

Short about me

- Name: Vidar Slåtten
- Enrolled in an “integrated PhD” program at NTNU, Trondheim, Norway
 - 4 year master, 2 year master and PhD, 3 year PhD
- Specialization (master’s): Software engineering
- Topic/interest (PhD): Formal methods applied to software engineering

Model checking service specifications

Model checking service specifications



Overview

- **SPACE: Composing (distributed) reactive systems from services**
- **Project thesis: Transforming service specifications to input for a model checker**
- **Master's thesis: Hiding the difficult bits from the user**

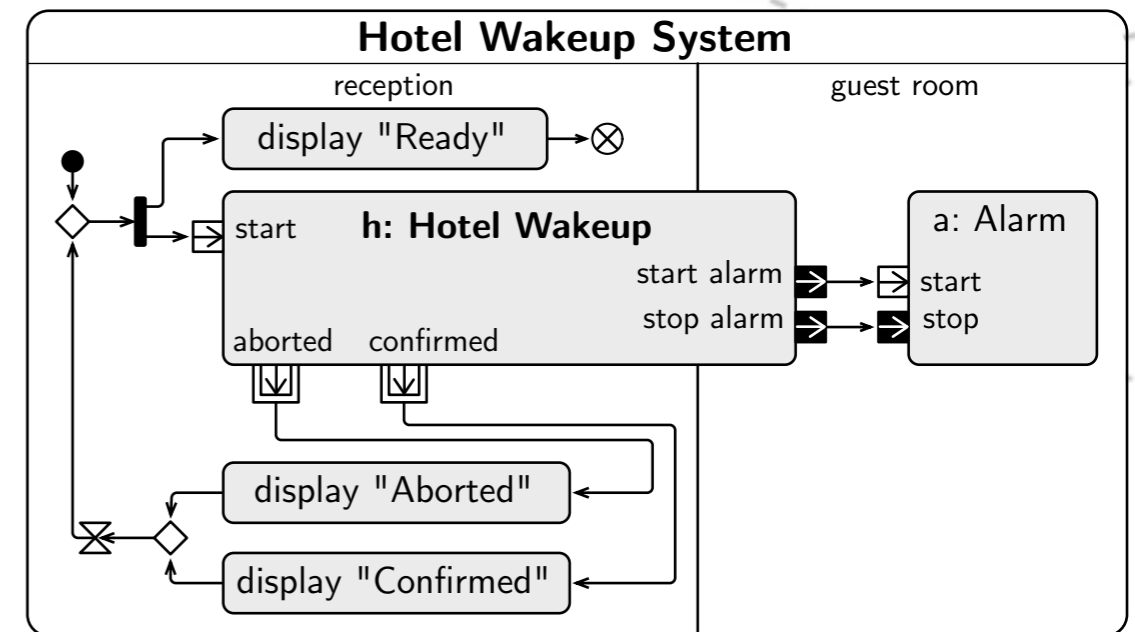
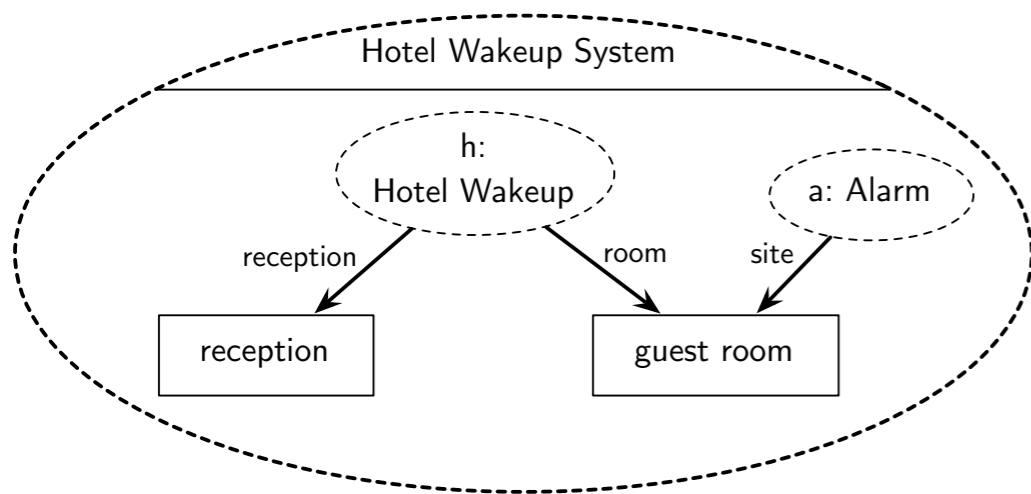
SPACE

- A method for creating distributed reactive systems
- Systems are composed from services
- Services can span several components
- Asynchronous communication

Why SPACE?

- Developing distributed reactive systems is a hard task
- Make the systems easier to understand
 - Reusable services
- Provide tools to resolve problems early, when they are still cheap to fix
- Use familiar syntax (UML)

Example

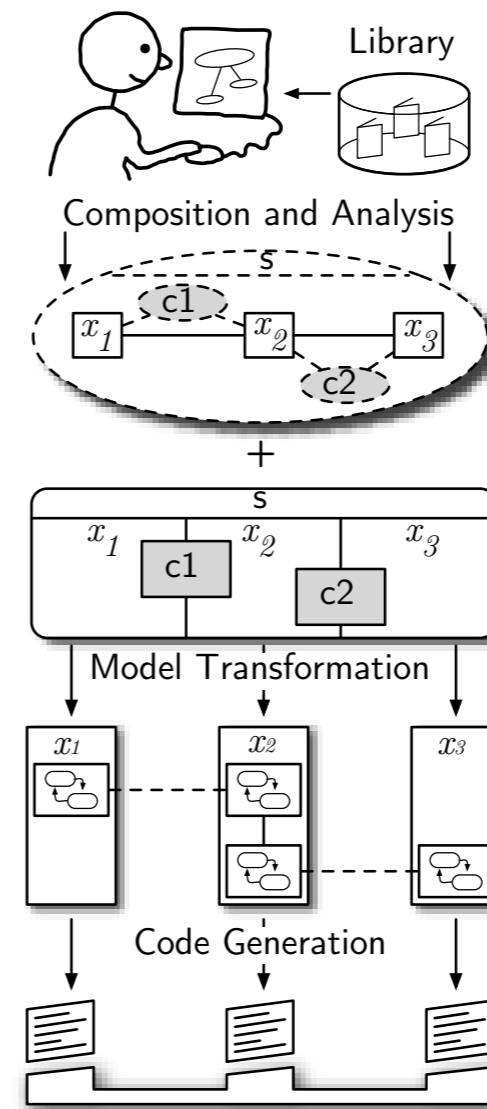


Structure – Which components take part in which services?

Behavior – How do services interact?

SPACE – the process

- Specify structure (collaboration diagrams)
- Specify behavior (activity diagrams)
- “Is it correct?”
- Transform to components
- Generate code (OSGi bundles)



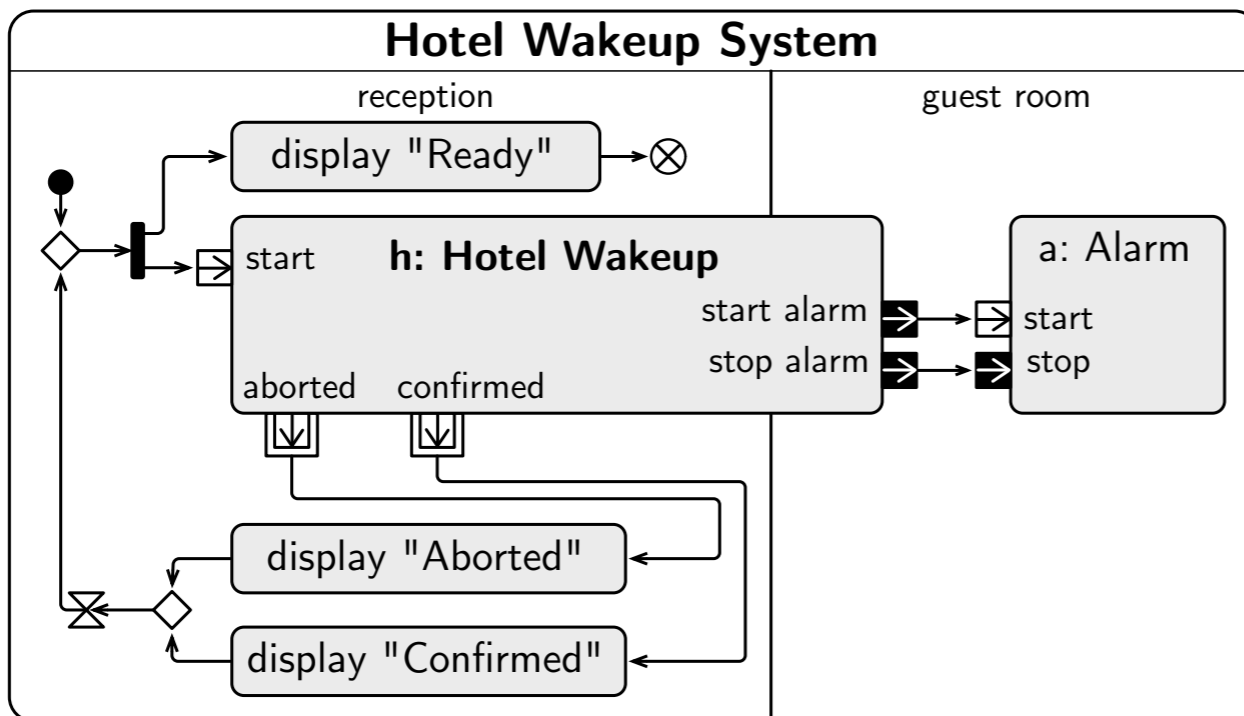
Service Engineering
Composition of Services
from Building Blocks

Service Specifications
UML Collaborations,
Activities

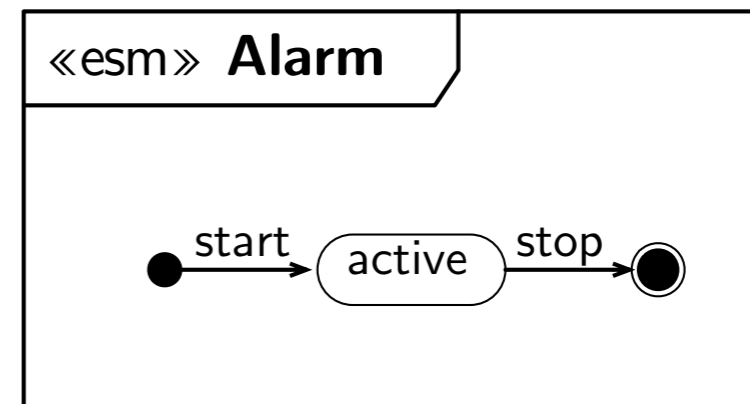
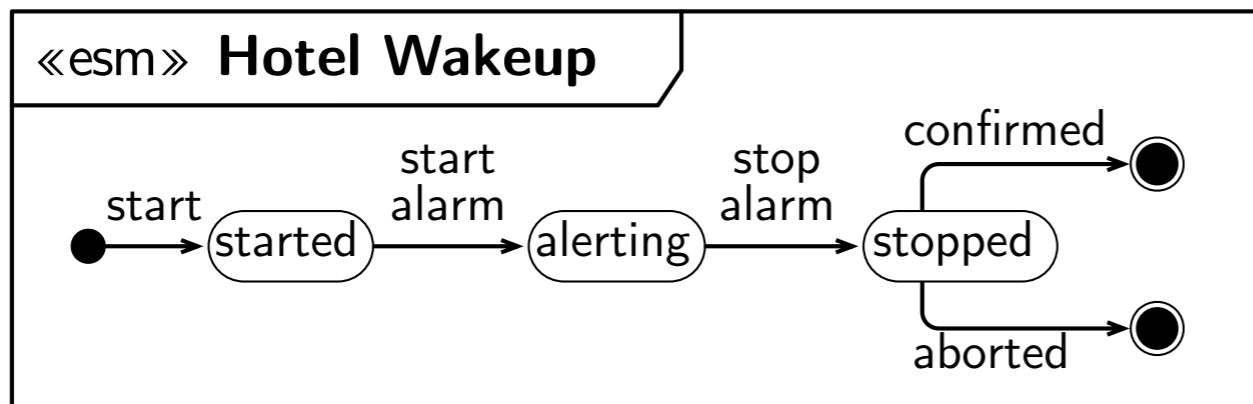
Service Components
UML State Machines,
Composite Structures

Executable System
Service Application Code
Execution Framework

External State Machines

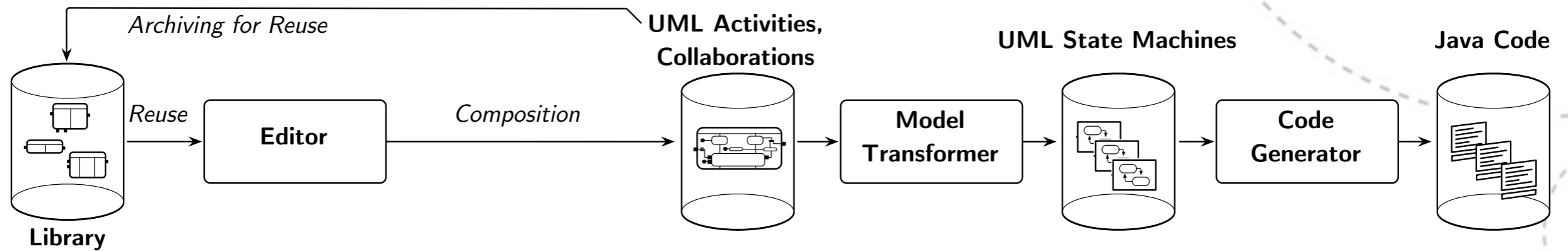


How do services interact?

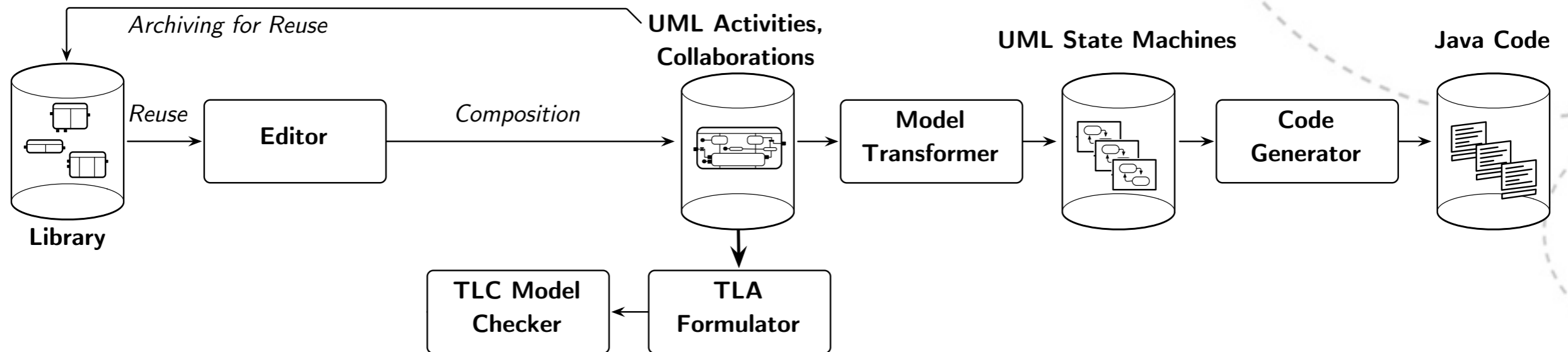


How does each services behave?

Arctis tool suite

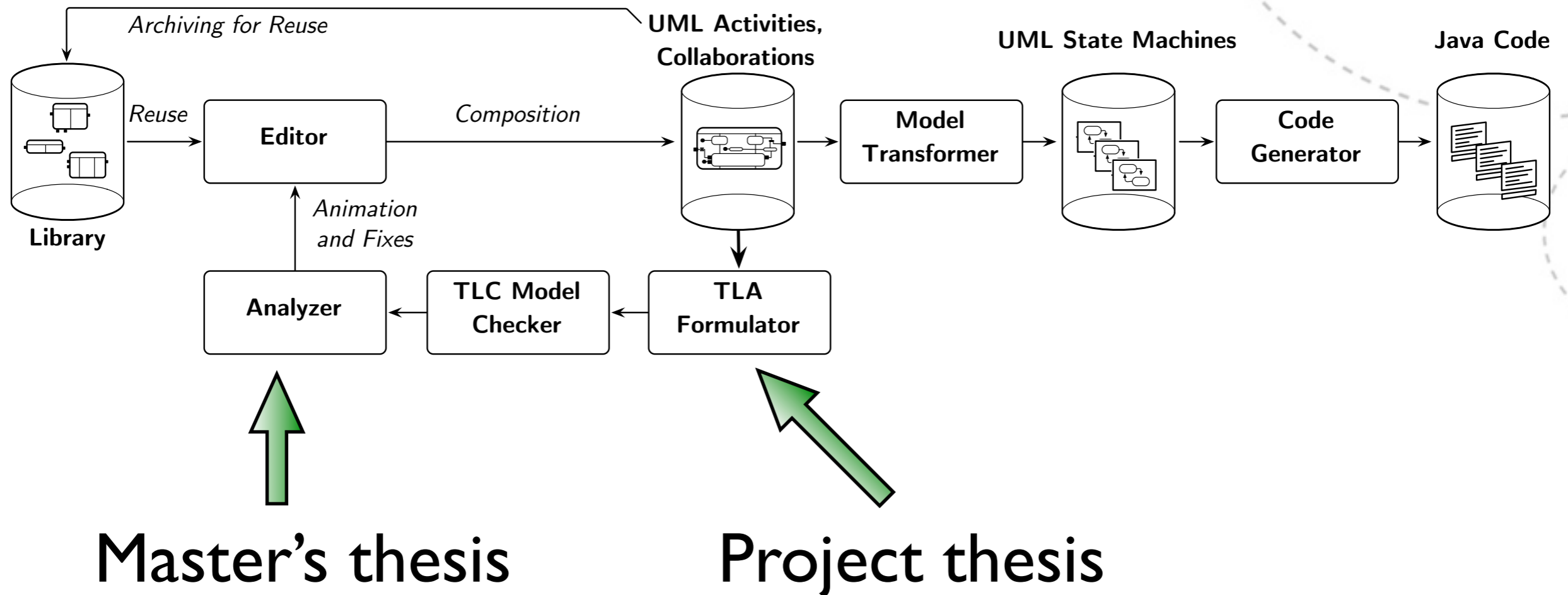


Arctis tool suite

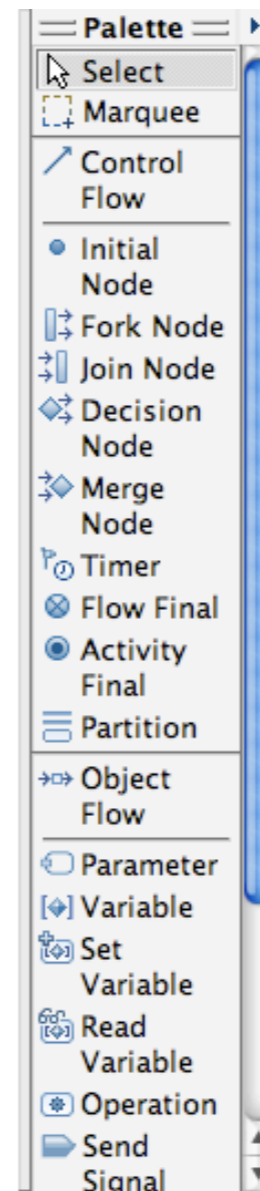
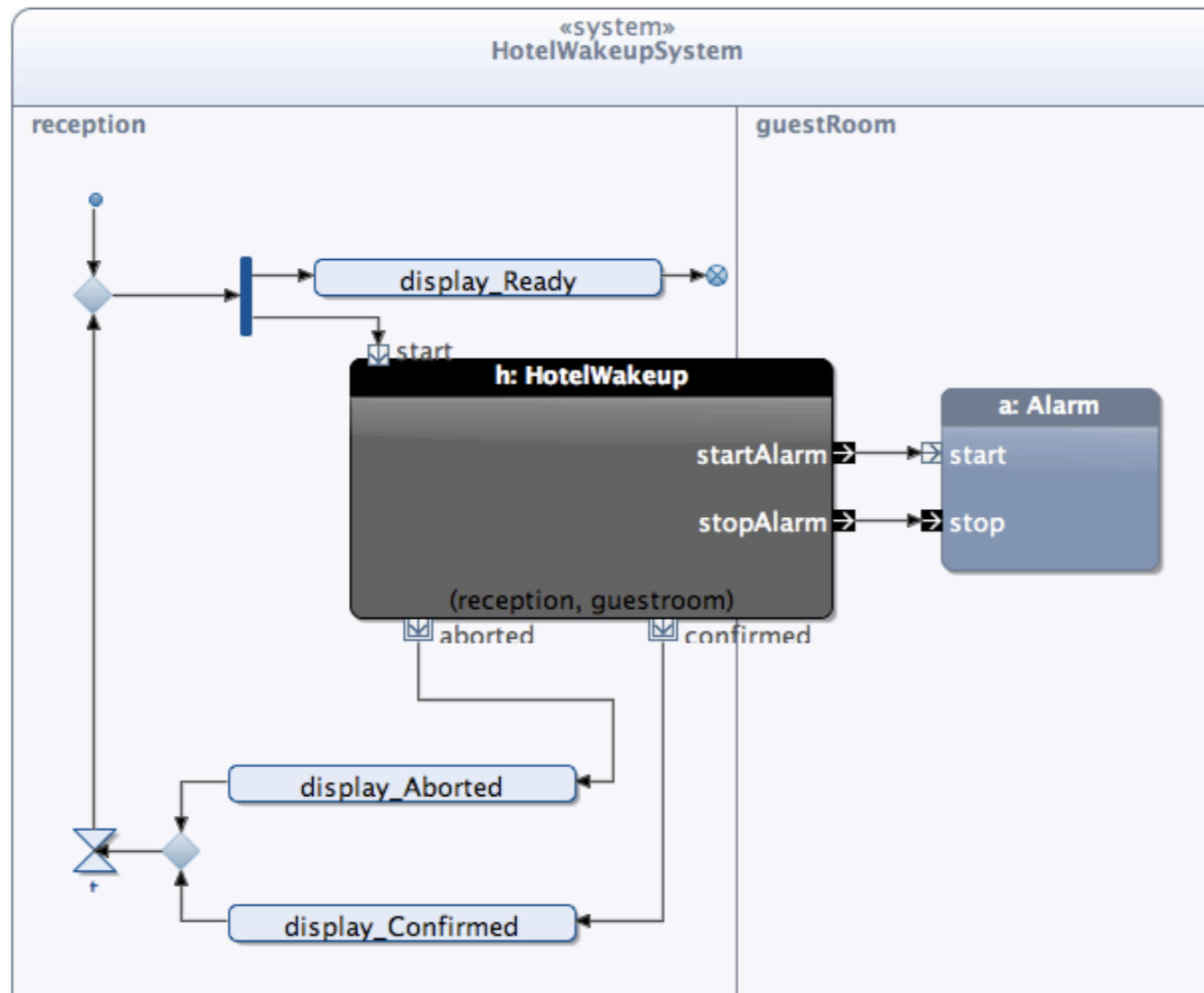
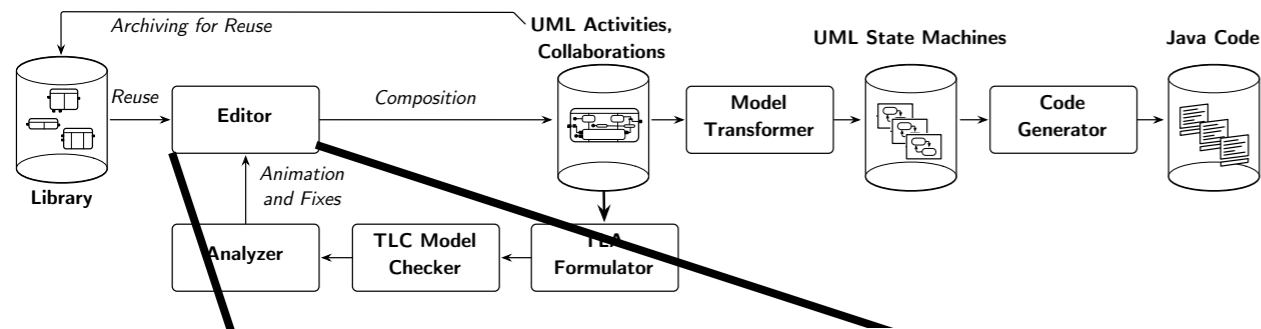


Project thesis

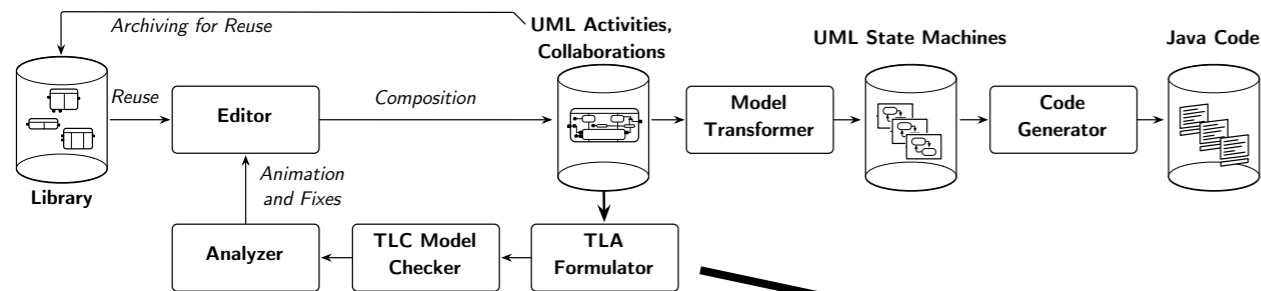
Arctis tool suite



The activity editor



TLA formulator



```

MODULE HotelWakeupSystem
EXTENDS Naturals
VARIABLES i, t, h, a

```

```

Init ≜
  ∧ i = 1 ∧ t = 0
  ∧ h = "off" ∧ a = "off"

```

```

initial ≜
  ∧ i = 1 ∧ i' = 0
  ∧ h = "off" ∧ h' = "started"
  ∧ UNCHANGED ⟨a, t⟩

```

```

startAlert ≜
  ∧ h = "started" ∧ h' = "alerting"
  ∧ a = "off" ∧ a' = "active"
  ∧ UNCHANGED ⟨i, t⟩

```

```

stopAlert ≜
  ∧ h = "alerting" ∧ h' = "stopped"
  ∧ a = "active" ∧ a' = "off"
  ∧ UNCHANGED ⟨i, t⟩

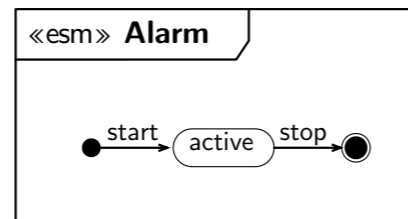
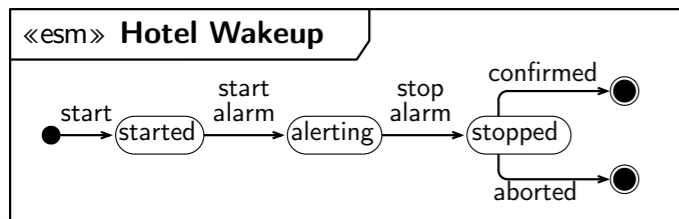
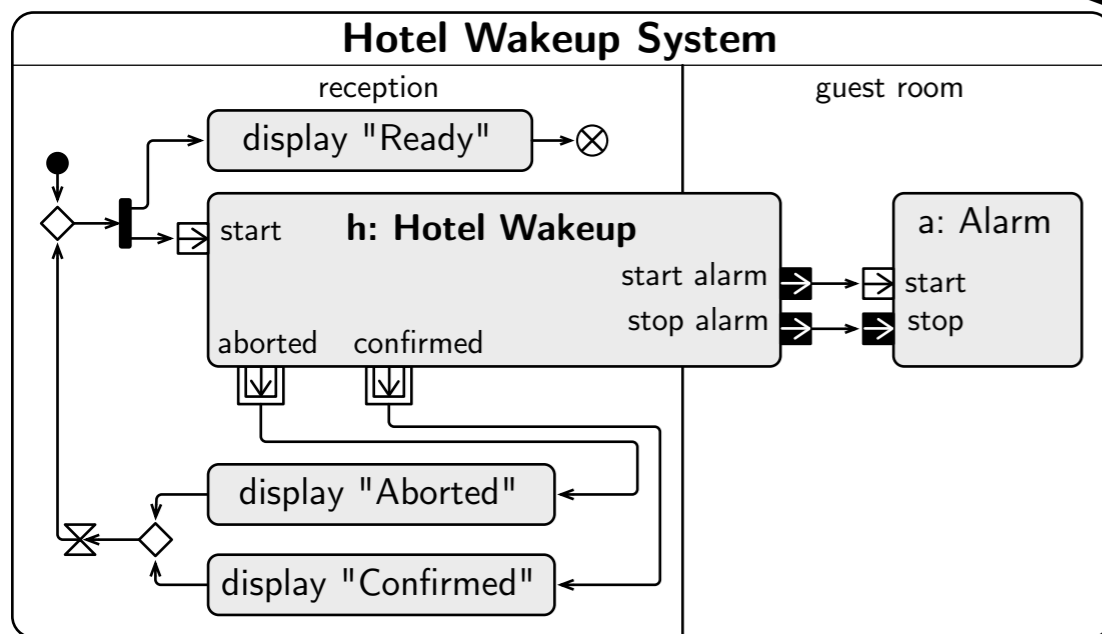
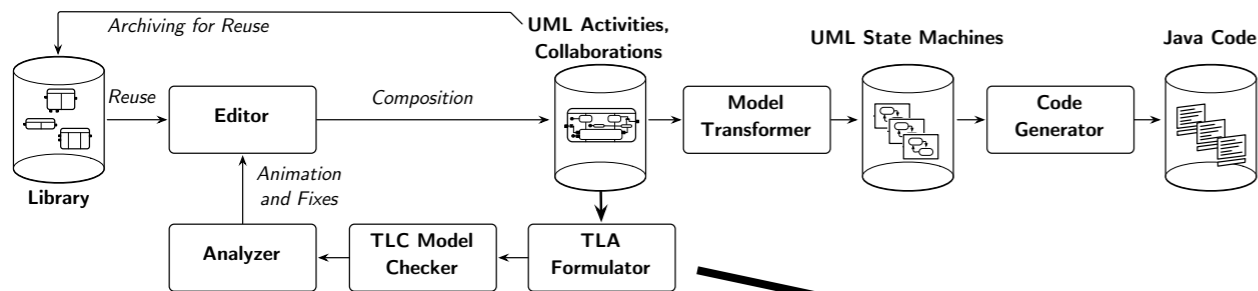
```

```

aborted ≜
  ∧ h = "stopped" ∧ h' = "off"

```

TLA formulator



MODULE *HotelWakeupSystem*

EXTENDS *Naturals*
 VARIABLES *i, t, h, a*

Init \triangleq
 $\wedge i = 1 \wedge t = 0$
 $\wedge h = \text{"off"} \wedge a = \text{"off"}$

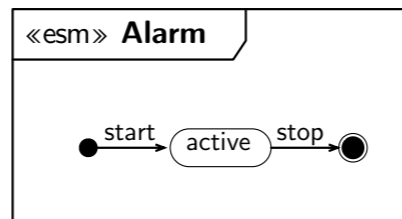
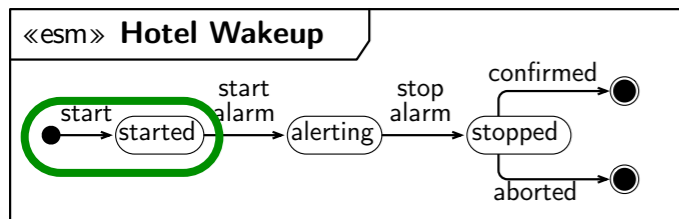
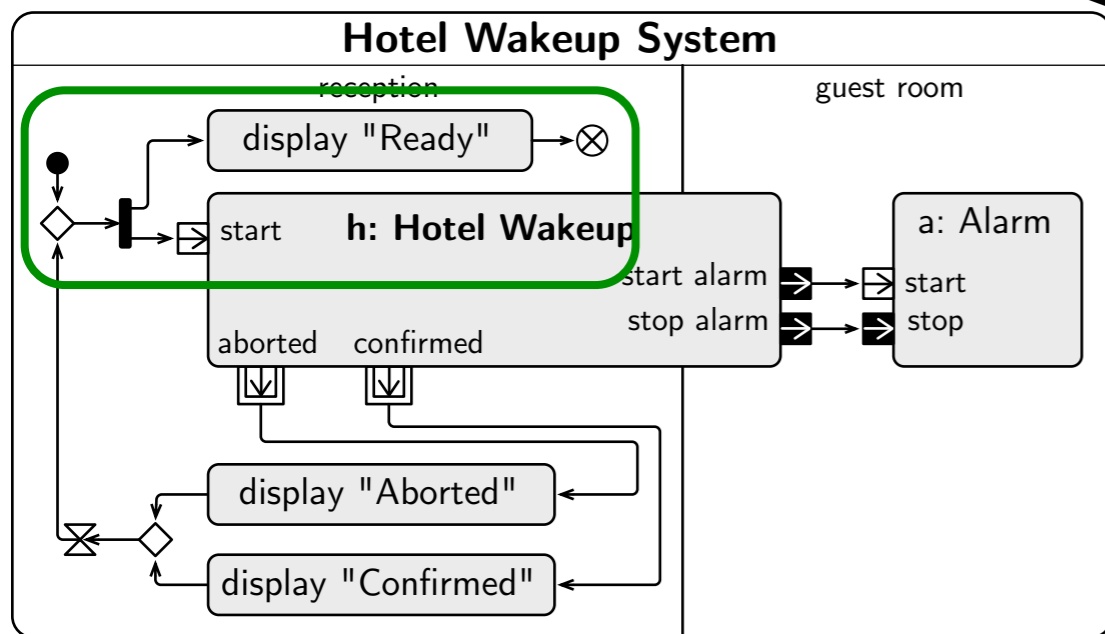
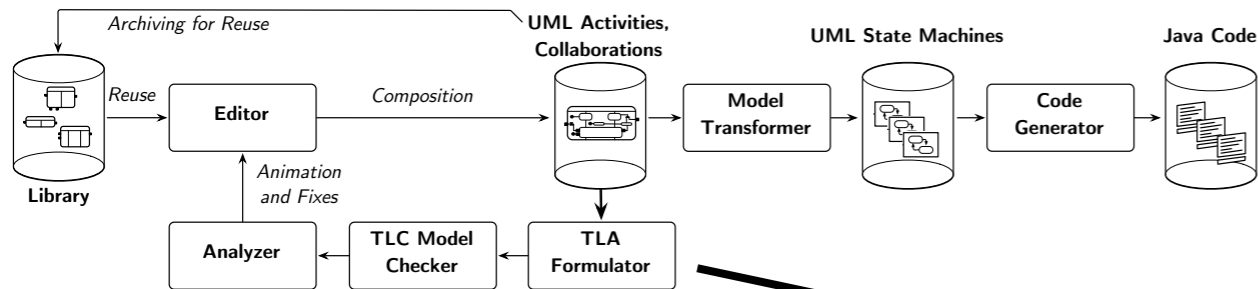
initial \triangleq
 $\wedge i = 1 \wedge i' = 0$
 $\wedge h = \text{"off"} \wedge h' = \text{"started"}$
 $\wedge \text{UNCHANGED } \langle a, t \rangle$

startAlert \triangleq
 $\wedge h = \text{"started"} \wedge h' = \text{"alerting"}$
 $\wedge a = \text{"off"} \wedge a' = \text{"active"}$
 $\wedge \text{UNCHANGED } \langle i, t \rangle$

stopAlert \triangleq
 $\wedge h = \text{"alerting"} \wedge h' = \text{"stopped"}$
 $\wedge a = \text{"active"} \wedge a' = \text{"off"}$
 $\wedge \text{UNCHANGED } \langle i, t \rangle$

aborted \triangleq
 $\wedge h = \text{"stopped"} \wedge h' = \text{"off"}$

TLA formulator



```

MODULE HotelWakeupSystem
EXTENDS Naturals
VARIABLES i, t, h, a
    
```

```

Init ≜
  ∧ i = 1 ∧ t = 0
  ∧ h = "off" ∧ a = "off"
    
```

```

initial ≜
  ∧ i = 1 ∧ i' = 0
  ∧ h = "off" ∧ h' = "started"
  ∧ UNCHANGED ⟨a, t⟩
    
```

```

startAlert ≜
  ∧ h = "started" ∧ h' = "alerting"
  ∧ a = "off" ∧ a' = "active"
  ∧ UNCHANGED ⟨i, t⟩
    
```

```

stopAlert ≜
  ∧ h = "alerting" ∧ h' = "stopped"
  ∧ a = "active" ∧ a' = "off"
  ∧ UNCHANGED ⟨i, t⟩
    
```

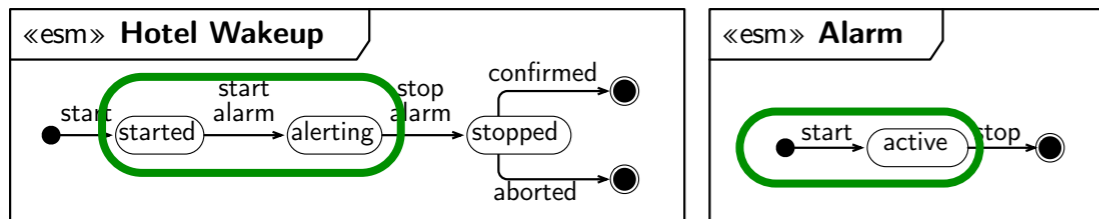
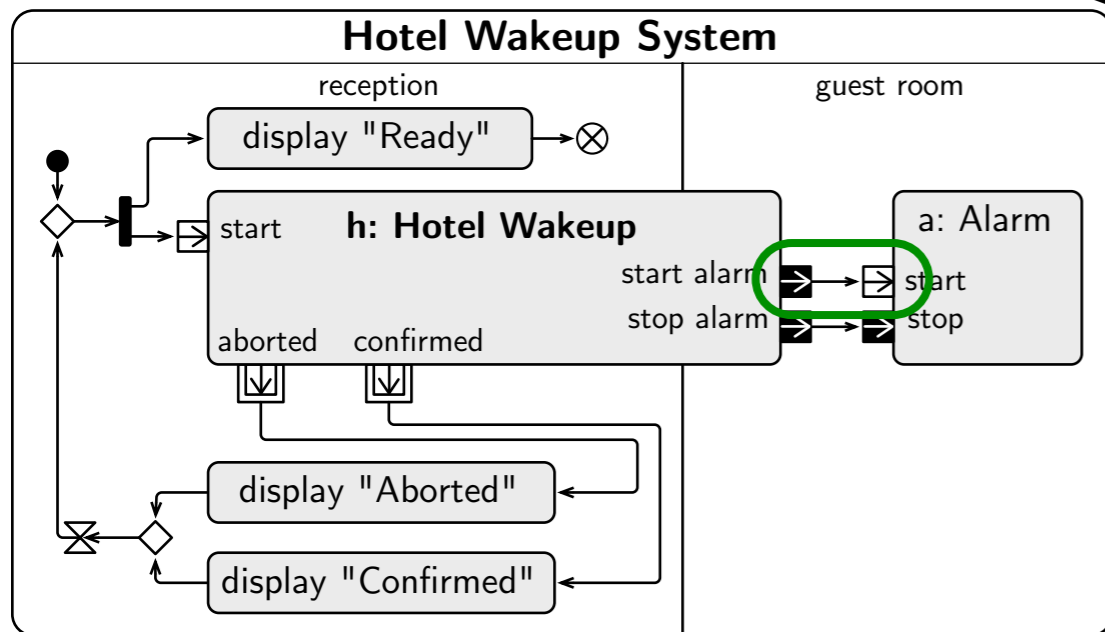
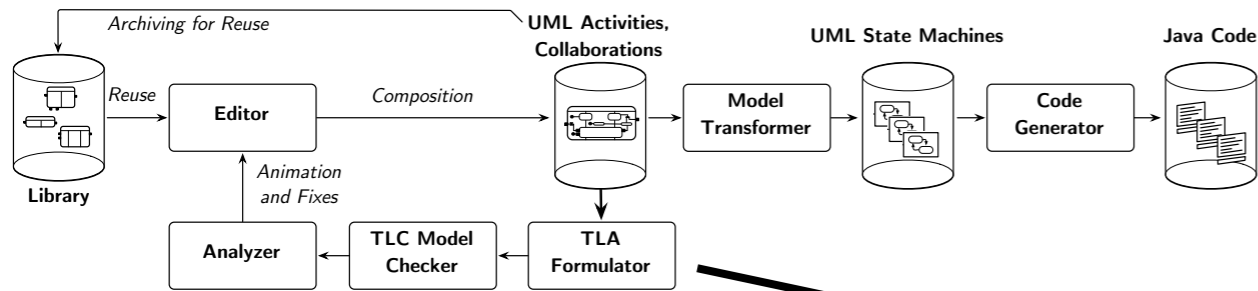
```

aborted ≜
  ∧ h = "stopped" ∧ h' = "off"
    
```

```

initial ≜
  ∧ i = 1 ∧ i' = 0
  ∧ h = "off" ∧ h' = "started"
  ∧ UNCHANGED ⟨a, t⟩
    
```


TLA formulator



```

MODULE HotelWakeupSystem
EXTENDS Naturals
VARIABLES i, t, h, a
    
```

```

Init ≜
  ∧ i = 1 ∧ t = 0
  ∧ h = "off" ∧ a = "off"
    
```

```

initial ≜
  ∧ i = 1 ∧ i' = 0
  ∧ h = "off" ∧ h' = "started"
  ∧ UNCHANGED ⟨a, t⟩
    
```

```

startAlert ≜
  ∧ h = "started" ∧ h' = "alerting"
  ∧ a = "off" ∧ a' = "active"
  ∧ UNCHANGED ⟨i, t⟩
    
```

```

stopAlert ≜
  ∧ h = "alerting" ∧ h' = "stopped"
  ∧ a = "active" ∧ a' = "off"
  ∧ UNCHANGED ⟨i, t⟩
    
```

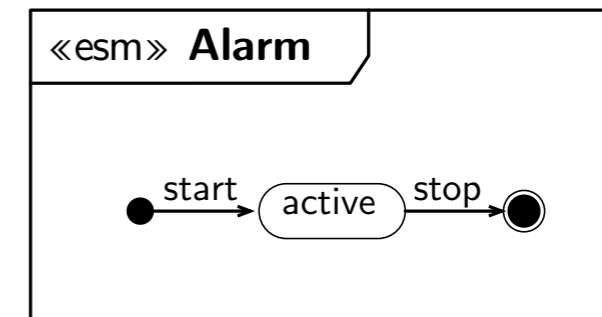
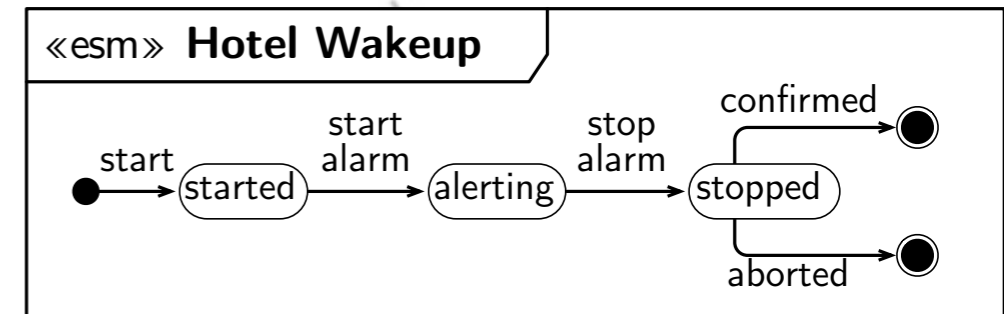
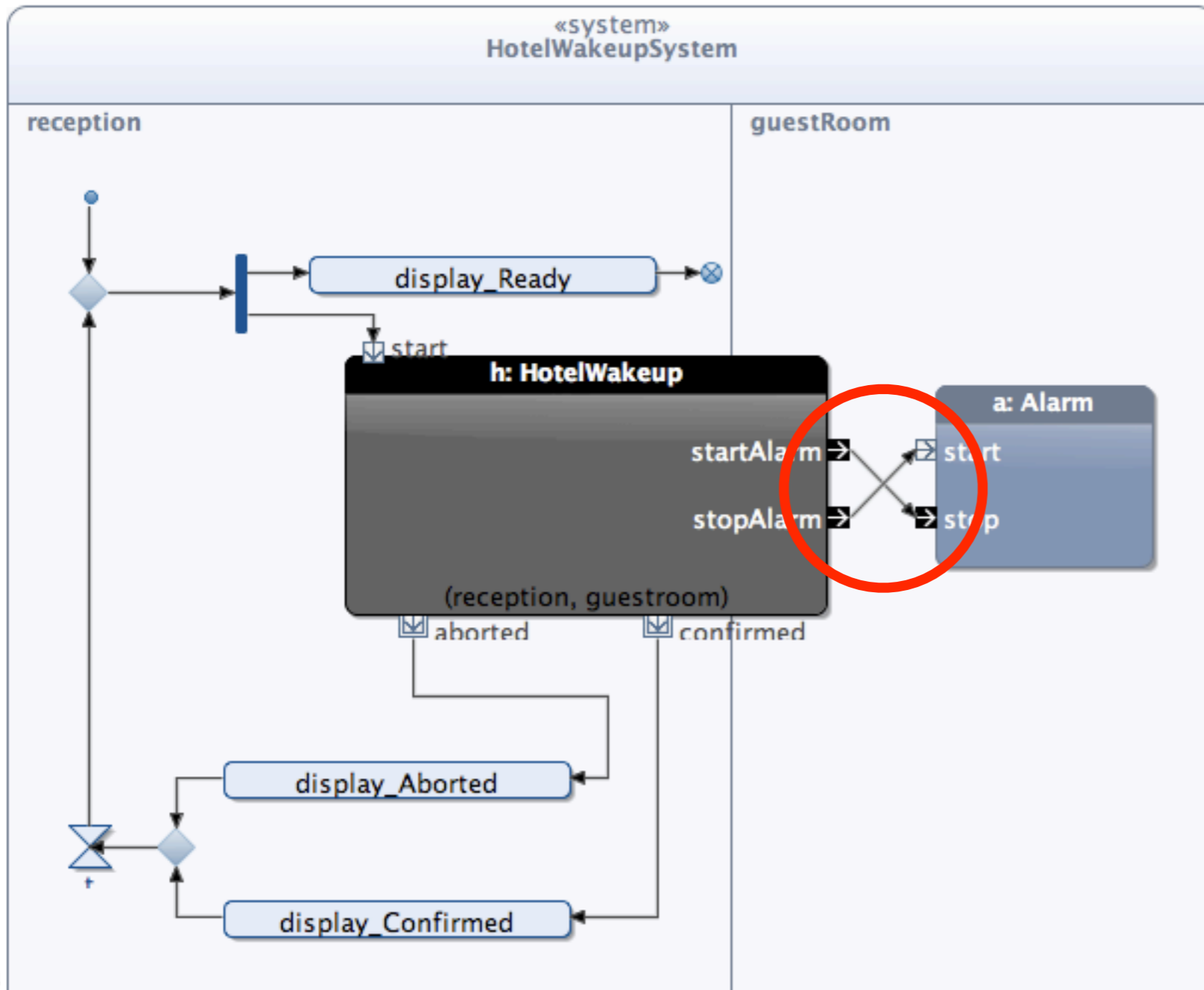
```

aborted ≜
  ∧ h = "stopped" ∧ h' = "off"
    
```

```

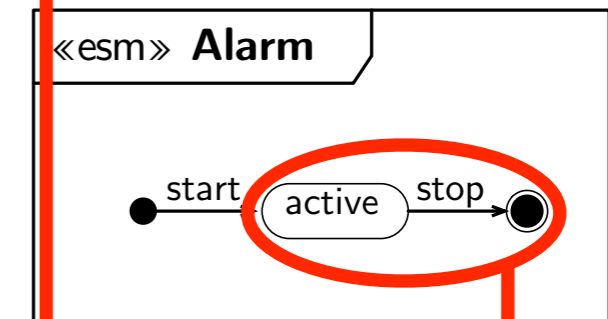
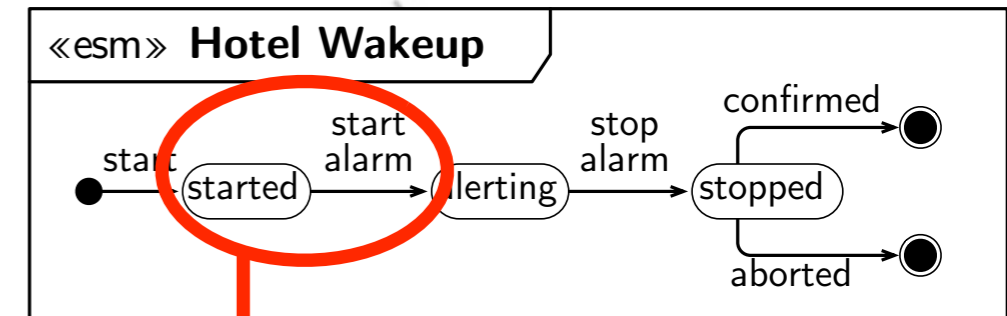
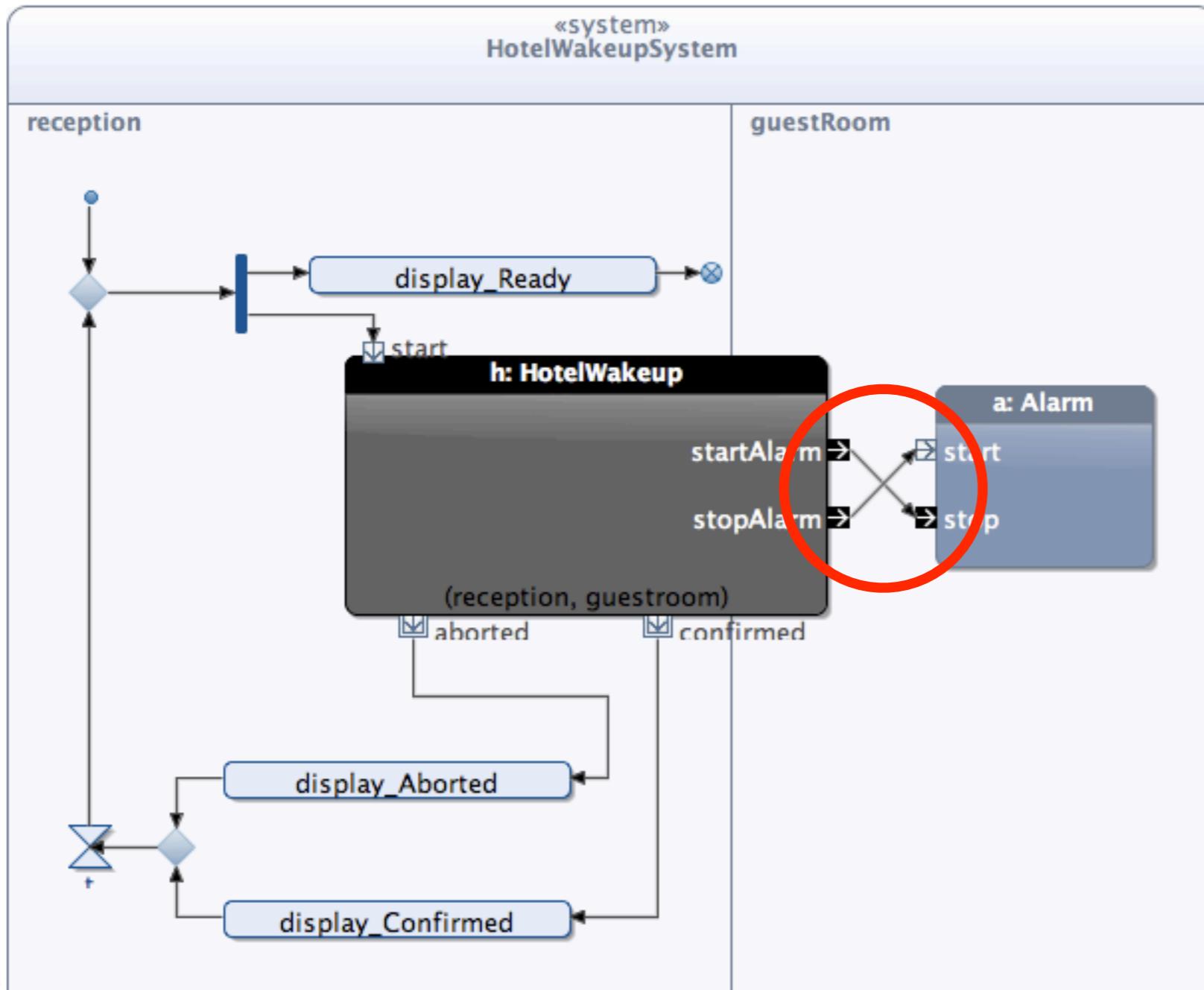
startAlert ≜
  ∧ h = "started" ∧ h' = "alerting"
  ∧ a = "off" ∧ a' = "active"
  ∧ UNCHANGED ⟨i, t⟩
    
```

Introduce an error



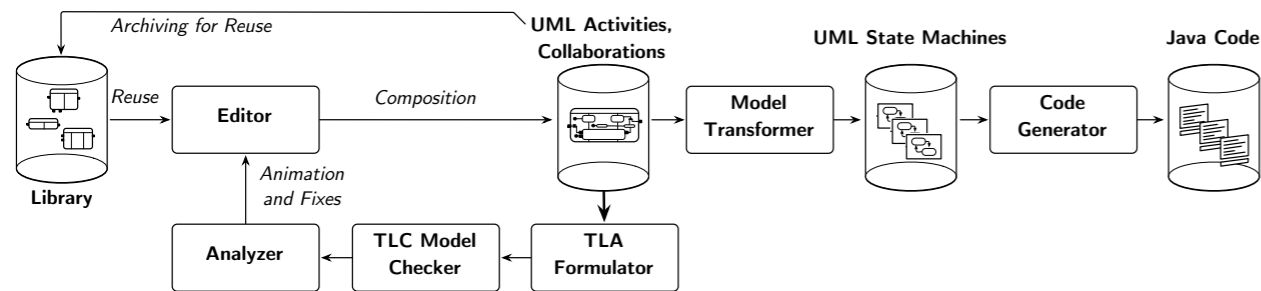
theorem__status_h_a ==
 $\square ((h = \text{"started"}) \Rightarrow (a = \text{"active"}))$

Introduce an error



theorem__status_h_a ==
 $\square ((h = \text{"started"}) \Rightarrow (a = \text{"active"}))$

TLC error trace



Error: Invariant theorem_status_h_a is violated. The behavior up to this point is:

STATE 1: <Initial predicate>

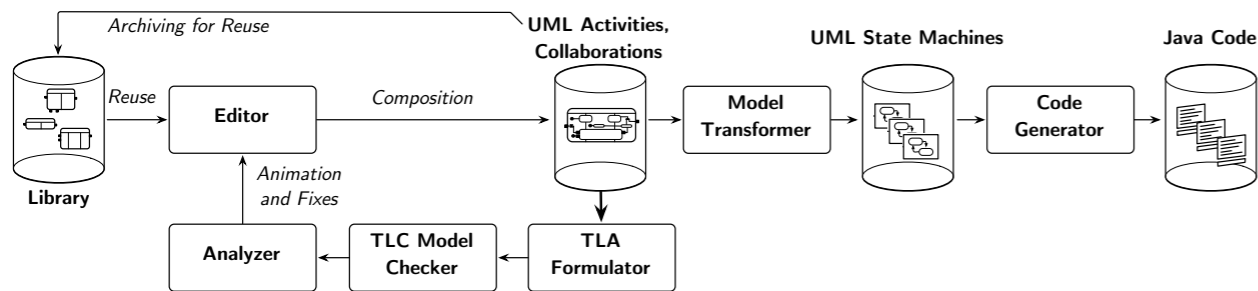
```
& h_counter = 0
& a_counter = 0
& t_counter = 0
& a = "_initial"
& _status = "pre_execution"
& h = "_initial"
& display_Aborted_counter = 0
& t = 0
& display_Confirmed_counter = 0
& display_Ready_counter = 0
```

STATE 2: <Action line 46, col 3 to line 55, col 60 of module HotelWakeupSystem>

```
& h_counter = 1
& a_counter = 0
& t_counter = 0
& a = "_initial"
& _status = "executing"
& h = "started"
& display_Aborted_counter = 0
& t = 0
& display_Confirmed_counter = 0
& display_Ready_counter = 1
```

2 states generated, 2 distinct states found, 1 states left on queue.
The depth of the complete state graph search is 2.

TLC error trace



`theorem__status_h_a ==`
`[] ((h = "started") => (a = "active"))`

Error: Invariant `theorem__status_h_a` is violated. The behavior up to this point is:

STATE 1: <Initial predicate>

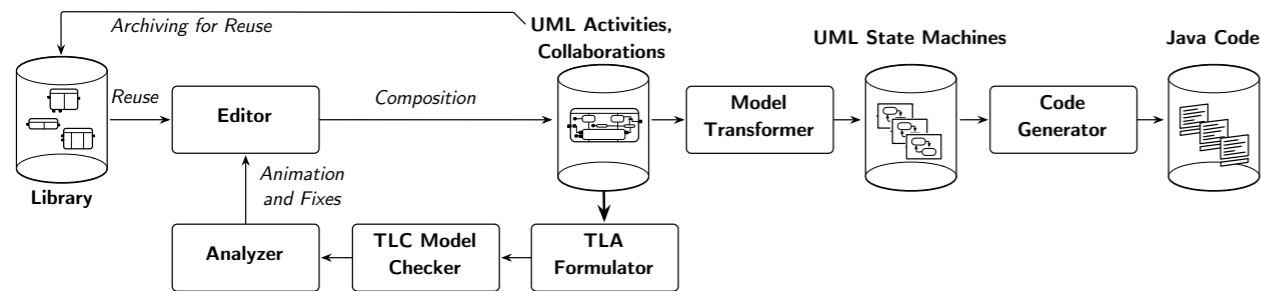
```
& h_counter = 0
& a_counter = 0
& t_counter = 0
& a = "_initial"
& _status = "pre_execution"
& h = "_initial"
& display_Aborted_counter = 0
& t = 0
& display_Confirmed_counter = 0
& display_Ready_counter = 0
```

STATE 2: <Action line 46, col 3 to line 55, col 60 of module HotelWakeupSystem>

```
& h_counter = 1
& a_counter = 0
& t_counter = 0
& a = "_initial"
& _status = "executing"
& h = "started"
& display_Aborted_counter = 0
& t = 0
& display_Confirmed_counter = 0
& display_Ready_counter = 1
```

2 states generated, 2 distinct states found, 1 states left on queue.
The depth of the complete state graph search is 2.

TLC error trace



```
theorem__status_h_a ==  
[] (( h = "started" ) => ( a = "active" ))
```

Error: Invariant theorem__status_h_a is violated. The behavior up to this point is:

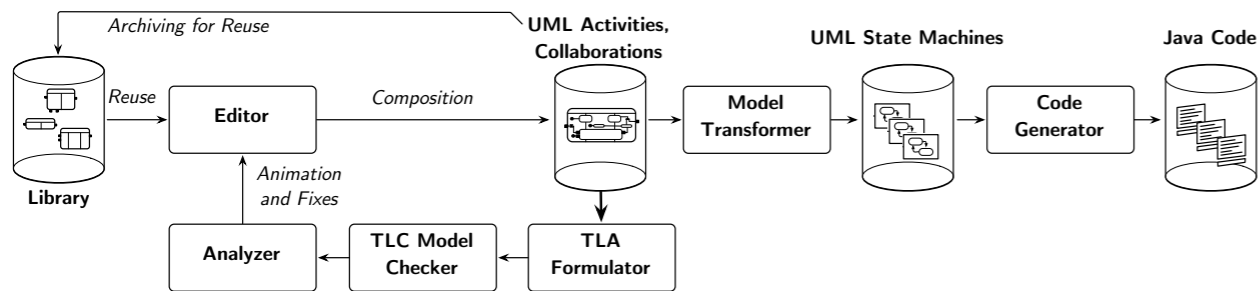
STATE 1: <Initial predicate>

```
^ h_counter = 0  
^ a_counter = 0  
^ t_counter = 0  
^ a = "_initial"  
^ _status = "pre_execution"  
^ h = "_initial"  
^ display_Aborted_counter = 0  
^ t = 0  
^ display_Confirmed_counter = 0  
^ display_Ready_counter = 0
```

```
STATE 2: <Action line 46, col 3 to line 55, col 60 of module HotelWakeupSystem>  
^ h_counter = 1  
^ a_counter = 0  
^ t_counter = 0  
^ a = "_initial"  
^ _status = "executing"  
^ h = "started"  
^ display_Aborted_counter = 0  
^ t = 0  
^ display_Confirmed_counter = 0  
^ display_Ready_counter = 1
```

```
2 states generated, 2 distinct states found, 1 states left on queue.  
The depth of the complete state graph search is 2.
```

TLC error trace



theorem__status_h_a ==
 $\square ((h = \text{"started"}) \Rightarrow (a = \text{"active"}))$

```

Error: Invariant theorem__status_h_a is violated. The behavior up to this point is:
STATE 1: <Initial predicate>
^ h_counter = 0
^ a_counter = 0
^ t_counter = 0
^ a = "_initial"
^ _status = "pre_execution"
^ h = "_initial"
^ display_Aborted_counter = 0
^ t = 0
^ display_Confirmed_counter = 0
^ display_Ready_counter = 0
    
```

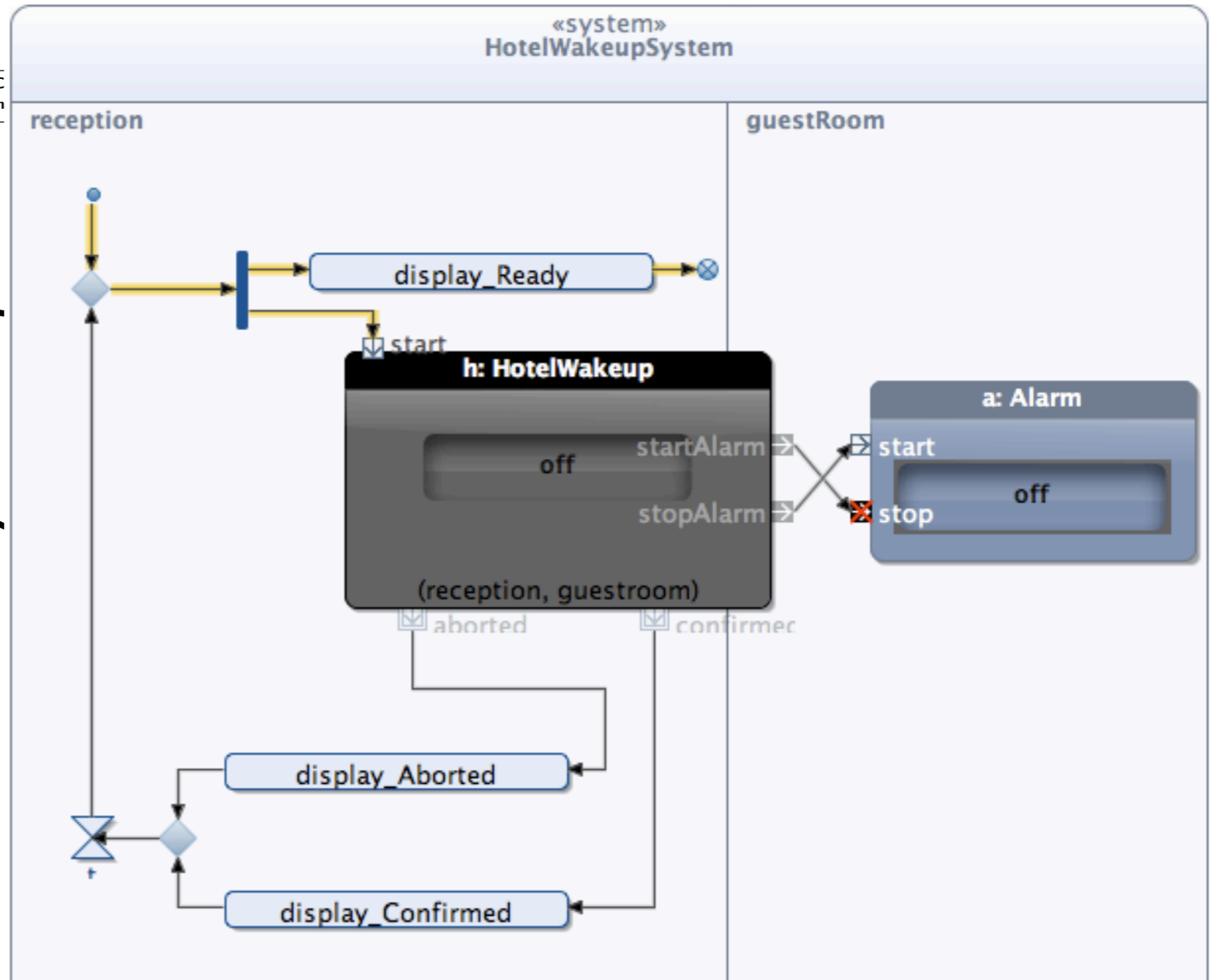
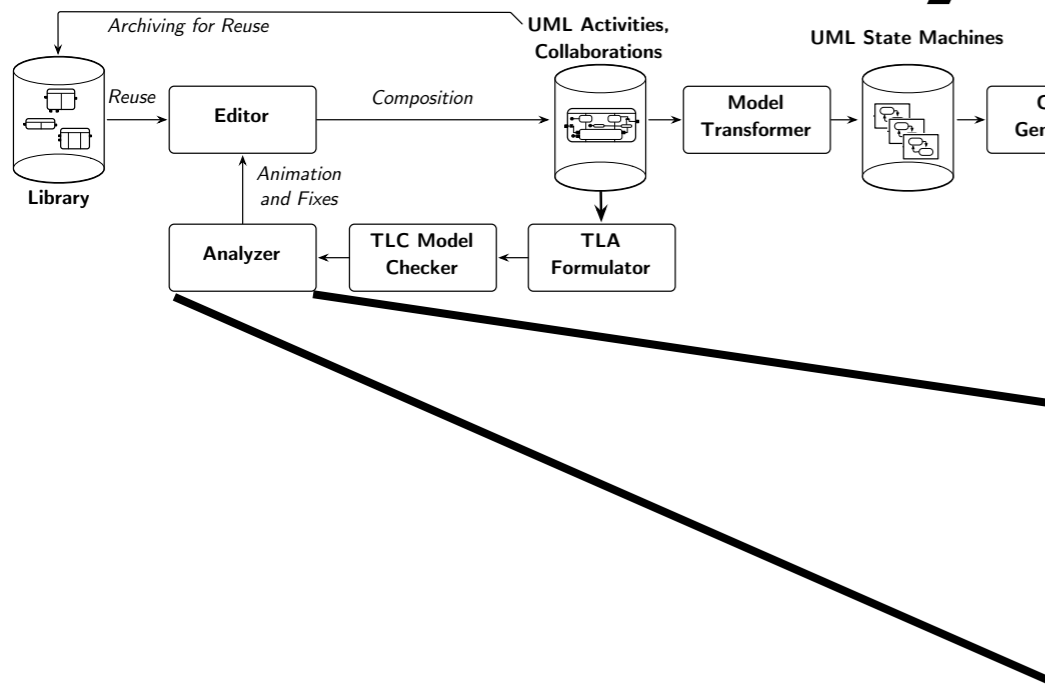
STATE 2: <Action line 46, col 3 to line 55, col 60 of module HotelWakeupSystem>

```

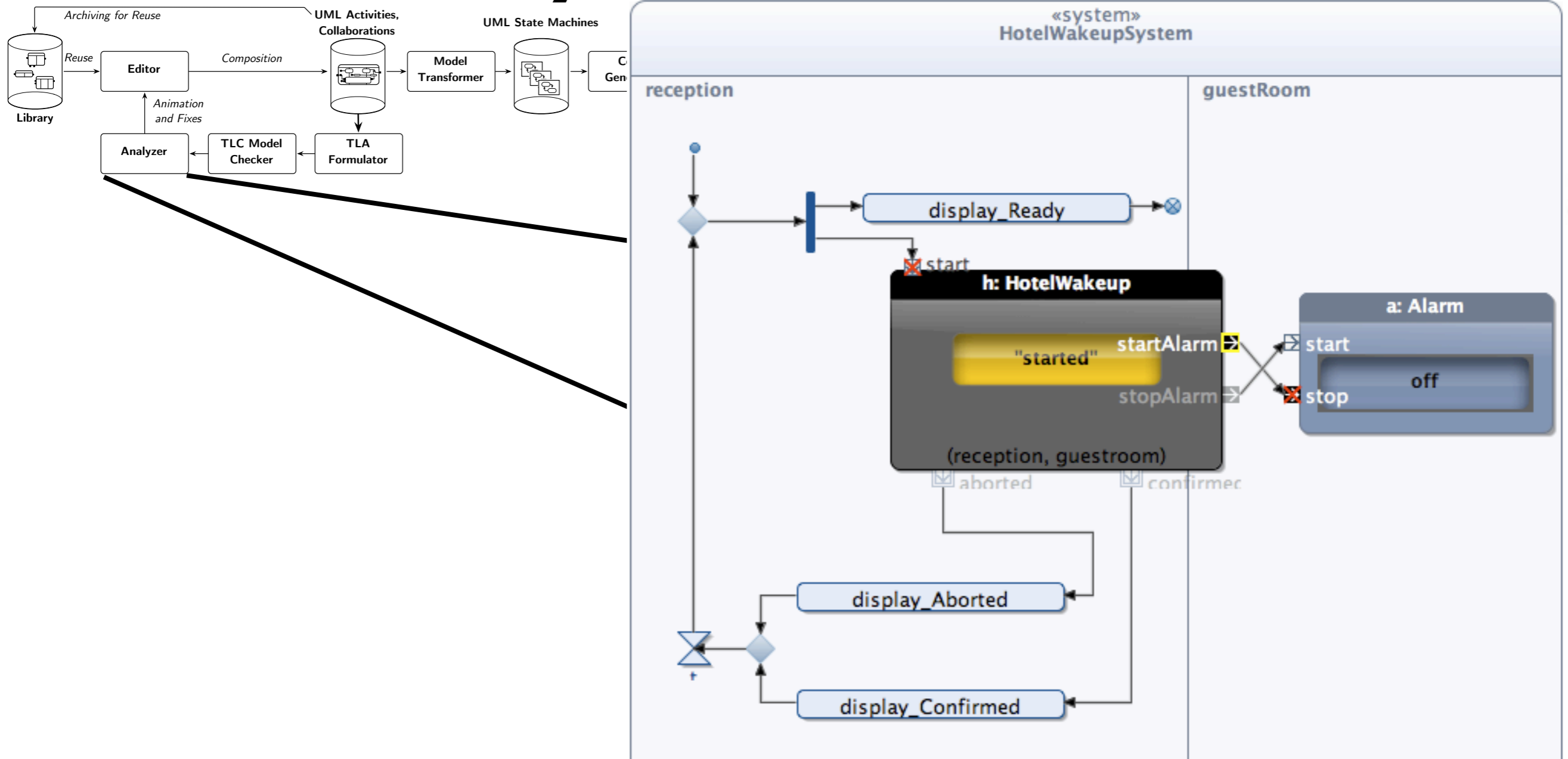
^ h_counter = 1
^ a_counter = 0
^ t_counter = 0
^ a = "_initial"
^ _status = "executing"
^ h = "started"
^ display_Aborted_counter = 0
^ t = 0
^ display_Confirmed_counter = 0
^ display_Ready_counter = 1
    
```

2 states generated, 2 distinct states found, 1 states left on queue.
 The depth of the complete state graph search is 2.

Analyzer feedback

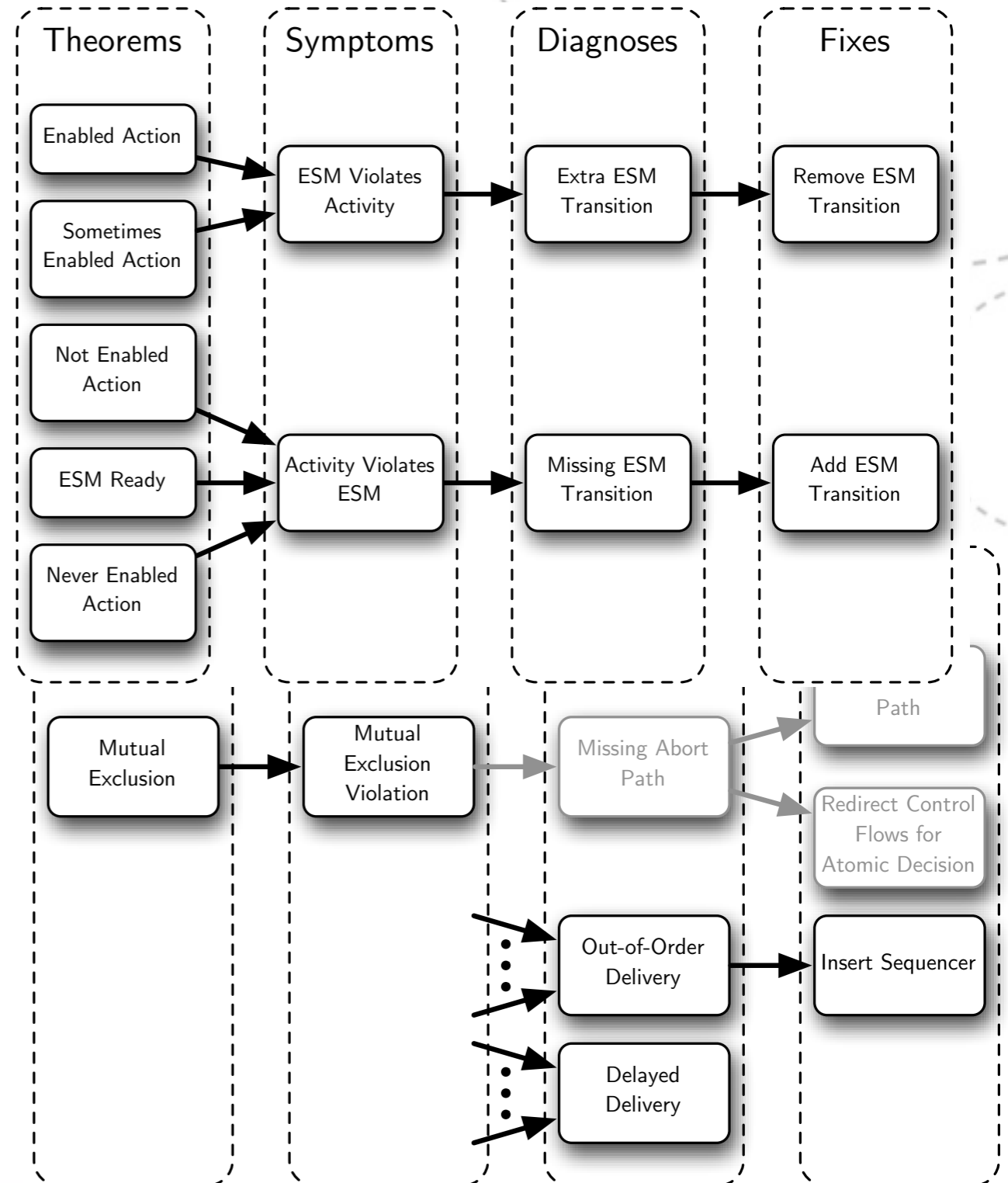
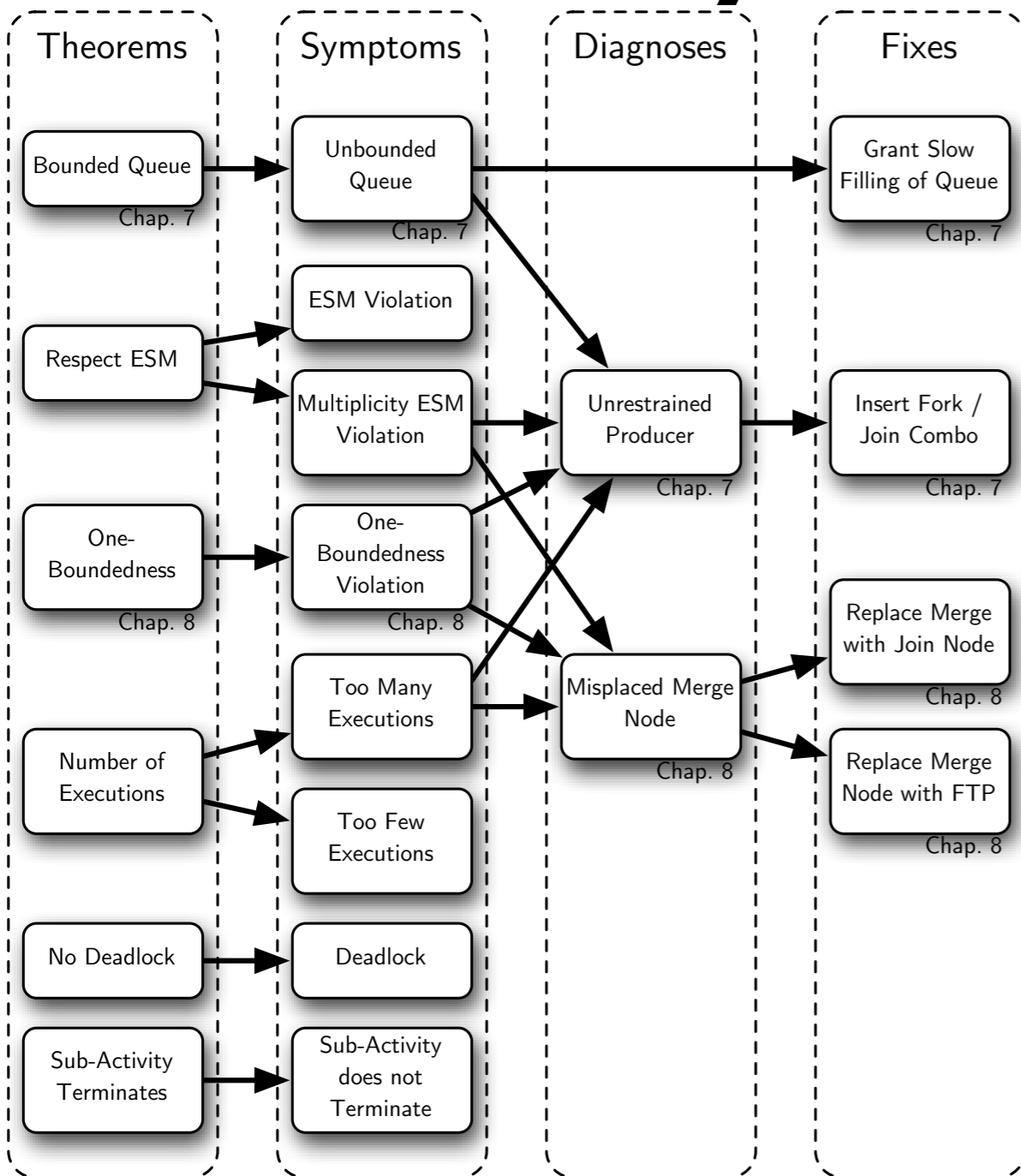


Analyzer feedback



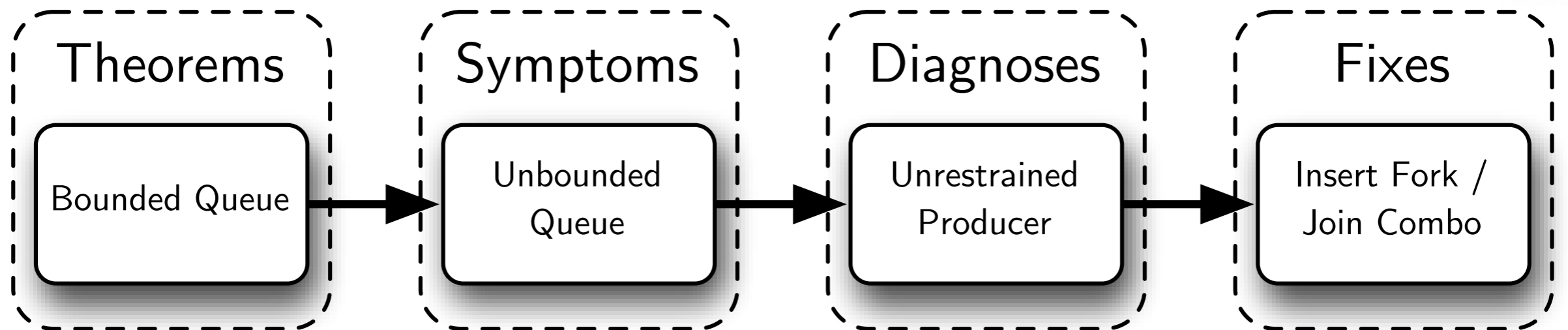
Analysis framework

Analysis framework

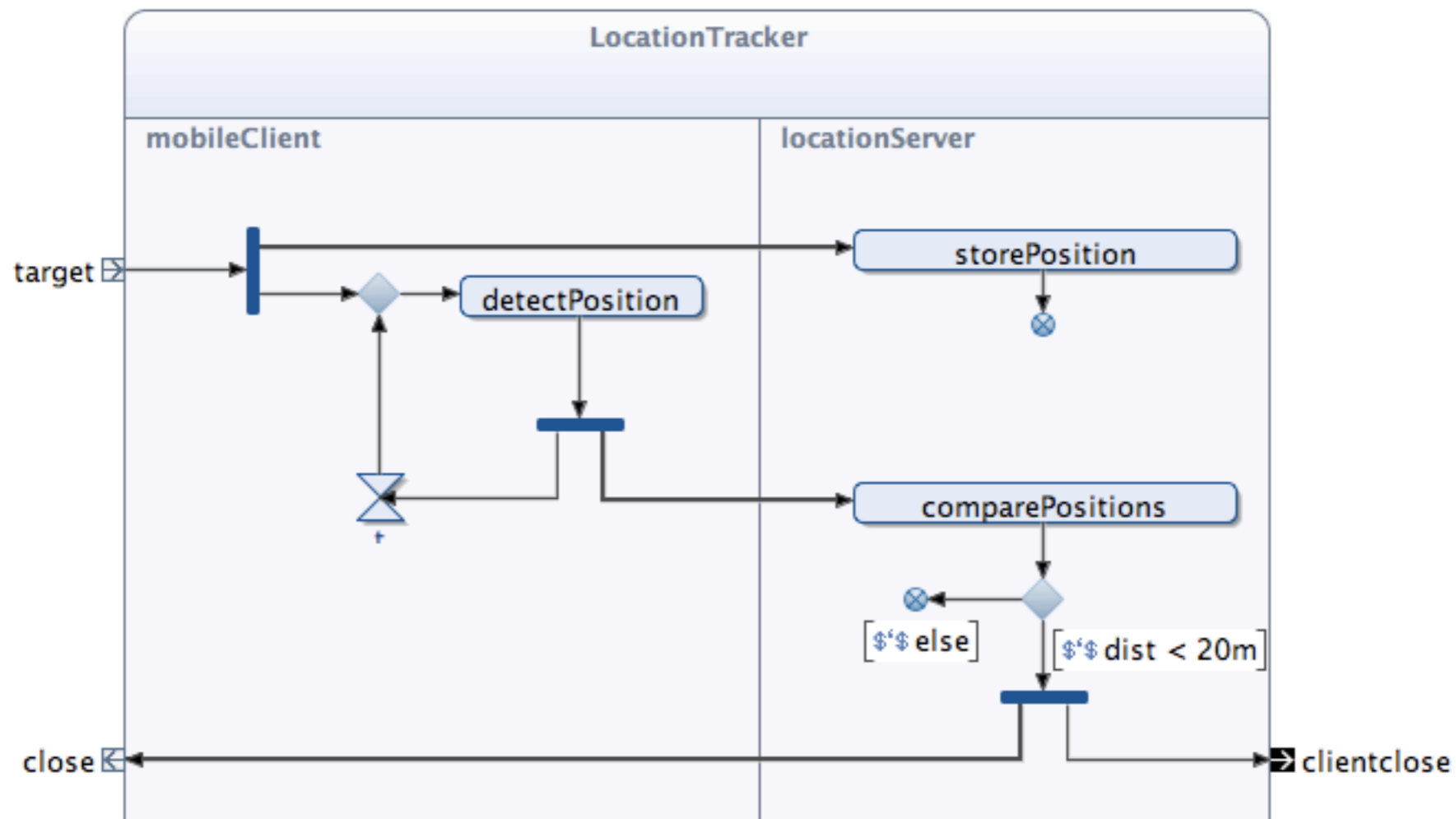


Analysis framework

Analysis framework

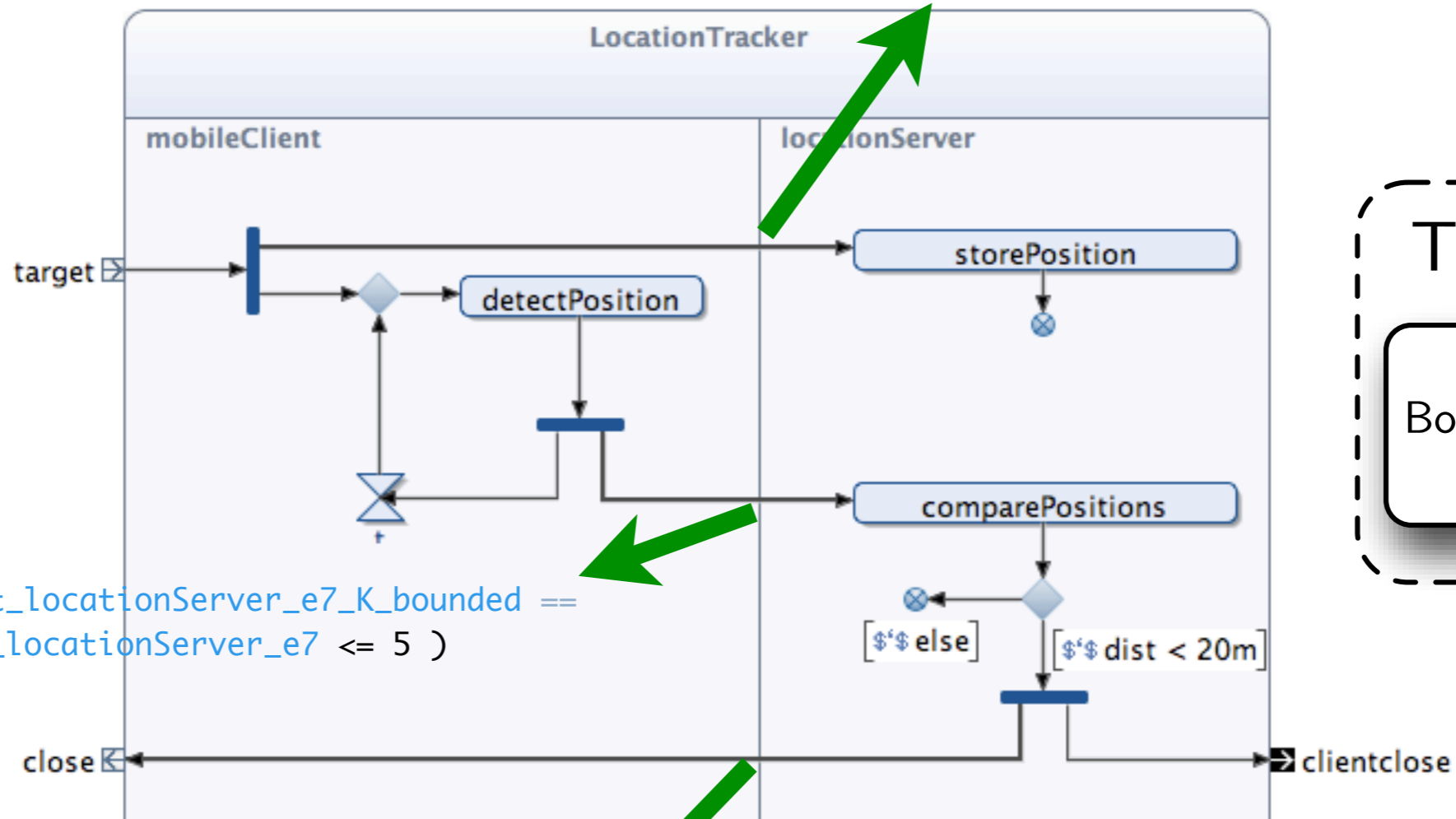


Example 2



Example 2

theorem_mobileClient_locationServer_e3_K_bounded ==
 $\square (\text{mobileClient_locationServer_e3} \leq 5)$



Theorems

Bounded Queue

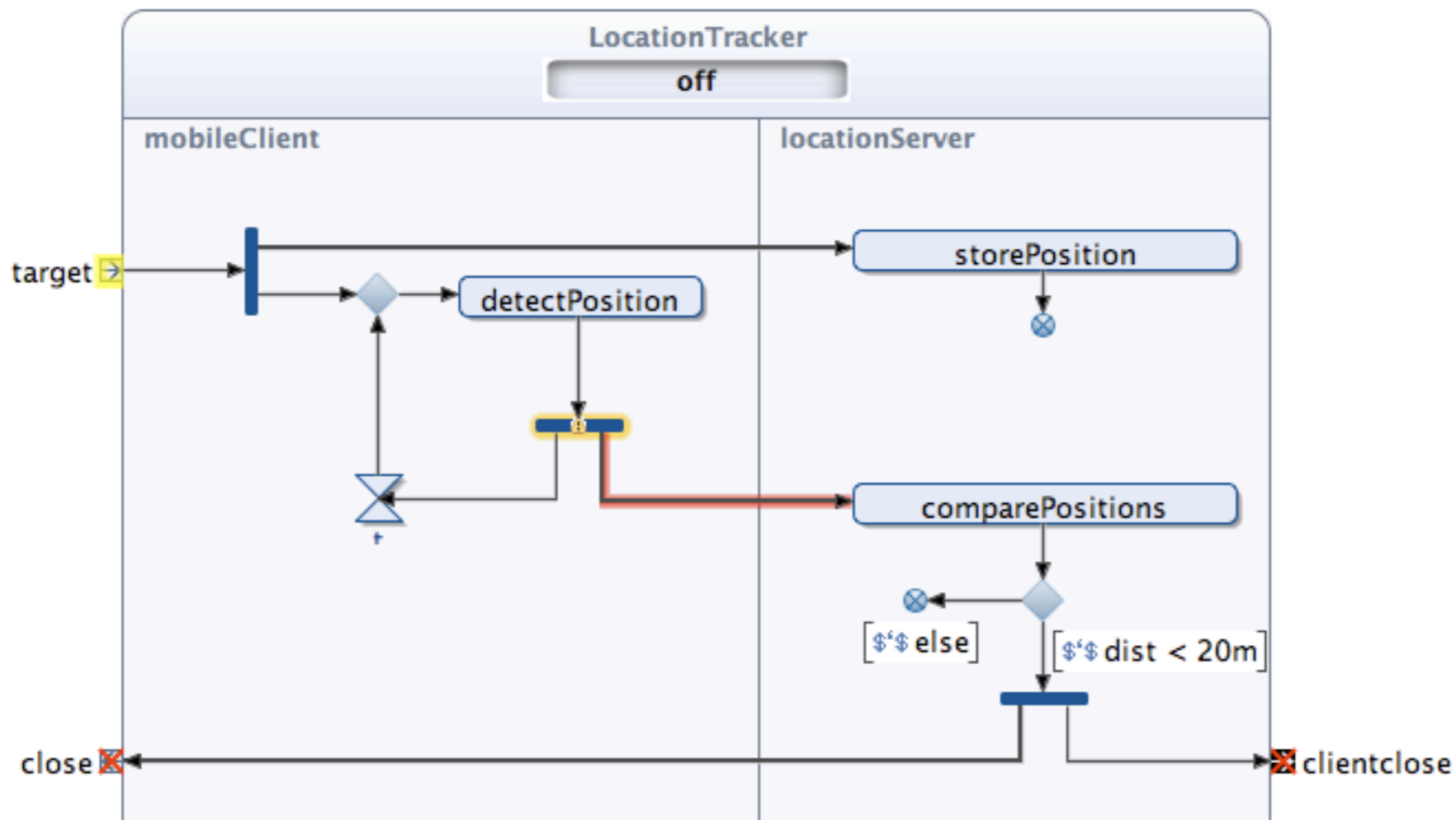
theorem_mobileClient_locationServer_e7_K_bounded ==
 $\square (\text{mobileClient_locationServer_e7} \leq 5)$

theorem_locationServer_mobileClient_e12_K_bounded ==
 $\square (\text{locationServer_mobileClient_e12} \leq 5)$

Trace (State I)

Internal Behavior The queue mobileClient locationServer e7 is not bounded.

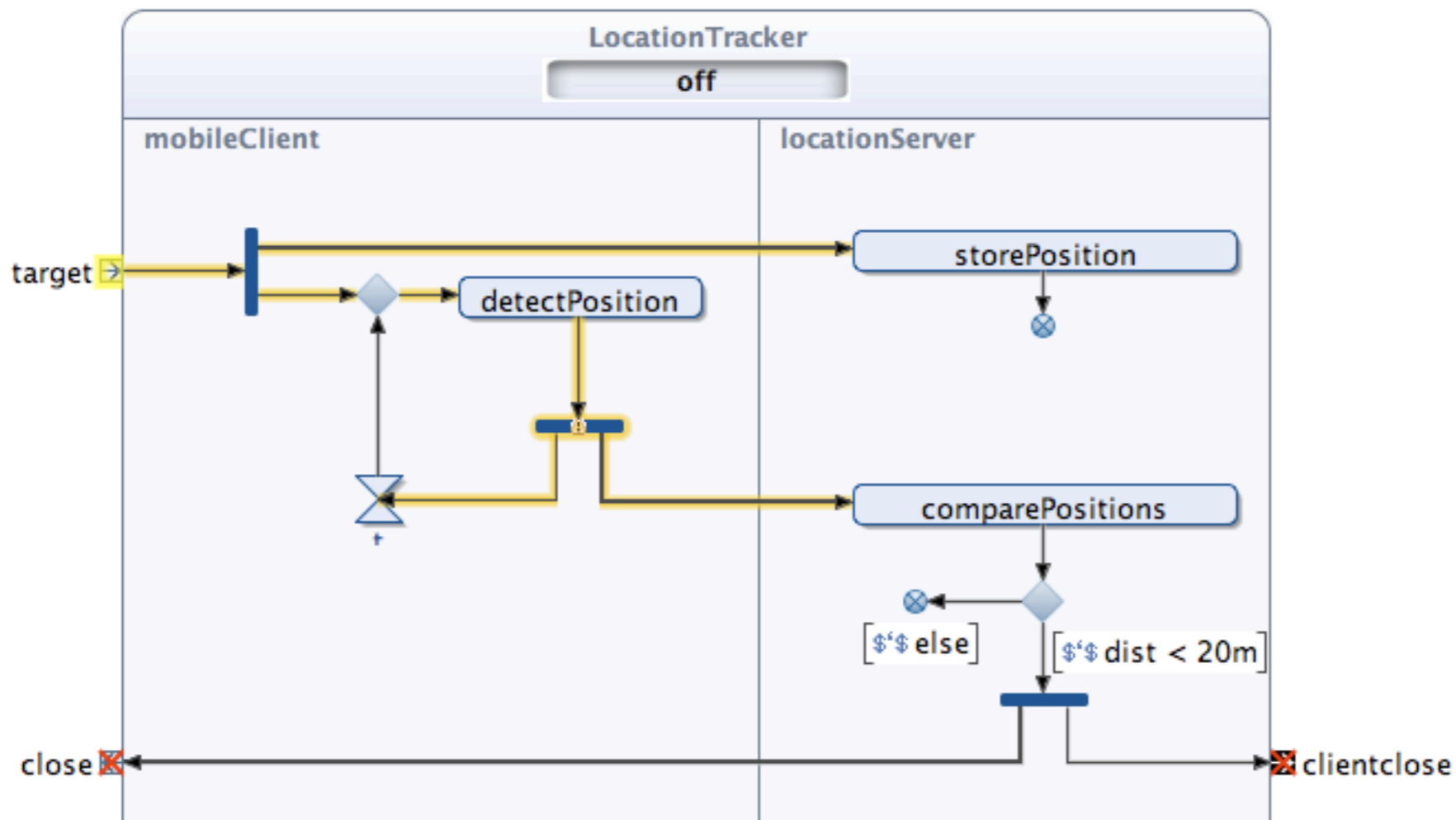
Trace: 1 / 7



Trace (Transition 1)

Internal Behavior The queue mobileClient locationServer e7 is not bounded.

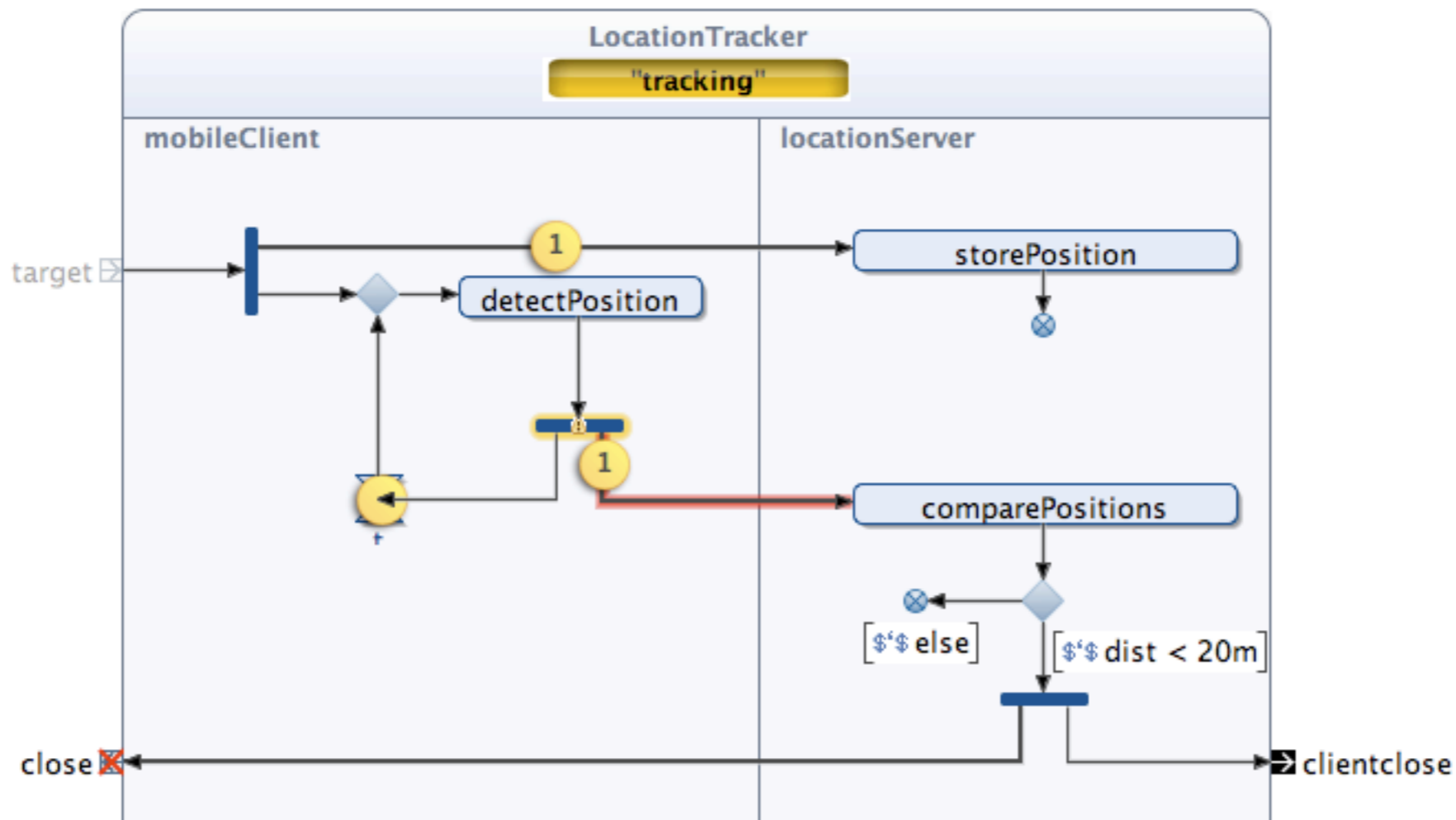
Trace: 1 / 7



Trace (State 2)

Internal Behavior The queue mobileClient locationServer e7 is not bounded.

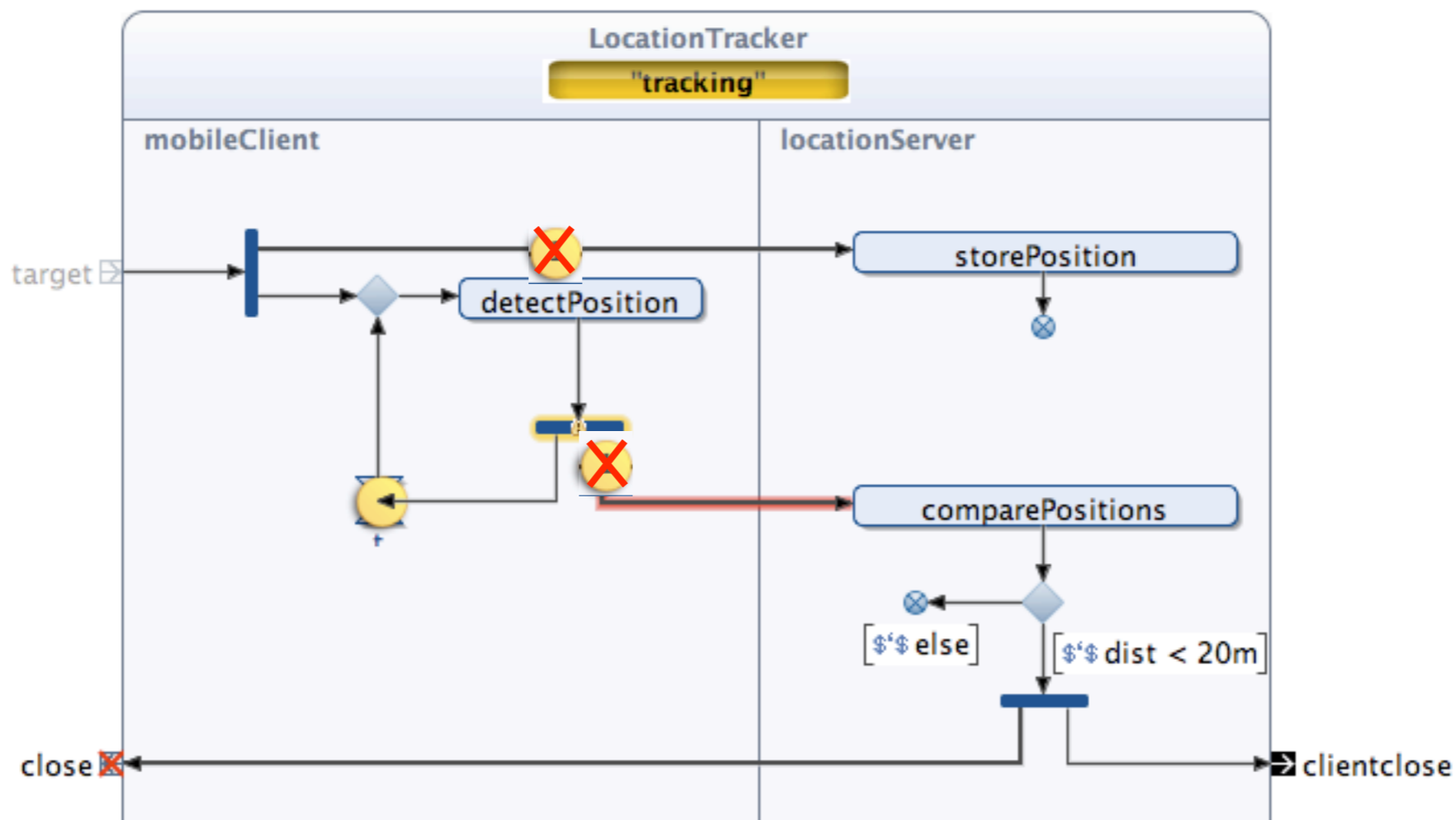
Trace: [Navigation icons] 2 / 7 [Close icon]



Trace (State 2)

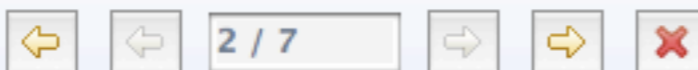
Internal Behavior The queue mobileClient locationServer e7 is not bounded.

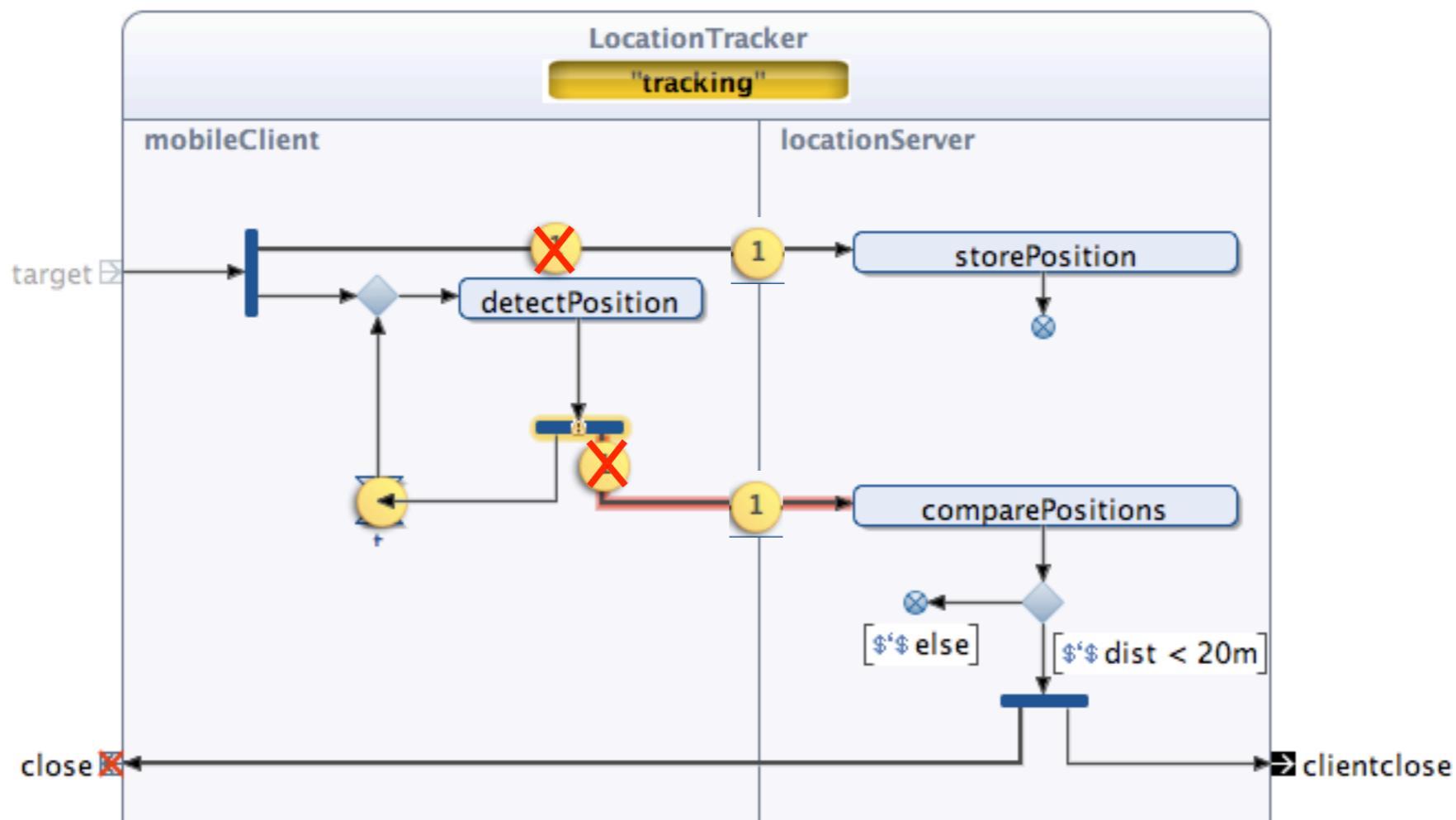
Trace: 2 / 7



Trace (State 2)

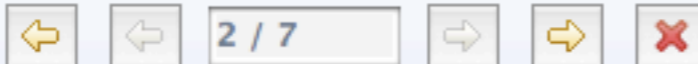
Internal Behavior The queue mobileClient locationServer e7 is not bounded.

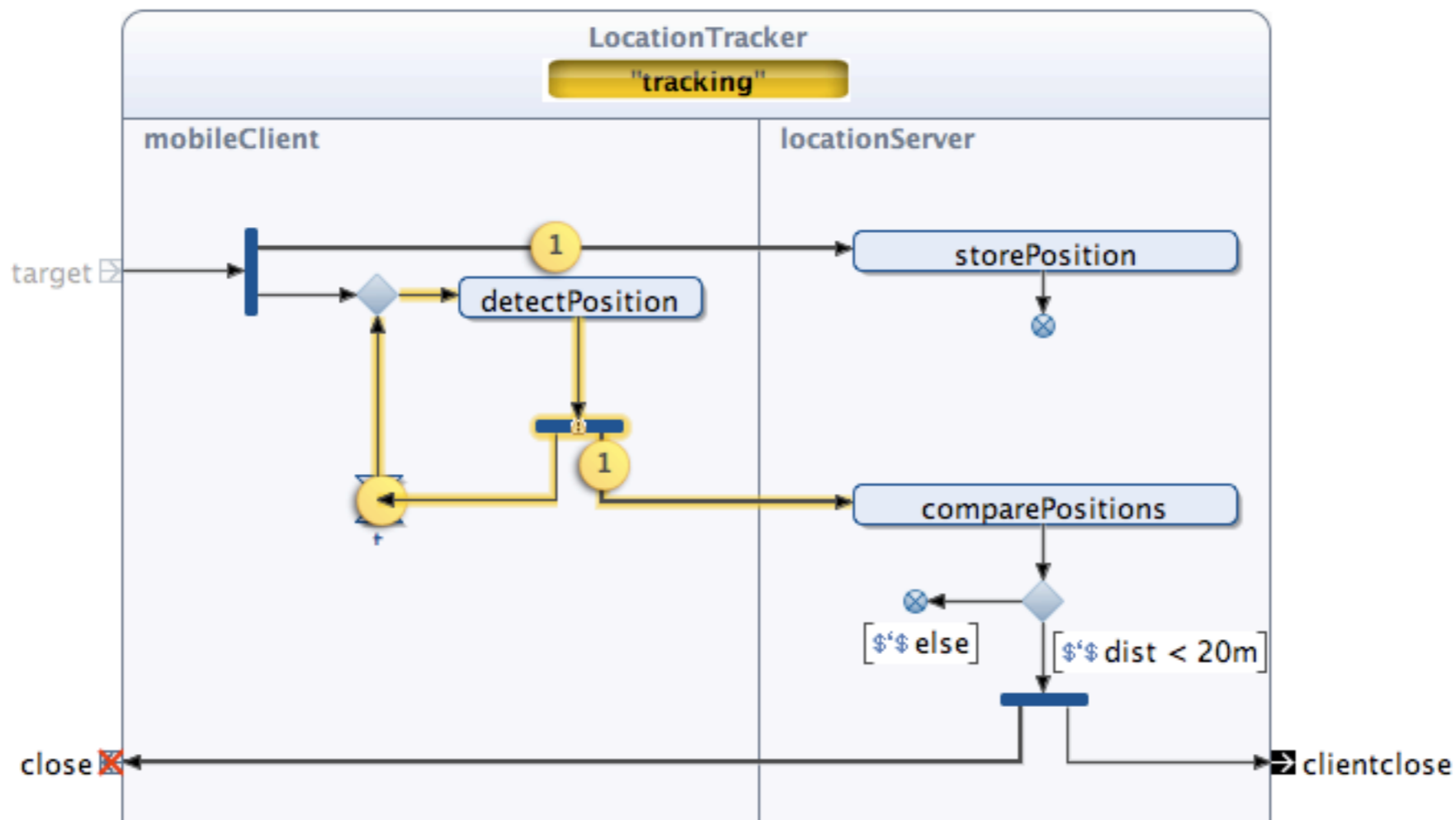
Trace:  2 / 7



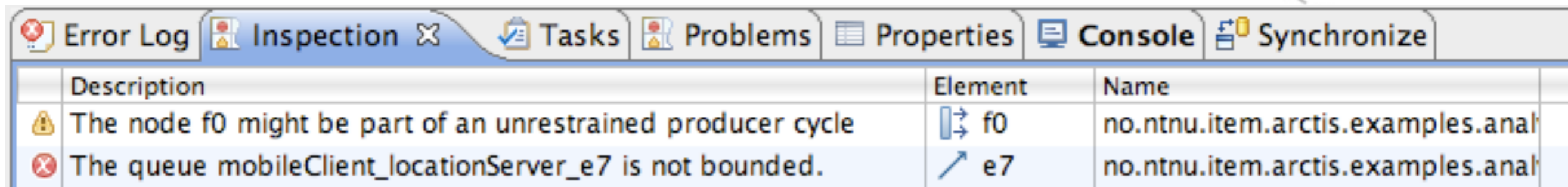
Trace (Transition 2)





Internal Behavior The queue mobileClient locationServer e7 is not bounded.

Trace:  2 / 7



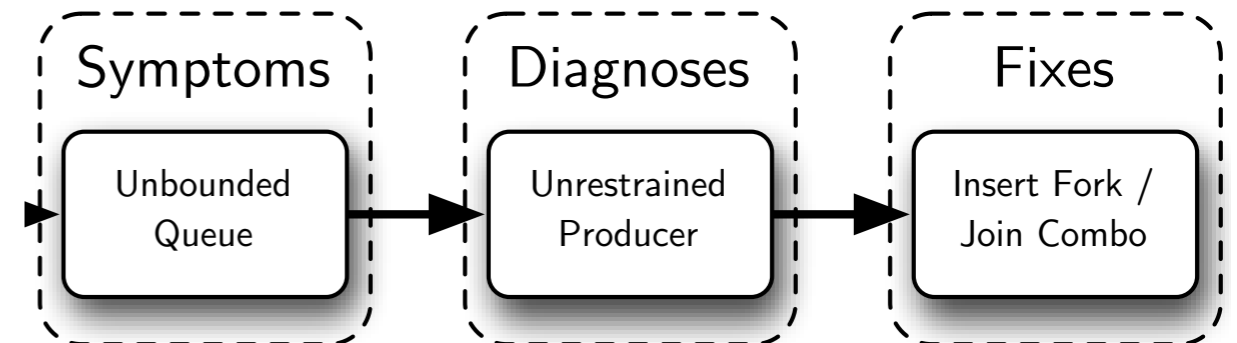
Symptom, diagnosis and fix



Description	Element	Name
 The node f0 might be part of an unrestrained producer cycle	 f0	no.ntnu.item.arctis.examples.anal
 The queue mobileClient_locationServer_e7 is not bounded.	 e7	no.ntnu.item.arctis.examples.anal

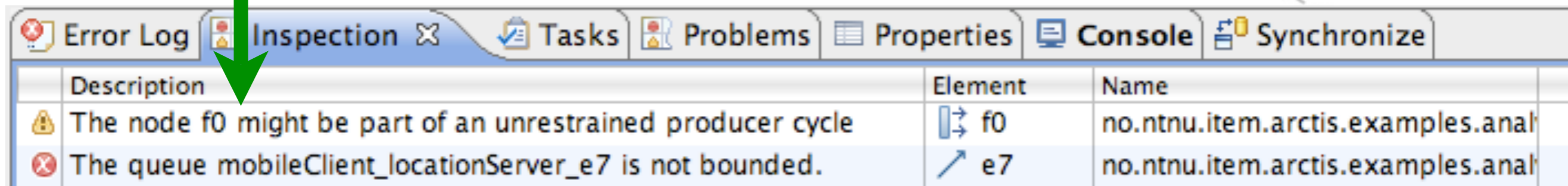
▼ So what should I do?

-  [Insert Fork/Join Combo](#)
Inserts a fork right after the overflown node / crossing, a join right after the triggering node / crossing and an edge between them. This creates a feedback loop.



Symptom, diagnosis and fix

Diagnosis



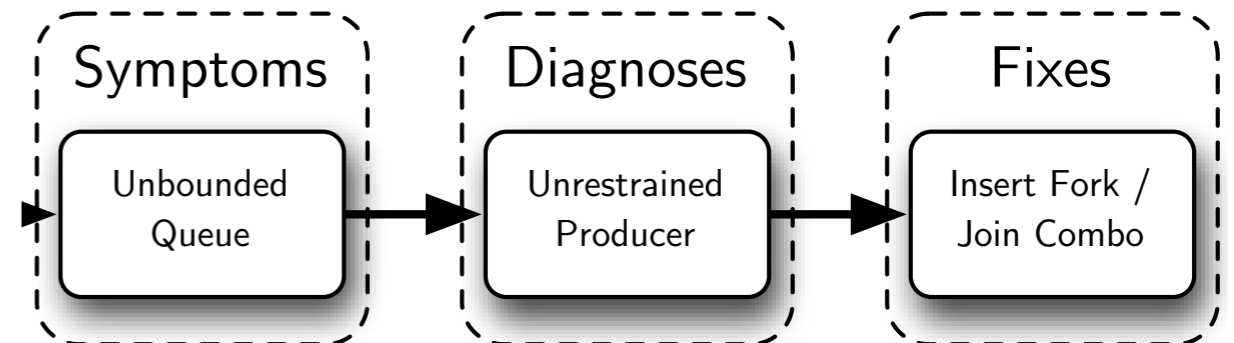
Description	Element	Name
⚠ The node f0 might be part of an unrestrained producer cycle	f0	no.ntnu.item.arctis.examples.anal
✖ The queue mobileClient_locationServer_e7 is not bounded.	e7	no.ntnu.item.arctis.examples.anal

So what should I do?

- 💡 [Insert Fork/Join Combo](#)
Inserts a fork right after the overflown node / crossing, a join right after the triggering node / crossing and an edge between them. This creates a feedback loop.

Symptom

Fix



Full screenshot

The screenshot displays a UML modeling tool interface. The main window shows a UML diagram for a **LocationTracker** system, divided into two lifelines: **mobileClient** and **locationServer**. The **mobileClient** lifeline starts with a **target** message, leading to a **detectPosition** activity. This activity sends a message to the **locationServer** lifeline, which performs **storePosition** and then **comparePositions**. The **comparePositions** activity has a decision diamond with two paths: **[\$\$ else]** and **[\$\$ dist < 20m]**. The **[\$\$ dist < 20m]** path leads to a join node, which then sends a **close** message back to the **mobileClient** lifeline. A red arrow points to a node labeled **f0** in the **comparePositions** activity, indicating a warning. The **Diagnosis** panel on the right provides details about this warning.

Internal Behavior The queue **mobileClient_locationServer_e7** is not bounded.

Trace: 1 / 7

Diagnosis

Unknown Source

What's wrong?

In a few words:
The node **f0** might be part of an unrestrained producer cycle

Which elements are affected?

Name: **f0**

Path: **no.ntnu.item.arctis.examples.analyzertests::LocationTracker::**

Type: **org.eclipse.uml2.uml.Fork**

Project: **P/no.ntnu.item.arctis.exarr**

So what should I do?

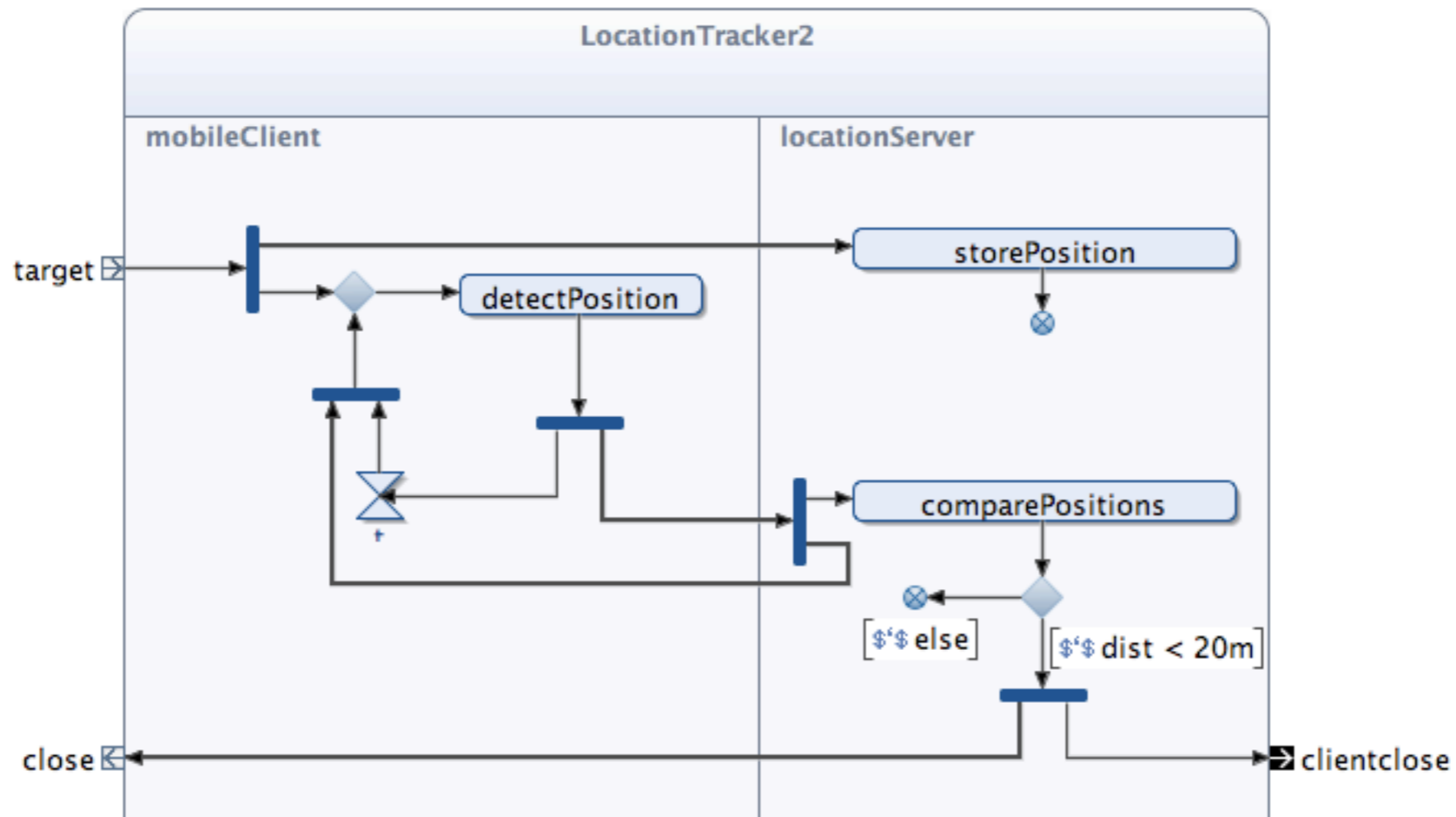
Insert Fork/Join Combo
Inserts a fork right after the overflow node / crossing, a join right after the triggering node / crossing and an edge between them. This creates a feedback loop.

More Help?

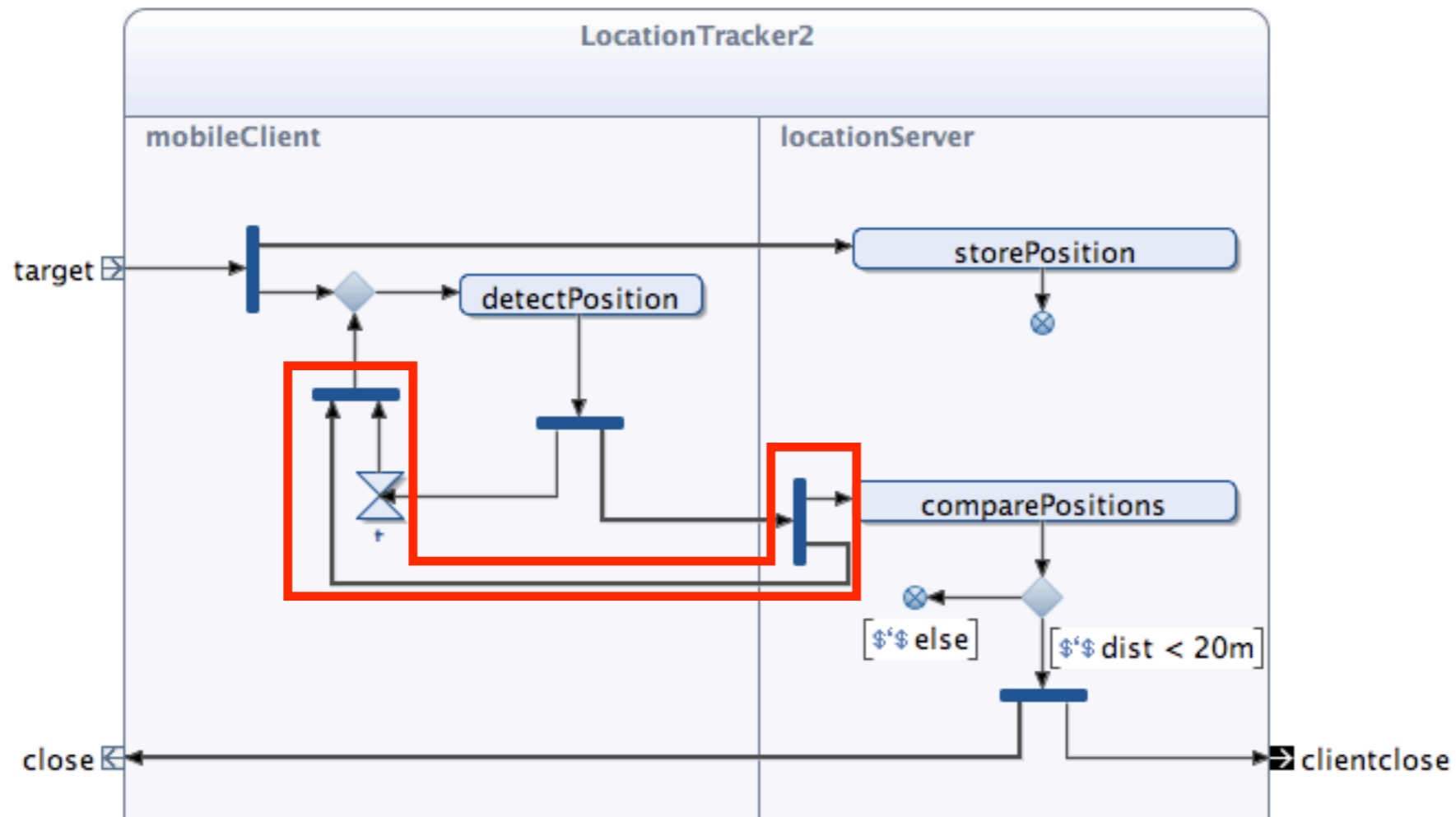
No help available for context .
More help:
[Report Errors with Inspectors](#)
[Contribute your own Inspectors](#)

Description	Element	Name
The node f0 might be part of an unrestrained producer cycle	f0	no.ntnu.item.arctis
The queue mobileClient_locationServer_e7 is not bounded.	e7	no.ntnu.item.arctis

Fix applied



Fix applied



Bibliography

- Arctis: <http://www.item.ntnu.no/arctis>
- SPACE
 - Kraemer, F.A. & Herrmann, P. (2006), Service Specification by Composition of Collaborations --- An Example, in 'Proceedings of the 2006 WI-IAT Workshops (2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology)', pp. 129--133.
 - Kraemer, F.A.; Bræk, R. & Herrmann, P. (2007), Synthesizing Components with Sessions from Collaboration-Oriented Service Specifications, in SDL 2007, volume 4745 of Lecture Notes in computer Science, pages 166–185.
 - Kraemer, F.A.; Slåtten, V. & Herrmann, P. (2007), Engineering Support for UML Activities by Automated Model-Checking --- An Example, in Proceedings of the 4th International Workshop on Rapid Integration of Software Engineering Techniques (RISE)
- Formulator and Analyzer
 - Slåtten, V. (2007), 'Model Checking Collaborative Service Specifications in TLA with TLC', Project Thesis, Norwegian University of Science and Technology, Trondheim, Norway.
 - Slåtten, V. (2008), 'Automatic Detection and Correction of Flaws in Service Specifications', Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.