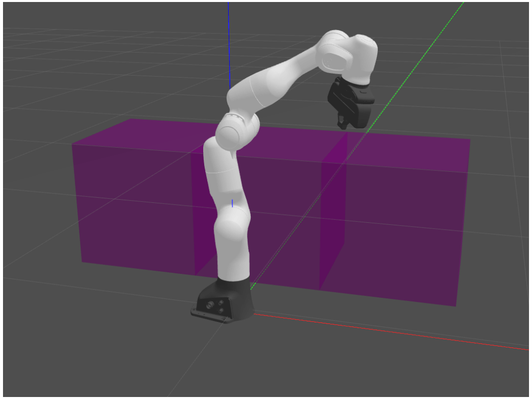
René Schöne, Johannes Mey, Sebastian Ebert, Uwe Aßmann

Connecting Conceptual Models using Relational Reference Attribute Grammars

connector.relational-rags.eu

October 16th 2020

Challenges when Designing Cyber-Physical Systems

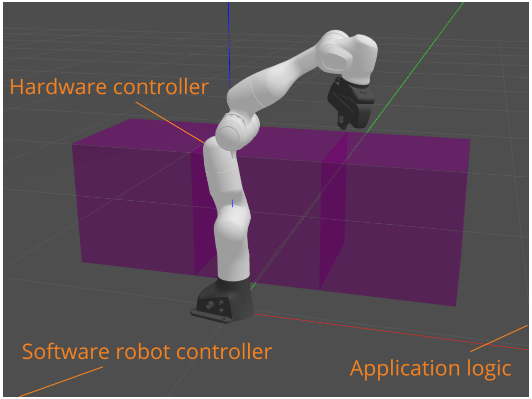


Distribution: transparent communication with locally and remotely accessible models

Multi-Paradigm: support for different paradigms and (programming as well as modelling) languages

Fast, reactive behaviour: changes in input lead to automatic re-computation for fast reaction

Challenges when Designing Cyber-Physical Systems

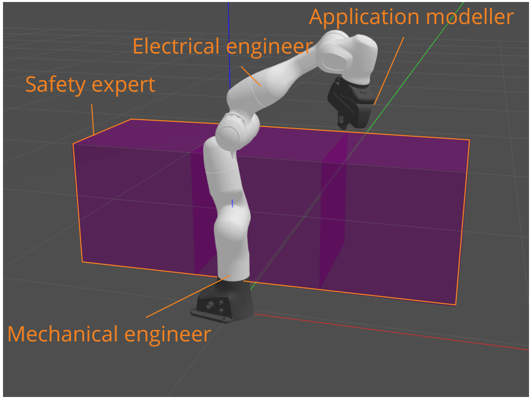


Distribution: transparent communication with locally and remotely accessible models

Multi-Paradigm: support for different paradigms and (programming as well as modelling) languages

Fast, reactive behaviour: changes in input lead to automatic re-computation for fast reaction

Challenges when Designing Cyber-Physical Systems

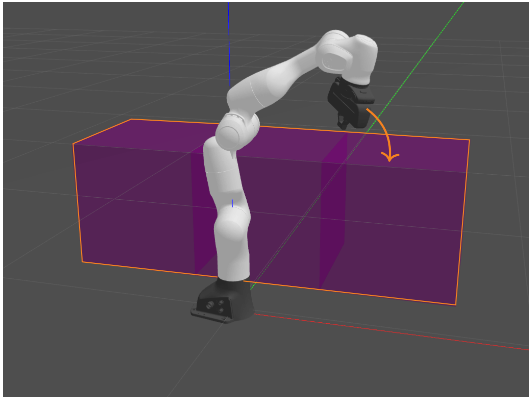


Distribution: transparent communication with locally and remotely accessible models

Multi-Paradigm: support for different paradigms and (programming as well as modelling) languages

Fast, reactive behaviour: changes in input lead to automatic re-computation for fast reaction

Challenges when Designing Cyber-Physical Systems

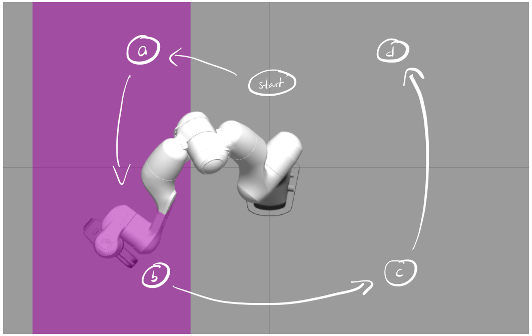


Distribution: transparent communication with locally and remotely accessible models

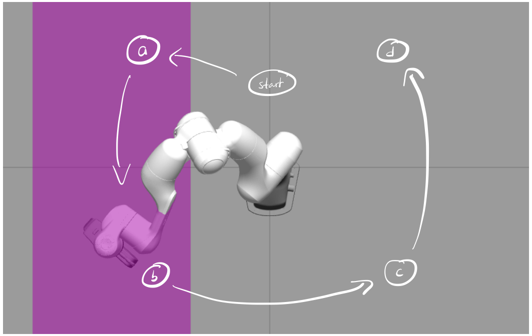
Multi-Paradigm: support for different paradigms and (programming as well as modelling) languages

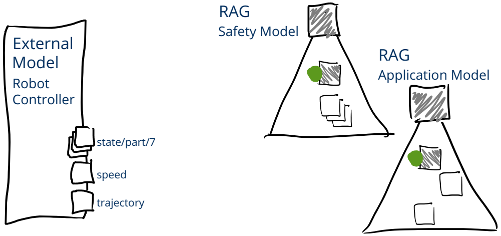
Fast, reactive behaviour: changes in input lead to automatic re-computation for fast reaction

Use Case: An Robotic Application

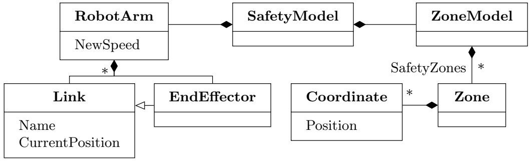


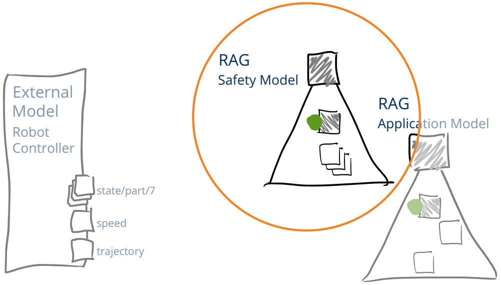
Use Case: An Robotic Application





Use Case: Relational RAG [Mey2020] Safety Model



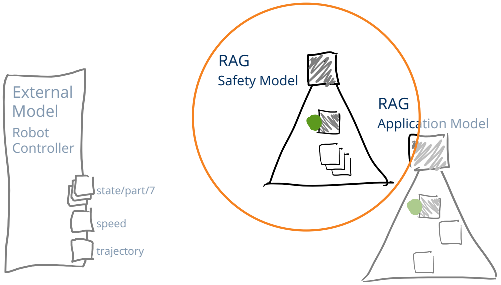


```
Model ::= RobotArm ZoneModel ;
ZoneModel ::= <Size:IntPosition> SafetyZone:Zone* ;
Zone ::= Coordinate* ;
RobotArm ::= Link* EndEffector /<NewSpeed:double>/ ;
Link ::= <Name:String> <CurrentPosition:IntPosition> ;
EndEffector : Link;
Coordinate ::= <Position:IntPosition> ;
```

[Mey2020] Johannes Mey, René Schöne, Görel Hedin, Emma Söderberg, Thomas Kühn, Niklas Fors, Jesper Öqvist, and Uwe Aßmann. Relational Reference Attribute Grammars: Improving Continuous Model Validation. Journal of Computer Languages (Jan. 2020). https://doi.org/10.1016/j.cola.2019.100940

Use Case: Relational RAG [Mey2020] Safety Model

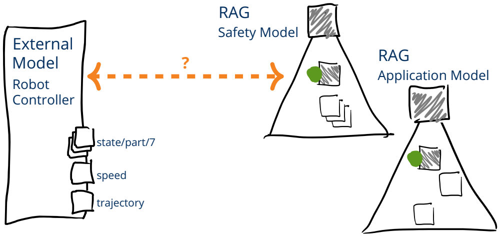
```
syn boolean RobotArm.isInSafetyZone() {
  for (Link link : getLinkList())
    if (model().getZoneModel()
        .isInSafetyZone(link.getCurrentPosition()))
      return true;
  return model().getZoneModel().isInSafetyZone(
    getEndEffector().getCurrentPosition());
}
syn boolean ZoneModel.isInSafetyZone(IntPosition pos) {
  for (Zone sz : getSafetyZoneList())
    for (Coordinate coordinate : sz.getCoordinateList())
      if (coordinate.getPosition().equals(pos))
        return true;
  return false;
}
syn double RobotArm.getNewSpeed() {
  return isInSafetyZone() ? LOW_SPEED : NORMAL_SPEED;
}
```



[Mey2020] Johannes Mey, René Schöne, Görel Hedin, Emma Söderberg, Thomas Kühn, Niklas Fors, Jesper Öqvist, and Uwe Aßmann. Relational Reference Attribute Grammars: Improving Continuous Model Validation. Journal of Computer Languages (Jan. 2020). https://doi.org/10.1016/j.cola.2019.100940

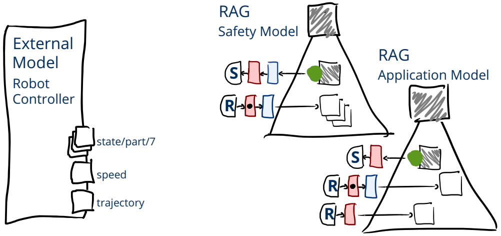
Use Case: Relational RAG [Mey2020] Safety Model

```
syn boolean RobotArm.isInSafetyZone() {
  for (Link link : getLinkList())
    if (model().getZoneModel()
          .isInSafetyZone(link.getCurrentPosition()))
      return true;
  return model().getZoneModel().isInSafetyZone(
    getEndEffector().getCurrentPosition());
}
syn boolean ZoneModel.isInSafetyZone(IntPosition pos) {
  for (Zone sz : getSafetyZoneList())
    for (Coordinate coordinate : sz.getCoordinateList())
      if (coordinate.getPosition().equals(pos))
        return true;
  return false;
}
syn double RobotArm.getNewSpeed() {
  return isInSafetyZone() ? LOW_SPEED : NORMAL_SPEED;
}
```

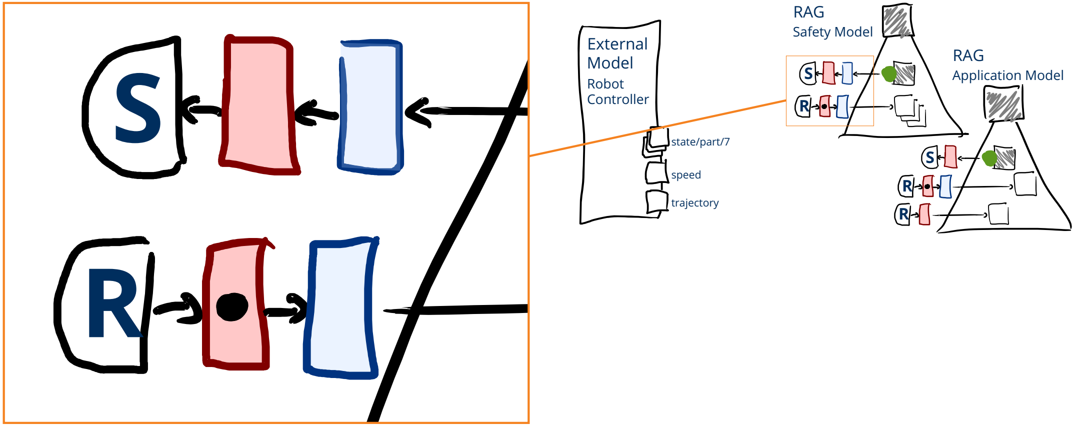


[Mey2020] Johannes Mey, René Schöne, Görel Hedin, Emma Söderberg, Thomas Kühn, Niklas Fors, Jesper Öqvist, and Uwe Aßmann. Relational Reference Attribute Grammars: Improving Continuous Model Validation. Journal of Computer Languages (Jan. 2020). https://doi.org/10.1016/j.cola.2019.100940

Idea: Explicit Specification of Connections



Idea: Explicit Specification of Connections



Solution: The DSL “RagConnect”

```
receive Link.CurrentPosition using ParseState, Transform;

  ParseState maps byte[] bytes to RobotState {:
    return RobotState.parseFrom(bytes);
  :}

 Transform maps RobotState rs to IntPosition {:
   RobotState.Position p = rs.getPosition();
   return IntPosition.of((int) (Math.round(p.getX() * 2)), (int) (Math.
     round(p.getY() * 2)), (int) (Math.round(p.getZ() * 2 - 0.5)));
 :}

send RobotArm.NewSpeed using
CreateSpeedMessage, SerializeRobotConfig;
```

Solution: The DSL “RagConnect”

```
receive Link.CurrentPosition using ParseState, Transform;

 ParseState maps byte[] bytes to RobotState {:
   return RobotState.parseFrom(bytes);
 :}

Transform maps RobotState rs to IntPosition {:
  RobotState.Position p = rs.getPosition();
  return IntPosition.of((int) (Math.round(p.getX() * 2)), (int) (Math.
    round(p.getY() * 2)), (int) (Math.round(p.getZ() * 2 - 0.5)));
:}

send RobotArm.NewSpeed using
CreateSpeedMessage, SerializeRobotConfig;
```

Solution: The DSL “RagConnect”

```
receive Link.CurrentPosition using ParseState, Transform;

  ParseState maps byte[] bytes to RobotState {:
    return RobotState.parseFrom(bytes);
  :}

 Transform maps RobotState rs to IntPosition {:
   RobotState.Position p = rs.getPosition();
   return IntPosition.of((int) (Math.round(p.getX() * 2)), (int) (Math.
     round(p.getY() * 2)), (int) (Math.round(p.getZ() * 2 - 0.5)));
 :}

send RobotArm.NewSpeed using
CreateSpeedMessage, SerializeRobotConfig;
```

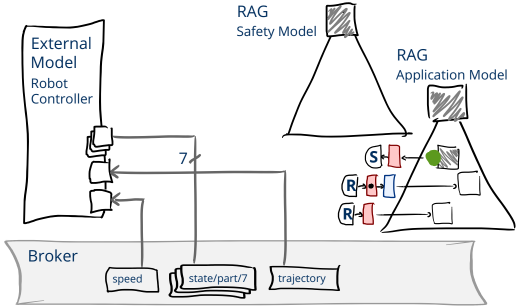
Solution: The DSL “RagConnect”

```
receive Link.CurrentPosition using ParseState, Transform;

  ParseState maps byte[] bytes to RobotState {:
    return RobotState.parseFrom(bytes);
  :}
Transform maps RobotState rs to IntPosition {:
  RobotState.Position p = rs.getPosition();
  return IntPosition.of((int) (Math.round(p.getX() * 2)), (int) (Math.
    round(p.getY() * 2)), (int) (Math.round(p.getZ() * 2 - 0.5)));
:}
send RobotArm.NewSpeed using
CreateSpeedMessage, SerializeRobotConfig;
```

shared by both models

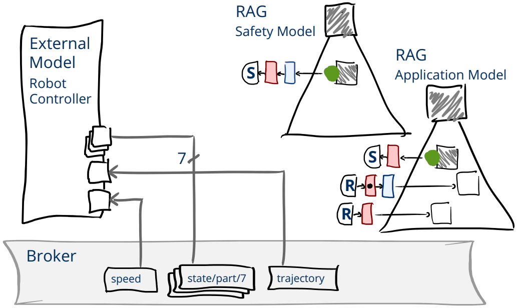
Using the Generated API



```
RobotArm robotArm = ...;
Link link1 = ...;
robotArm.addLink(link1);

link1.connectCurrentPosition("state/part/7");
robotArm.connectNewSpeed("robot/speed", true);
```

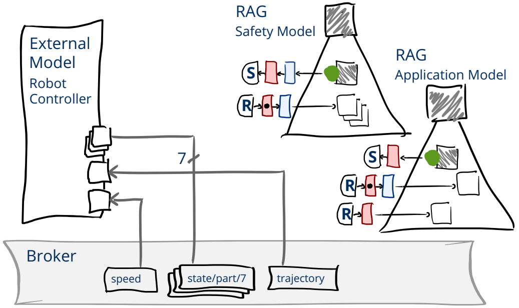
Using the Generated API



```
RobotArm robotArm = ...;
Link link1 = ...;
robotArm.addLink(link1);

link1.connectCurrentPosition("state/part/7");
robotArm.connectNewSpeed("robot/speed", true);
```

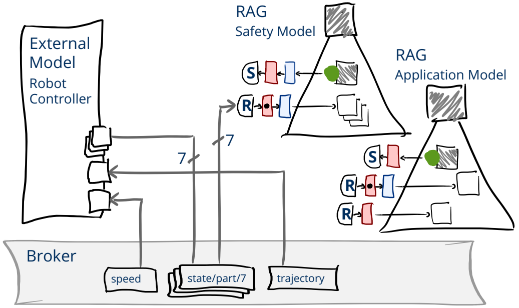
Using the Generated API



```
RobotArm robotArm = ...;
Link link1 = ...;
robotArm.addLink(link1);

link1.connectCurrentPosition("state/part/7");
robotArm.connectNewSpeed("robot/speed", true);
```

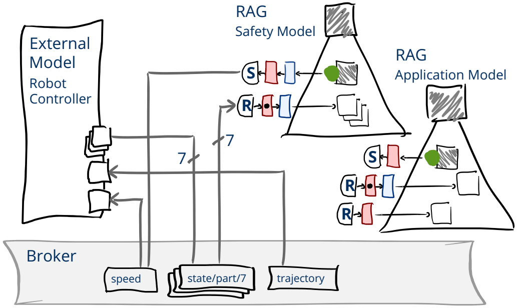
Using the Generated API



```
RobotArm robotArm = ...;
Link link1 = ...;
robotArm.addLink(link1);

link1.connectCurrentPosition("state/part/7");
robotArm.connectNewSpeed("robot/speed", true);
```

Using the Generated API

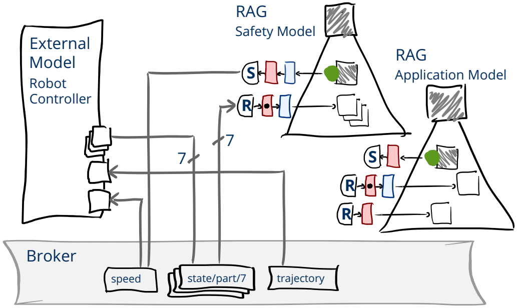


```
RobotArm robotArm = ...;
Link link1 = ...;
robotArm.addLink(link1);

link1.connectCurrentPosition("state/part/7");
robotArm.connectNewSpeed("robot/speed", true);
```

send current value immediately

Using the Generated API

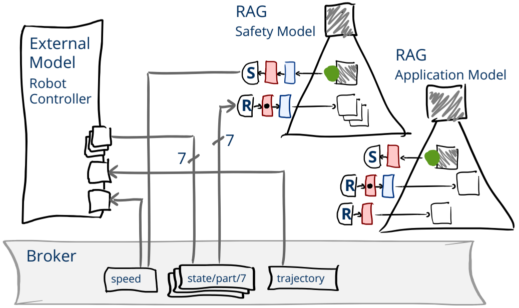


```
RobotArm robotArm = ...;
Link link1 = ...;
robotArm.addLink(link1);

link1.connectCurrentPosition("state/part/7");
robotArm.connectNewSpeed("robot/speed", true);
```

send current value immediately

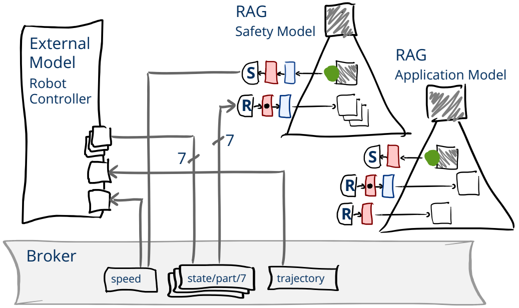
Using the Generated API



```
RobotArm robotArm = ...;
Link link1 = ...;
robotArm.addLink(link1);
robotArm.addDependency1(link1);
link1.connectCurrentPosition("state/part/7");
robotArm.connectNewSpeed("robot/speed", true);
```

send current value immediately

Using the Generated API



not necessary in future versions

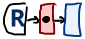
```
RobotArm robotArm = ...;
Link link1 = ...;
robotArm.addLink(link1);
robotArm.addDependency1(link1);
link1.connectCurrentPosition("state/part/7");
robotArm.connectNewSpeed("robot/speed", true);
```

send current value immediately

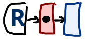
Inner Workings

Robot Controller				`isInSafetyZone?`		**Message**
(0.01, 0.03, 0.02)	$\rightarrow$	(0, 0, 0)	$\rightarrow$	`false`	$\rightarrow$	`speed = NORMAL`
(0.02, 0.04, 0.06)	$\rightarrow$	(0, 0, 0)	.			
⋮						
(0.02, 0.50, 0.06)	$\rightarrow$	(0, 1, 0)	$\rightarrow$	`false`	.	
(0.02, 0.52, 0.06)	$\rightarrow$	(0, 1, 0)	.			
⋮						
(1.33, 1.00, 0.73)	$\rightarrow$	(3, 2, 1)	$\rightarrow$	`true`	$\rightarrow$	`speed = LOW`
$\underbrace{\qquad}_{38\,000}$		$\underbrace{\qquad}_{54}$				$\underbrace{\qquad}_{6}$

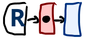
Inner Workings

Robot Controller				`isInSafetyZone?`		Message
(0.01, 0.03, 0.02)	→	(0, 0, 0)	→	`false`	→	`speed = NORMAL`
(0.02, 0.04, 0.06)	→	(0, 0, 0)	.			
⋮						
(0.02, 0.50, 0.06)	→	(0, 1, 0)	→	`false`	.	
(0.02, 0.52, 0.06)	→	(0, 1, 0)	.			
⋮						
(1.33, 1.00, 0.73)	→	(3, 2, 1)	→	`true`	→	`speed = LOW`
38 000		54				6

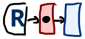
Inner Workings

Robot Controller				isInSafetyZone?		Message
(0.01, 0.03, 0.02)	→	(0, 0, 0)	→	`false`	→	`speed = NORMAL`
(0.02, 0.04, 0.06)	→	(0, 0, 0)	.			
⋮						
(0.02, 0.50, 0.06)	→	(0, 1, 0)	→	`false`	.	
(0.02, 0.52, 0.06)	→	(0, 1, 0)	.			
⋮						
(1.33, 1.00, 0.73)	→	(3, 2, 1)	→	`true`	→	`speed = LOW`
38 000				54		6

Inner Workings

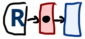
Robot Controller				`isInSafetyZone?`		Message
(0.01, 0.03, 0.02)	→	(0, 0, 0)	→	`false`	→	`speed = NORMAL`
(0.02, 0.04, 0.06)	→	(0, 0, 0)	.			
⋮						
(0.02, 0.50, 0.06)	→	(0, 1, 0)	→	`false`	.	
(0.02, 0.52, 0.06)	→	(0, 1, 0)	.			
⋮						
(1.33, 1.00, 0.73)	→	(3, 2, 1)	→	`true`	→	`speed = LOW`
$\underbrace{\qquad\qquad}_{38\,000}$		$\underbrace{\qquad\qquad\qquad\qquad}_{54}$				$\underbrace{\qquad\qquad}_{6}$

Inner Workings

Robot Controller				`isInSafetyZone?`		Message
(0.01, 0.03, 0.02)	→	(0, 0, 0)	→	`false`	→	`speed = NORMAL`
(0.02, 0.04, 0.06)	→	(0, 0, 0)	.			
⋮						
(0.02, 0.50, 0.06)	→	(0, 1, 0)	→	`false`	.	
(0.02, 0.52, 0.06)	→	(0, 1, 0)	.			
⋮						
(1.33, 1.00, 0.73)	→	(3, 2, 1)	→	`true`	→	`speed = LOW`

$\underbrace{\qquad\qquad}_{38\,000}$ $\underbrace{\qquad\qquad\qquad\qquad}_{54}$ $\underbrace{\qquad\qquad}_{6}$

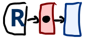
Inner Workings

Robot Controller				isInSafetyZone?		Message
(0.01, 0.03, 0.02)	→	(0, 0, 0)	→	`false`	→	`speed = NORMAL`
(0.02, 0.04, 0.06)	→	(0, 0, 0)	.			
⋮						
(0.02, 0.50, 0.06)	→	(0, 1, 0)	→	`false`	.	
(0.02, 0.52, 0.06)	→	(0, 1, 0)	.			
⋮						
(1.33, 1.00, 0.73)	→	(3, 2, 1)	→	`true`	→	`speed = LOW`
38 000			54			6

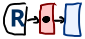
Inner Workings

Robot Controller				`isInSafetyZone?`		**Message**
(0.01, 0.03, 0.02)	→	(0, 0, 0)	→	`false`	→	`speed = NORMAL`
(0.02, 0.04, 0.06)	→	(0, 0, 0)	.			
⋮						
(0.02, 0.50, 0.06)	→	(0, 1, 0)	→	`false`	.	
(0.02, 0.52, 0.06)	→	(0, 1, 0)	.			
⋮						
(1.33, 1.00, 0.73)	→	(3, 2, 1)	→	`true`	→	`speed = LOW`
38 000				54		6

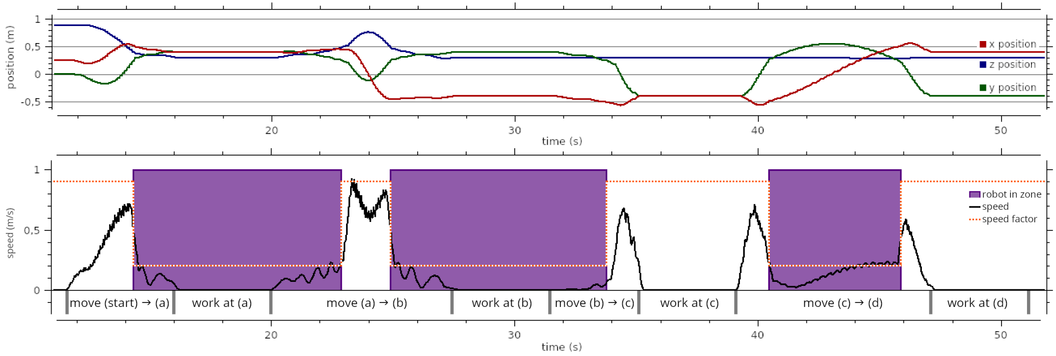
Inner Workings

Robot Controller				`isInSafetyZone?`		Message
$(0.01, 0.03, 0.02)$	$\rightarrow$	$(0, 0, 0)$	$\rightarrow$	`false`	$\rightarrow$	`speed = NORMAL`
$(0.02, 0.04, 0.06)$	$\rightarrow$	$(0, 0, 0)$	.			
$\vdots$						
$(0.02, 0.50, 0.06)$	$\rightarrow$	$(0, 1, 0)$	$\rightarrow$	`false`	.	
$(0.02, 0.52, 0.06)$	$\rightarrow$	$(0, 1, 0)$	.			
$\vdots$						
$(1.33, 1.00, 0.73)$	$\rightarrow$	$(3, 2, 1)$	$\rightarrow$	`true`	$\rightarrow$	`speed = LOW`
38 000		54				6

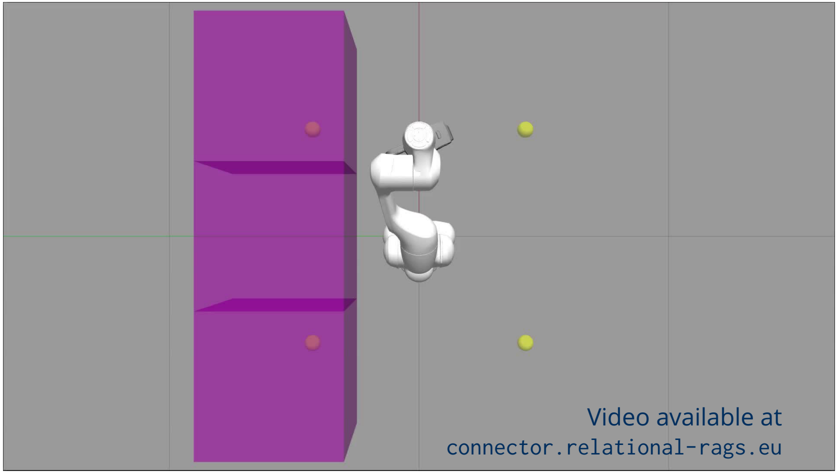
Inner Workings

Robot Controller				`isInSafetyZone?`		Message
(0.01, 0.03, 0.02)	→	(0, 0, 0)	→	`false`	→	`speed = NORMAL`
(0.02, 0.04, 0.06)	→	(0, 0, 0)	.			
⋮						
(0.02, 0.50, 0.06)	→	(0, 1, 0)	→	`false`	.	
(0.02, 0.52, 0.06)	→	(0, 1, 0)	.			
⋮						
(1.33, 1.00, 0.73)	→	(3, 2, 1)	→	`true`	→	`speed = LOW`
38 000		54				6

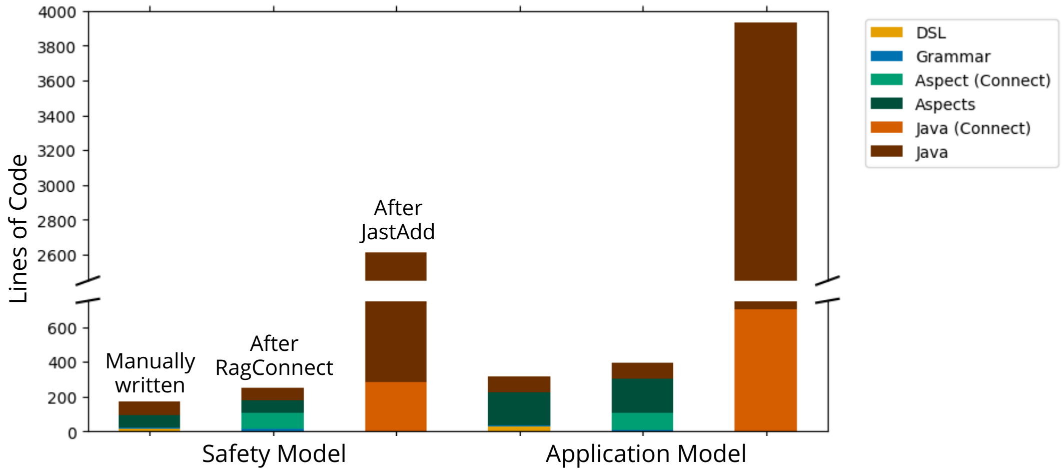
Inner Workings (Evaluation)



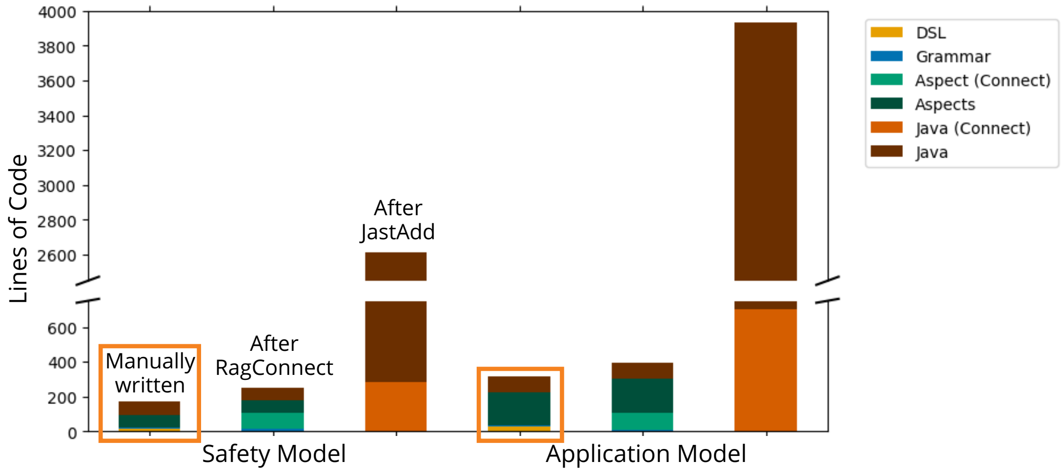
Inner Workings (Evaluation)



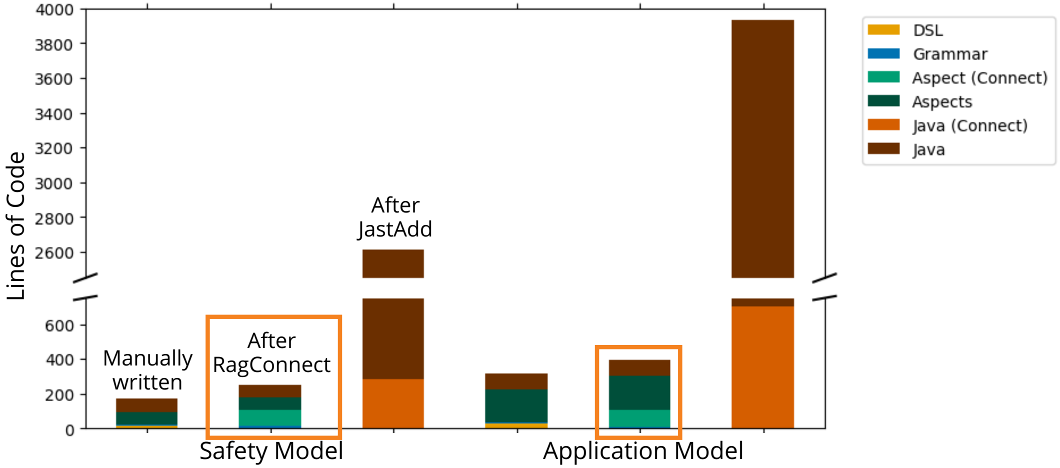
Minimize Development Effort



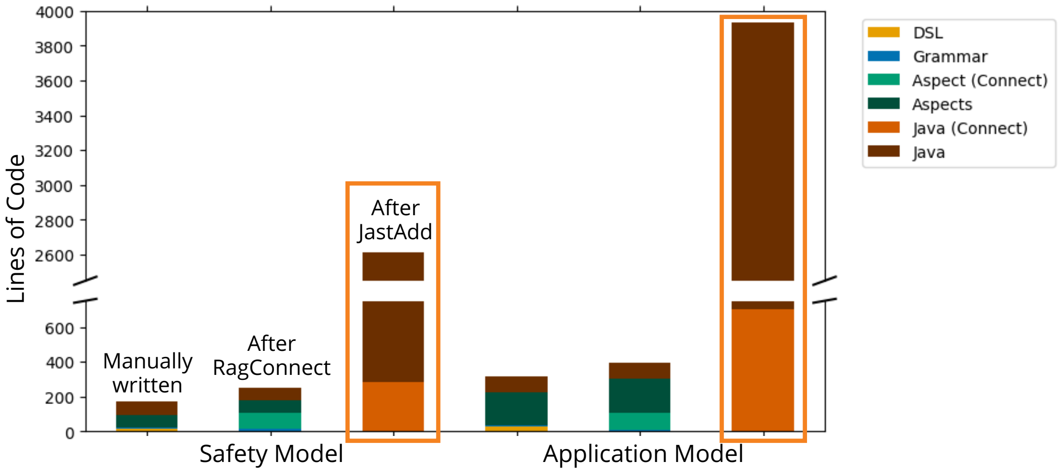
Minimize Development Effort



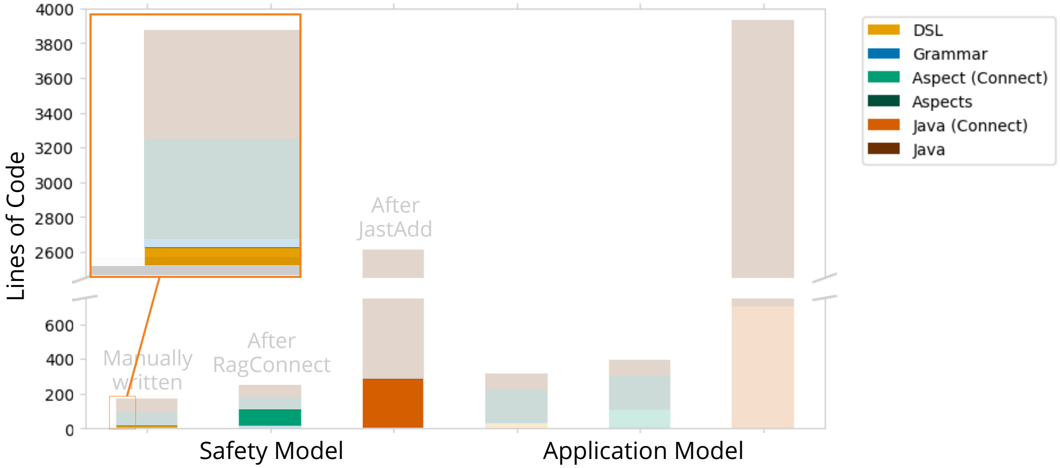
Minimize Development Effort



Minimize Development Effort

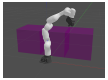


Minimize Development Effort



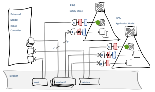
Summary and Future Work

Problem: Constructing cyber-physical systems is difficult
Challenges: Distribution • Multi-Paradigm • Fast, reactive behaviour

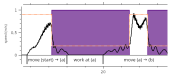


Solution: Generation of model connectors

```
receive Link.CurrentPosition using ParseState, Transform;
```



Use-Case: Robot with workflow and safety model
- Development: 18 (28) DSL-code → 281 (701) Java-code
- Runtime: 38 000 position updates → 54 re-computation → 6 speed messages



Future Work:
- Remove unnecessary dependency definitions
- Support additional communication protocols
- Update complex parts of model (instead of only tokens)

René Schöne, Johannes Mey, Sebastian Ebert, Uwe Aßmann

Connecting Conceptual Models using Relational Reference Attribute Grammars

connector.relational-rags.eu

October 16th 2020