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?

Figures from Kraemer, F.A.; Slåtten, V. & Herrmann, P (2007)
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

---

www.ntnu.no

Thursday, 3 July, 2008
External State Machines

How do services interact?

Hotel Wakeup System

- Reception
  - Display "Ready"
  - Start
  - Aborted
  - Confirmed

- Guest Room
  - Display "Aborted"
  - Display "Confirmed"
  - Start alarm
  - Stop alarm

Hotel Wakeup

- Start
  - Started
  - Alerting
  - Confirmed
  - Stopped
  - Aborted

Alarm

- Start
  - Active
  - Stop

How does each service behave?

Figures from Kraemer, F.A.; Slåtten, V. & Herrmann, P (2007)
Arctis tool suite

Diagram:

- Library
  - Reuse
  - Archiving for Reuse

- Editor
  - Composition
  - UML Activities, Collaborations

- Model Transformer
- UML State Machines

- Code Generator
- Java Code
Arctis tool suite

![Diagram of Arctis tool suite]

Project thesis
Arctis tool suite

Archiving for Reuse

Reuse

Library

Editor

Composition

UML Activities, Collaborations

Analyzer

TLC Model Checker

TLA Formulator

Model Transformer

UML State Machines

Code Generator

Java Code

Master’s thesis

Project thesis
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"
Hotel Wakeup System

**tor**

- **reception**
  - display "Ready"
  - start
    - aborted
    - confirmed
  - stop alarm
  - start alarm

- **guest room**
  - a: Alarm
    - start
    - stop
    - aborted
    - confirmed
  - display "Confirmed"
  - display "Aborted"

**Hotel Wakeup**

**Alarm**

- start
- stop
- alerted
- confirmed
- aborted

---

**Module HotelWakeSystem**

EXTENDS Naturals

**Variables**

i, t, h, a

**Init**

\[i = 1 \land t = 0\]
\[h = \text{"off"} \land a = \text{"off"}\]

**Initial**

\[i = 1 \land i' = 0\]
\[h = \text{"off"} \land h' = \text{"started"}\]
\[\land \text{UNCHANGED} \langle i, t \rangle\]

**start**

\[h = \text{"started"} \land h' = \text{"alerting"}\]
\[a = \text{"off"} \land a' = \text{"active"}\]
\[\land \text{UNCHANGED} \langle i, t \rangle\]

**stop**

\[h = \text{"alerting"} \land h' = \text{"stopped"}\]
\[a = \text{"active"} \land a' = \text{"off"}\]
\[\land \text{UNCHANGED} \langle i, t \rangle\]

**aborted**

\[h = \text{"stopped"} \land h' = \text{"off"}\]
**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 \langle a, t \rangle

startAlert ≡
∧ h = "started" ∧ h' = "alerting"
∧ a = "off" ∧ a' = "active"
∧ UNCHANGED \langle i, t \rangle

stopAlert ≡
∧ h = "alerting" ∧ h' = "stopped"
∧ a = "active" ∧ a' = "off"
∧ UNCHANGED \langle i, t \rangle

aborted ≡
∧ h = "stopped" ∧ h' = "off"
```
module HotelWakeUpSystem

extends Naturals

variables i, t, h, a

init \( \triangleq \)
\( \& i = 1 \& t = 0 \)
\( \& h = "off" \& a = "off" \)

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

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

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

aborted \( \triangleq \)
\( \& h = "stopped" \& h' = "off" \)
Introduce an error

\[
\text{theorem\_status\_h\_a} \equiv \left( \left( h = \text{"started"} \right) \Rightarrow \left( a = \text{"active"} \right) \right)
\]
Introduce an error

\[
\text{theorem}\_\text{status}_h_a = (h = \text{"started"}) \Rightarrow (a = \text{"active"})
\]
Introduce an error

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

STATE 1: <Initial predicate>
\( h\_\text{counter} = 0 \)
\( a\_\text{counter} = 0 \)
\( t\_\text{counter} = 0 \)
\( a = "\_\text{initial}" \)
\( \_\text{status} = "\text{preexecution}" \)
\( h = "\_\text{initial}" \)
\( \text{display}\_\text{Aborted}\_\text{counter} = 0 \)
\( t = 0 \)
\( \text{display}\_\text{Confirmed}\_\text{counter} = 0 \)
\( \text{display}\_\text{Ready}\_\text{counter} = 0 \)

STATE 2: <Action line 46, col 3 to line 55, col 60 of module HotelWakeupsystem>
\( h\_\text{counter} = 1 \)
\( a\_\text{counter} = 0 \)
\( t\_\text{counter} = 0 \)
\( a = "\_\text{initial}" \)
\( \_\text{status} = "\text{executing}" \)
\( h = "\text{started}" \)
\( \text{display}\_\text{Aborted}\_\text{counter} = 0 \)
\( t = 0 \)
\( \text{display}\_\text{Confirmed}\_\text{counter} = 0 \)
\( \text{display}\_\text{Ready}\_\text{counter} = 1 \)

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>
\(\forall\ h\_counter = 0\)
\(\forall\ a\_counter = 0\)
\(\forall\ t\_counter = 0\)
\(\forall\ a = "\_initial"\)
\(\forall\ _status = "pre\_execution"\)
\(\forall\ h = "\_initial"\)
\(\forall\ display\_Aborted\_counter = 0\)
\(\forall\ t = 0\)
\(\forall\ display\_Confirmed\_counter = 0\)
\(\forall\ display\_Ready\_counter = 0\)

STATE 2: <Action line 46, col 3 to line 55, col 60 of module HotelWakeupSystem>
\(\forall\ h\_counter = 1\)
\(\forall\ a\_counter = 0\)
\(\forall\ t\_counter = 0\)
\(\forall\ a = "\_initial"\)
\(\forall\ _status = "executing"\)
\(\forall\ h = "\_initial"\)
\(\forall\ display\_Aborted\_counter = 0\)
\(\forall\ t = 0\)
\(\forall\ display\_Confirmed\_counter = 0\)
\(\forall\ display\_Ready\_counter = 1\)

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.
Archiving for Reuse

Composition in Kraemer et al.

A system of state machines we use temporal logic as well. Similar to our proceeding case, the same sequence of transitions follows as depicted in Sect. X. We have defined formal semantics for the initiative of actions in the inter-component interactions. Where actions are inter-component, we use temporal logic as well. Similar to our proceeding case, the same sequence of transitions follows as depicted in Sect. X. We have defined formal semantics for the initiative of actions in the inter-component interactions.

In the next section, we explain later. If a theorem is violated, the analyzer tries to identify possible reasons and presents an error report.
Analyzer feedback

Analyzer for Reuse
Analyzer

Archiving for Reuse

Library

Editor

Animation and Fixes

Composition

Model Transformer

TLA Formulator

TLC Model Checker

UML Activities, Collaborations

UML State Machines

Model

C

Components

House

Analyzer

Figure 10. Executable state machine as generated by Arctis

«system»
HotelWakeupSystem

reception
display_Ready

start

display_Aborted

display_Confirmed

guestRoom

startAlarm

stopAlarm

start

stop

waiting

Alarm

off

(breaking, guestroom)

(reception, guestroom)

"started"

end

start

stop

abort

confirm
Analysis framework
Analysis framework
Analysis framework

Theorems
- Bounded Queue

Symptoms
- Unbounded Queue

Diagnoses
- Unrestrained Producer

Fixes
- Insert Fork / Join Combo

Thursday, 3 July, 2008
Example 2
Example 2

\[
\text{theorem}_\text{mobileClient}_\text{locationServer}_\text{e3}_\text{K}_\text{bounded} == \top (\text{mobileClient}_\text{locationServer}_\text{e3} \leq 5)
\]

\[
\text{theorem}_\text{locationServer}_\text{mobileClient}_\text{e12}_\text{K}_\text{bounded} == \top (\text{locationServer}_\text{mobileClient}_\text{e12} \leq 5)
\]

\[
\text{theorem}_\text{mobileClient}_\text{locationServer}_\text{e7}_\text{K}_\text{bounded} == \top (\text{mobileClient}_\text{locationServer}_\text{e7} \leq 5)
\]
Trace (State 1)

The queue mobileClient.locationServer.e7 is not bounded.
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: 2 / 7
Trace (State 2)

Internal Behavior: The queue mobileClient.locationServer.e7 is not bounded.

Trace:

---

www.ntnu.no

Thursday, 3 July, 2008
Trace (State 2)

Internal Behavior: The queue mobileClient.locationServer.e7 is not bounded.
Trace (Transition 2)

**Internal Behavior**: The queue `mobileClient.locationServer.e7` is not bounded.

```
Trace: [Diagram of a state transition diagram showing the interactions between `mobileClient` and `locationServer`.

- The diagram includes states such as `detectPosition`, `storePosition`, `comparePositions`, and a decision node with conditions `dist < 20m` and `else`.

- Events include `target`, `close`, and `clientclose`.
```
Symptom, diagnosis and fix

The node f0 might be part of an unrestrained producer cycle.
The queue mobileClient_locationServer_e7 is not bounded.

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.

Symptoms
- Unbounded Queue

Diagnoses
- Unrestrained Producer

Fixes
- Insert Fork / Join Combo
Symptom, diagnosis and fix

**Symptom**

The node f0 might be part of an unrestrained producer cycle
The queue mobileClient_locationServer_e7 is not bounded.

**Diagnosis**

- Unbounded Queue
- Unrestrained Producer

**Fix**

- Insert Fork / Join Combo

---

Thursday, 3 July, 2008
Full screenshot
Fix applied
Fix applied
Bibliography

• Arctis: http://www.item.ntnu.no/arctis

• SPACE
  


• Formulator and Analyzer
  


www.ntnu.no

Thursday, 3 July, 2008