QoS Modelling and Verification with UML Statecharts

Holger Hermanns
Universität des Saarlandes, Germany

The StoCharts Approach

joint work with

David N. Jansen
Universiteit Twente, the Netherlands

Joost-Pieter Katoen
RWTH Aachen, Germany
Overview

Introduction to QoS modeling and analysis

- Introduction to Statecharts

- StoCharts
  - Introduction
  - Semantics
  - Applications

- Conclusions and future outlook
Quality of Service?

- Some first remarks on QoS.
- Let’s take one of the classical OO examples.
Quality of Service?

- Some first remarks on QoS.
- Let’s take one of the classical OO examples.

The good old Hotel reservation system.

How about QoS here?
Quality of Service?

- Some first remarks on QoS.
- Let’s take one of the classical OO examples.

The good old Hotel reservation system.

How about QoS here?
Quality vs. Quantity of Service

- UML is suggested as the method to design systems.

- How about its support for 'model driven' QoS?
Quality vs. Quantity of Service

- UML is suggested as the method to design systems.

- How about its support for ‘model driven’ QoS?

- What’s the point?

- Any QoS property we can think of comes with a metric, or at least, a scale.

  Or: one needs quantitative models to assess quality.
What QoS properties do we think of?

- **Specific properties (but pretty vague still)**

- **Abstract properties**
  - **Time-related**
    - Date/time, time delay, latency, etc
  - ** Dependability-related**
    - Failure rates, message losses, availability, etc
  - **Capacity-related**
    - Throughput, processor load, power consumption, etc
  - **Security-related characteristics**
    - Protection, access control, authentication, confidentiality, etc

- and so on ...
A brief guide through QoS Modelling

QoS models are stochastic in nature to model (or abstract from)
- message buffering,
- interdependencies due to media sharing,
- communication characteristics,
- component/link failures,
- hardware circuit inaccuracies
- etc.

What we have out there
- queueing networks,
- Petri net extensions,
- hierarchical formalisms,
- Compositional formalisms (process algebra),
- Annotated design methods (SDL, ..., UML-Statecharts).
Stochastic models

What I will tell you about (very briefly)

- their ingredients
- their analysis
- their construction
Stochastic models

Their ingredients?

- states
- transitions
- labels
  - of states
  - of transitions
States and transitions

- stochastic models describe an abstract view of a real system

- **states** are abstract views of system configurations

- **transitions** describe changes from one system configuration to another as time progresses

- **labels** represent relevant information

- Precise semantics of transition labels induces a precise description of stochastic behaviour. (Prerequisite for faithful analysis!)
A snackbar in Eindhoven

- Customers arrive at a certain frequency, say approximately 1 customer per five minutes.
  \[ \text{arrival rate} \quad \lambda = \frac{1}{5} \text{ min} \]

- Service requires, say, three minutes.
  \[ \text{service rate} \quad \mu = \frac{1}{3} \text{ min} \]

- At most six customers can wait inside the snackbar.
States and transitions

- **states** are abstract views of system configurations
  here: number of customers in the snackbar

- **transitions** describe changes from one system configuration to another as time progresses
  here: arrival and service of customers
What is this?

- a *stochastic process*

- more precise:
  a *Markov chain* (named after A.A. Markov, 1909)

- again more precise:
  a finite homogeneous continuous-time Markov chain
Continuous-time Markov chains

- (finite state) automata,
- all times are exponentially distributed,
- very well investigated class of stochastic processes,
- widely used in practice,
- best guess, if only mean values are known,
- efficient and numerically stable algorithms for stationary and transient analysis are available.

sojourn time in states are memory-less,
Continuous time, *but* memory

- and many, many others

- actually:
  
  *absence of memory is* rare;

- but:
  
  it makes life (i.e. modelling and analysis) *a lot* simpler;

- and:
  
  it is often an appropriate simplification.
Beyond Markov Chains: Stochastic models with memory

- states, transitions

- labels
  - of states
  - of transitions

- clocks
  - serve as the memory of time.
  - are sampled from distributions, count down
A light in the stairway

- The light is turned on if someone enters the stairway.

- It goes off after exactly 10.3 minutes.

- People arrive randomly, at least every 15 minutes, with equal probability for each time instant.
Phase-type distributions

- The time it takes to reach an 'absorbing' state in a finite CTMC is phase-type distributed.

- Phase-type distributions can approximate arbitrary distributions with arbitrary precision.
Stochastic models

What I will tell you about (very briefly)

- their ingredients
- their analysis
- their construction
A snackbar in Eindhoven (revisited)

- Customers arrive at a certain frequency, say approximately 1 customer per five minutes.

  \[ \text{arrival rate } \lambda = \frac{1}{5} \text{ min} \]

- Service requires, say, three minutes.

  \[ \text{service rate } \mu = \frac{1}{3} \text{ min} \]

- At most six customers can wait inside the snackbar.

![Diagram of a snackbar with states and rates]
Snackbar (cont.)

What’s the utilization of the snackbar?

Solve

$$\tilde{Q}_\pi = 0$$

$$\sum_s \pi(s) = 1$$
Analysing stochastic models

What would you like to know?

- Or, what is your measure of performance?
  - mean number of customers waiting in the snackbar?
  - mean time a customer has to wait?
  - percentage of time snackbar is utilised by someone?
  - number of customers served per minute?
  - percentage of customers that are lost, due to lack of space?
  - profit made?
  - number of wealthy customers lost?
  - ...

Standard performance measures

- mean queue length,
- mean waiting time,
- throughput,
- probability of loss,
- utilization

... and fault tolerance measures

- mean time to system failure,
- mean time between failures,
- system availability,
- ...

\[ \lambda \quad \lambda \quad \lambda \quad \lambda \quad \lambda \quad \lambda \quad \lambda \]

\[ \mu \quad \mu \quad \mu \quad \mu \quad \mu \quad \mu \quad \mu \]
Calculating performance measures

- all these performance (and fault tolerance) measures can be computed on the basis of
  - state probabilities,
  - state labels, and/or
  - transition labels.

- Computation of *state probabilities* is the main technical issue.

- Recall: state probabilities describe the likelihood of being in a certain state (at a certain time instant)
Calculating state probabilities

There are three fundamentally different ways to calculate state probabilities

- *analytical* solution,
- *numerical* solution,
- *simulation*. 
Analytical solution

- express the state probabilities (or even measures directly) as *closed formulae* in the parameters of the model

**example:** utilization of the snack bar \( U(\lambda, \mu) = \frac{\lambda}{\mu} \)

   provided that \( \lambda < \mu \), and that the queue length may become larger than 6, namely infinite

- **pros:**
  - very accurate
  - very fast, and simple

- **cons:**
  - only for highly restricted classes of stochastic processes
  - requires study of scientific literature, to find specific formulae
Numerical solution

- state probabilities are obtained by an exact algorithm where model parameters are instantiated with numerical values.

**Example:** state probabilities of the snackbar are obtained by (e.g.) Gauss elimination of a $7 \times 7$ matrix based on $1/3$ and $1/5$ entries.

- **Pros:**
  - accurate, up to numerical precision

- **Cons:**
  - only reasonable for finite Markov chains
  - number of states is a limiting factor (about $10^8$)
Simulation

- the stochastic model is mimicked by a simulator rolling dices and producing statistics of simulation time spent in states. The fraction of simulation time spent in a particular state is used as an estimate for the state probability.

example: Let a lot of (virtual) people use the (virtual) light bulb, and compute the fraction of time where the light is on. Do this 100000 times faster than real time.

- **pros:**
  - very general, suitable for arbitrary stochastic models

- **cons:**
  - good accuracy usually requires long (often very long) simulation runs
Rules of thumb

- **Analytical solution** allows very quick and very precise insight in your model, but the model tends to be a very loose approximation of reality.

- **Simulation** allows relatively slow and rough insight in a single instance of your model, but the model can have a close correspondence to reality.

- **Numerical solution** allows quick and precise insight in a single instance of your Markov chain model, which usually is an approximation of reality (due to absence of memory).
Stochastic modelling and analysis

The standard procedure:
- construct a model (...),
- determine your performance measure of interest,
- choose a solution method:
  - analytical,
  - numerical, or
  - simulation,
- fix model parameters ($\lambda, \mu, ...$),
- derive measure.

play with the model
Why play with model parameters?

- to pose “what if” questions
  
  *perturbation analysis*

- to see how performance changes if parameters change
  
  *sensitivity analysis*

- to find the best performance (tuning)
  
  *optimisation*
Rules of thumb revisited

- **Analytical solution** allows very **quick** and very **precise** insight in your model, but the model tends to be a very **loose approximation** of reality.

- **Simulation** allows relatively **slow** and **rough** insight in a **single** instance of your model, but the model can have a close **correspondance** to reality.

- **Numerical solution** allows **quick** and **precise** insight in a **single** instance of your **Markov chain** model, which usually is an **approximation** of reality (due to absence of memory).

In order to **optimise** (etc.) that computation has to be repeated many times
Overview

- Introduction to QoS modeling and analysis
- Introduction to Statecharts
  - StoCharts
    - Introduction
    - Semantics
    - Applications
- Conclusions and future outlook
Statecharts

Material courtesy of David N. Jansen
Example: Cruise Control

start button [brake released] / desired speed := speed

stop button

Off

On

Active

Suspended

brake

resume button [brake released]
Basic Ideas

- Statecharts :=
  finite state machines +
  hierarchy +
  parallel behaviours

- Originally developed by David Harel
Possible Uses

- Control automata of embedded systems
  - developed to specify avionics of an Israeli aircraft
  - also used e. g. in automotive systems

- Object life cycle in OO software
  - part of the UML

- Protocol specification
  - in UML 2.0
Tools and Methods

- Commercial and research tools available
  - Statemate
  - several UML tools, e.g. Rhapsody

- There are many variants
  - RSML
  - Stateflow
  - Argos / SyncCharts
  - UML Statecharts

- Used in industry
  - e.g. by DaimlerChrysler, Siemens, Airbus
Basic Syntax

- **Nodes**
  - A

- **Transitions**
  - event [guard] / action

- **Initial node**
  - B

- **OR nodes**
  - C
    - D
    - E

- **AND nodes**
  - F
    - G
    - H
    - K
    - I
    - J
Informal Semantics

- The system is always in a configuration
  - Configuration := set of nodes
    (multiple nodes if there is parallelism)

- Upon reception of an event, it takes a step
  - Step := set of transitions that are taken simultaneously
    (multiple transitions if there is parallelism)

- The system...
  - leaves the source node(s)
  - executes the actions of the transitions
  - enters the target node(s)
Some Intricacies
Example: RTnet Protocol

- RTnet is a simple realtime network
  - nodes share the bandwidth of the network
  - real-time guarantees

- A token is passed around
  - Its owner may send data for some time (THT)

- The monitor checks the active node
  - After being active node, a node becomes monitor
  - If the new active node dies, the monitor reconfigures the system
Example: RTnet Protocol

Not token owner

Monitoring

On time →Delayed

/ send token; stop monitoring

Active

Sending → Dispatch

after(THT)

/ poll

stop monitoring

Idle

receive token

after(THT) / poll

stop monitoring
Assignement: RTnet Protocol

Not token owner

Monitoring

On time

Delayed

stop monitoring

Idle

/ send token;
stop monitoring

receive token

Active

Sending

Dispatch

after(THT)

after(THT) / poll
Object-Oriented Statecharts

- Statecharts for object life cycles
- variant of Statechart semantics

- Inheritance / overriding of behaviour?
  - Yes: refine states
  - Perhaps: redirect transitions
  - No: change transition triggers

- Priorities in accordance with inheritance
Example of Behaviour Inheritance

- **Off**
  - **Stop button**
  - **Start button**
  - **Resume button [brake released]**
- **Active**
  - **Start button**
  - **Accelerate button**
  - **Accelerating**
    - **Entry / desired speed := \infty**
    - **Exit / desired speed := speed**
- **Suspended**
  - **Brake**
  - **Resume button [brake released]**
- **On**
  - **Start button [brake released]**
  - **Desired speed := speed**
Run-To-Completion Semantics

- procedure call-like definition of transition:
  - One transition may invoke another transition
  - State of the invoking Statechart is undefined during this invocation
  - may lead to deadlock

- We recommend to abandon this semantics
Semantics of Statecharts

Material courtesy of Joost-Pieter Katoen
What is a UML StateChart?

A UML StateChart $\mathcal{SC} = (\text{Nodes}, \text{Ev}, \text{Edges})$ with:

- A set $\text{Nodes}$ of $\text{nodes}$ structured in a tree

- A (finite) set $\text{Ev}$ of $\text{events}$
  - there is a pseudo-event $\text{after}(d)$ denoting a delay of $d$ time units
  - $\perp \not\in \text{Ev}$ stands for “no event”

- A (finite) set $\text{Edges}$ of edges (in fact, $\text{hyperedges}$)
Syntactic sugar

*This is an elementary form; the UML allows more constructs that can be defined in terms of these basic elements*

- Deferred events: simulate by regeneration
- Parametrised events: simulate by set of parameter-less events
- Activities that take time: simulate by start and end event
- Dynamic choice points: simulate by intermediate state
- Synchronization states: use a hyperedge with a counter
- History states: (re)define an entry point
More about nodes

• Nodes are structured in a tree
  – $\text{children}(x)$ is the set of children of node $x$
  – $\text{root}$ is the unique root node of
  – $x \unlhd y$ means node $x$ is a descendant of node $y$

• Nodes are typed, $\text{type}(x) \in \{\text{BASIC, AND, OR}\}$ such that:
  – the root node is of type OR
  – each leaf node is of type BASIC
  – each child of an AND-node is of type OR
  – each OR-node has a default (initial) node
Foundations of StoCharts

Edges

An edge is a tuple \((X, e, g, A, Y)\), notation \(X \xrightarrow{e[g]/A} Y\), where

- \(X\) and \(Y\) are non-empty sets of nodes
- \(X\) is the source and \(Y\) the target
- \(e \in Ev\) is the trigger event
- \(g\) is a guard, i.e., a Boolean expression
- \(A\) is a set of actions (such as \(send\) \(j.e\) or \(v := \text{expr}\))

“if currently in \(X\) and \(g\) holds, on occurrence of \(e\), actions \(A\) can be performed while evolving into \(Y\)”
Assumptions in our semantics

adopted from [Eshuis & Wieringa 2000]

- Input is event set (and not a queue)
- System reacts to all input events (≠ first); not consumed events “die”
- Instantaneous communication and instantaneous actions
- Unlimited concurrency
- Perfect communication (no loss)
- System reacts to internal events generated in step $i$ in step $i+1$
  - this is called a sequential step semantics
  $\Rightarrow$ yields a causality-preserving semantics
What does a single StateChart mean?

Intuitive semantics as a transition system:

- State = one or more nodes (“current control”) + the values of variables
- Edge is enabled if guard holds in current state
- Executing an edge = perform actions $A$, consume event $e$,
  - leave source state and switch to target state
  $\Rightarrow$ events are unordered, and considered as a set
- Principle: execute as many edges at once (without conflict)
  $\Rightarrow$ this is called a *step*

How to compute a step?
States and configurations

• A configuration $C$ is a set of nodes satisfying
  - $\text{root} \in C$
  - for each OR-node in $C$, there is a unique child in $C$
  - for each AND-node in $C$, all children are in $C$

• State $(C, I, V)$ is a tuple where
  - $C$ is a configuration
  - $I \subseteq Ev$ is the set of events to be processed
  - $V$ is the valuation of the variables
Enabling of an edge

Edge \((X, e, g, A, Y)\) is *enabled* in state \((C, I, V)\) whenever:

- \(X \subseteq C\), i.e. all source nodes are in configuration \(C\)
- guard \(g\) holds in \(C_1\) through \(C_n\) with events \(I_1\) through \(I_n\)
- \(e \in I\), i.e. the input event needs to be present (or equals \(\perp\))

\(En(C, I, V)\) denotes the set of enabled edges in state \((C, I, V)\)
Scope of an edge

Scope of edge $X \longrightarrow Y =$ most nested OR-node containing $X$ and $Y$

The most nested node that is unaffected by executing the edge

```
scope(A → D) = root and scope(A → C) = G and scope(A → B) = F
```
Priority

Priority is a partial order on edges; different priority schemes exist, e.g.

\[ \text{prio}(ed) \geq \text{prio}(ed') \quad \text{if} \quad \text{scope}(ed) \subseteq \text{scope}(ed') \quad \text{UML} \]

\[ \text{prio}(ed) \leq \text{prio}(ed') \quad \text{if} \quad \text{scope}(ed) \subseteq \text{scope}(ed') \quad \text{STATEMATE} \]

\[ \text{prio}(A \rightarrow C) > \text{prio}(A \rightarrow D) \quad \text{since} \quad \text{scope}(A \rightarrow C) = G \subseteq \text{root} = \text{scope}(A \rightarrow D) \]
What is now a step?

A step is a set of enabled edges such that:

- all edges in a step are enabled
- all edges are pairwise consistent
  - they are identical or
  - scopes are (descendants of) different children of the same AND-node
- enabled edge $ed$ is not in step implies
  $$\exists ed' \in \text{step. } (ed \text{ inconsistent with } ed' \text{ and } /_{prio}(ed') < \text{prio}(ed))$$
- a step must be maximal (wrt. set inclusion)
Computing the set $T$ of steps

Initialize $T := \emptyset$

While

there is an enabled edge in $\text{En}(C, I, V) - T$
which is consistent with all edges in $T$

Do

choose one of these edges which has maximal priority
and add this edge to $T$

Return $T$

$\text{Steps}(\text{En}(C, I, V))$ denotes the set of steps in state $(C, I, V)$
Executing a step

\{ \text{root}, A, C \} \xrightarrow{\{t\}} \{ \text{root}, I, D, E, F, H \}

i.e., the \textit{default completion of} \{ \text{root}, I \}

Notation: \( \text{Exec}(C_{1..n}, T_{1..n}) = ((C'_1, I'_1, V'_1), \ldots, (C'_n, I'_n, V'_n)) \)
I/O automata

[ Lynch & Tuttle 1987]

An input-output automaton is a tuple \((L, \ell_0, A, \rightarrow)\) with

- \(L\) is a finite non-empty set of locations with initial location \(\ell_0 \in L\)
- \(A\) is a set of actions partitioned into \(A^{in}\) and \(A^{out}\)
- \(\rightarrow \subseteq L \times A \times L\), the transition relation satisfying
  - for any \(\ell \in L\) and \(a \in A^{in}\), \(\exists \ell' \in L\) such that \(\ell \xrightarrow{a} \ell'\) \(\text{input enabledness}\)

as internal actions are not used here, they are omitted
Foundations of StoCharts

From StateCharts to an I/O automaton

- \( L = (Conf_1 \times 2^{Ev_1} \times Val_1) \times (Conf_2 \times 2^{Ev_2} \times Val_2) \)

- \( \ell_0 = ((C_{0,1}, \emptyset, V_{0,1}), (C_{0,2}, \emptyset, V_{0,2})) \)
  - where \( C_{0,i} \) is the default completion of \( \{ \text{root}_i \} \)

- \( A^{\text{in}} = 2^{Ev_1 \cup Ev_2} - \{ \emptyset \} \)

- \( A^{\text{out}} = 2^E \) with \( E = \{ \text{send} \ j.e \in SC_1 \cup SC_2 \mid j \neq 1, 2 \} \)

- \( E \in A^{\text{in}} \) with \( \ell_j = (C_j, I_j, V_j) \) and \( \ell'_j = (C_j, I_j \cup (E \cap Ev_j), V_j) \)

- \( T_i \in \text{Steps}(En(s_i)) \) for all \( i \)
  - with \( E = \) outputs in \( T_i \) sent to \( SC_j, j \neq 1, 2 \)

\( \odot \) JPK\textcopyright{16}
Overview

- Introduction to QoS modeling and analysis
- Introduction to Statecharts

StoCharts
- Introduction
- Semantics
- Applications

- Conclusions and future outlook
Look, a UML-Statechart!

It models a car damage assessment process.
This Statechart has means to model stochastic phenomena.

We call it a Stoc(hastic Statec)hart, or StoChart for short.
QoS primitives in StoCharts

- **Probabilistic branching**
  - restricted to the *effect* of edges (output)
  - input stays reactive & non-probabilistic
  - Notation uses a 'P-pseudonode' (denoted \(\mathbb{P}\))

- **Random delays**
  - randomising the 'after'-construct of the UML
Random delays

Randomising the 'after'-construct of the UML

Some measured random delay

after(9 min)/end

after(UNIF[5min,15min])/end
QoS primitives in StoCharts

- **Probabilistic branching**
  - restricted to the *effect* of edges (output)
  - input stays reactive & non-probabilistic

![Diagram of probabilistic branching]

- **Random delays**
  - randomising the ‘after’-construct of the UML

```plaintext
after(UNIF[5min,15min])/end
```

That's all.
Overview

- Introduction to QoS modeling and analysis
- Introduction to Statecharts

StoCharts
- Introduction
- Semantics
- Applications

Conclusions and future outlook
Semantics of Stocharts

Material courtesy of Joost-Pieter Katoen
The (small) adaptations needed for StoCharts

This is what's all about!
A StoChart

Foundations of StoCharts

☎ JPK
What is a StoChart?

A StoChart $SC = (Nodes, Ev, PEdges)$ with:

- A set $Nodes$ of nodes structured in a tree
- A (finite) set $Ev$ of events
  - pseudo-event $after(F)$ denotes a random delay determined by distribution $F$
  - $\ldots$ the probability to wait at most $d$ time units is $F(d)$
  - $\bot \notin Ev$ stands for “no event”
- A (finite) set $PEdges$ of probabilistic edges (in fact, hyperedges)
Probabilistic edges

A P-edge is a tuple \((X, e, g, P)\), notation \(X \xrightarrow{e[g]} P\), where

- \(X\) is a non-empty (source) set of nodes
- \(e \in Ev\) is the trigger event, possibly \(after(F)\)
- \(g\) is a guard, i.e., a Boolean expression
- \(P\) is a function assigning probabilities to pairs \((A, Y)\)

“if currently in \(X\) and \(g\) holds, on occurrence of \(e\),
with probability \(P(A, Y)\) target \(Y\) is selected while performing actions \(A\)”
A simple StoChart

\[
\begin{align*}
\text{playing} & \quad \text{toss} \quad 1/2 \ / \ 1.\text{heads} \\
& \quad \text{toss} \quad 1/2 \ / \ 1.\text{tails} \\
& \quad \text{ignore} \\

\{ \text{playing} \}, \text{toss}, \text{true}, P' & \quad \text{with} \quad P'(\emptyset, \{ \text{playing} \}) = 1 \\
\{ \text{playing} \}, \text{toss}, \text{true}, P & \quad \text{with} \quad P(\{ 1.\text{heads} \}, \{ \text{won} \}) = \frac{1}{2} \\
& \quad \text{and} \quad P(\{ 1.\text{tails} \}, \{ \text{playing} \}) = \frac{1}{2}
\end{align*}
\]
Turning a StateChart into a StoChart

A StateChart

- Contacting garage
  - Normal
  - VIP
  - Negotiating with garage
  - Repairing

Assessing
- start
- after(3 days)
- after(1 day)

Normal
- start
- after(3 days)

VIP
- after(1 day)

Negotiating with garage
- after(15 min)

Repairing

turned into a StoChart

- Contacting garage
  - Normal
  - VIP
  - Negotiating with garage
  - Repairing

Assessing
- start
- after(EXP[3 days])
- after(EXP[1 day])

Normal
- after(EXP[3 days])
- after(UNIF[10 min, 20 min])

VIP
- after(EXP[1 day])

Negotiating with garage
- after(UNIF[10 min, 20 min])

Repairing
P-edges, edges and scope

\((j, A, Y)\) is an edge of P-edge \(j = (X, e, g, P)\) if \(P_j(A, Y) > 0\)

For simplicity: all edges of a P-edge must have the same scope
Stochastic I/O automata

I/O automata extended with:

- **Timers** to model probabilistic delays
  - initialized by samples of probability distributions
  - they run backwards, all at the same pace
  - and expire when their value becomes 0

- Three types of transition relations
  - a deterministic input relation (= function)
  - a delay transition relation
  - a *probabilistic* output relation
Stochastic I/O automata

A stochastic I/O-automaton is a tuple \((L, \ell_0, A, T, I, O, \Delta)\) with

- a finite non-empty set \(L\) of locations with initial location \(\ell_0 \in L\)
- a set \(T\) of timers, such that each timer \(t \in T\) has a pdf \(F_t\)
- an input function \(I : L \times A^{in} \rightarrow L\) satisfying
  - input-enabledness
- an output transition relation \(O \subseteq L \times \text{Prob}(A^{out} \times 2^T \times L)\)
- a delay transition relation \(\Delta \subseteq L \times T \times L\)

for output-compatible IOSA, composition can be easily defined
Foundations of StoCharts

An example IOSA

Idle

Preparing to check

Checking

Check finished

Check positive

Error

i: request

o: set t

i: withdraw

0.98, allow

0.02, deny

i: withdraw

i: withdraw

i: withdraw

i: withdraw

i: withdraw
Interpretation of an IOSA

- A state = current location + values of all timers (= timer valuation)

- When is a transition enabled?
  - an output transition is always enabled
  - an input transition is enabled whenever its input action is present
  - a delay transition is enabled whenever its timer has expired

- No transition enabled?
  - time may pass (no change of location) until some transition is enabled

- Taking an output transition emanating from location $l$:
  - nondeterministically select a probability space $\mathcal{P}$ with $(l, \mathcal{P})$ in $O$
  - select one of the possible targets $(a, T, l')$ in $\mathcal{P}$ probabilistically
  - generate output $a$, reset all timers in $T$ and evolve to location $l'$
What kind of stochastic process?

- An IOSA is in fact a generalized semi-Markov decision process
- Each output-deterministic IOSA can be mapped onto a GSMP
  - this is a well-studied class of stochastic processes
  - . . . . . for which efficient discrete-event simulation algorithms exist
  - subclasses of this model are amenable to numerical solutions
    * continuous-time Markov chains (CTMCs)
    * Markov decision processes (MDPs)
    * continuous-time Markov decision processes (CTMDPs)

In most cases we obtain a stochastic process that can be analyzed!
What does a single StoChart mean?

Intuitive semantics as a transition system:

- State = one or more nodes ("current control") + the values of variables
- P-edge is enabled if guard holds in current state
- Executing a P-edge = consume event $e$, and
  - probabilistically select a target state and actions
  - execute actions, leave source state and switch to target state
- Principle: execute as many edges at once (without conflict)
  ⇒ this is called a *step*

Isn’t this the same as before? Yes, almost! Only the probabilistic selection
What is now a step?

A **step** is a set of enabled edges such that:

- all edges in a step are enabled
- all edges are pairwise consistent
  - scopes are (descendants of) different children of the same AND-node
- enabled edge $ed$ is not in step implies $\exists ed' \in \text{step. } (ed \text{ inconsistent with } ed' \text{ and } prio(ed') \geq prio(ed))$
- a step must be **maximal** (wrt. set inclusion)

Isn’t this the same as before? Yes, indeed!
Computing the set of steps

1. Calculate the set of enabled P-edges

2. Calculate all maximal, prioritized, consistent sets of P-edges thereof

3. Select nondeterministically one of them

4. Draw samples from the probability space of the P-edges
   - this results in a set of edges together forming a step
   - the probability of the step is simply the product of the individual edges

Isn’t this the same as before? Well almost, except the last step!
From StoCharts to an IOSA

- \( L = (Conf_1 \times 2^{Ev_1} \times Val_1) \times (Conf_2 \times 2^{Ev_2} \times Val_2) \)

- \( \ell_0 = (s_{0,1}, s_{0,2}) \), where \( s_{0,i} = (C_{0,i}, \emptyset, V_{0,i}) \) is the initial location of \( SC_i \)

- For each P-edge with label \( (F_{ij}) \), \( \exists \) timer \( t_{ij} \in T \) with cdf \( F_{ij} \)

- \( A^{in} = 2^{Ev_1 \cup Ev_2} - \{ \emptyset \} \)

- \( A^{out} = 2^E \) with \( E = \{ \text{send } j.e \in SC_1 \cup SC_2 \mid j \neq 1, 2 \} \)

- \( \frac{E \in A^{in}}{(\ell_1, \ell_2) \xrightarrow{E} (\ell'_1, \ell'_2)} \) with \( s_j = (C_j, I_j, V_j) \) and \( \ell'_j = (C_j, I_j \cup (E \cap Ev_j), V_j) \)

© JPK
The delay transition relation

\[
X \subseteq C_1 \quad (X, \text{after}(F_{1,j}), g, P) \in PEdges_1
\]

\[
((C_1, I_1, V_1), (C_2, I_2, V_2)) \xrightarrow{t_{1,j}} ((C_1, I_1 \cup \{\text{after}(F_{1,j})\}, V_1), (C_2, I_2, V_2))
\]

and symmetrically for \( SC_2 \)
Output transition relation

Let \( PT_i = (2^{\text{Edges}_i}, P_i) \in PSteps(\text{EnP}(C_i, I_i, V_i)) \) for each \( i \)

Then \((C_i, I_i, V_i)_{i=1}^2, (A \times 2^T \times L, P)) \in O \) with probability measure:

\[
P\left(\{(E, T, \text{Exec}(C_{1...2}, T_{1...2}, V_{1...2}))\}\right) = P_1(\{T_1\}) \cdot P_2(\{T_2\})
\]

where \( E \) is the set of events that are sent to the environment, and

\[
T = \{t_{ij} \mid \nu_i(j) \text{ becomes enabled} \}
\]

Let \( P(\{(E, T, \omega)\}) = 0 \) in all other cases
Put in a nutshell

- Semantics StoChart ≈ your favourite Statechart semantics
  - illustrated here for the requirements-level semantics [Eshuis & Wieringa]
  + probabilistic branching and some timers for random delays
  - . . . . . indeed it is as simple as that!

- Formal semantics precisely defines the obtained stochastic process

⇒ This constitutes the basis for trustworthy QoS analysis

- . . . . . Applications indicate what you can do with this framework
  - the proof of the pudding is in the eating!
Overview

- Introduction to QoS modeling and analysis
- Introduction to Statecharts
- StoCharts
  - Introduction
  - Semantics
  - Applications
- Conclusions and future outlook
Applications

Material courtesy of David N. Jansen
Kiezen op Afstand

- KOA = Televote via WWW
- Dutch government project
- pilot (2004):
  ±8.000 voters living abroad e-voted
- final goal:
  allow all voters to e-vote on Vote Day

high availability (and performance)

in a very limited period (12 h)
Quality Goal

- Long downtimes are unacceptable
- Long downtime := 10 minutes
- Unacceptable := once in $10^3$ votes

Q: What is the probability that a televote system has an unacceptable downtime?

A: ... using a StoChart model ...
A Televote System
Firewall model

- Initializing
  - after(1 min)
  - after(EXP[10 min])
- Working
  - after(EXP[500 d])
- Down
  - after(EXP[10 min])
Televote: Assignment

- **Model the Vote and Backup servers**

- **Vote server characteristics:**
  - startup time: 2 min
  - MTTF: 500 days
  - MTTR: 10 min

- **Backup server characteristics:**
  - startup time, MTTF and MTTR as above
  - from hot standby to full operation: 1 min
Televote: Solution
Televote: System Analysis

- Simulated using the tool Ymer
  - Statistical sampling:
    generate large number of runs and estimate the desired probability
  - Analyzes GSMPs
  - Developed by Håkan Younes at CMU

- Result: Yes!
  This Televote system is reliable enough
One-Armed Bandit

- Gambling machine
- 3 reels show fruit symbols etc.
- Some combinations incur a prize

- Prize probability prescribed by law
One-Armed Bandit: Assignment

- The 3 reels spin in parallel and (almost) independently
- Easy to model as a StoChart

Assignment:
draw a model of the reels
One-Armed Bandit: Solution
One-Armed Bandit: System Analysis

- Model checked using the tool Prism
  - numerical solution of the Markov decision process
  - developed by Kwiatkowska’s group (Birmingham)

- Desired properties checked: legal requirements on win and loss
Example Properties

- In 30 games, the player loses $\leq 15$ coins with probability $>0.5$

- In 30 games, the player wins $<10$ coins with probability $\geq 0.99$

- In 1000 games, the probability that the automaton cannot pay out is $\leq 10^{-20}$
European Train Control System
Train Interoperability

- Interoperability:
  one railway’s train runs on another railway’s track
- Some interoperability is given

- Broken by different security systems
Securing Trains: Practice

- Signals show movement authorities to the driver
- Some protection against human error
  - Indicate passage of danger points

Eurobalise
- Trackside transceiver
- Transmits position, movement authorities etc.

Different national systems
ETCS: Next-generation trains

- Future standard for cross-European trains
- Defines train interoperability, not train internals
- Moves functionality inside train, to improve track utilisation
- Train-trackside communication via 'GSM-R'
Speaking technically

- **Eurobalise**
  - trackside transceiver
  - transmit position and movement authorities etc.

- **GSM-R**
  - a variant of GSM
  - transmit movement authorities etc.
  - communicate with trackside radio block center (RBC)

- **Cab signalling and on-board integrity check**
  - train internals
  - only a few aspects specified

ETCS Level 1

ETCS Level 2+3

ETCS Level 3
Is GSM-R reliable enough?

- Various standard documents detail reliability, dependability and availability requirements to be fulfilled by
  - GSM-R
  - ETCS level 1, 2, 3

- We take the GSM-R specs as guarantees, and verify the ETCS requirements

- Level 3:
  - ‘Moving block operation’
  - ‘private’ track area moves with train
  - train informs track-side dispatcher at regular intervals about
    - position
    - train integrity
  - dispatcher notifies following train

Why? Is expected to allow shorter headways.
ETCS Level-3: Moving Block Operation

- Enabled by on-board integrity check
- Each fraction of the rails is freed immediately after the train has passed...
- ... and can be reserved for the next train without delay
- shorter headway thus better track utilisation

block of following train  |  block of preceding train

Integrity Check
ETCS Level-3: Moving Block Operation

- Enabled by on-board integrity check
- Each fraction of the rails is freed immediately after the train has passed...
- ... and can be reserved for the next train without delay
- shorter headway thus better track utilisation
ETCS train radio reliability

Question: Can GSM-R radio handle train communications?
- fast ($\geq 300$ km/h)
- in dense traffic (headway $\approx 1$ min)
- with high reliability (99.95%)

Answer: Yes

So, that's good. Let's skip the details then.
Assumptions and Guarantees

- “Design by Contract” paradigm

- If the environment keeps the assumptions, the system is guaranteed to fulfil its duty.

- Our assumptions: GSM-R works as specified
  - e. g. “a GSM-R connection is established within 5 sec with 95% probability.”

- Our guarantees: ETCS radio is as dependable as specified
  - e. g. “the communication succeeds with 99.95% probability”
Sender Model

Sender

- after(5 sec)
- position report ready
- in(connected) / transmit
- position report sent
- unconnected entry / try
- connect
- detect connection loss
- connected
- h_start
- h_end
- cell handover
ETCS: Assignment

- Complete the connection timing in this transmission medium model.
Overview

- Context: Train signalling
- European Train Control System Standard
- Models and Modelling
- ETCS Communication Reliability

Experiments and Results

Conclusion
Model Analysis

StoChart model → ProVer → Set of runs → Postprocessor → Desired guarantee

translational

operational

IOSA

MoDeST spec

MoToR/Mobius

MoToR/Mobius

MoToR/Mobius

Result
**ProVer**

- model checker look-alike simulation tool:
  checks whether a probabilistic property is satisfied

  - e. g.: Is the probability of a failure less than 1%?
  - Possible answer: Yes, with confidence 0.99.

- tailored to GSMPs (deterministic IOSA)

- developed at CMU by Håkan Younes
MoDeST tool support

- **MoToR:**
  a frontend for MoDeST developed at University of Twente

- **Mobius:**
  - flexible modelling and analysis environment for discrete event systems
  - developed in the PERFORM group at UIUC Urbana-Champaign
  - rooted in Petri nets, but quite component-oriented
Communication Reliability

- Is the communication reliable enough?

- Required by the spec: 99.95%

Does the next message arrive at the RBC within $t$ sec?

<table>
<thead>
<tr>
<th>$t$</th>
<th>PROVER direct</th>
<th>PROVER +POSTPROCESSOR</th>
<th>MoDEST +POSTPROCESSOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 sec</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10 sec</td>
<td>0.98267±9</td>
<td>0.98271±6</td>
<td>0.9840</td>
</tr>
<tr>
<td>15 sec</td>
<td>0.999700±9</td>
<td>0.999688±8</td>
<td>0.9997</td>
</tr>
<tr>
<td>20 sec</td>
<td>0.9999944±6</td>
<td>0.9999950±10</td>
<td>0.9999</td>
</tr>
</tbody>
</table>
Delayed Trains

- How often do GSM-R failures cause delays?

- Challenging scenario:
  Two trains at minimal distance
  - for a full trip (~ 1 hour)
  - at maximum speed (300 km/h)
  - with moving block operation

```
block of following train
```
```
block of preceeding train
```
Delayed Trains

- **Every 5 seconds**
- **4 train pairs per hour**
- **< 1 train per month delayed**

### Table

<table>
<thead>
<tr>
<th>max $\Delta t$</th>
<th>ProVer direct</th>
<th>ProVer $+$ Postprocessor</th>
<th>MoDEST $+$ Postprocessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 5 sec</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>&gt; 10 sec</td>
<td>0.9562± 9</td>
<td>0.9551± 13</td>
<td>0.9338± 16</td>
</tr>
<tr>
<td>&gt; 15 sec</td>
<td>0.101± 2</td>
<td>0.1006± 9</td>
<td>0.0784± 18</td>
</tr>
<tr>
<td>&gt; 20 sec</td>
<td>0.0036± 4</td>
<td>0.0041± 4</td>
<td>0.0023± 3</td>
</tr>
<tr>
<td>&gt; 25 sec</td>
<td>0.00034±11</td>
<td>0.00029±11</td>
<td>0.00004± 4</td>
</tr>
<tr>
<td>&gt; 75 sec</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Hawks and Doves
Simulation Games

- Model used in biology
- Conflicts between animals
  - Fight for some advantage (e.g., food, territory)
  - Measured in points

- Strategies considered:
  - Hawk: Fight forcefully, don't give up unless severely injured
  - Dove: Fight with limited effort, or give up before getting injured
System Model

- Arbiter selects who is to fight nondeterministically
- Three individuals select hawk/dove strategy probabilistically
- Arbiter decides who wins
Individual StoChart

- \( p = \text{probability to choose dove strategy} \)
Hawks and Doves: Assignment

- Complete the Arbiter StoChart

Diagram:

- Idle
  - I1 fights I2
  - I1 fights I3
  - I2 fights I3

States:
- [I1.hawk and I3.hawk] / ...
- [I1.hawk and I3.dove] / ...
- [I1.dove and I3.dove] / ...
- [I1.dove and I3.hawk] / ...
- ?
Hawks and Doves: Solution
Hawk and Doves: Analysis

- Model checked using the tool Prism

- Situations checked:
  - 3 peaceful animals
  - 3 aggressive animals
  - 1 aggressive and 2 peaceful animals
Hawks and Doves: Analysis

- The probability to die is < 1%
  - “to die” = “pts < 1”

  - 3 peaceful: yes
  - 3 aggressive: no

- The probability to earn 20 points in 100 steps is < 75%.
  - 3 peaceful: no
  - 3 aggressive: yes
  - mixed: the hawk has an advantage
Overview

- Introduction to QoS modeling and analysis
- Introduction to Statecharts
- StoCharts
  - Introduction
  - Semantics
  - Applications

Conclusions and future outlook
Conclusions
Conclusion

What did you learn?

- Stochastic modelling principles
- Statechart principles

StoCharts

- Principles
- Applications