Vienna University of Technology

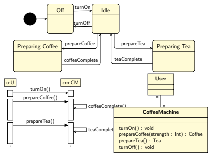
Towards a Uniform Framework to Support the Evolution of Software Models

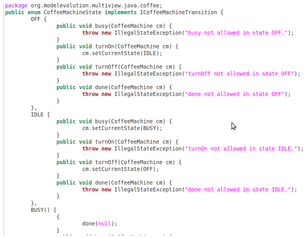
Magdalena Widl

Business Informatics Group
Vienna University of Technology

Motivation and scope

Model-driven engineering

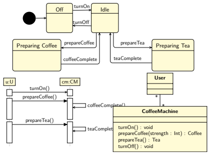


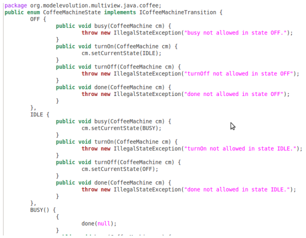


```
package org.modelevolution.multiview.java.coffee;
public enum CoffeeMachineState implements ICoffeeMachineTransition {
        OFF {
                public void busy(CoffeeMachine cm) {
                        throw new IllegalStateException("busy not allowed in state OFF.");
                }
                public void turnOn(CoffeeMachine cm) {
                        cm.setCurrentState(IDLE);
                }
                public void turnOff(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOff not allowed in state OFF");
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state OFF");
                }
        },
        IDLE {
                public void busy(CoffeeMachine cm) {
                        cm.setCurrentState(BUSY);
                }
                public void turnOn(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOn not allowed in state IDLE.");
                }
                public void turnOff(CoffeeMachine cm) {
                        cm.setCurrentState(OFF);
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state IDLE.");
                }
        },
        BUSY() {
                {
                        done(null);
                }
```

Motivation and scope

Model-driven engineering



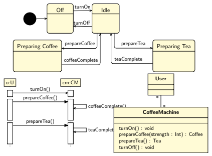


```
package org.modelevolution.multiview.java.coffee;
public enum CoffeeMachineState implements ICoffeeMachineTransition {
        OFF {
                public void busy(CoffeeMachine cm) {
                        throw new IllegalStateException("busy not allowed in state OFF.");
                }
                public void turnOn(CoffeeMachine cm) {
                        cm.setCurrentState(IDLE);
                }
                public void turnOff(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOff not allowed in state OFF");
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state OFF");
                }
        },
        IDLE {
                public void busy(CoffeeMachine cm) {
                        cm.setCurrentState(BUSY);
                }
                public void turnOn(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOn not allowed in state IDLE.");
                }
                public void turnOff(CoffeeMachine cm) {
                        cm.setCurrentState(OFF);
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state IDLE.");
                }
        },
        BUSY() {
                {
                        done(null);
                }
```

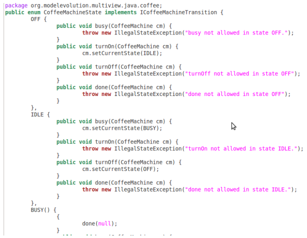
Software verification, testing

Motivation and scope

Model-driven engineering



$\dashrightarrow$



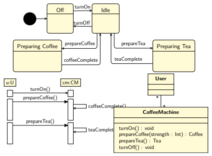
```
package org.modelevolution.multiview.java.coffee;
public enum CoffeeMachineState implements ICoffeeMachineTransition {
        OFF {
                public void busy(CoffeeMachine cm) {
                        throw new IllegalStateException("busy not allowed in state OFF.");
                }
                public void turnOn(CoffeeMachine cm) {
                        cm.setCurrentState(IDLE);
                }
                public void turnOff(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOff not allowed in state OFF");
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state OFF");
                }
        },
        IDLE {
                public void busy(CoffeeMachine cm) {
                        cm.setCurrentState(BUSY);
                }
                public void turnOn(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOn not allowed in state IDLE.");
                }
                public void turnOff(CoffeeMachine cm) {
                        cm.setCurrentState(OFF);
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state IDLE.");
                }
        },
        BUSY() {
                {
                        done(null);
                }
```

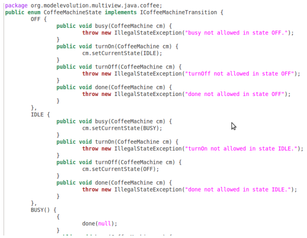
Model verification?

Software verification, testing

Motivation and scope

Model-driven engineering




```
package org.modelevolution.multiview.java.coffee;
public enum CoffeeMachineState implements ICoffeeMachineTransition {
        OFF {
                public void busy(CoffeeMachine cm) {
                        throw new IllegalStateException("busy not allowed in state OFF.");
                }
                public void turnOn(CoffeeMachine cm) {
                        cm.setCurrentState(IDLE);
                }
                public void turnOff(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOff not allowed in state OFF");
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state OFF");
                }
        },
        IDLE {
                public void busy(CoffeeMachine cm) {
                        cm.setCurrentState(BUSY);
                }
                public void turnOn(CoffeeMachine cm) {
                        throw new IllegalStateException("turnOn not allowed in state IDLE.");
                }
                public void turnOff(CoffeeMachine cm) {
                        cm.setCurrentState(OFF);
                }
                public void done(CoffeeMachine cm) {
                        throw new IllegalStateException("done not allowed in state IDLE.");
                }
        },
        BUSY() {
                {
                        done(null);
                }
```

$$\dashrightarrow$$

Model verification?

Software verification, testing

Idea

Find errors on *model* level

Then work only on *consistent* models

Motivation and scope

Inconsistencies

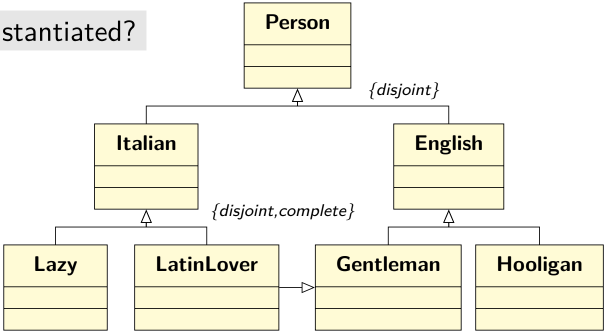
Static diagrams **Dynamic diagrams** **Both**

Motivation and scope

Inconsistencies

Static diagrams Dynamic diagrams Both

Can this be instantiated?



example from *Calvanese et al., slides from ESSLI Summer School 2003, Vienna*

Motivation and scope

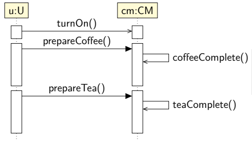
Inconsistencies

Static diagrams **Dynamic diagrams** **Both**

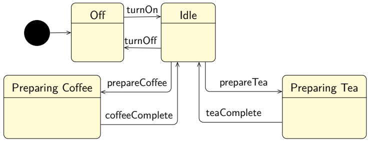
Motivation and scope

Inconsistencies

Static diagrams **Dynamic diagrams** Both



implemented by



?

Motivation and scope

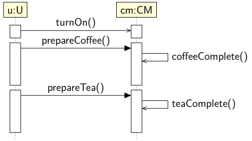
Inconsistencies

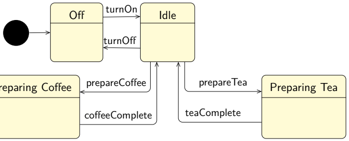
Static diagrams Dynamic diagrams **Both**

Motivation and scope

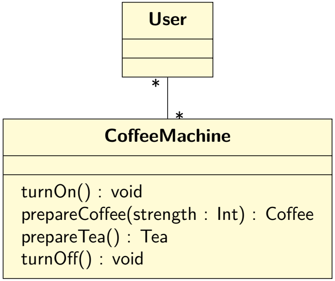
Inconsistencies

Static diagrams Dynamic diagrams **Both**





defined by



?

Motivation and scope

Model evolution

Evolution is multi-view, multidimensional

May introduce *inconsistencies*

Different evolution tasks may introduce different types of inconsistencies

Motivation and scope

Model evolution

Evolution is multi-view, multidimensional

May introduce *inconsistencies*

Different evolution tasks may introduce different types of inconsistencies

Idea

Establish a classification of changes in models and find which inconsistencies they may cause

Motivation and scope

We consider a *multi-view subset* of UML relevant for MDE:

- State machines
- Sequence diagrams
- Class diagrams

Motivation and scope

We consider a *multi-view subset* of UML relevant for MDE:

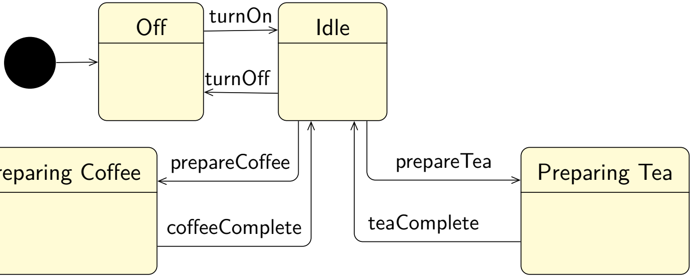
- State machines
- Sequence diagrams
- Class diagrams

Many complex constructs are omitted, but will be gradually added.

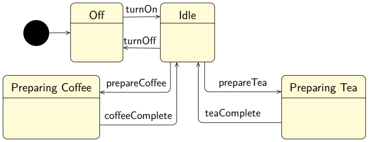
Example

Example

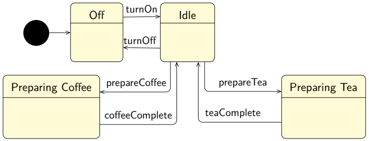
Example

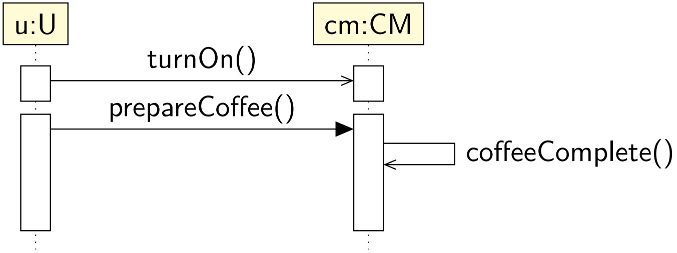

Off
Idle
turnOn
turnOff
Preparing Coffee
Preparing Tea
prepareCoffee
coffeeComplete
prepareTea
teaComplete

Example

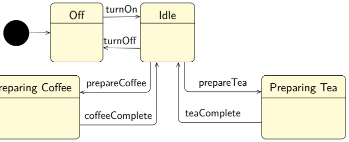

Off
Idle
turnOn
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

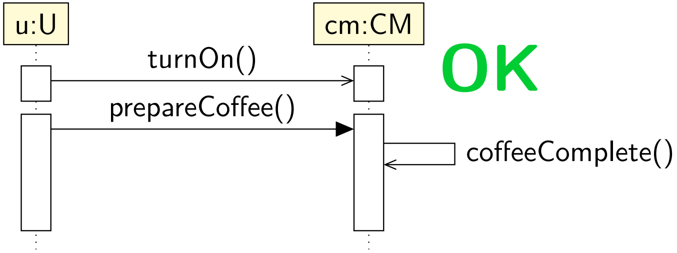
Example


Off
Idle
turnOn
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea


u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()

Example

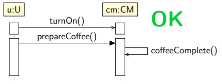

Off
Idle
turnOn
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
Preparing Tea
prepareTea
teaComplete


u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()

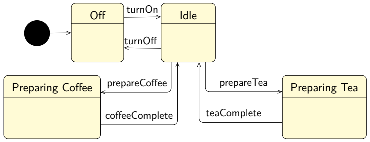
OK

Example

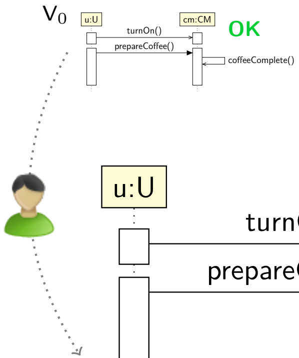
V_0

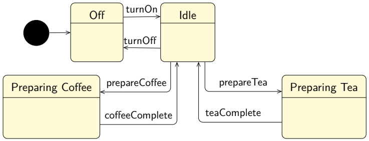


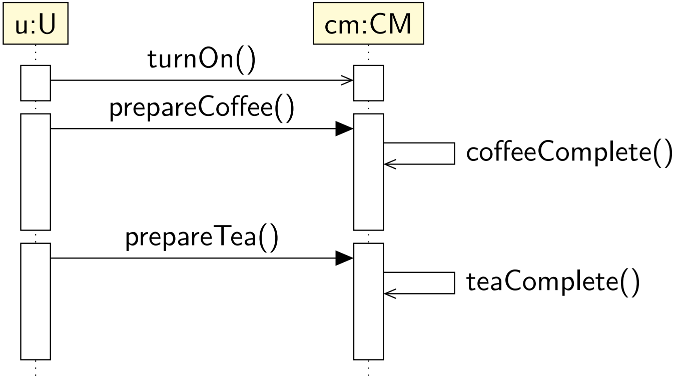
OK



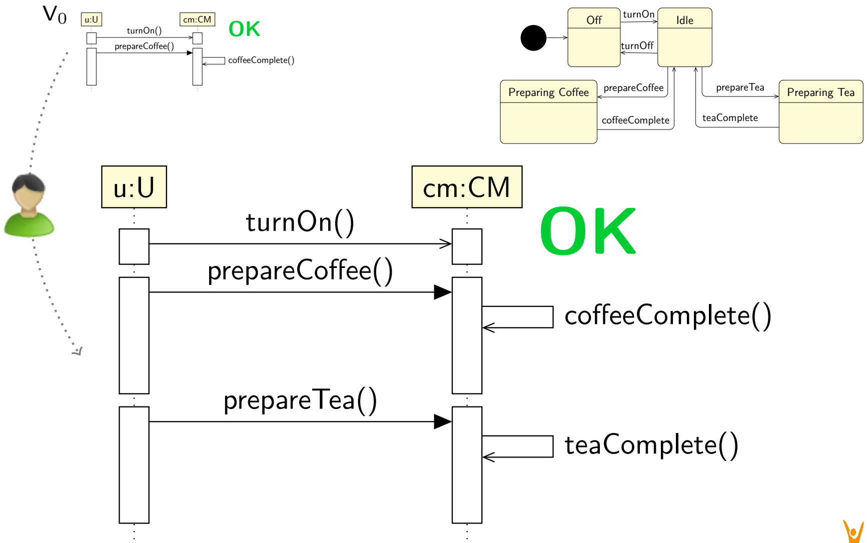
Example


V_0
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
OK

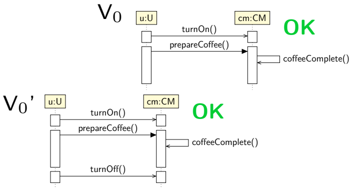

Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

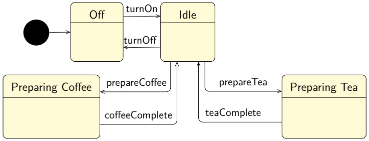

u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()

Example

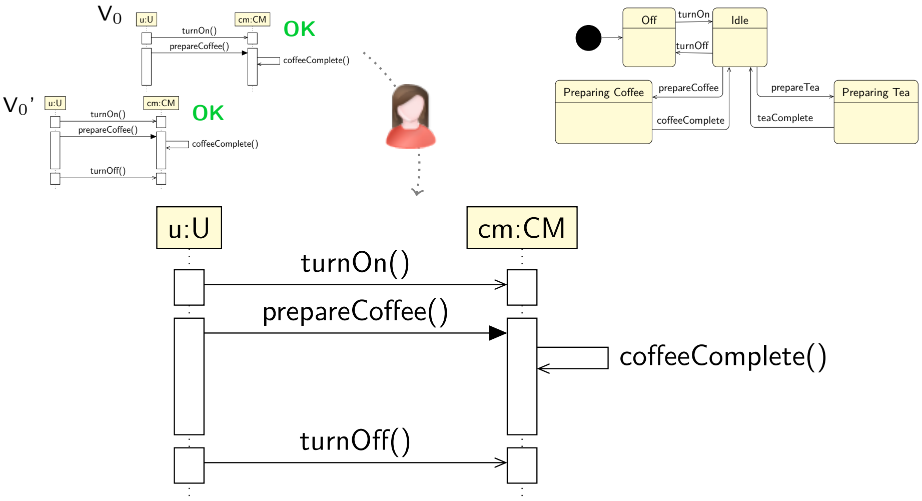

V_0
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
OK
Off
Idle
turnOn
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()
OK

Example

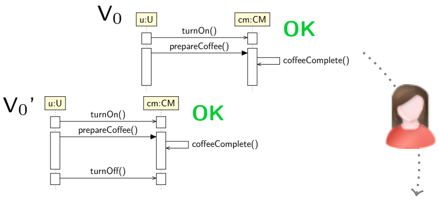

V_0
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
OK
V_0'
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
OK

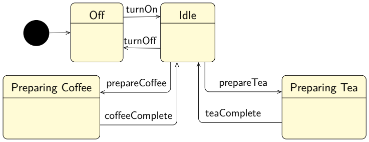

Off
Idle
turnOn
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
Preparing Tea
prepareTea
teaComplete

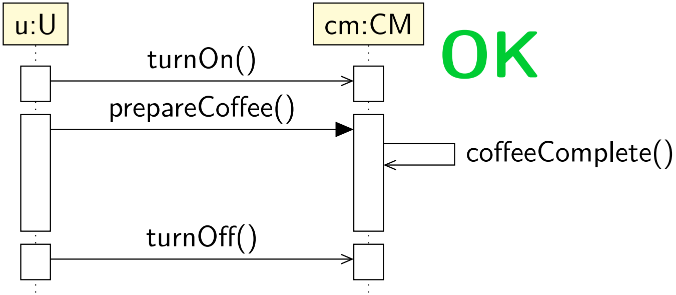
Example


V_0
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
OK
V_0'
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
OK
Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()

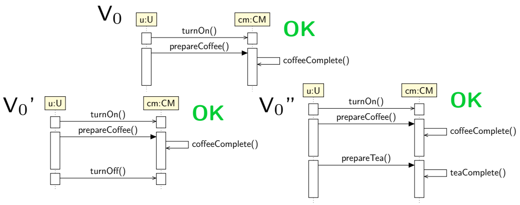
Example

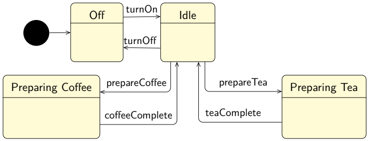

V0
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
OK
V0'
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
OK


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea


u:U
cm:CM
OK
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()

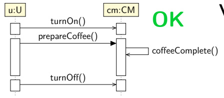
Example


V_0
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
OK
V_0'
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
OK
V_0''
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()
OK


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

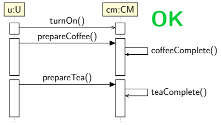
Example

V_0'

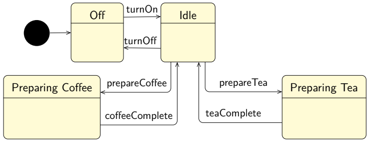


OK

V_0''

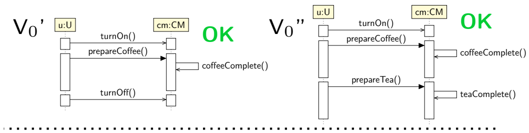


OK

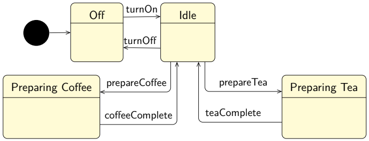


V_1?

Example

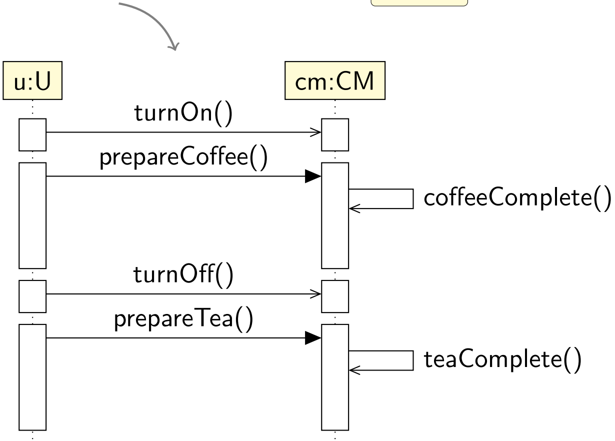


V_0'
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
OK
V_0''
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()
OK



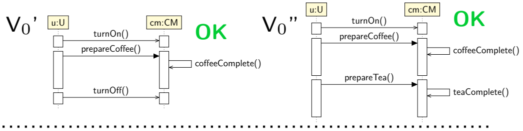
Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

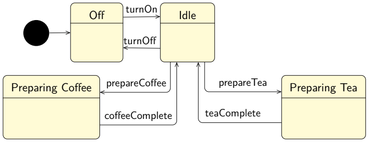
V_1?

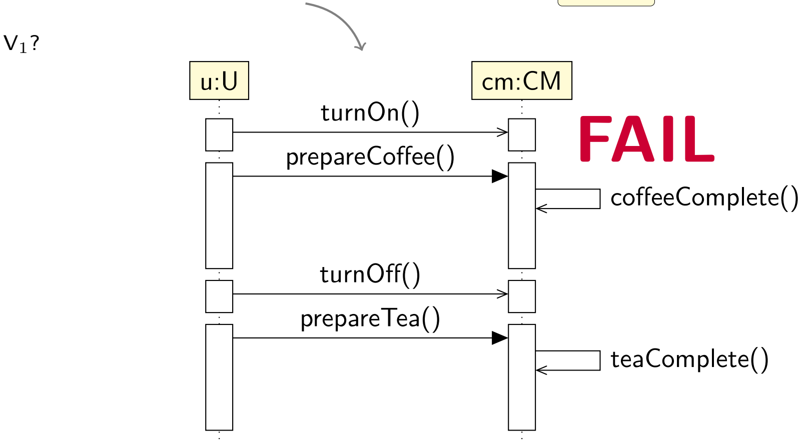


u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
prepareTea()
teaComplete()

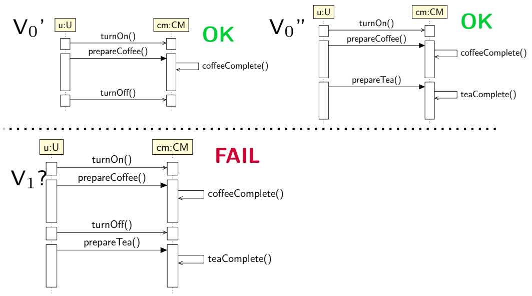
Example

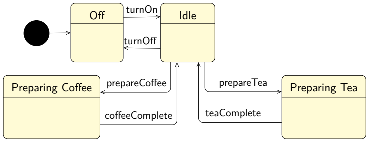

V0'
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
OK
V0''
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()
OK


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

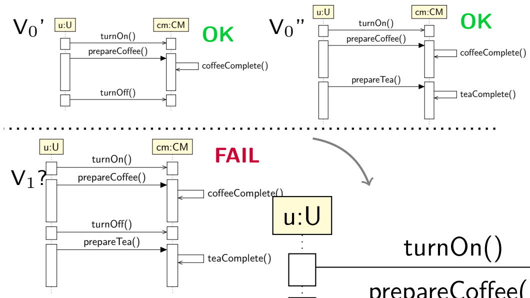

V1?
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
prepareTea()
teaComplete()
FAIL

Example

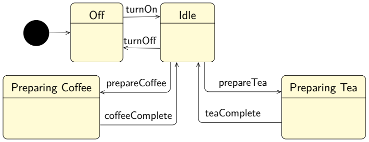

V0'
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
OK
V0''
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()
OK
u:U
cm:CM
V1?
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
prepareTea()
teaComplete()
FAIL


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

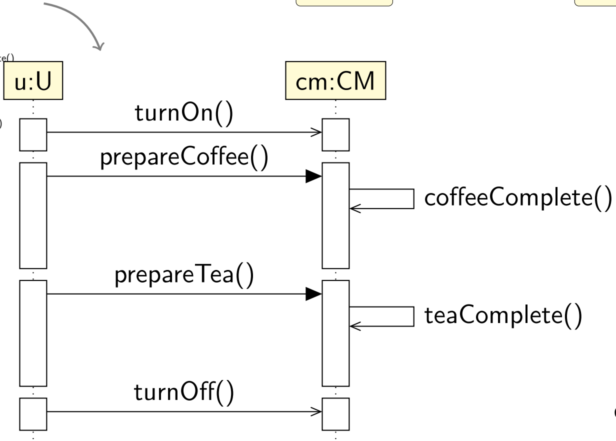
Example



V_0'
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
OK
V_0''
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()
OK
V_1?
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
prepareTea()
teaComplete()
FAIL

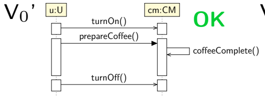


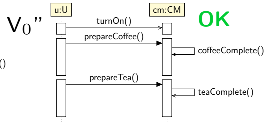
Off
Idle
turnOn
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
teaComplete
Preparing Tea

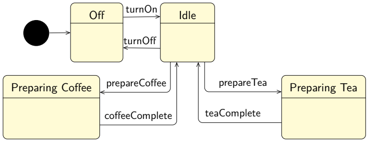


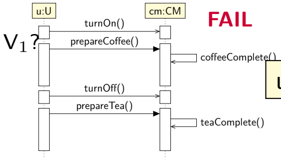
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()
turnOff()

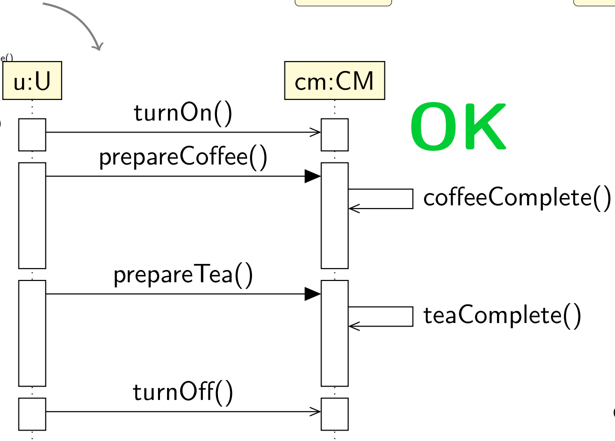
Example


V0'
u:U
cm:CM
OK
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()

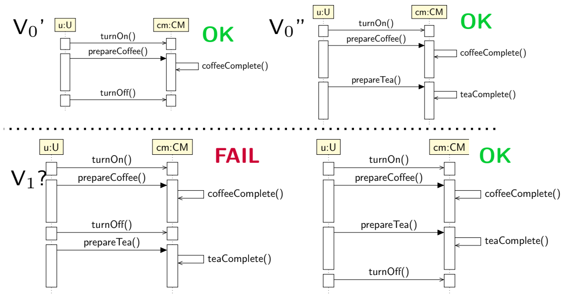

V0''
u:U
cm:CM
OK
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()

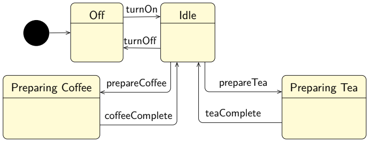

Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
prepareTea
Preparing Tea
coffeeComplete
teaComplete


V1?
u:U
cm:CM
FAIL
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
prepareTea()
teaComplete()

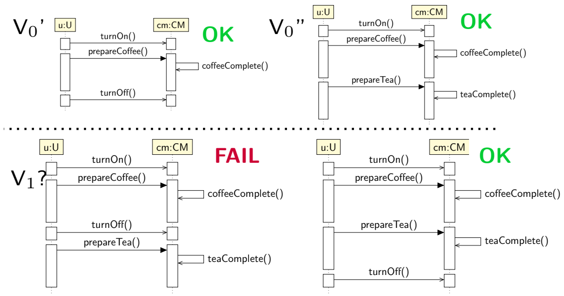

u:U
cm:CM
OK
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()
turnOff()

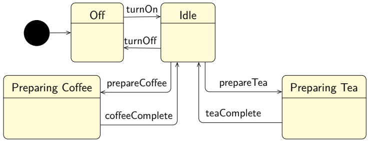
Example


V$_0$'
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
OK
V$_0$''
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()
OK
V$_1$?
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
turnOff()
prepareTea()
teaComplete()
FAIL
u:U
cm:CM
turnOn()
prepareCoffee()
coffeeComplete()
prepareTea()
teaComplete()
turnOff()
OK


Off
turnOn
Idle
turnOff
Preparing Coffee
prepareCoffee
coffeeComplete
prepareTea
Preparing Tea
teaComplete

Example





Problem

Many syntactically correct merges possible.
But how to avoid inconsistency with state machine?

Related work

- Model Evolution

Related work

- Model Evolution
  - *Mens et al.* Challenges in Software Evolution
  - *Mens et al.* Challenges in Model-Driven Software Engineering
  - *Brosch et al.* Model Versioning, Change Management

Related work

- Model Evolution
  - *Mens et al.* Challenges in Software Evolution
  - *Mens et al.* Challenges in Model-Driven Software Engineering
  - *Brosch et al.* Model Versioning, Change Management
- Model Verification

Related work

- Model Evolution
  - *Mens et al.* Challenges in Software Evolution
  - *Mens et al.* Challenges in Model-Driven Software Engineering
  - *Brosch et al.* Model Versioning, Change Management
- Model Verification
  - *Maoz et al.* Semantic model differencing
  - *Knapp et al., Schaefer et al., Eshuis et al.* Model checking dynamic UML diagrams
  - *v.d. Straeten et al.* Description Logics

Related work

- Model Evolution
  - *Mens et al.* Challenges in Software Evolution
  - *Mens et al.* Challenges in Model-Driven Software Engineering
  - *Brosch et al.* Model Versioning, Change Management
- Model Verification
  - *Maoz et al.* Semantic model differencing
  - *Knapp et al., Schaefer et al., Eshuis et al.* Model checking dynamic UML diagrams
  - *v.d. Straeten et al.* Description Logics
- Formal Semantics of UML

Related work

- Model Evolution
  - *Mens et al.* Challenges in Software Evolution
  - *Mens et al.* Challenges in Model-Driven Software Engineering
  - *Brosch et al.* Model Versioning, Change Management
- Model Verification
  - *Maoz et al.* Semantic model differencing
  - *Knapp et al., Schaefer et al., Eshuis et al.* Model checking dynamic UML diagrams
  - *v.d. Straeten et al.* Description Logics
- Formal Semantics of UML
  - *Rumpe et al.* System Model
  - *Eshuis et al.* Activity Diagrams
  - *Luettgen and Mendler* Statechart Semantics via Intuitionistic Kripke Models

Approach and methodology

State of the art survey

Approach and methodology

State of the art survey

- Model evolution
- Model verification
- Semantics of UML

Approach and methodology

State of the art survey

Taxonomy of change

Approach and methodology

State of the art survey

Taxonomy of change

- Change in model evolution
- Definitions of inconsistencies
- Relations between change and inconsistency

Approach and methodology

State of the art survey

Taxonomy of change

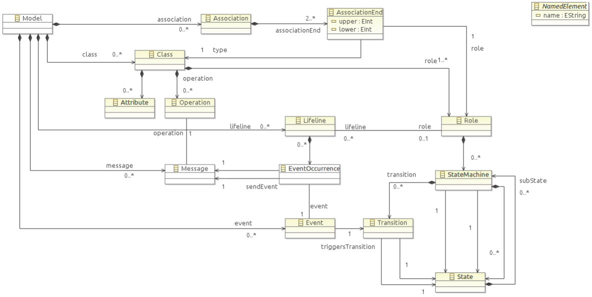
UML subset

Approach and methodology

State of the art survey

Taxonomy of change

UML subset



Approach and methodology

State of the art survey

Taxonomy of change

UML subset

Verification methods

Approach and methodology

State of the art survey

Taxonomy of change

UML subset

Verification methods

- Model checking (focus on dynamic view)
- Analysis of state space
- Own model checker for software models?

Approach and methodology

State of the art survey

Taxonomy of change

UML subset

Verification methods

Handling complexity

Approach and methodology

State of the art survey

Taxonomy of change

UML subset

Verification methods

Handling complexity

- Identify complex tasks
- Incremental verification

Approach and methodology

State of the art survey

Taxonomy of change

UML subset

Verification methods

Handling complexity

Evaluation

Approach and methodology

State of the art survey

Taxonomy of change

UML subset

Verification methods

Handling complexity

Evaluation
- Eclipse-based implementation
- Benchmarks from previous project
- Students in “Model Engineering” lab

Approach and methodology

State of the art survey

Taxonomy of change

UML subset

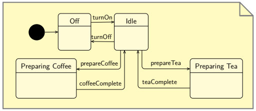
Verification methods

Handling complexity

Evaluation

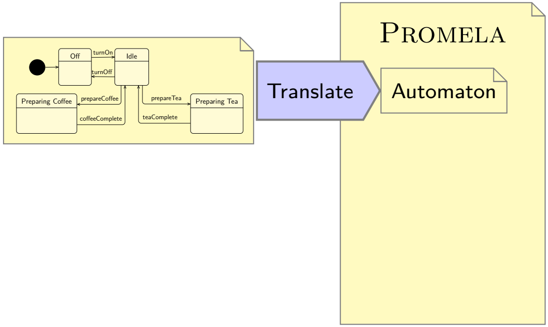
Semantics-aware model versioning

Using the SPIN model checker



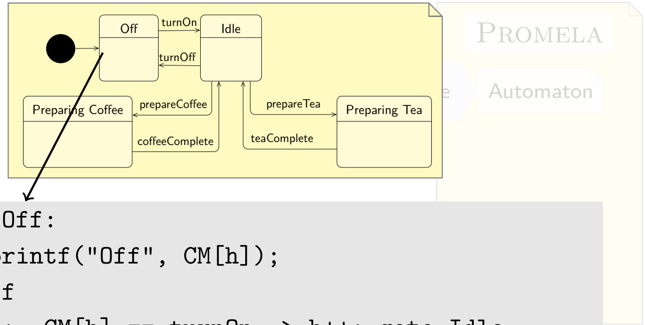
Semantics-aware model versioning

Using the SPIN model checker



Semantics-aware model versioning

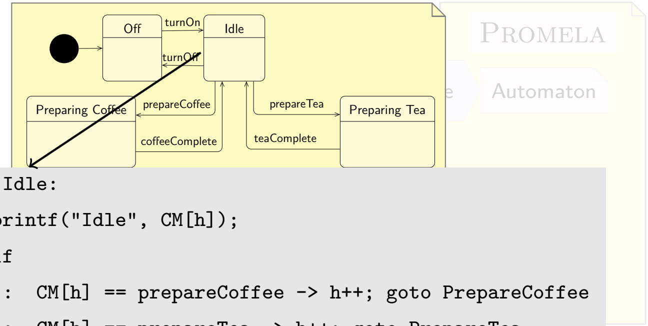
Using the SPIN model checker



```
 Off:
printf("Off", CM[h]);
if
::  CM[h] == turnOn -> h++; goto Idle
::  CM[h] == acc -> goto end
fi;
```

Semantics-aware model versioning

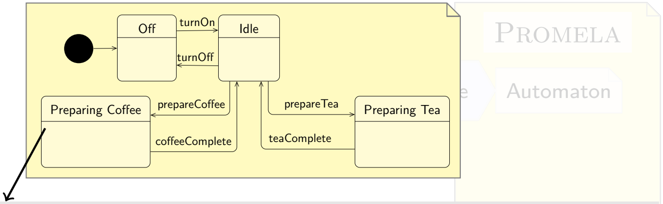
Using the SPIN model checker



```
 Idle:
printf("Idle", CM[h]);
if
::  CM[h] == prepareCoffee -> h++; goto PrepareCoffee
::  CM[h] == prepareTea -> h++; goto PrepareTea
::  CM[h] == turnOff -> h++; goto Off
::  CM[h] == acc -> goto end
fi;
```

Semantics-aware model versioning

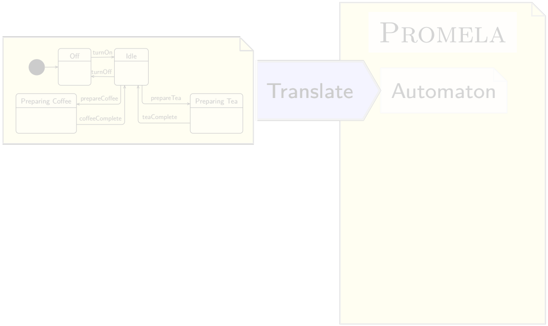
Using the SPIN model checker

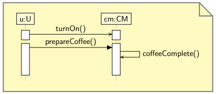


```
 PreparingCoffee:
printf("PrepareCoffee", CM[h]);
if
::  CM[h] == coffeeComplete -> h++; goto Idle
::  CM[h] == acc -> goto end
fi;
```

Semantics-aware model versioning

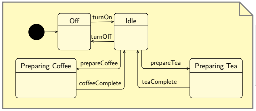
Using the SPIN model checker

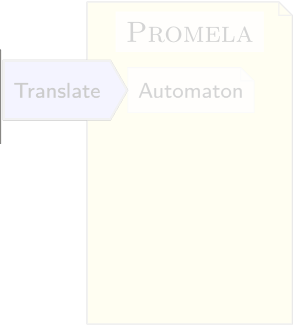


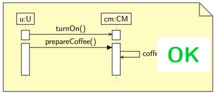


Semantics-aware model versioning

Using the SPIN model checker

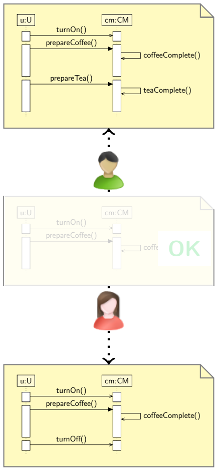


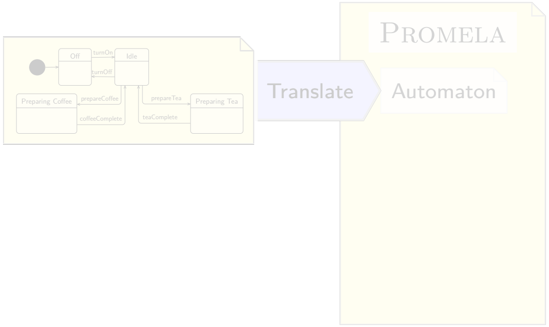




Semantics-aware model versioning

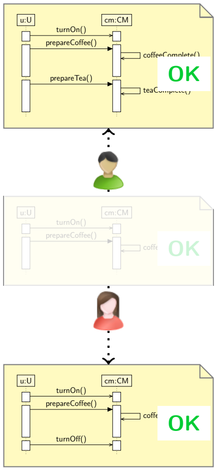
Using the SPIN model checker

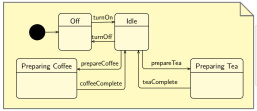


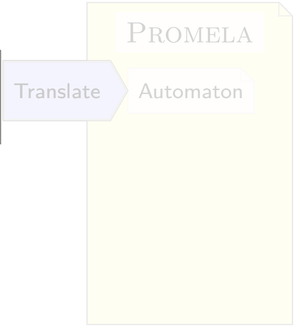


Semantics-aware model versioning

Using the SPIN model checker

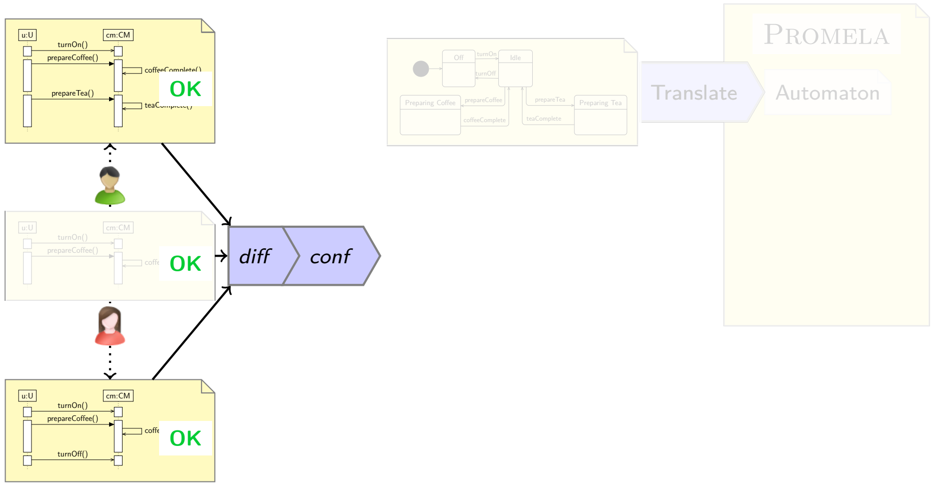






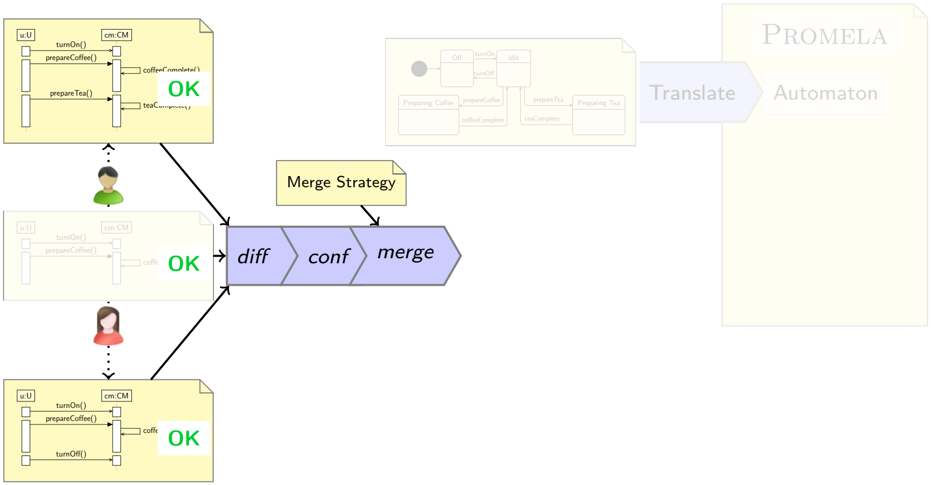
Semantics-aware model versioning

Using the SPIN model checker



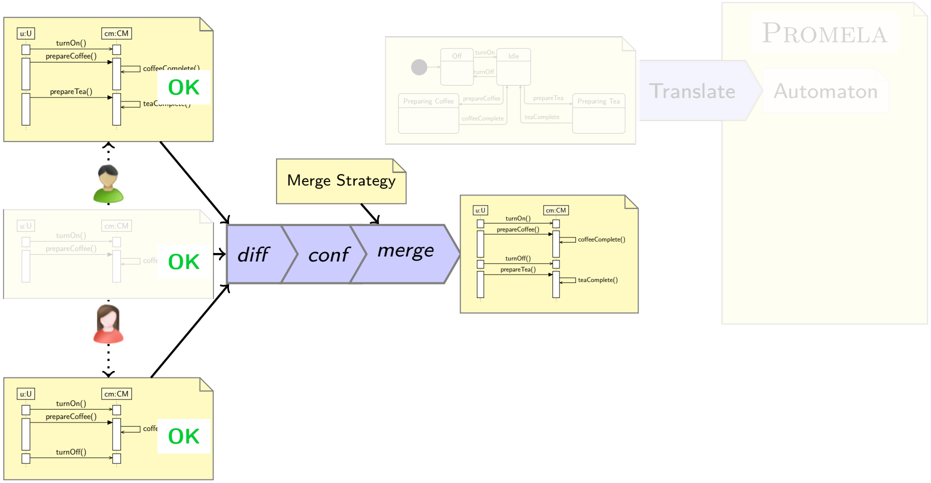
Semantics-aware model versioning

Using the SPIN model checker



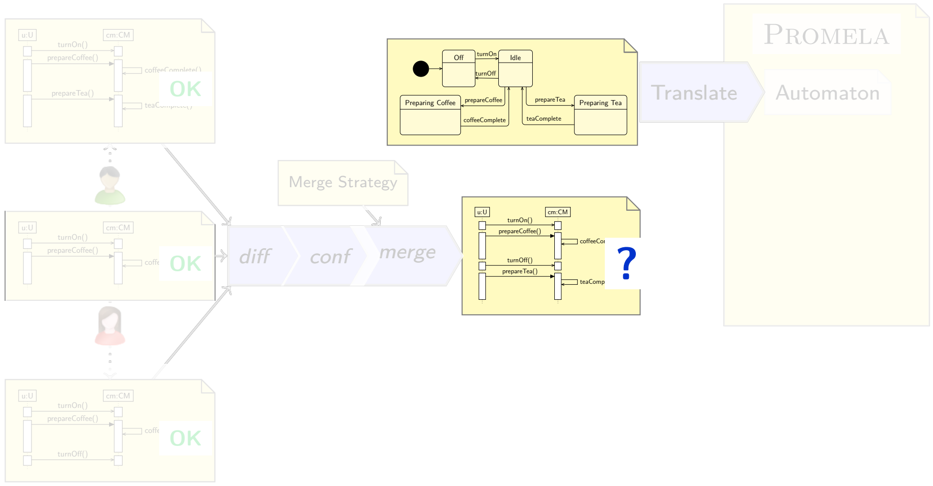
Semantics-aware model versioning

Using the SPIN model checker



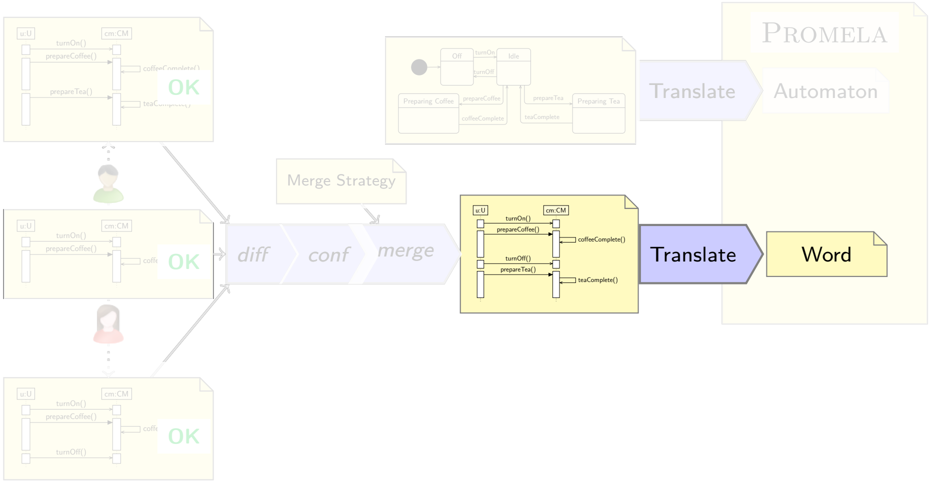
Semantics-aware model versioning

Using the SPIN model checker



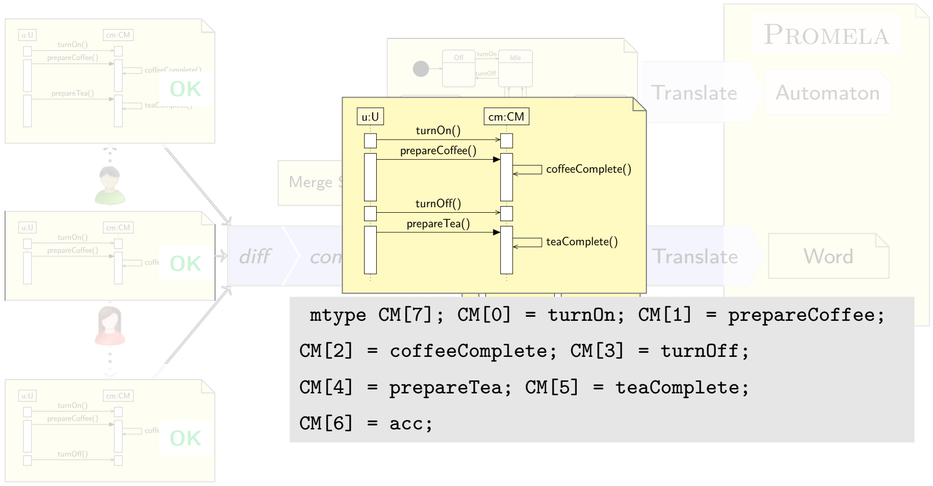
Semantics-aware model versioning

Using the SPIN model checker



Semantics-aware model versioning

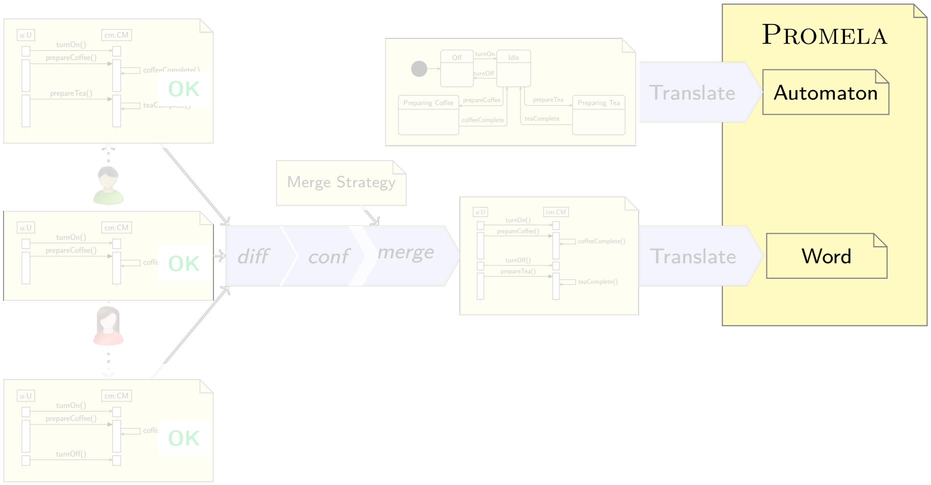
Using the SPIN model checker



```
 mtype CM[7]; CM[0] = turnOn; CM[1] = prepareCoffee;
CM[2] = coffeeComplete; CM[3] = turnOff;
CM[4] = prepareTea; CM[5] = teaComplete;
CM[6] = acc;
```

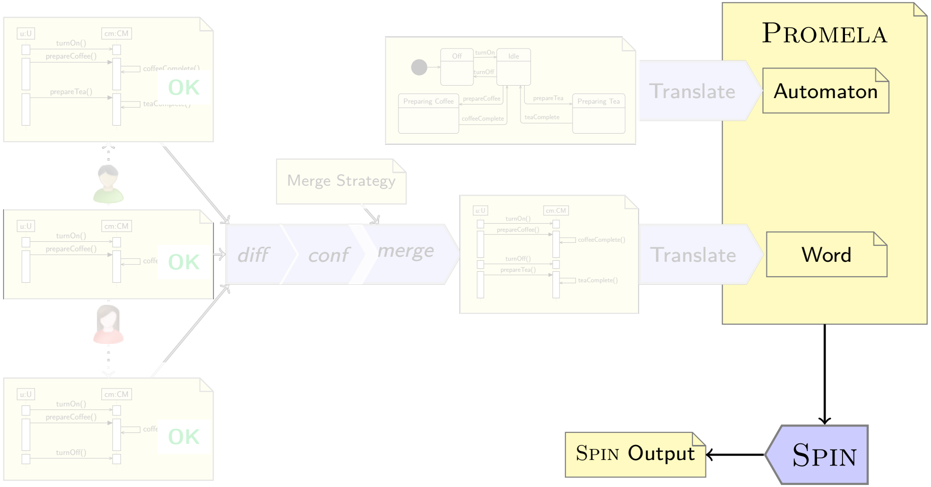
Semantics-aware model versioning

Using the SPIN model checker



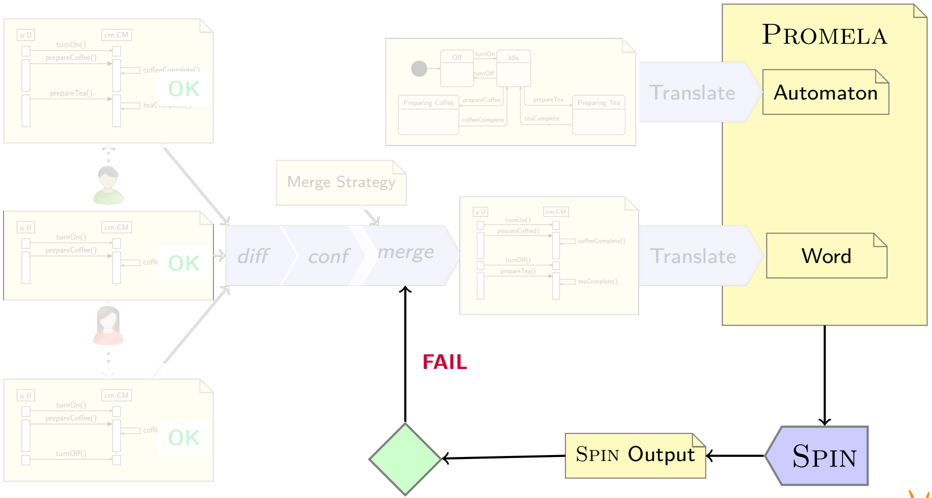
Semantics-aware model versioning

Using the SPIN model checker



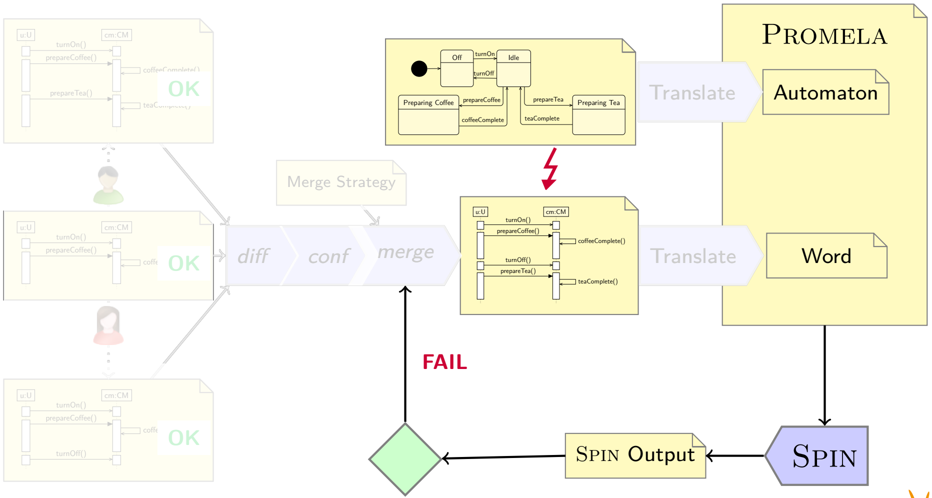
Semantics-aware model versioning

Using the SPIN model checker



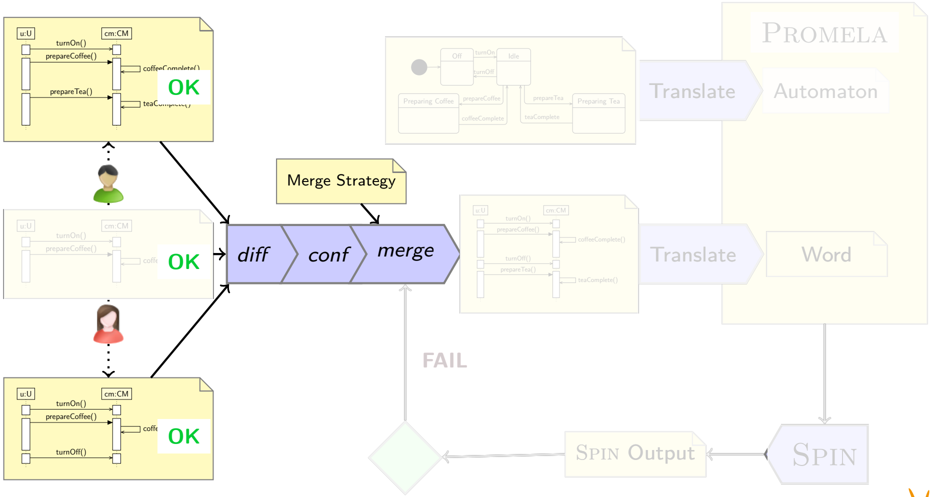
Semantics-aware model versioning

Using the SPIN model checker



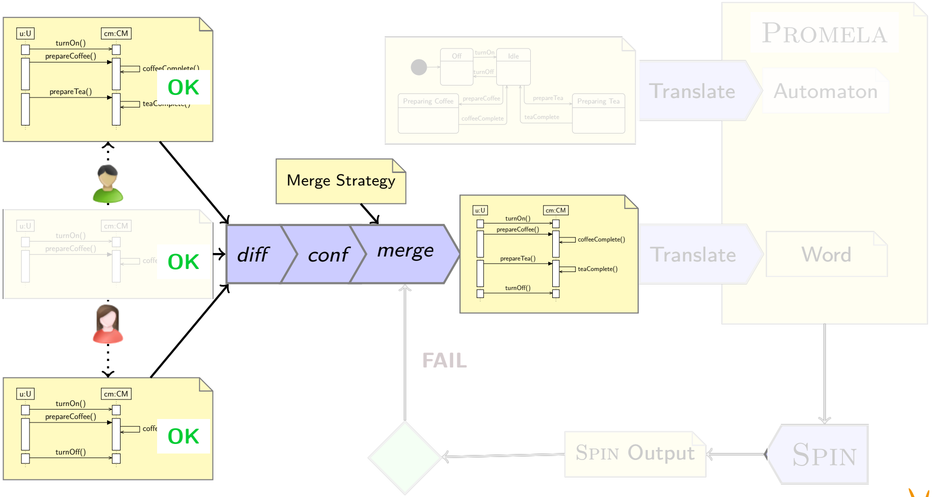
Semantics-aware model versioning

Using the SPIN model checker



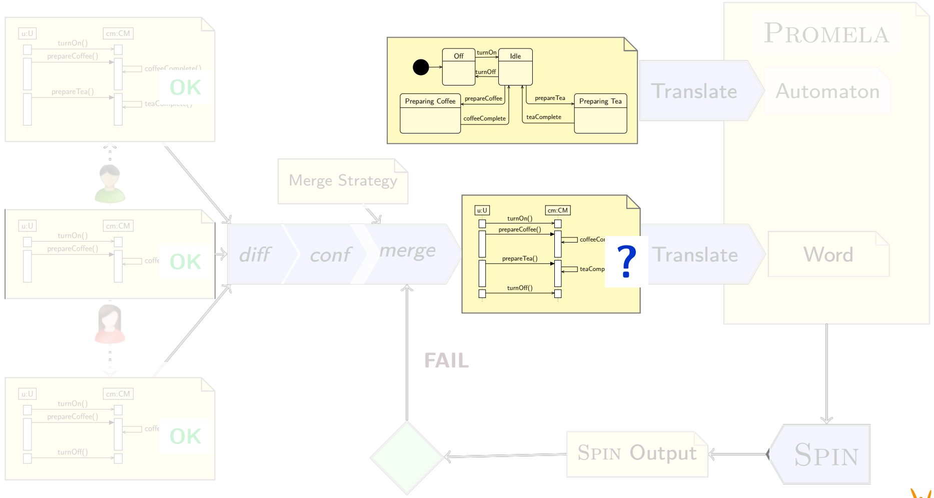
Semantics-aware model versioning

Using the SPIN model checker



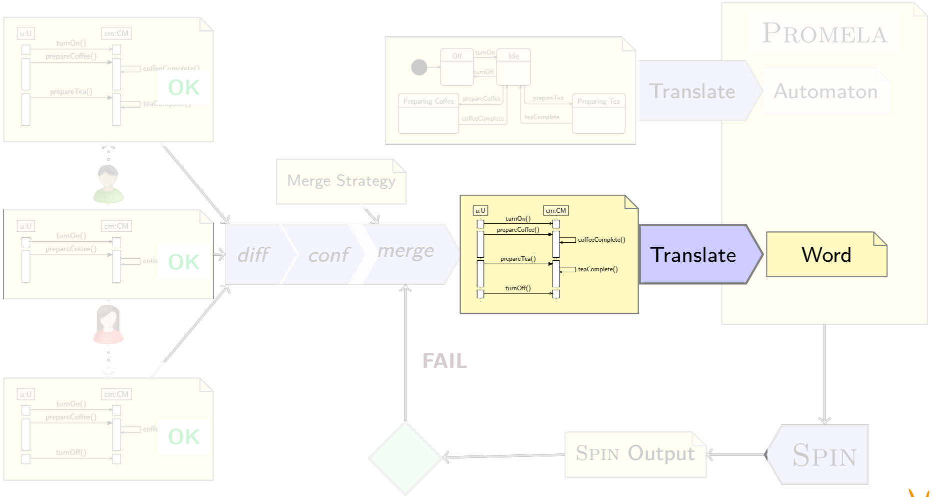
Semantics-aware model versioning

Using the SPIN model checker



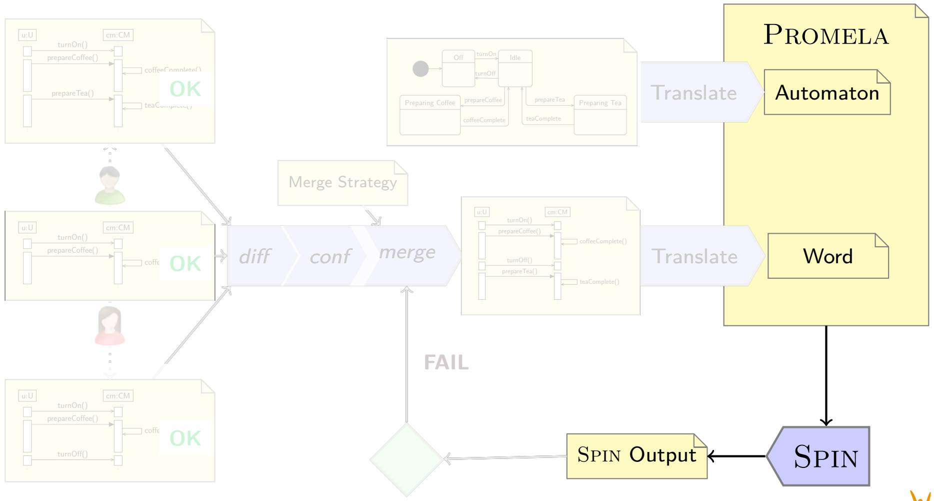
Semantics-aware model versioning

Using the SPIN model checker



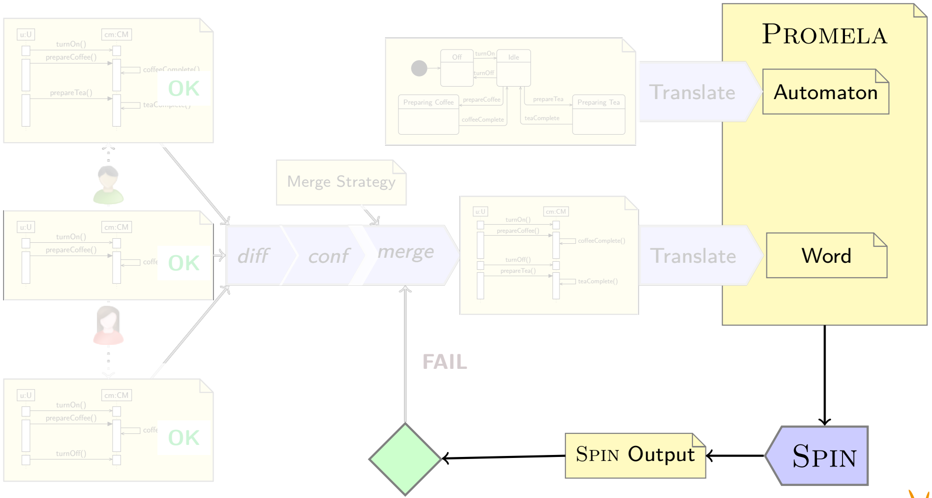
Semantics-aware model versioning

Using the SPIN model checker



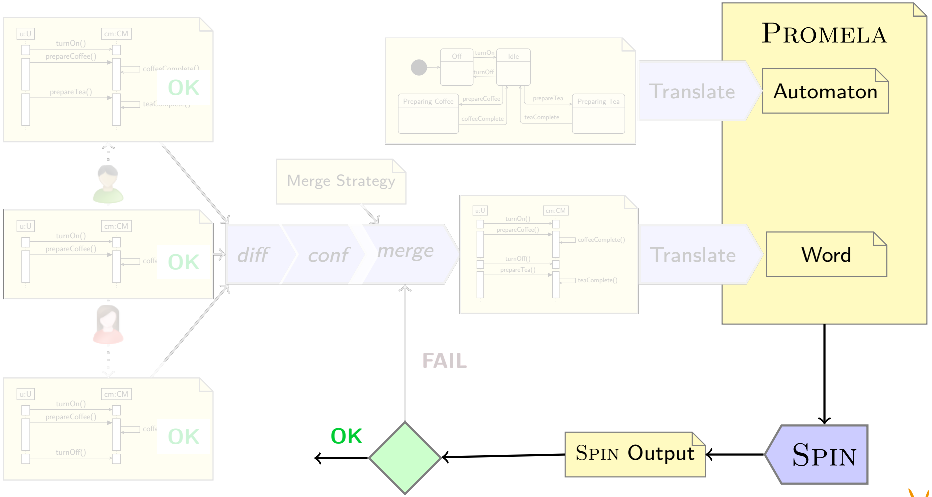
Semantics-aware model versioning

Using the SPIN model checker



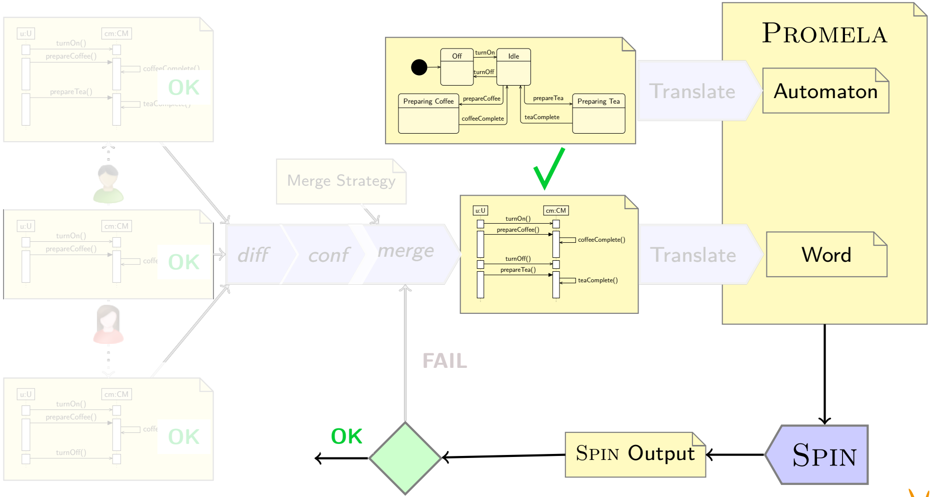
Semantics-aware model versioning

Using the SPIN model checker



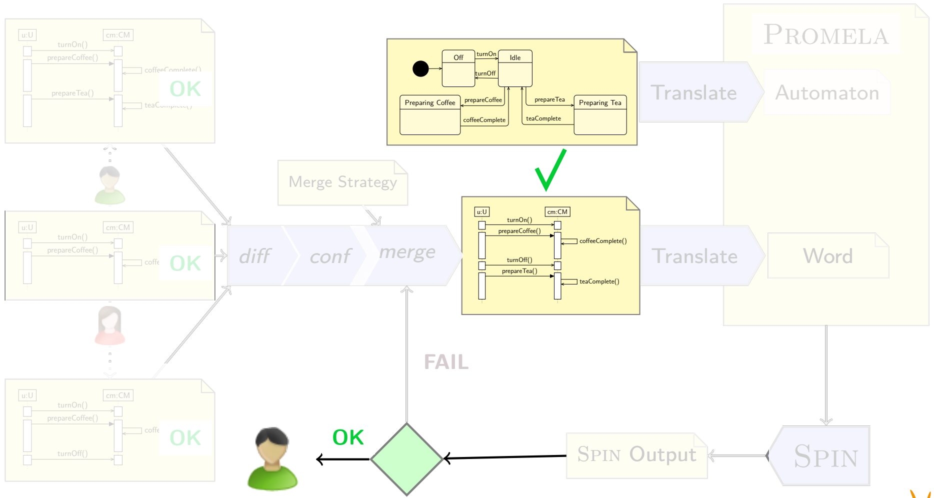
Semantics-aware model versioning

Using the SPIN model checker



Semantics-aware model versioning

Using the SPIN model checker



Expected Contributions

- Survey on model evolution
- Taxonomy of change and inconsistencies
- Verification methods
- Integration into a formal framework to assist MDE