Qualitative and Quantitative Analysis in Automotive Scenarios – A preview of D8.6 –

Maurice ter Beek

ISTI-CNR
Pisa, Italy

SENSORIA workshop
München, 11–14 March 2008
1. Aim of SENSORIA Deliverable 8.6
2. This Presentation’s Origins
3. Logics for Qualitative and Quantitative Analysis
4. Qualitative Analysis of On Road Assistance Scenario
5. Towards Quantitative Analysis of Accident Assistance Scenario
6. Future Work
Aim of SENSORIA Deliverable 8.6 (due M36)

Relations between case studies and Theme 2 results

D8.6 will describe the relations between the four SENSORIA case studies of WP8 and the more technical work as carried out within the WPs of Theme 2: *Mathematical analysis and verification techniques and tools for system behaviour and quality of service properties*

D8.6 will show how the theoretical approaches to qualitative and quantitative aspects of services (WP3 and WP4) are applied to scenarios of the automotive, finance, telecommunications and course management & e-learning case studies (WP8)

Preview of D8.6: two exemplary approaches

- Qualitative analysis of the on road assistance scenario
- Towards quantitative analysis of the accident assistance scenario
Relations between case studies and Theme 2 results

D8.6 will describe the relations between the four SENSORIA case studies of WP8 and the more technical work as carried out within the WPs of Theme 2: *Mathematical analysis and verification techniques and tools for system behaviour and quality of service properties*

D8.6 will show how the theoretical approaches to qualitative and quantitative aspects of services (WP3 and WP4) are applied to scenarios of the automotive, finance, telecommunications and course management & e-learning case studies (WP8)

Preview of D8.6: two exemplary approaches

- Qualitative analysis of the on road assistance scenario
- Towards quantitative analysis of the accident assistance scenario
Analyses of Two Automotive Scenarios

UML 2.0 specification using a UML Profile

N. Koch and D. Berndl, Requirements Modelling and Analysis of Selected Scenarios of the Automotive Case Study. SENSORIA Deliverable 8.2a, September 2007

Qualitative analysis: On road assistance


Towards quantitative analysis: Accident assistance (a.k.a. airbag)

Analyses of Two Automotive Scenarios

UML 2.0 specification using a UML Profile

N. Koch and D. Berndl, Requirements Modelling and Analysis of Selected Scenarios of the Automotive Case Study. SENSORIA Deliverable 8.2a, September 2007

Qualitative analysis: On road assistance


Towards quantitative analysis: Accident assistance (a.k.a. airbag)

On Road Assistance Scenario

- On the road, a vehicle’s diagnostic system reports a low oil level
- Triggers in-vehicle diagnostic system to report problem with pressure of cylinder heads (car no longer driveable) and to send this diagnostic data and vehicle’s GPS coordinates to repair server
- Given driver’s preferences, service discovery system identifies and selects appropriate services (garage/tow truck/rental car) in area
- When driver makes appointment with garage, results of in-vehicle diagnosis are automatically sent along, allowing garage to identify the spare parts needed to repair car
- Similarly, when driver orders tow truck and rental car, the vehicle’s GPS coordinates are sent along
- Obviously, driver is required to deposit a security payment before being able to order any service
- Each service can be denied or cancelled, causing an appropriate compensation activity
On Road Assistance Scenario

- On the road, a vehicle’s diagnostic system reports a low oil level.
- Triggers in-vehicle diagnostic system to report problem with pressure of cylinder heads (car no longer driveable) and to send this diagnostic data and vehicle’s GPS coordinates to repair server.
- Given driver’s preferences, service discovery system identifies and selects appropriate services (garage/tow truck/rental car) in area.
- When driver makes appointment with garage, results of in-vehicle diagnosis are automatically sent along, allowing garage to identify the spare parts needed to repair car.
- Similarly, when driver orders tow truck and rental car, the vehicle’s GPS coordinates are sent along.
- Obviously, driver is required to deposit a security payment before being able to order any service.
- Each service can be denied or cancelled, causing an appropriate compensation activity.
On Road Assistance Scenario

- On the road, a vehicle’s diagnostic system reports a low oil level.
- Triggers in-vehicle diagnostic system to report problem with pressure of cylinder heads (car no longer driveable) and to send this diagnostic data and vehicle’s GPS coordinates to repair server.
- Given driver’s preferences, service discovery system identifies and selects appropriate services (garage/tow truck/rental car) in area.
- When driver makes appointment with garage, results of in-vehicle diagnosis are automatically sent along, allowing garage to identify the spare parts needed to repair car.
- Similarly, when driver orders tow truck and rental car, the vehicle’s GPS coordinates are sent along.
- Obviously, driver is required to deposit a security payment before being able to order any service.
- Each service can be denied or cancelled, causing an appropriate compensation activity.
On Road Assistance Scenario

- On the road, a vehicle’s diagnostic system reports a low oil level.
- Triggers in-vehicle diagnostic system to report problem with pressure of cylinder heads (car no longer driveable) and to send this diagnostic data and vehicle’s GPS coordinates to repair server.
- Given driver’s preferences, service discovery system identifies and selects appropriate services (garage/tow truck/rental car) in area.
- When driver makes appointment with garage, results of in-vehicle diagnosis are automatically sent along, allowing garage to identify the spare parts needed to repair car.
- Similarly, when driver orders tow truck and rental car, the vehicle’s GPS coordinates are sent along.
- Obviously, driver is required to deposit a security payment before being able to order any service.
- Each service can be denied or cancelled, causing an appropriate compensation activity.
On Road Assistance Scenario

- On the road, a vehicle’s diagnostic system reports a low oil level
- Triggers in-vehicle diagnostic system to report problem with pressure of cylinder heads (car no longer driveable) and to send this diagnostic data and vehicle’s GPS coordinates to repair server
- Given driver’s preferences, service discovery system identifies and selects appropriate services (garage/tow truck/rental car) in area
- When driver makes appointment with garage, results of in-vehicle diagnosis are automatically sent along, allowing garage to identify the spare parts needed to repair car
- Similarly, when driver orders tow truck and rental car, the vehicle’s GPS coordinates are sent along
- Obviously, driver is required to deposit a security payment before being able to order any service
- Each service can be denied or cancelled, causing an appropriate compensation activity
On the road, a vehicle’s diagnostic system reports a low oil level.

Triggers in-vehicle diagnostic system to report problem with pressure of cylinder heads (car no longer driveable) and to send this diagnostic data and vehicle’s GPS coordinates to repair server.

Given driver’s preferences, service discovery system identifies and selects appropriate services (garage/tow truck/rental car) in area.

When driver makes appointment with garage, results of in-vehicle diagnosis are automatically sent along, allowing garage to identify the spare parts needed to repair car.

Similarly, when driver orders tow truck and rental car, the vehicle’s GPS coordinates are sent along.

Obviously, driver is required to deposit a security payment before being able to order any service.

Each service can be denied or cancelled, causing an appropriate compensation activity.
On Road Assistance Scenario

- On the road, a vehicle’s diagnostic system reports a low oil level.
- Triggers in-vehicle diagnostic system to report problem with pressure of cylinder heads (car no longer driveable) and to send this diagnostic data and vehicle’s GPS coordinates to repair server.
- Given driver’s preferences, service discovery system identifies and selects appropriate services (garage/tow truck/rental car) in area.
- When driver makes appointment with garage, results of in-vehicle diagnosis are automatically sent along, allowing garage to identify the spare parts needed to repair car.
- Similarly, when driver orders tow truck and rental car, the vehicle’s GPS coordinates are sent along.
- Obviously, driver is required to deposit a security payment before being able to order any service.
- Each service can be denied or cancelled, causing an appropriate compensation activity.
Components of UML Specification

- Engine: causes low oil level alert
- Discovery engine: discovers services needed
- Reasoner: selects best services
- Orchestrator: composes services to achieve goal
- Driver: calls garage, tow truck, rental car and bank
- GPS: sends vehicle’s current coordinates to services
- Garage: receives diagnostic data about vehicle
- Tow truck: receives GPS coordinates of vehicle
- Rental car: receives GPS coordinates of driver
- Bank: receives deposit from driver
Operational Model: Communicating State Machines

Subcomponent **BankCommunication**

**BankComm** (Defers bankrevoke)

- **b1**
  - bankcharge / bank.requestCardCharge(...)
  - chargeResponseFail / self.bankFail

- **b2**
  - chargeResponseOK(chargeID) / chargedID := chargeID; self.bankOK

- **b3**
  - bankrevoke / bank.revokeCardCharge(self, chargedID)

- **b4**
  - bankrevokeOK

- **b5**
  - bankrevoke

- **b6**
  - bankrevoke
Modelling Assumptions

- We do not define a separate state machine for each component, but rather structure some of them as subcomponents of others
- We abstract altogether from a remote discovery state machine (to search for services in a remote repository)
- **LocalDiscovery** returns at most one choice of services for on-road assistance
- Compensations are explicitly modelled as requests to cancel operations (viz. bankrevoke and garagerevoke)
- All communications between Car’s subcomponents, and those between these subcomponents and Bank and RoadAssistance (i.e. all service invocations), are modelled as pairs of request/response signals; whereas this is necessary for the former (synchronous operation calls would deadlock), the latter might equally well be modelled as synchronous operation calls.
Modelling Assumptions

- We do not define a separate state machine for each component, but rather structure some of them as subcomponents of others.
- We abstract altogether from a remote discovery state machine (to search for services in a remote repository).
- `LocalDiscovery` returns at most one choice of services for on-road assistance.
- Compensations are explicitly modelled as requests to cancel operations (viz. `bankrevoke` and `garagerevoke`).
- All communications between Car's subcomponents, and those between these subcomponents and Bank and RoadAssistance (i.e. all service invocations), are modelled as pairs of request/response signals; whereas this is necessary for the former (synchronous operation calls would deadlock), the latter might equally well be modelled as synchronous operation calls.
Modelling Assumptions

- We do not define a separate state machine for each component, but rather structure some of them as subcomponents of others.
- We abstract altogether from a remote discovery state machine (to search for services in a remote repository).
- **LocalDiscovery** returns at most one choice of services for on-road assistance.
- Compensations are explicitly modelled as requests to cancel operations (viz. `bankrevoke` and `garagerevoke`).
- All communications between Car’s subcomponents, and those between these subcomponents and Bank and RoadAssistance (i.e. all service invocations), are modelled as pairs of request/response signals; whereas this is necessary for the former (synchronous operation calls would deadlock), the latter might equally well be modelled as synchronous operation calls.
Modelling Assumptions

- We do not define a separate state machine for each component, but rather structure some of them as subcomponents of others.
- We abstract altogether from a remote discovery state machine (to search for services in a remote repository).
- **LocalDiscovery** returns at most one choice of services for on-road assistance.
- Compensations are explicitly modelled as requests to cancel operations (viz. bankrevoke and garagerevoke).
- All communications between Car's subcomponents, and those between these subcomponents and Bank and RoadAssistance (i.e. all service invocations), are modelled as pairs of request/response signals; whereas this is necessary for the former (synchronous operation calls would deadlock), the latter might equally well be modelled as synchronous operation calls.
Modelling Assumptions

We do not define a separate state machine for each component, but rather structure some of them as subcomponents of others.

We abstract altogether from a remote discovery state machine (to search for services in a remote repository).

**LocalDiscovery** returns at most one choice of services for on-road assistance.

Compensations are explicitly modelled as requests to cancel operations (viz. `bankrevoke` and `garagerevoke`).

All communications between Car’s subcomponents, and those between these subcomponents and Bank and RoadAssistance (i.e. all service invocations), are modelled as pairs of request/response signals; whereas this is necessary for the former (synchronous operation calls would deadlock), the latter might equally well be modelled as synchronous operation calls.
Modelling Assumptions

- We do not define a separate state machine for each component, but rather structure some of them as subcomponents of others.
- We abstract altogether from a remote discovery state machine (to search for services in a remote repository).
- **LocalDiscovery** returns at most one choice of services for on-road assistance.
- Compensations are explicitly modelled as requests to cancel operations (viz. `bankrevoke` and `garagerevoke`).
- All communications between Car’s subcomponents, and those between these subcomponents and Bank and RoadAssistance (i.e. all service invocations), are modelled as pairs of request/response signals; whereas this is necessary for the former (synchronous operation calls would deadlock), the latter might equally well be modelled as synchronous operation calls.
Structure Diagram of State Machines

⇒ Validation by hand is not feasible: 535 states and 814 transitions
The emerging trend in industry is to use UML (state diagrams)

- Ease of expressiveness w.r.t. pure action- or state-based logics
- Their use often leads to a reduced state space, small memory needs and less time spent for verification

UCTL defined as an extension of action-based CTL (ACTL [DNV90])

- UCTL includes both CTL and ACTL
- Formally represent all possible system evolutions by \( L^2 TS \) [DNV95]
  - states represent the various system configurations
  - edges represent the possible evolutions of a system configuration

The service-oriented logic S\(OC\)L is a recent [FASE’08] specialization of UCTL meant to capture peculiar aspects of services
To use the full potential of specification languages (like UML) that allow both action and state changes to be modelled, various action- and state-based logics have been developed.

- Ease of expressiveness w.r.t. pure action- or state-based logics
- Their use often leads to a reduced state space, small memory needs and less time spent for verification

UCTL defined as an extension of action-based CTL (ACTL [DNV90])

- UCTL includes both CTL and ACTL
- Formally represent all possible system evolutions by $L^2TS$ [DNV95]
  - states represent the various system configurations
  - edges represent the possible evolutions of a system configuration

The service-oriented logic SoCL is a recent [FASE’08] specialization of UCTL meant to capture peculiar aspects of services
To use the full potential of specification languages (like UML) that allow both action and state changes to be modelled, various action- and state-based logics have been developed.

- Ease of expressiveness w.r.t. pure action- or state-based logics
- Their use often leads to a reduced state space, small memory needs and less time spent for verification

UCTL defined as an extension of action-based CTL (ACTL [DNV90])
- UCTL includes both CTL and ACTL
- Formally represent all possible system evolutions by $L^2TS$ [DNV95]
  - states represent the various system configurations
  - edges represent the possible evolutions of a system configuration

The service-oriented logic SocL is a recent [FASE’08] specialization of UCTL meant to capture peculiar aspects of services.
To use the full potential of specification languages (like UML) that allow both action and state changes to be modelled, various action- and state-based logics have been developed.

- Ease of expressiveness w.r.t. pure action- or state-based logics
- Their use often leads to a reduced state space, small memory needs and less time spent for verification

UCTL defined as an extension of action-based CTL (ACTL [DNV90])

- UCTL includes both CTL and ACTL
- Formally represent all possible system evolutions by $L^2TS$ [DNV95]
  - states represent the various system configurations
  - edges represent the possible evolutions of a system configuration

The service-oriented logic SocL is a recent [FASE’08] specialization of UCTL meant to capture peculiar aspects of services.
To use the full potential of specification languages (like UML) that allow both action and state changes to be modelled, various action- and state-based logics have been developed.

- Ease of expressiveness w.r.t. pure action- or state-based logics
- Their use often leads to a reduced state space, small memory needs and less time spent for verification

UCTL defined as an extension of action-based CTL (ACTL [DNV90])
- UCTL includes both CTL and ACTL
- Formally represent all possible system evolutions by \( L^2 \text{TS} \) [DNV95]
  - States represent the various system configurations
  - Edges represent the possible evolutions of a system configuration

The service-oriented logic SOC\( L \) is a recent [FASE’08] specialization of UCTL meant to capture peculiar aspects of services.
To use the full potential of specification languages (like UML) that allow both action and state changes to be modelled, various action- and state-based logics have been developed.

- Ease of expressiveness w.r.t. pure action- or state-based logics
- Their use often leads to a reduced state space, small memory needs and less time spent for verification

UCTL defined as an extension of action-based CTL (ACTL [DNV90])

- UCTL includes both CTL and ACTL
- Formally represent all possible system evolutions by $L^2TS$ [DNV95]
  - states represent the various system configurations
  - edges represent the possible evolutions of a system configuration

The service-oriented logic SOCL is a recent [FASE’08] specialization of UCTL meant to capture peculiar aspects of services.
To use the full potential of specification languages (like UML) that allow both action and state changes to be modelled, various action-and state-based logics have been developed.

- Ease of expressiveness w.r.t. pure action- or state-based logics
- Their use often leads to a reduced state space, small memory needs and less time spent for verification

UCTL defined as an extension of action-based CTL (ACTL [DNV90])

- UCTL includes both CTL and ACTL
- Formally represent all possible system evolutions by $L^2TS$ [DNV95]
  - states represent the various system configurations
  - edges represent the possible evolutions of a system configuration

The service-oriented logic SocL is a recent [FASE’08] specialization of UCTL meant to capture peculiar aspects of services
On-the-fly model checker to verify SoCL formulae (specifying action- and state-based properties) over communicating UML state machines

Current experimentation via web interface at http://fmt.isti.cnr.it/{c/u}mc ⇒ planned integration into the SENSORIA CASE tool

Service responsiveness

A service is *responsive* if it guarantees a response to each received request

SoCL property

\[
\begin{align*}
AG \left[ request(\text{CardCharge}, \$\text{carID}) \right] \\
AF \{\text{ResponseOK}(\text{CardCharge}, \%\text{carID}) \lor \text{ResponseFail}(\text{CardCharge}, \%\text{carID}) \} \text{ true}
\end{align*}
\]
Model Checker for SoCLoL

On-the-fly model checker to verify SoCLoL formulae (specifying action- and state-based properties) over communicating UML state machines

Current experimentation via web interface at http://fmt.isti.cnr.it/{c/u}mc ⇒ planned integration into the SENSORIA CASE tool

Service responsiveness

A service is responsive if it guarantees a response to each received request

SoCLoL property

\[ AG \left[ \text{request}(\text{CardCharge}, \$\text{carID}) \right] \]
\[ AF \{ \text{ResponseOK}(\text{CardCharge}, \%\text{carID}) \lor \text{ResponseFail}(\text{CardCharge}, \%\text{carID}) \} \] true
Model Checker for SocL

On-the-fly model checker to verify SocL formulae (specifying action-and state-based properties) over communicating UML state machines.

Current experimentation via web interface at http://fmt.isti.cnr.it/{c/u}mc ⇒ planned integration into the SENSORIA CASE tool.

Service responsiveness

A service is *responsive* if it guarantees a response to each received request.

SocL property

\[
\begin{align*}
AG \left[ \text{request}(\text{CardCharge}, $\text{carID}) \right] \\
AF \left\{ \text{ResponseOK}(\text{CardCharge}, %\text{carID}) \lor \text{ResponseFail}(\text{CardCharge}, %\text{carID}) \right\} \text{ true}
\end{align*}
\]
Model Checker for SoC\textit{L}

On-the-fly model checker to verify SoC\textit{L} formulae (specifying action- and state-based properties) over communicating UML state machines

Current experimentation via web interface at http://fmt.isti.cnr.it/{c/u}mc ⇒ planned integration into the SENSORIA CASE tool

**Service responsiveness**

A service is *responsive* if it guarantees a response to each received request

SoC\textit{L} property

\[
\begin{aligned}
AG \ [ \text{request}(\text{CardCharge}, $\text{carID}$)] \\
AF \ { \text{ResponseOK}(\text{CardCharge}, \%\text{carID}) \lor \\
\text{ResponseFail}(\text{CardCharge}, \%\text{carID}) } \quad \text{true}
\end{aligned}
\]
Model Checker for SocL

On-the-fly model checker to verify SocL formulae (specifying action- and state-based properties) over communicating UML state machines

Current experimentation via web interface at http://fmt.isti.cnr.it/{c/u}mc

⇒ planned integration into the SENSORIA CASE tool

Service responsiveness

If **Car** requests **Bank** to charge a credit card, then

**Bank** will reply by notifying either a successful or a failed attempt to charge the credit card

SocL property

\[
AG \left[ \text{request}(\text{CardCharge}, \$.carID) \right] \\
AF \left\{ \text{ResponseOK}(\text{CardCharge}, %.carID) \lor \text{ResponseFail}(\text{CardCharge}, %.carID) \right\} \text{ true}
\]
Model Checker for SocL

On-the-fly model checker to verify SocL formulae (specifying action- and state-based properties) over communicating UML state machines

Current experimentation via web interface at http://fmt.isti.cnr.it/{c/u}mc
⇒ planned integration into the SENSORIA CASE tool

Service responsiveness

If **Car** requests **Bank** to charge a credit card, then **Bank** will reply by notifying either a successful or a failed attempt to charge the credit card

SocL property

\[
AG \left[ \text{request}(\text{CardCharge}, \$\text{carID}) \right]
AF \left\{ \text{ResponseOK}(\text{CardCharge}, \%\text{carID}) \lor \text{ResponseFail}(\text{CardCharge}, \%\text{carID}) \right\} \text{true}
\]

- A *temporal* logic (dynamic evolution)
- An *action*- and *state*-based logic
- A *real-time* logic (real-time bounds)
- A *probabilistic* logic (performance and dependability aspects)
- A *spatial* logic (spatial structure of the network)
MoSL: Mobile Stochastic Logic (for STOKLAIM)


- A *temporal* logic (dynamic evolution)
- An *action* - and *state*-based logic
- A *real-time* logic (real-time bounds)
- A *probabilistic* logic (performance and dependability aspects)
- A *spatial* logic (spatial structure of the network)
Origins of UCTL, MoSL and SocL

- UCTL: [De Nicola et al.]
- MoSL: [De Nicola et al.]
- aCTL: [Baier et al.]
- asCSL: [Baier et al.]
- aCSL: [Hermanns et al.]
- CSL: [Aziz et al., Baier et al.]
- CTL: [Clarke et al.]
- SocL: [Fantechi et al.]
- MoSL: [De Nicola et al.]
- asCSL: [Baier et al.]
- aCSL: [Hermanns et al.]
- CSL: [Aziz et al., Baier et al.]
- CTL: [Clarke et al.]
- SocL: [Fantechi et al.]
**Availability**

“Probability is larger than 0.99 that server $X$ is not faulty in the long run”

$$S_{>0.99}(\neg(\langle \text{Fault} \rangle@X))$$

$S$ is the probabilistic steady-state operator and the fault is modelled with the presence of token Fault at the server's data repository.

**Reliability**

“Probability is less than 0.02 that a fault occurs within 3600 time units and when the system enters the faulty state, it’s due to the execution in server $X$ of an input operation from component $Y$’s repository”

$$P_{<0.02}(\neg(\langle \text{Fault} \rangle@X) \ U_{<3600}^{\{X:\text{IN}(z)@Y\}} \langle \text{Fault} \rangle@X)$$

$P$ probabilistic path operator, $U$ (dense) time-bounded until operator (its subscript indicates the specific action that causes the fault).
Exemplary Requirements Expressable in MoSL

Availability

“Probability is larger than 0.99 that server $X$ is not faulty in the long run”

\[ S_{>0.99}(\neg(\langle \text{Fault} \rangle @X)) \]

$S$ is the probabilistic steady-state operator and the fault is modelled with the presence of token $\text{Fault}$ at the server’s data repository.

Reliability

“Probability is less than 0.02 that a fault occurs within 3600 time units and when the system enters the faulty state, it’s due to the execution in server $X$ of an input operation from component $Y$’s repository”

\[ P_{<0.02}(\neg(\langle \text{Fault} \rangle @X) \until^{3600}_{\{X:IN(z)@Y\}} \langle \text{Fault} \rangle @X) \]

$P$ probabilistic path operator, $\until$ (dense) time-bounded until operator (its subscript indicates the specific action that causes the fault).
An accident occurs to a car registered with the Accident Assistance Service and the airbag of the car deploys. Triggers an automated message to the Accident Assistance Server, containing the vehicle’s GPS data, the vehicle identification number and a collection of sensor data. The Accident Assistance Server calls the driver’s mobile phone. If, due to injuries, the driver is unable to answer, and the severity of the accident is confirmed also by the sensor data, then the emergency services are alerted and the vehicle location is communicated to them. Other cars approaching the accident are alerted as well.
An accident occurs to a car registered with the Accident Assistance Service and the airbag of the car deploys.

Triggers an automated message to the Accident Assistance Server, containing the vehicle’s GPS data, the vehicle identification number and a collection of sensor data.

The Accident Assistance Server calls the driver’s mobile phone.

If, due to injuries, the driver is unable to answer, and the severity of the accident is confirmed also by the sensor data, then the emergency services are alerted and the vehicle location is communicated to them.

Other cars approaching the accident are alerted as well.
Accident Assistance Scenario (a.k.a. Airbag Scenario)

- An accident occurs to a car registered with the Accident Assistance Service and the airbag of the car deploys.
- Triggers an automated message to the Accident Assistance Server, containing the vehicle’s GPS data, the vehicle identification number and a collection of sensor data.
- The Accident Assistance Server calls the driver’s mobile phone.
  - If, due to injuries, the driver is unable to answer, and the severity of the accident is confirmed also by the sensor data, then the emergency services are alerted and the vehicle location is communicated to them.
  - Other cars approaching the accident are alerted as well.
An accident occurs to a car registered with the Accident Assistance Service and the airbag of the car deploys.

- Triggers an automated message to the Accident Assistance Server, containing the vehicle’s GPS data, the vehicle identification number and a collection of sensor data.

- The Accident Assistance Server calls the driver’s mobile phone.

- If, due to injuries, the driver is unable to answer, and the severity of the accident is confirmed also by the sensor data, then the emergency services are alerted and the vehicle location is communicated to them.

- Other cars approaching the accident are alerted as well.
Accident Assistance Scenario (a.k.a. Airbag Scenario)

- An accident occurs to a car registered with the Accident Assistance Service and the airbag of the car deploys.
- Triggers an automated message to the Accident Assistance Server, containing the vehicle’s GPS data, the vehicle identification number and a collection of sensor data.
- The Accident Assistance Server calls the driver’s mobile phone.
- If, due to injuries, the driver is unable to answer, and the severity of the accident is confirmed also by the sensor data, then the emergency services are alerted and the vehicle location is communicated to them.
- Other cars approaching the accident are alerted as well.
Modelling Assumptions

Simplifications

- Airbag deployment is *not considered*
- If the driver does not answer the phone call, the emergency services are alerted *anyway*
- Alerting cars approaching the accident area are *not considered*
Modelling Assumptions

Simplifications

- Airbag deployment is *not considered*
- If the driver does not answer the phone call, the emergency services are alerted *anyway*
- Alerting cars approaching the accident area are *not considered*
Modelling Assumptions

Simplifications

- Airbag deployment is *not considered*
- If the driver does not answer the phone call, the emergency services are alerted *anyway*
- Alerting cars approaching the accident area are *not considered*
Simplifications

- Airbag deployment is *not considered*
- If the driver does not answer the phone call, the emergency services are alerted *anyway*
- Alerting cars approaching the accident area are *not considered*
Modelling Assumptions

- Each entity of interest is (assumed modelled as a STOKLAIM site and is) provided with its physical address
  - Accident Assistance Server address: AccAssSrv
  - Phone Server address: PhSrv
  - Emergency Server address: EmSrv
  - Each car has its own address
- Each car uniquely identified by car identifier $CarID \in CID$, finite
- Accident notification:
  - storing $(CarID, GPSData, SenData) @ AccAssSrv$
  - $PhNr : CID \rightarrow PhN$;
  - $PhNr(carID) = carID$ driver’s mobile phone nr.
- Making a phone call: storing phone number to site PhSrv
- The Accident Assistance Server alerts the Emergency Server by uploading the GPS data to site EmSrv
Modelling Assumptions

- Each entity of interest is (assumed modelled as a STOKCLAIM site and is) provided with its physical address
  - Accident Assistance Server address: \textit{AccAssSrv}
  - Phone Server address: \textit{PhSrv}
  - Emergency Server address: \textit{EmSrv}
  - Each car has its own address

- Each car uniquely identified by car identifier \textit{CarID} \in \textit{CID}, \textit{finite}

- Accident notification: storing (\textit{CarID}, \textit{GPSData}, \textit{SenData}) @ \textit{AccAssSrv}

- \textit{PhNr} : \textit{CID} \rightarrow \textit{PhN};
  \textit{PhNr(carID)} = \textit{carID} driver’s mobile phone nr.

- Making a phone call: storing phone number to site \textit{PhSrv}

- The Accident Assistance Server alerts the Emergency Server by uploading the GPS data to site \textit{EmSrv}
Modelling Assumptions

- Each entity of interest is (assumed modelled as a STOKLAIM site and is) provided with its physical address
  - Accident Assistance Server address: $\text{AccAssSrv}$
  - Phone Server address: $\text{PhSrv}$
  - Emergency Server address: $\text{EmSrv}$
  - Each car has its own address

- Each car uniquely identified by car identifier $\text{CarID} \in \text{CID}$, finite

- Accident notification:
  storing $(\text{CarID}, \text{GPSData}, \text{SenData})$ @ $\text{AccAssSrv}$

- $\text{PhNr} : \text{CID} \rightarrow \text{PhN}$;
  $\text{PhNr(carID)} = \text{carID}$ driver’s mobile phone nr.

- Making a phone call: storing phone number to site $\text{PhSrv}$

- The Accident Assistance Server alerts the Emergency Server by uploading the GPS data to site $\text{EmSrv}$
Modelling Assumptions

- Each entity of interest is (assumed modelled as a STOKLAIM site and is) provided with its physical address
  - Accident Assistance Server address: $\text{AccAssSrv}$
  - Phone Server address: $\text{PhSrv}$
  - Emergency Server address: $\text{EmSrv}$
  - Each car has its own address

- Each car uniquely identified by car identifier $\text{CarID} \in \text{CID}$, finite

- Accident notification:
  - storing $(\text{CarID}, \text{GPSData}, \text{SenData})$ @ $\text{AccAssSrv}$

- $\text{PhNr} : \text{CID} \rightarrow \text{PhN}$;
  - $\text{PhNr(carID)} = \text{carID}$ driver’s mobile phone nr.

- Making a phone call: storing phone number to site $\text{PhSrv}$

- The Accident Assistance Server alerts the Emergency Server by uploading the GPS data to site $\text{EmSrv}$
Modelling Assumptions

Each entity of interest is (assumed modelled as a STOKCLAIM site and is) provided with its physical address:
- Accident Assistance Server address: \( \text{AccAssSrv} \)
- Phone Server address: \( \text{PhSrv} \)
- Emergency Server address: \( \text{EmSrv} \)
- Each car has its own address

Each car uniquely identified by car identifier \( \text{CarID} \in \text{CID}, \text{finite} \)

Accident notification:
- storing \((\text{CarID}, \text{GPSData}, \text{SenData}) @ \text{AccAssSrv}\)

\( \text{PhNr} : \text{CID} \rightarrow \text{PhN}; \)

\( \text{PhNr(carID)} = \text{carID} \) driver’s mobile phone nr.

Making a phone call: storing phone number to site \( \text{PhSrv} \)

The Accident Assistance Server alerts the Emergency Server by uploading the GPS data to site \( \text{EmSrv} \)
Modelling Assumptions

- Each entity of interest is (assumed modelled as a STOKLAIM site and is) provided with its physical address
  - Accident Assistance Server address: AccAssSrv
  - Phone Server address: PhSrv
  - Emergency Server address: EmSrv
  - Each car has its own address
- Each car uniquely identified by car identifier $CarID \in CID$, finite
- Accident notification:
  storing $(CarID, GPSData, SenData) @ AccAssSrv$
- $PhNr : CID \rightarrow \text{PhN};$
  $PhNr(carID) = carID$ driver’s mobile phone nr.
- Answering a phone call: removing phone number from site PhSrv
  - The Accident Assistance Server alerts the Emergency Server by uploading the GPS data to site EmSrv
Modelling Assumptions

- Each entity of interest is (assumed modelled as a STOKCLAIM site and is) provided with its physical address
  - Accident Assistance Server address: AccAssSrv
  - Phone Server address: PhSrv
  - Emergency Server address: EmSrv
  - Each car has its own address

- Each car uniquely identified by car identifier \( \text{CarID} \in CID, \text{finite} \)

- Accident notification:
  - storing \((\text{CarID}, \text{GPSData}, \text{SenData}) @ \text{AccAssSrv}\)

- \( \text{PhNr} : CID \rightarrow \text{PhN}; \)
  - \( \text{PhNr(carID)} = \text{carID} \) driver’s mobile phone nr.

- Answering a phone call: removing phone number from site \( \text{PhSrv} \)

- The Accident Assistance Server alerts the Emergency Server by uploading the GPS data to site \( \text{EmSrv} \)
Response time

Guarantee acceptable timing for detecting an accident that has seriously injured the driver as well as for rescue alerting

- First decompose the scenario’s requirements into simpler ones
- Subsequently make each of these more precise by making time bounds explicit, using time-bounded until operator making statements realistic, using probabilistic operators

We now illustrate this process by applying it to one of the sub-requirements of the responsiveness requirement.
Performance Property

**Response time**

Guarantee acceptable timing for detecting an accident that has seriously injured the driver as well as for rescue alerting.

- First decompose the scenario’s requirements into simpler ones.
- Subsequently make each of these more precise by:
  - making time bounds explicit, using time-bounded until operator
  - making statements realistic, using probabilistic operators.

We now illustrate this process by applying it to one of the (sub-)requirements of the responsiveness requirement.
Guarantee acceptable timing for detecting an accident that has seriously injured the driver as well as for rescue alerting

- First decompose the scenario’s requirements into simpler ones
- Subsequently make each of these more precise by
  - making time bounds explicit, using time-bounded until operator
  - making statements realistic, using probabilistic operators

→ We now illustrate this process by applying it to one of the (sub-)requirements of the responsiveness requirement
Response time

Guarantee acceptable timing for detecting an accident that has seriously injured the driver as well as for rescue alerting

- First decompose the scenario’s requirements into simpler ones
- Subsequently make each of these more precise by
  - making time bounds explicit, using time-bounded until operator
  - making statements realistic, using probabilistic operators

We now illustrate this process by applying it to one of the (sub-)requirements of the responsiveness requirement
Guarantee acceptable timing for detecting an accident that has seriously injured the driver as well as for rescue alerting

First decompose the scenario’s requirements into simpler ones
Subsequently make each of these more precise by
  making time bounds explicit, using time-bounded until operator
  making statements realistic, using probabilistic operators

⇒ We now illustrate this process by applying it to one of the (sub-)requirements of the responsiveness requirement
Verifying Performance (Response Time)

- Suppose it has been detected that an accident involving car $car_{id}$ has taken place and that the driver is seriously injured.
- Then Accident Assistance Server is supposed to send, by means of proper output action, an alert to the Emergency Service.

Ideal requirement

Emergency Service alerted within maximal time $t_{alert}$

MoSL property

$$\mathcal{P} \geq 0.997(\text{true} \uparrow U_{\langle t_{alert} \rangle}^{\text{true}} \{\text{AccAssSrv} : o((car_{id},gps), EmSrv)\})$$
Suppose it has been detected that an accident involving car $car_{id}$ has taken place and that the driver is seriously injured.

Then Accident Assistance Server is supposed to send, by means of proper output action, an alert to the Emergency Service.

**Ideal requirement**

Emergency Service alerted within maximal time $t_{alert}$

**MoSL property**

$$\mathcal{P}_{\geq 0.997}(\text{true} \quad \mathcal{U}^{t_{alert}}_{\{\text{AccAssSrv} \circ ((car_{id},gps),\text{EmSrv})\}} \text{true})$$
Suppose it has been detected that an accident involving car \( car_{id} \) has taken place and that the driver is seriously injured.

Then Accident Assistance Server is supposed to send, by means of proper output action, an alert to the Emergency Service.

**Realistic requirement**

- Emergency Service alerted within maximal time \( t_{alert} \) in at least 99.7% of the cases.

**MoSL property**

\[
P_{\geq 0.997}(true \cup_{\{AccAssSrv:o((car_{id},gps),EmSrv)\}}^{t_{alert}} \{true\})
\]
Verifying Performance (Response Time)

- Suppose it has been detected that an accident involving car $\text{car}_{id}$ has taken place and that the driver is seriously injured.
- Then Accident Assistance Server is supposed to send, by means of proper output action, an alert to the Emergency Service.

**Realistic requirement**

Emergency Service alerted within maximal time $t_{alert}$ in at least 99.7% of the cases.

**MoSL property**

$$\mathcal{P}_{\geq 0.997}(true \quad \top \quad \mathcal{U}^{t_{alert}} \{\text{AccAssSrv}:o((\text{car}_{id},gps),\text{EmSrv})\} \quad true)$$
Future Work

- Relax modelling assumptions made w.r.t. requirements models
- Verify more complex scenarios (drivers competing for tow trucks)
- Quantitative analysis of on road assistance scenario (via MoSL?)
- Perform quantitative analysis of accident assistance scenario
- Integrate model checker for SoCL into the SENSORIA CASE tool

Collect your contributions for D8.6 through the SENSORIA website!
Future Work

- Relax modelling assumptions made w.r.t. requirements models
- Verify more complex scenarios (drivers competing for tow trucks)
- Quantitative analysis of on road assistance scenario (via MoSL?)
- Perform quantitative analysis of accident assistance scenario

- Integrate model checker for SoCL into the SENSORIA CASE tool

Collect your contributions for D8.6 through the SENSORIA website!
Future Work

- Relax modelling assumptions made w.r.t. requirements models
- Verify more complex scenarios (drivers competing for tow trucks)
- Quantitative analysis of on road assistance scenario (via MoSL?)
- Perform quantitative analysis of accident assistance scenario
- Integrate model checker for SoCIL into the SENSORIA CASE tool

Collect your contributions for D8.6 through the SENSORIA website!
Relax modelling assumptions made w.r.t. requirements models

Verify more complex scenarios (drivers competing for tow trucks)

Quantitative analysis of on road assistance scenario (via MoSL?)

Perform quantitative analysis of accident assistance scenario

Integrate model checker for SoCL into the SENSORIA CASE tool

Collect your contributions for D8.6 through the SENSORIA website!