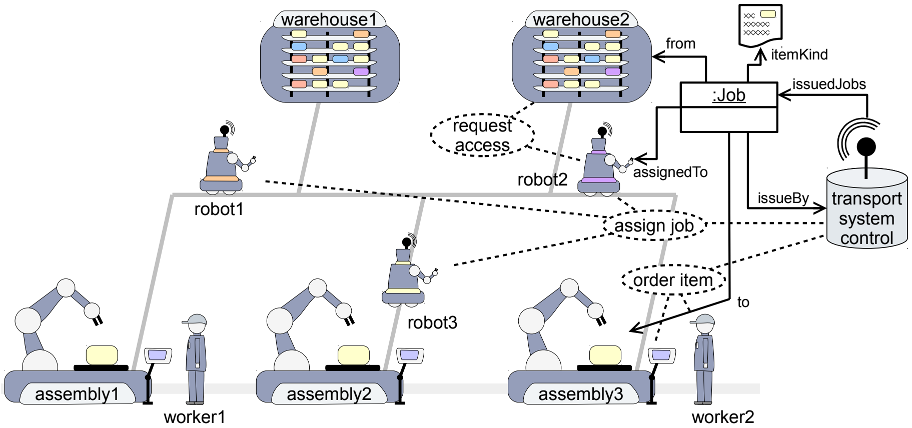
Integrating Graph Transformations and Modal Sequence Diagrams for Specifying Structurally Dynamic Reactive Systems

Sabine Winetzhammer[1], Joel Greenyer[2] and Matthias Tichy[3]

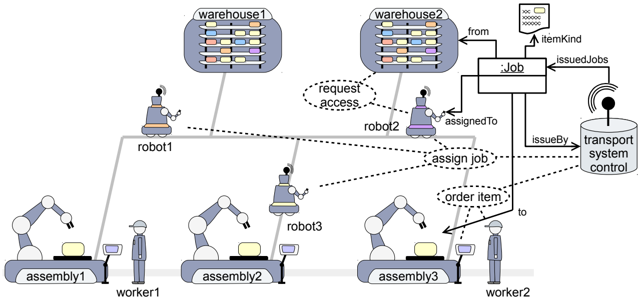
[1]Chair of Applied Computer Science 1 - Software Engineering, Bayreuth University, Germany
[2]Software Engineering Group, Leibniz Universität Hannover, Germany.
[3] Software Engineering Division, Chalmers | University of Gothenburg, Sweden.

September 29th, 2014

What's happening...?

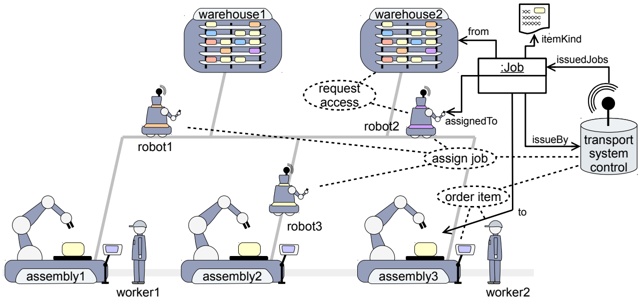


What's happening...?



- Multiple physically distributed mechatronic components
- Interaction between these components
- Changes in physical and logical structure

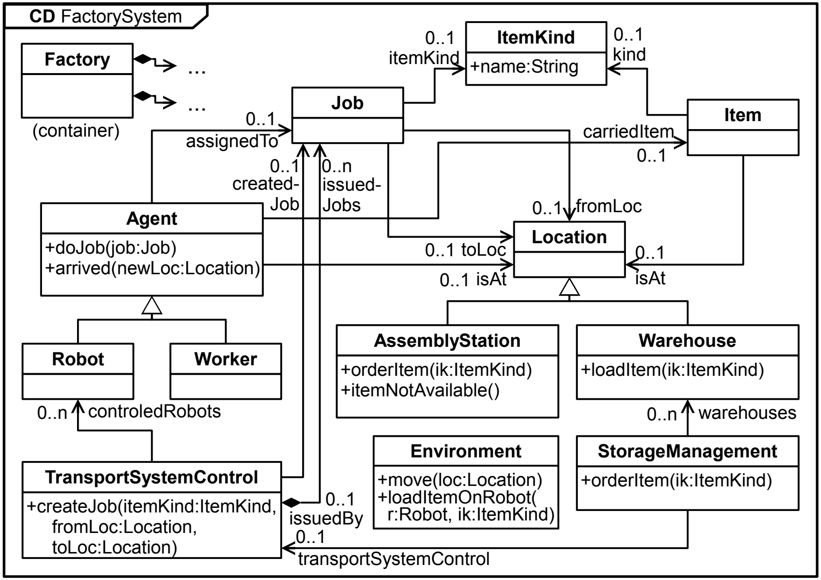
What's happening...?



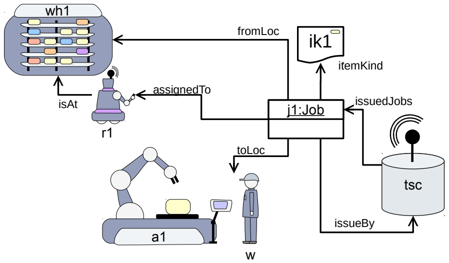
- Multiple physically distributed mechatronic components
- Interaction between these components
- Changes in physical and logical structure

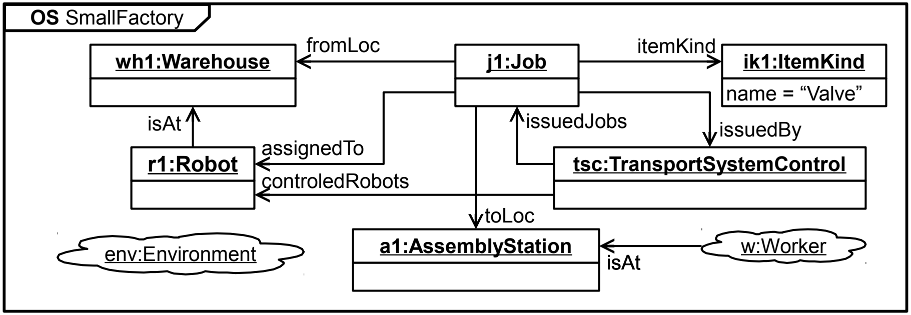
$\Rightarrow$ Structurally dynamic (distributed) reactive system

Class Diagram

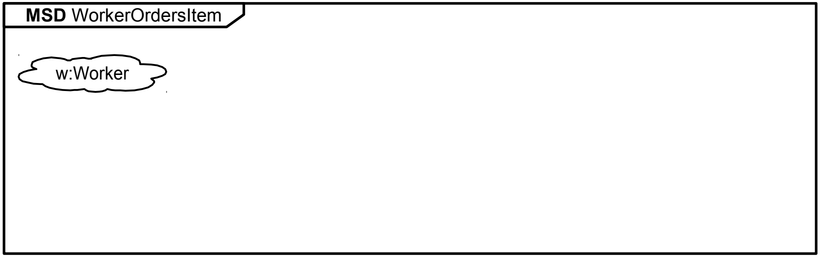


Object System

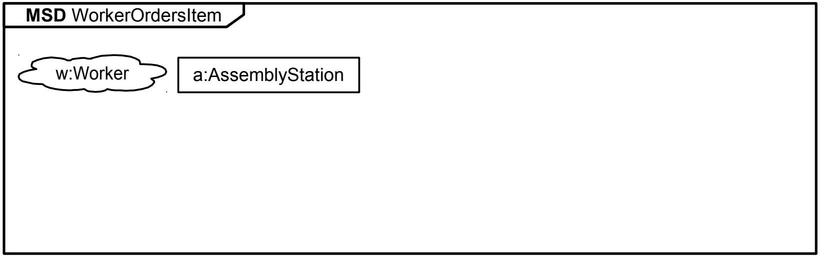




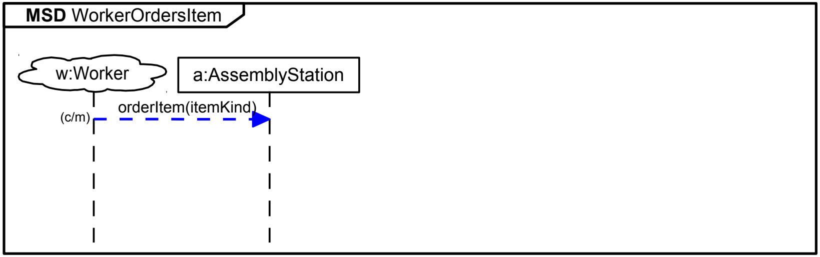
Modal Sequence Diagrams



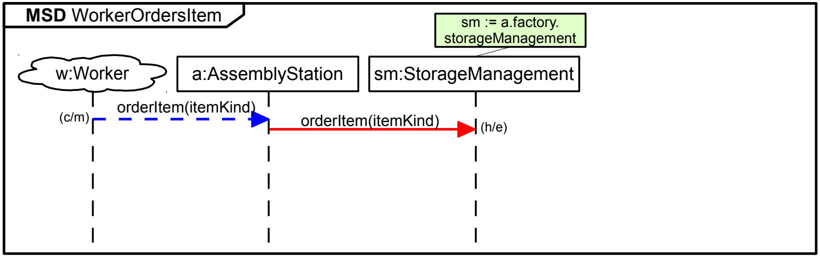
Modal Sequence Diagrams



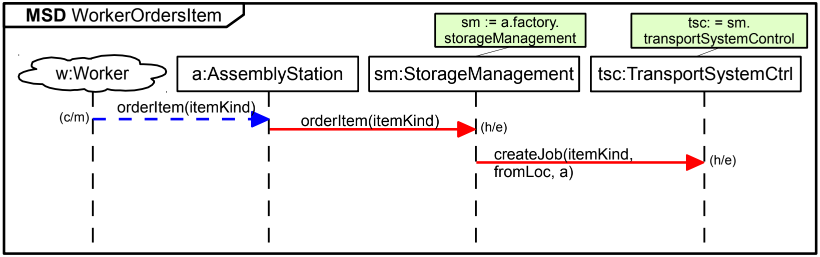
Modal Sequence Diagrams



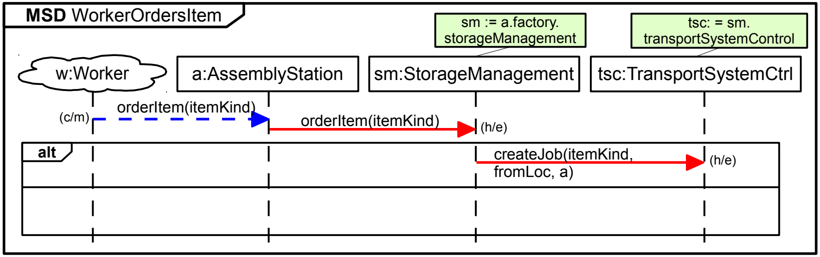
Modal Sequence Diagrams



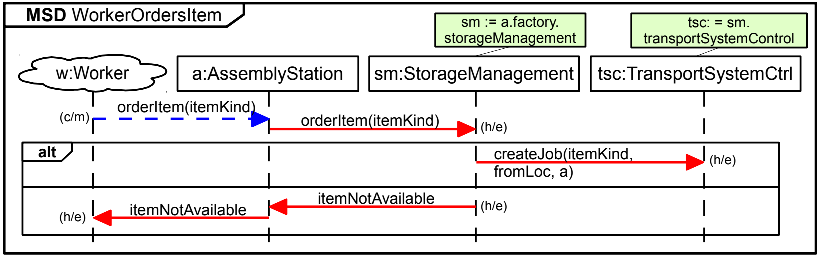
Modal Sequence Diagrams



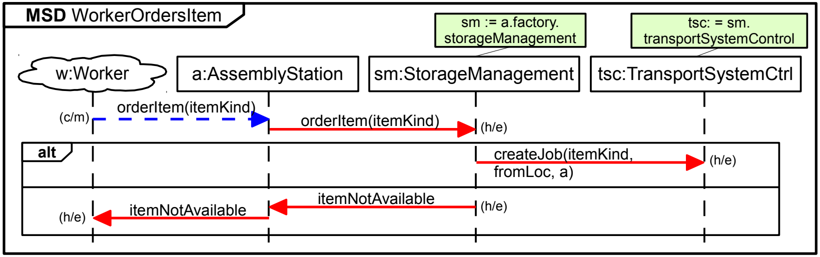
Modal Sequence Diagrams



Modal Sequence Diagrams

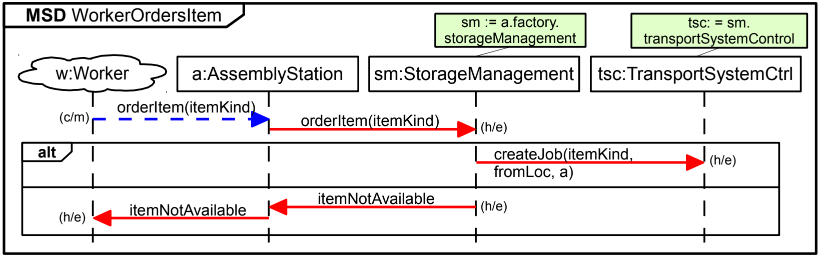


Modal Sequence Diagrams



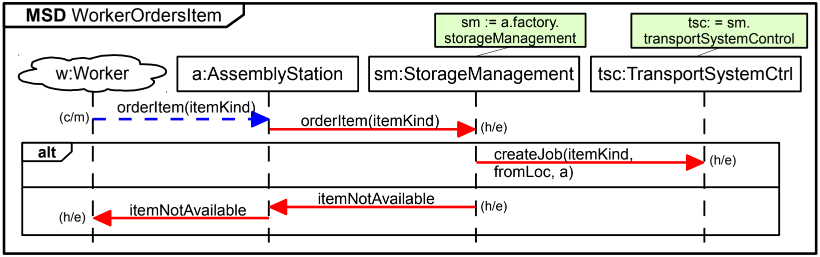
- Requirements

Modal Sequence Diagrams



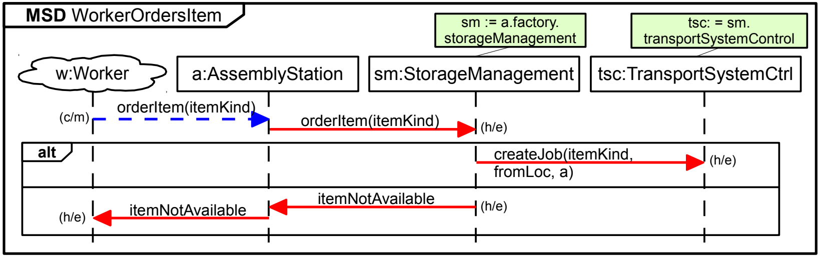
- Requirements
- Assumptions

Modal Sequence Diagrams



- Requirements
- Assumptions
- Simple structural changes

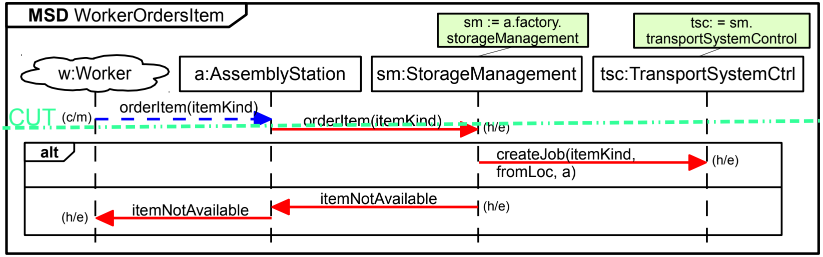
Modal Sequence Diagrams



- Requirements
- Assumptions
- Simple structural changes

& Play-out

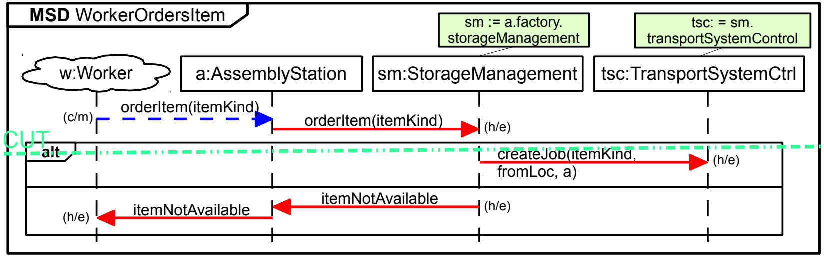
Modal Sequence Diagrams



- Requirements
- Assumptions
- Simple structural changes

& Play-out

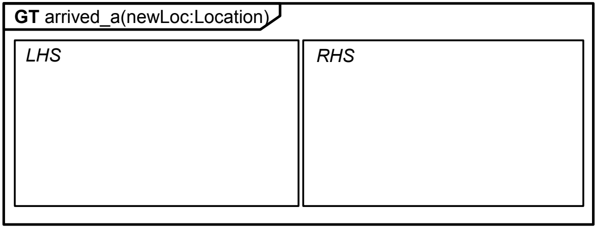
Modal Sequence Diagrams



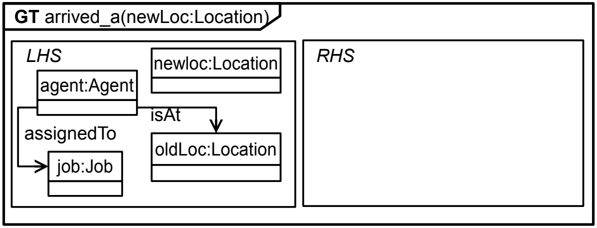
- Requirements
- Assumptions
- Simple structural changes

& Play-out

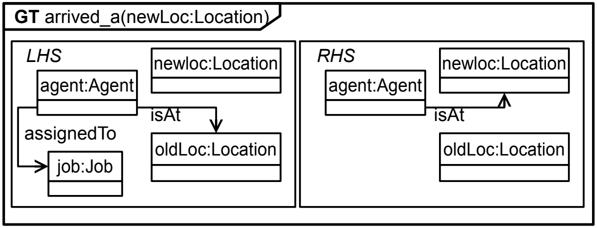
Graph Transformation Rules



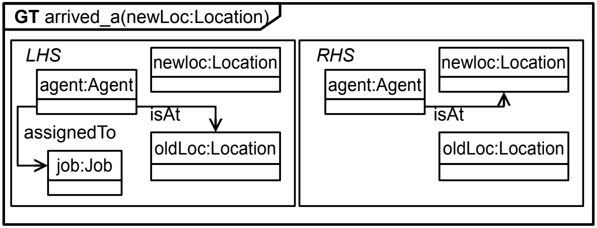
Graph Transformation Rules



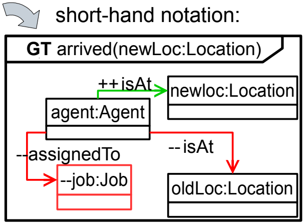
Graph Transformation Rules



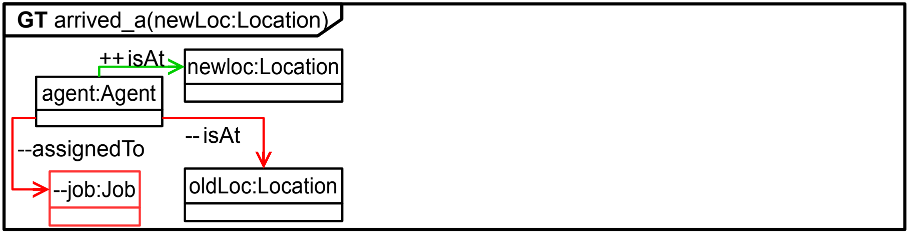
Graph Transformation Rules



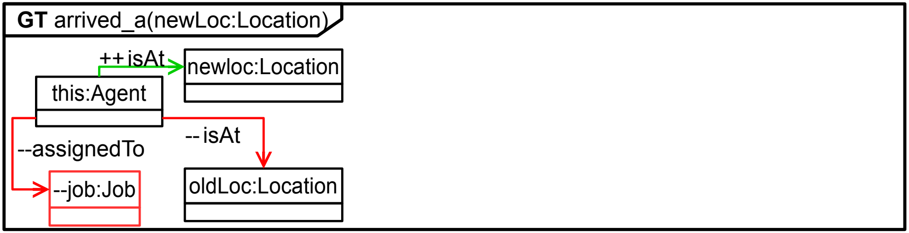
short-hand notation:



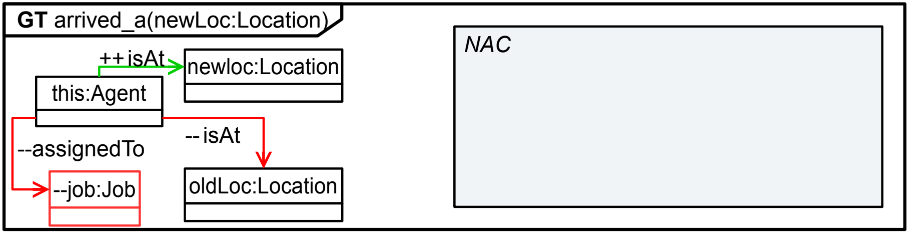
Graph Transformation Rules



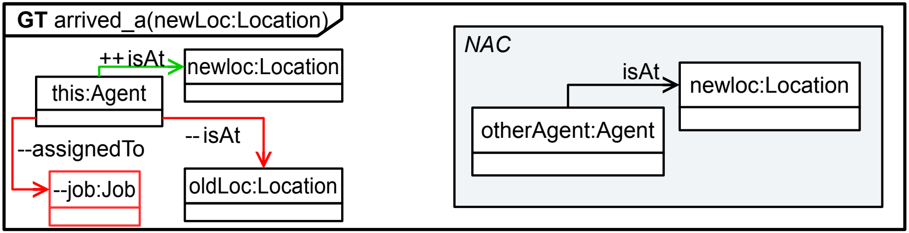
Graph Transformation Rules



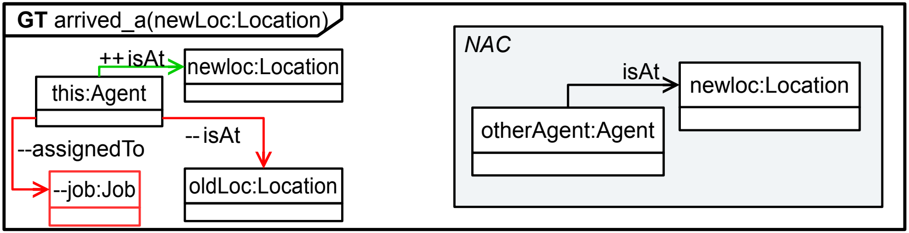
Graph Transformation Rules



Graph Transformation Rules

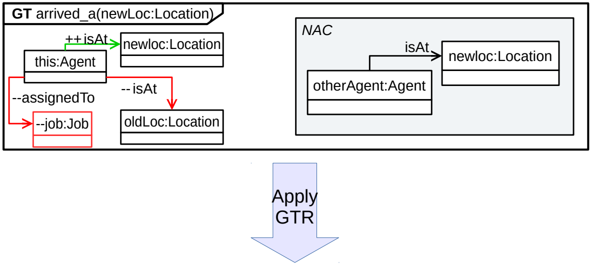


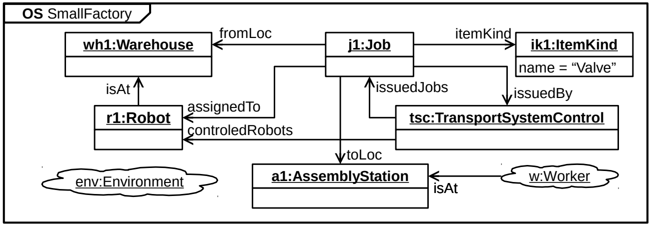
Graph Transformation Rules



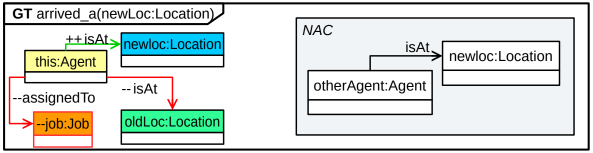
- Complex structural changes (as implementation of an operation)

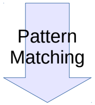
Applying Graph Transformation Rules

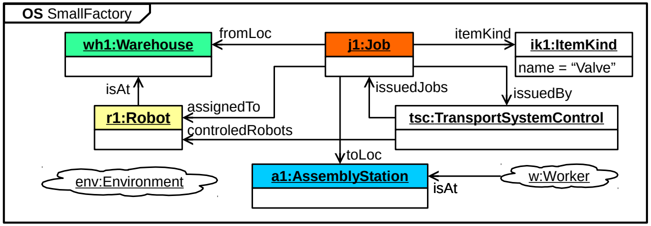




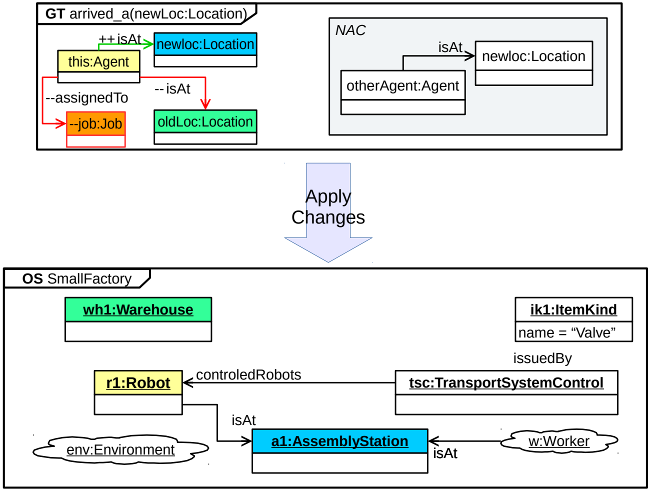
Applying Graph Transformation Rules



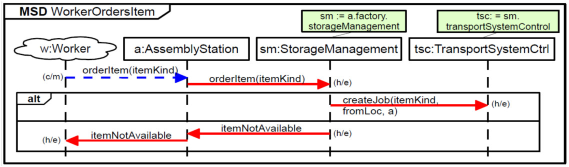




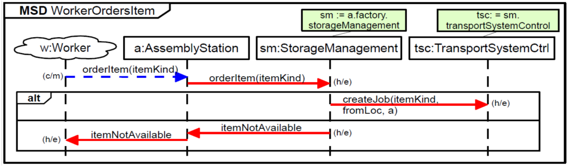
Applying Graph Transformation Rules

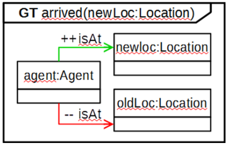


Integration?

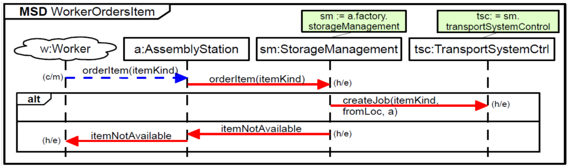


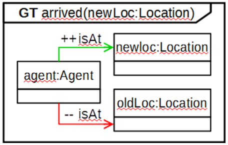
Integration?



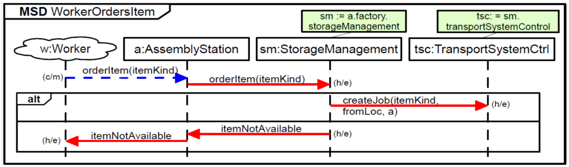


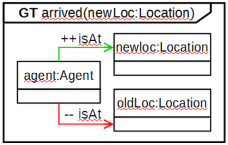
Integration?





Integration?





Concept of Integration

Integration by influence: MSDs and GTRs influence each other!

Concept of Integration

Integration by influence: MSDs and GTRs influence each other!

- Still use GTRs to describe implementations of operations
- GTRs are executed as side-effects of message events occurring during a system run
- Synchronous execution

$\Rightarrow$ **Events trigger GTRs**

Concept of Integration

Integration by influence: MSDs and GTRs influence each other!

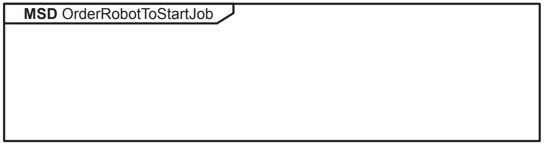
- Still use GTRs to describe implementations of operations
- GTRs are executed as side-effects of message events occurring during a system run
- Synchronous execution

$\Rightarrow$ **Events trigger GTRs**

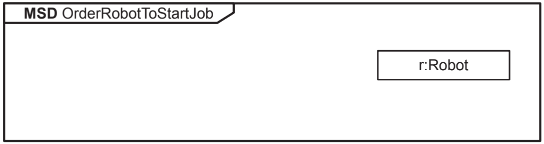
- GTRs constrain allowed sequences of events
- If precondition for applying a GTR is not satisfed

$\Rightarrow$ **An inexecutable GTR leads to a safety violation**

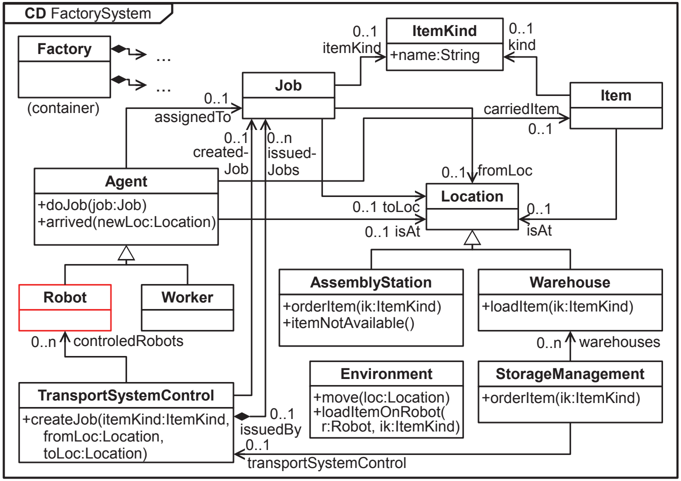
Integration by Example: Order a Robot to Start a Job



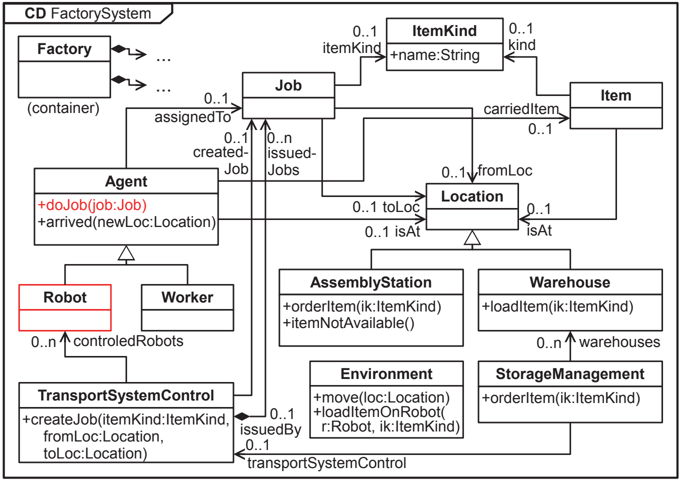
Integration by Example: Order a Robot to Start a Job



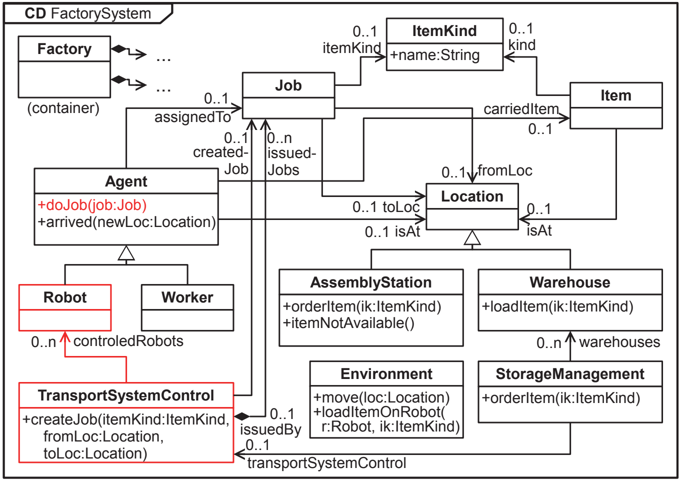
Integration by Example: Order a Robot to Start a Job



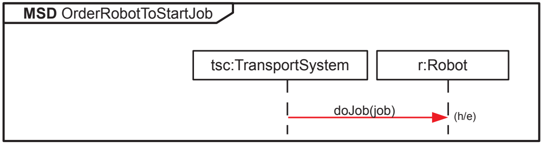
Integration by Example: Order a Robot to Start a Job



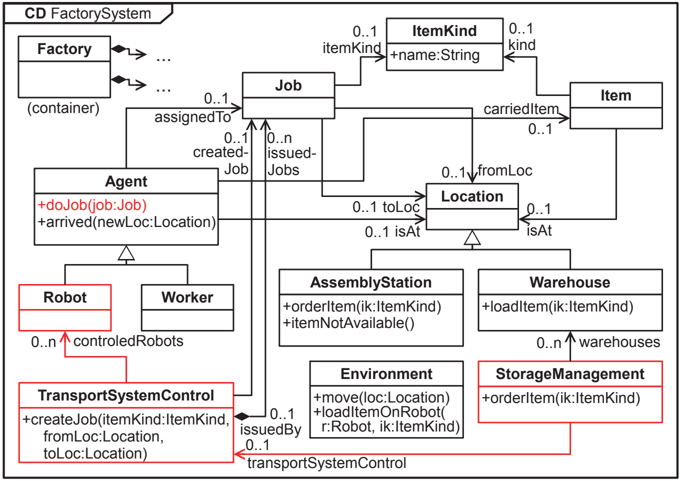
Integration by Example: Order a Robot to Start a Job



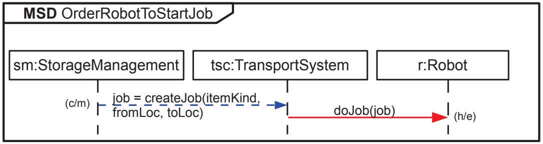
Integration by Example: Order a Robot to Start a Job



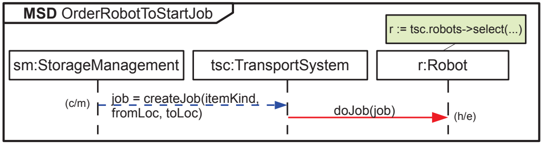
Integration by Example: Order a Robot to Start a Job



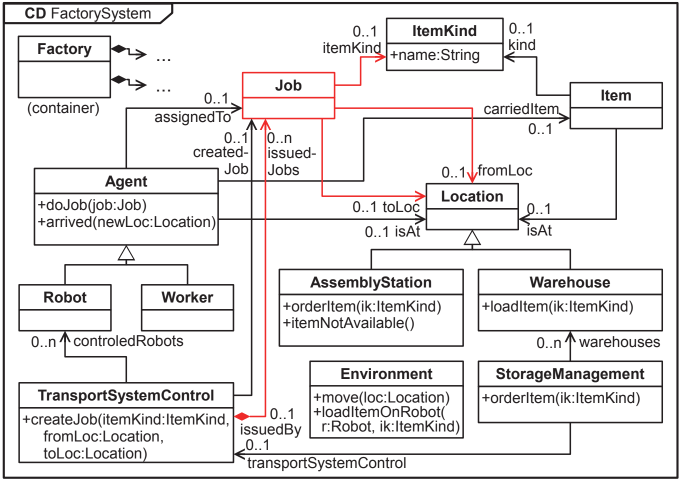
Integration by Example: Order a Robot to Start a Job



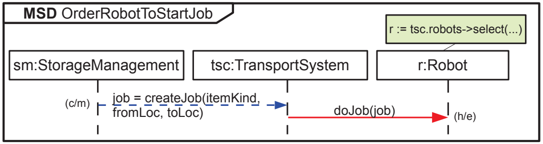
Integration by Example: Order a Robot to Start a Job

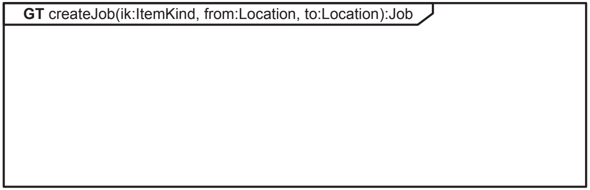


Integration by Example: Order a Robot to Start a Job

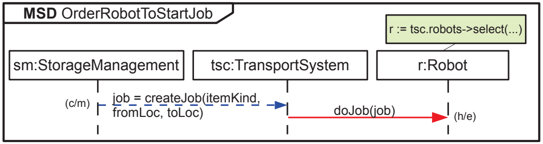


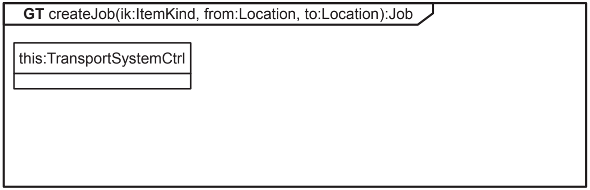
Integration by Example: Order a Robot to Start a Job



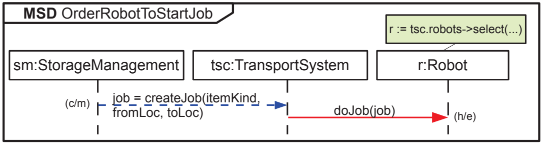


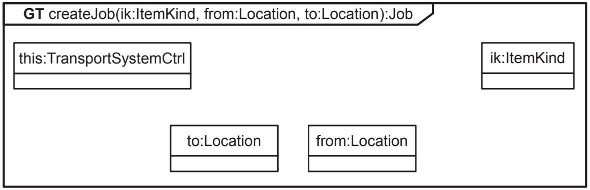
Integration by Example: Order a Robot to Start a Job



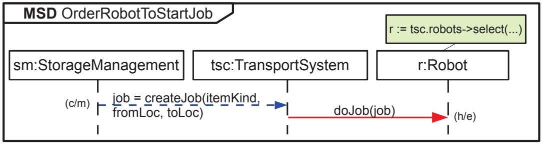


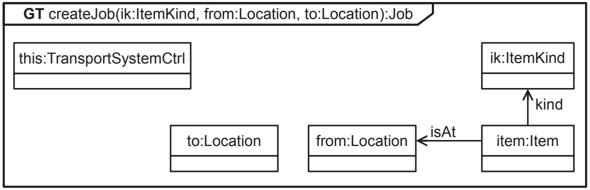
Integration by Example: Order a Robot to Start a Job



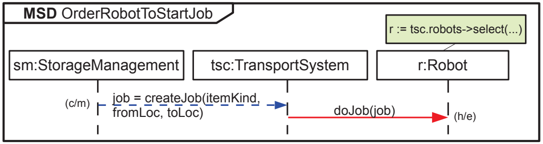


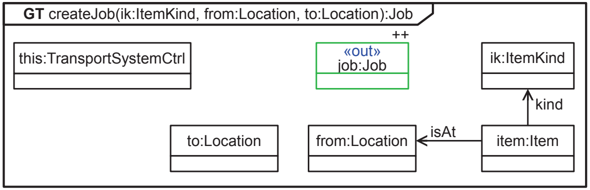
Integration by Example: Order a Robot to Start a Job



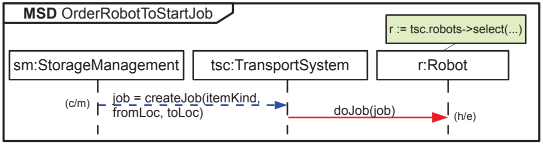


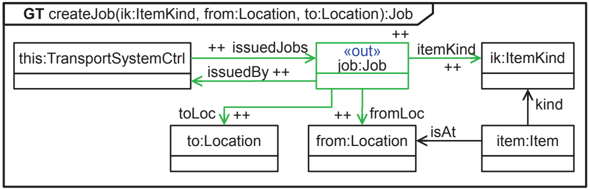
Integration by Example: Order a Robot to Start a Job



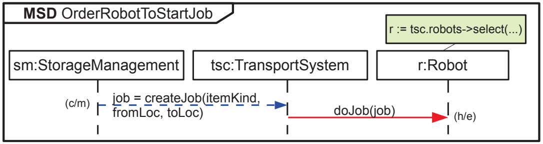


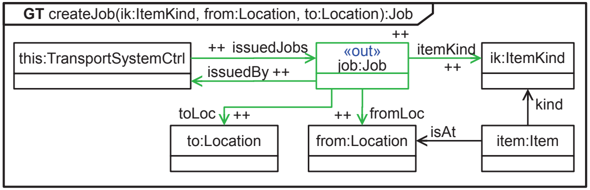
Integration by Example: Order a Robot to Start a Job

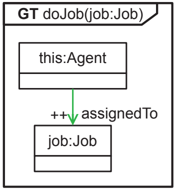




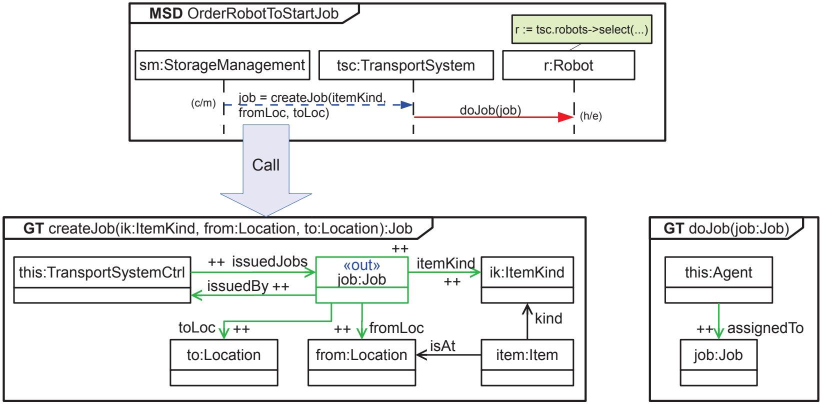
Integration by Example: Order a Robot to Start a Job



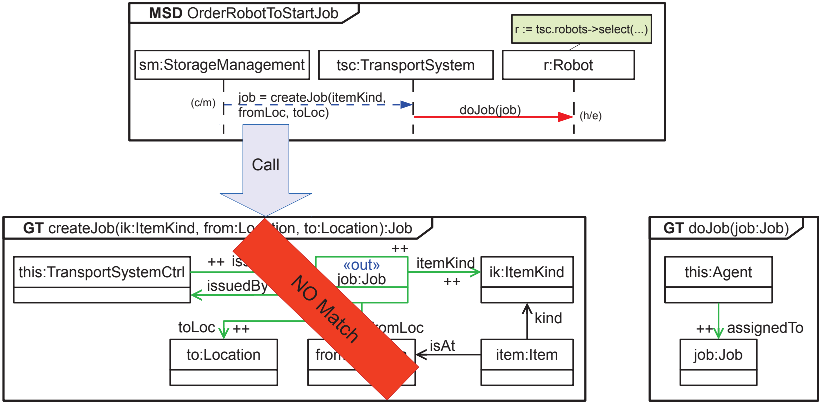




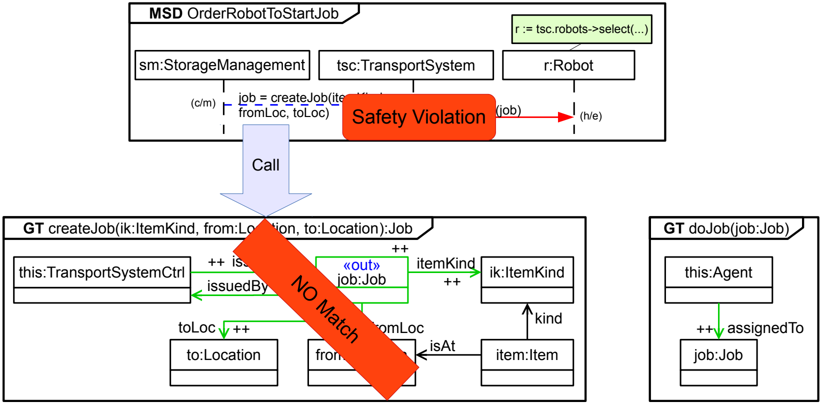
Integration by Example: Execution



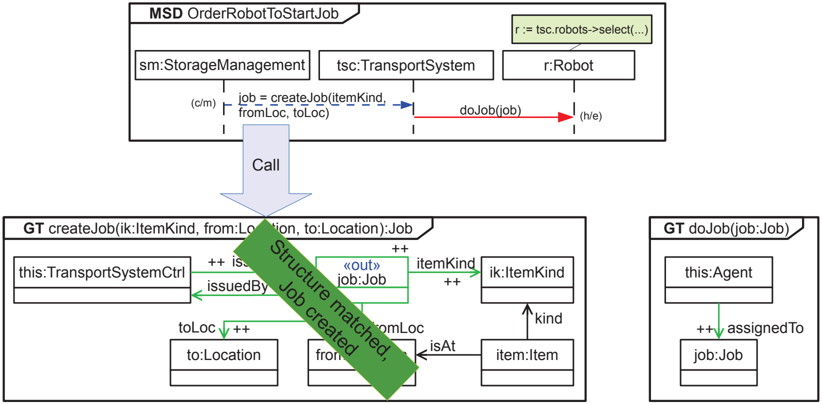
Integration by Example: Execution



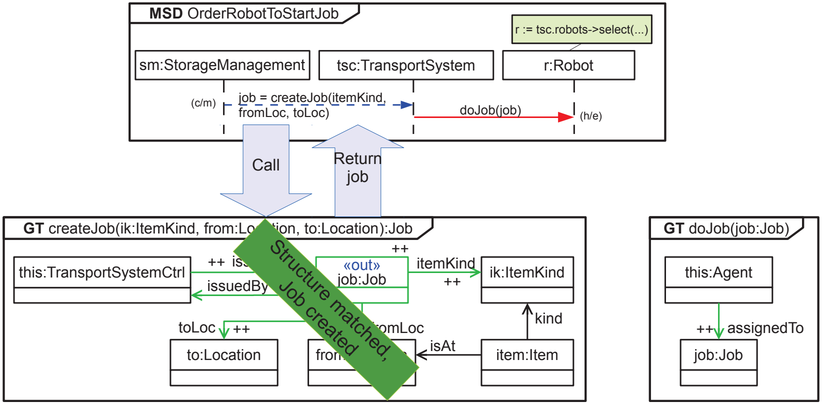
Integration by Example: Execution



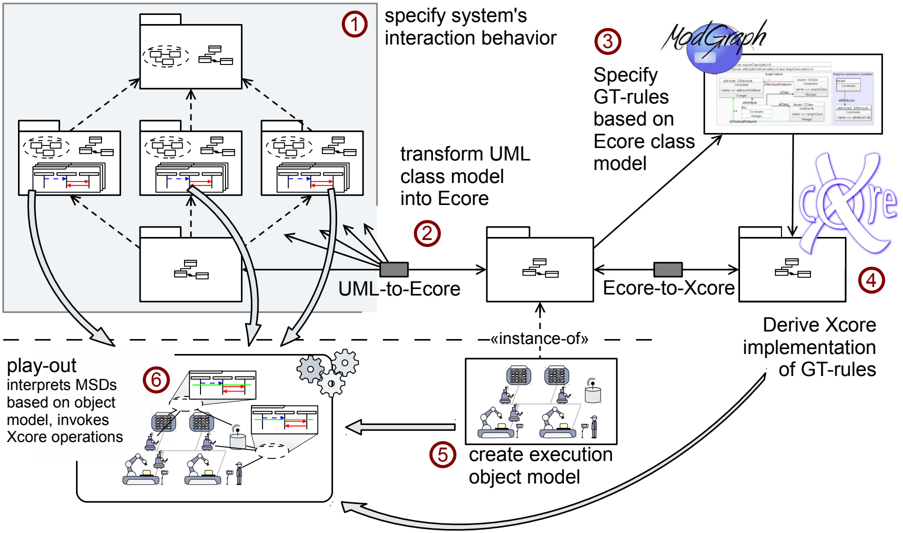
Integration by Example: Execution



Integration by Example: Execution



Tool Interaction


1
specify system's interaction behavior
2
transform UML class model into Ecore
UML-to-Ecore
3
Specify GT-rules based on Ecore class model
ModGraph
Xcore
4
Ecore-to-Xcore
Derive Xcore implementation of GT-rules
«instance-of»
5
create execution object model
6
play-out
interprets MSDs based on object model, invokes Xcore operations

Related Work

- MechatronicUML:
  - A design method for self-adaptive mechatronic systems
  - Interaction is defined by intra-component state machines

$\Leftrightarrow$ We use inter-component scenario models

Related Work

- MechatronicUML:
  - A design method for self-adaptive mechatronic systems
  - Interaction is defined by intra-component state machines

$\Leftrightarrow$ We use inter-component scenario models

- Turning collaboration diagram strips into storycharts
  - Use a set of simple graph transformation scenarios as input
  - All similar graph transformations are mapped to a common state in the state machine
  - No consideration that the graph transformations can change the object structure

$\Leftrightarrow$ The two-folded interaction of GTRs and MSDs is one of our main considerations

Related Work

- MechatronicUML:
  - A design method for self-adaptive mechatronic systems
  - Interaction is defined by intra-component state machines

  $\Leftrightarrow$ We use inter-component scenario models
- Turning collaboration diagram strips into storycharts
  - Use a set of simple graph transformation scenarios as input
  - All similar graph transformations are mapped to a common state in the state machine
  - No consideration that the graph transformations can change the object structure

  $\Leftrightarrow$ The two-folded interaction of GTRs and MSDs is one of our main considerations

None of the them rigorously supports the reconfiguration of the participating objects or components at run-time

Conclusion

Sound modeling and analysis approach integrating scenario-based specifications using MSDs with graph transformation:

Conclusion

Sound modeling and analysis approach integrating scenario-based specifications using MSDs with graph transformation:

- MSDs support an incremental refinement and extension of the message-based interaction behavior

Conclusion

Sound modeling and analysis approach integrating scenario-based specifications using MSDs with graph transformation:

- MSDs support an incremental refinement and extension of the message-based interaction behavior
- GTRs offer an easy to understand, declarative, pattern-oriented means for expressing structural change

Conclusion

Sound modeling and analysis approach integrating scenario-based specifications using MSDs with graph transformation:

- MSDs support an incremental refinement and extension of the message-based interaction behavior
- GTRs offer an easy to understand, declarative, pattern-oriented means for expressing structural change

$\Rightarrow$ Support for interactive and incremental specification of message-based interaction behavior and structural system reconfiguration behavior

$\Rightarrow$ **Integration of intuitive modeling paradigms, supported by prototype Tool.**

Thanks!

Thank you for your attention!

Questions