Overview

An introduction to Statecharts

- StoCharts
- Syntax
- Semantics
- Case Study: A Beverages dispenser
- Conclusion
The UML perspective - A scenario

Name: Mr Troy aka Mr T
Age: 29
Position: Software Engineer

Name: Mr McNamara aka Mr M
Age: 27
Position: Visual designer

T seems to have an idea!
The UML perspective - Getting the message across

Mr. T has a great design idea

but ...

Mr. M doesn’t seem to understand T’s design

What do you think M?

What is that T?
The UML perspective - The Problem

Mr. T has an idea in his head

but ...

Mr. M forms a completely different one in his head

This is ... NOT good
The UML perspective - Enter UML

Why does Mr. M get it?

UML provides a Lingua franca that promotes communication between designers.

Mr. T redesigns with UML....

Ok what about this?

Mr. M gets it !!!

I see it now!

Lingua franca =
Common Language
Introducing StoCharts – An Intuition

StoChart Example: 6 wins the game

(1) Dice starts rolling
(takes some time to stop: 3 sec)

(2) Nondeterministic choice
- Dice cannot be read
- Dice can be read

(3) Probabilistic choice
- \{1,2,3,4,5\} face-up
  (start rolling again)
- 6 face-up
  (game over)

\[\perp\]: No need to wait to take edge
Some Distinctions – Probabilistic choice

(Example: Throwing a fair coin)

Probabilistic choice:

It is **not clear** which possibility out of a discrete set a system may take.

So what does “not clear” mean?

- Prediction: 50% of the time heads/tails will occur.
- **BUT** impossible to predict the actual outcome for each toss i.e., whether it’s a head or a tail.
Some Distinctions - Stochastic timing

(Example: Video Streaming)

Client

Video request

Webserver

Stream buffered

Scheduling of video packets

Video displayed

Stochastic timing?

- Some time needed to schedule video packets (server).
- Some time for video to be displayed (client).
Some Distinctions - Probabilistic vs (non) deterministic Phenomena

Probabilistic (Examples: Weather predictions - Probability of rain, snow etc; Rolling of dice in a board game)

Deterministic (Example: Forced moves in chess)

Nondeterministic (Examples: Free moves in chess and Tic Tac Toe)
Some Distinctions - Non/Not determinism

Nondeterministic ↔ Not deterministic

Not deterministic Phenomena
Only infers that a phenomenon is either probabilistic or nondeterministic (and nothing more !!)

An unfortunate terminology but one we have to live with :-(
Some Distinctions - System vs Environment randomness

Environment randomness

Environment randomness (Example: Reactive system - Bank ATM, Dispenser, etc)

Intuition: System exposed to external random stimuli and reacts.

System reaction is **NOT** random
Some Distinctions - System randomness

Requirement: Modems must transmit on different frequencies.

Frequencies = \{1200Hz, 1800Hz\}

Notation:
- \(H_j\): “Heads” for modem \(j\) (1 \(\leq j \leq 2\))
  [Semantics: Modem \(j\) transmits at 1200 Hz]
- \(T_j\): “Tails” for modem \(j\) (1 \(\leq j \leq 2\))
  [Semantics: Modem \(j\) transmits at 1800 Hz]

Randomised Algorithm (modem 1)

1. Throw a coin.
2. (i) Heads \(\Rightarrow\) Transmit at 1200 Hz.
   (ii) Tails \(\Rightarrow\) Transmit at 1800 Hz.
3. No response received (modem 2 using similar frequency) after some time \(\Rightarrow\) goto 1
   else end. (modems using different frequencies)
Overview

- An introduction to Statecharts

StoCharts

- Syntax
- Semantics

Case Study: A Beverages dispenser

Conclusion
UML Extensions - P-Statecharts & StoCharts

We can extend UML Statecharts: 2 fold Extension

Extension 1: Probabilistic choice
(Example: Heads/Tails choice on a coin toss)

Extension 2: Stochastic timing
(Example: Timing in a Drink dispenser)

\[
\text{Statecharts} + \text{Extension 1} \\
= \text{Probabilistic Statecharts (aka P-Statecharts)}
\]

\[
\text{Statecharts} + \text{Extension 1} + \text{Extension 2} \\
= \text{Stochastic Statecharts (aka StoCharts)}
\]
UML Extensions - Random probability distributions

- **DET [x min]**
  A deterministic distribution with a duration of x minutes

- **EXP [x min]**
  A negative exponential distribution with a mean of x minutes

- **UNIF [x min, y min]**
  A uniform distribution in the interval from x to y minutes
UML Extensions - Random probability

- **DET[5min]**: Discrete function
- **EXP[1min]**: Exponential distribution
- **UNIF[5min,10min]**: Uniform distribution
In a *nutshell* ...

- UML Statecharts are trivial P-Statecharts
- UML Statecharts + Probabilistic choice => P-Statecharts
- P-Statecharts + Stochastic timing => StoCharts
- UML Statecharts are trivial StoCharts
Overview

- An introduction to Statecharts
- StoCharts
  - Syntax
  - Semantics
- Case Study: A Beverages dispenser
- Conclusion
A syntactical perspective - StoChart drawing

Nodes

Parent node C encloses its children A, B
( Default node: A )

F (AND) node partitioned by (OR)
children A and C
( Default nodes: B,D )
A syntactical perspective - StoChart drawing

Trivial P-edges

\[ e[g]/A \]

X : Set of source nodes
Y : set of target nodes
e : event
g : guard
A : action set

Here: Actions “A” have the form \((send) \ j.e\) (send event e to the external component identified by j)

Example (Router Login)

Check password[password=“admin”] /User.Logged in

Verifying User

Showing Admin Page
A syntactical perspective - StoChart drawing

Non trivial P-edges

X : Set of source nodes
Y : set of target nodes
e : event
g : guard
A : action set
p,p′: Probabilities
p=p′ is also possible

Example (File downloading)

Checking origin

Check country/

0.4/downloader.UK servers available

0.6/downloader.USA servers available

Showing UK Server 1

Showing USA Server 7
A syntactical perspective - System Perspective

- Intuition: A system is a finite collection of intercommunicating StoCharts

- Formally: Denotable by the set seen below.

\[ \{ SC_i \mid 1 \leq i \leq n, n \in N \} \]
A syntactical perspective - StoChart

StoChart 4-Tuple

\[ SC_i = (Nodes_i, Events_i, Vars_i, PEdges_i) \]

- **Nodes\_i**: A finite set of nodes structured in a tree
- **Events\_i**: A finite set of events
- **Vars\_i**: A finite set of local variables
- **PEdges\_i**: A finite set of P-edges
A syntactical perspective - StoChart Events

\[ SC_i = (Nodes_i, Events_i, Vars_i, PEdges_i) \]

- A finite set of events (Signals that may trigger a state transition)
  - \( \bot \notin Events_i \) - No event is required to trigger a P-edge
  - \( PSEvents_i \) - Set of Pseudo events
  - \textit{pseudo event} \( \text{after}(F) \) - stochastic random delays
A syntactical perspective - StoChart Variables

\[ SC_i = (Nodes_i, Events_i, Vars_i, PEdges_i) \]

- Intuition: A finite set of local variables with an initial value that assigns initial values to variables
- Formally: Denotable by the function below

\[ V_0 : Vars_i \rightarrow \mathbb{Z} \]
A syntactical perspective - StoChart P-edges

P-edge 4-Tuple

(X, e, g, P)

A finite set of P-edges

- \( X \subseteq \text{Nodes}_i \)  
  A non-empty set of source state nodes

- \( e \in \text{Events}_i \cup \text{PSEvents}_i \cup \{\perp\} \)  
  Triggering event that occurs

- \( g \in \text{Guards}_i \)  
  A guard existing on the edge

- \( P \) denotes the possible actions and target state nodes (formal description needed)
A syntactical perspective - StoChart P-edges

A formal description of $P$

$(X, e, g, P)$

$P$ can be formally described by a probability measure in a discrete probability space

$(\text{Pow}(\text{Actions}_i) \times (\text{Pow}(\text{Nodes}_i) - \{\emptyset\}), P)$

Denoting the set of possible outcomes

$\Omega$

$\text{Pow}$: Denotes a Power set
Intuition: The scope of a P-edge is the smallest (in the parent-child hierarchy) OR-node containing both the source nodes and the target nodes.

Scope(A→H) = root
Scope(A→D) = G
Scope(A→B) = C
Overview

- An introduction to Statecharts
- StoCharts
  - Syntax
  - Semantics
- Case Study: A Beverages dispenser
- Conclusion
A semantical perspective – Automaton approach

IOSA 7-Tuple

\((L, l_0, T, A, I, O, \Delta)\)

**IOSA = Stochastic I/O-Automaton**

- **L**
  - A set L of locations with an initial location \(l_0 \in L\)

- **T**
  - A set of timers (each possessing a cdf)

- **A**
  - A set of actions partitioned into input-actions (from external environment) and output-actions (system intrinsic)

- **I**
  - An input transition relation

- **O**
  - A probabilistic output transition relation

- **\(\Delta\)**
  - A delay transition relation
A semantical perspective - Automaton approach

IOSA 7-Tuple

$$(L, l_0, T, A, I, O, \Delta)$$

An input transition relation

$I : L \times A^{in} \rightarrow L$
A semantical perspective - Automaton approach

IOSA 7-Tuple

\[(L, l_0, T, A, I, O, \Delta)\]

A probabilistic output transition relation

\[O \subseteq L \times P(A^{out} \times \text{Pow}(T) \times L)\]
A semantical perspective - Automaton approach

**IOSA 7-Tuple**

\[(L, l_0, T, A, I, O, \Delta)\]

A delay transition relation

\[\Delta \subseteq L \times T \times L\]
### A semantical perspective - Enabled Transitions

**IOSA 7-Tuple**

$$(L, l_0, T, A, I, O, \Delta)$$

- **o**: output transition
- **i**: input transition
- **d**: delay transition

<table>
<thead>
<tr>
<th>Transition</th>
<th>Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Always</td>
</tr>
<tr>
<td>i</td>
<td>Only if Input action present (Input enabledness)</td>
</tr>
<tr>
<td>d</td>
<td>Only if timer expires (reaches zero)</td>
</tr>
</tbody>
</table>
A semantical perspective - IOSA example

**Bank behavior being modeled!**

**ATM**

- request $\rightarrow$ i
- allow or deny $\leftarrow$ o
- withdraw $\rightarrow$ i

**Input transition enabled**

**Output transitions always enabled**

- i: request
- i: withdraw

- o: output transitions
- d: delay transitions
- i: input transitions

- Timer set
- Timer has expired

- Bank behavior being modeled!

**No withdraw without positive verification!**

- 0.98, allow
- 0.02, deny

**Probabilistic choice**

**Check positive**
Overview

- An introduction to Statecharts
- StoCharts
  - QoS Extensions
- Syntax
- Semantics

Case Study: A Beverages dispenser

Conclusion
How does it work?

 Drinks offered?

“Espresso” coffee and “oolong” tea.

 Payment?

Money chips – 1 Chip per drink.
A beverage dispenser - Modeling perspective

What are we modeling?

The behavior of the system (system randomness), as seen by the customer.

What we are not modeling

The expected/desired behavior of the customer (environmental randomness) is not being considered here.
A naive UML Statechart - Events

- Events are sent from the customer to the HBD.
  - insert chip.
  - select espresso.
  - select oolong.

HBD = Hot Beverages Dispenser
A naive UML Statechart - Closer look

HBD - Hot Beverages Dispenser

- insert chip/
  Intuition: In the event of a chip being inserted, the system evolves to its next state

- select oolong/customer.brew oolong
  Intuition: In the event that oolong is selected, send the event “brew oolong” to the customer

- Analog for espresso
What’s wrong with our *naive* design?

- Incomplete – Drinking what we paid for?
- Completing the model
  - “Placing” of cups in cup holder
  - “Filling” of the cups
  - “waiting” for someone (not a system) to remove the cup

---

An improved UML Statechart – Closer look

**Improved HBD (IHBD)**

- Waiting for chip
  - insert chip/
  - Placing cup
  - select Espresso/
    - customer.brew espresso
  - select Oolong/
    - customer.brew oolong
  - Filling cup with espresso
    - cup full/customer.
      - notify drink ready
  - Filling cup with oolong
    - cup full/customer.
      - notify drink ready
  - Waiting for cup to be removed
  - remove cup/
Limitations of the IHBD

Modeling system requirements?

1. No cups => coin returned to customer
2. Return a chip exactly after 10 seconds (“timeout”)
3. Brewing takes about 5 seconds (“timeout”)

Realisation: Impossible to model our requirements without extensions !!!
Extending UML Statecharts - Modeling requirements

Requirements to be modeled

1. No cups $\Rightarrow$ coin returned to customer
2. Return a chip exactly after 10 seconds ("timeout")
3. Brewing takes about 5 seconds ("timeout")

IHBD extended
(probabilistic choice & stochastic timing)

"Probabilistic choice"
Enough cups vs not enough cups

"Stochastic timing"
after(DET [10s] ) /
customer.return chip

"Stochastic timing"
after(EXP [5s] ) /
customer.notify drink ready
Requirements to be modeled

1. No cups => coin returned to customer
2. Return a chip exactly after 10 seconds ("timeout")
3. Brewing takes about 5 seconds ("timeout")
Extending UML Statecharts - Statecharts compared

IHBD - without extensions.
Modeling requirements involving Stochastic behavior not possible !

S-IHBD (IHBD with extensions).
Modeling requirements involving Stochastic behavior now becomes possible !!
Overview

- An introduction to Statecharts
- StoCharts
  - QoS Extensions
- Syntax
- Semantics
- Case Study: A Beverages dispenser

Conclusion
What have we learnt?

- Awareness of **real** UML limitations and the need to extend them.

- Extending UML Statecharts with:
  
  - Probabilistic choice (**P-Statecharts**).
  
  - Stochastic timing (**StoCharts**).

- Informal and formal syntax of **StoCharts** (with examples).

- Informal semantics: Stochastic Input/Output Automata (**IOSA**).

- Application of UML extensions: Case study of beverage dispenser.
Some useful reading


Thank you for your attention!