Software Design, Modelling and Analysis in UML

Lecture 17: Live Sequence Charts I

2016-01-21

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

Last Lecture:
- Hierarchical state machines: the rest
- Deferred events
- Passive reactive objects

This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What are constructive and reflective descriptions of behaviour?
  - What are UML Interactions?
  - What is the abstract syntax of this LSC?
  - How is the semantics of LSCs constructed?
  - What is a cut, fired-set, etc.?

- **Content:**
  - Rhapsody code generation
  - Interactions: Live Sequence Charts
  - LSC syntax
  - Towards semantics
A Closer Look to Rhapsody Code Generation
Rhapsody

```
E
```

"DIO just moved from s₀ to s₁ by transition t."

```
run
```

```
build/make
COMPILED

DefaultComponent.exe
```
You are here.
Reflective Descriptions of Behaviour
Recall:

- The **semantics** of the UML model \( \mathcal{M} = (CD, IM, OD) \) is the **transition system** \((S, \to, S_0)\) constructed according to discard/dispatch/continue/etc.-rules.
- The **computations** of \( \mathcal{M} \), denoted by \([\mathcal{M}]\), are the computations of \((S, \to, S_0)\).

A **requirement** \( \vartheta \) is a property of computations; something which is either satisfied or not satisfied by a computation

\[
\pi = (\sigma_0, \varepsilon_0) \xrightarrow{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \xrightarrow{(cons_1, Snd_1)} \cdots \in [\mathcal{M}],
\]

denoted by \( \pi \models \vartheta \) and \( \pi \not\models \vartheta \), resp.

We write \( \mathcal{M} \models \vartheta \) if and only if \( \forall \pi \in [\mathcal{M}] \bullet \pi \models \vartheta \).

**Simplest case**: OCL constraint viewed as invariant.

But how to formalise

“if a user enters 50 cent and then (later) presses the water button (while there is water in stock), then (even later) the vending machine will dispense water”? 
Harel (1997) proposes to distinguish constructive and reflective descriptions:

- “A language is **constructive** if it contributes to the dynamic semantics of the model. That is, its constructs contain information needed in executing the model or in translating it into executable code.”

  A constructive description tells **how** things are computed (which can then be desired or undesired).

- “Other languages are **reflective** or **assertive**, and can be used by the system modeler to capture parts of the thinking that go into building the model – behavior included –, to derive and present views of the model, statically or during execution, or to set constraints on behavior in preparation for verification.”

  A reflective description tells **what** shall or shall not be computed.

**Note:** No sharp boundaries! (Would be too easy.)
In UML, reflective (temporal) descriptions are subsumed by interactions. A UML model \( M = (C, D, SM, OD, I) \) has a set of interactions \( I \).

- An interaction \( I \in I \) can be (OMG claim: equivalently) diagrammed as:
  - communication diagram (formerly known as collaboration diagram),
  - timing diagram, or
  - sequence diagram.
Interactions as Reflective Description

- In UML, reflective (temporal) descriptions are subsumed by interactions. A UML model $M = (C, I, O, D, I)$ has a set of interactions $I$.
- An interaction $I \in I$ can be (OMG claim: equivalently) diagrammed as
  - communication diagram (formerly known as collaboration diagram),
  - timing diagram, or
  - sequence diagram.

Figure 14.27 - Communication diagram

Figure 14.30 - Communication diagram

Figure 14.31 - Timing diagram with more than one Lifeline and with Messages

(OMG, 2007, 513)
**Most Prominent**: Sequence Diagrams — with **long history**:

- **Message Sequence Charts**, standardized by the ITU in different versions, often accused to lack a formal semantics.
- **Sequence Diagrams** of UML 1.x

Most severe **drawbacks** of these formalisms:

- unclear **interpretation**: example scenario or invariant?
- unclear **activation**: what triggers the requirement?
- unclear **progress** requirement: must all messages be observed?
- **conditions** merely comments
- no means to express **forbidden scenarios**
Thus: Live Sequence Charts

- **SDs of UML 2.x** address some issues, yet the standard exhibits unclarities and even contradictions Harel and Maoz (2007); Störrle (2003)

- For the lecture, we consider **Live Sequence Charts** (LSCs) Damm and Harel (2001); Klose (2003); Harel and Marelly (2003), who have a common fragment with UML 2.x SDs Harel and Maoz (2007)

- **Modelling guideline**: stick to that fragment.
Live Sequence Charts — Syntax
Let $\Theta = \{\text{hot, cold}\}$. An **LSC body** is a tuple

$$(I, (\mathcal{L}, \preceq), \sim, \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv})$$

- $I$ is a finite set of **instance lines**,
- $(\mathcal{L}, \preceq)$ is a finite, non-empty, **partially ordered** set of **locations**; each $l \in \mathcal{L}$ is associated with a temperature $\theta(l) \in \Theta$ and an instance line $i_l \in I$,
- $\sim \subseteq \mathcal{L} \times \mathcal{L}$ is an **equivalence relation** on locations, the **simultaneity** relation,
Let $\Theta = \{\text{hot, cold}\}$. An **LSC body** is a tuple $(I, (\mathcal{L}, \leq), \sim, \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv})$

- $I$ is a finite set of **instance lines**,
- $(\mathcal{L}, \leq)$ is a finite, non-empty, **partially ordered** set of **locations**; each $l \in \mathcal{L}$ is associated with a temperature $\theta(l) \in \Theta$ and an instance line $i_l \in I$,
- $\sim \subseteq \mathcal{L} \times \mathcal{L}$ is an **equivalence relation** on locations, the **simultaneity** relation,
- $\mathcal{I} = (\mathcal{I}, \mathcal{C}, V, \text{atr}, \mathcal{E})$ is a signature,
- $\text{Msg} \subseteq \mathcal{L} \times \mathcal{E} \times \mathcal{L}$ is a set of **asynchronous messages** with $(l, b, l') \in \text{Msg}$ only if $l \leq l'$,
  **Not:** instantaneous messages — could be mapped to method/operation calls.
- $\text{Cond} \subseteq (2^{\mathcal{L}} \setminus \emptyset) \times \text{Expr}_\mathcal{F} \times \Theta$ is a set of **conditions** where $\text{Expr}_\mathcal{F}$ are OCL expressions over $W = I \cup \{\text{self}\}$ with $(L, \text{expr}, \theta) \in \text{Cond}$ only if $l \sim l'$ for all $l, l' \in L$,
- $\text{LocInv} \subseteq \mathcal{L} \times \{\circ, \bullet\} \times \text{Expr}_\mathcal{F} \times \Theta \times \mathcal{L} \times \{\circ, \bullet\}$ is a set of **local invariants**,
Well-Formedness

**Bondedness/no floating conditions:** (could be relaxed a little if we wanted to)

- For each location \( l \in \mathcal{L} \), **if** \( l \) is the location of
  - a **condition**, i.e. \( \exists (L, expr, \theta) \in \text{Cond} : l \in L \), or
  - a **local invariant**, i.e. \( \exists (l_1, i_1, expr, \theta, l_2, i_2) \in \text{LocInv} : l \in \{l_1, l_2\} \), or

then there is a location \( l' \) **equivalent** to \( l \), i.e. \( l \sim l' \), which is the location of

- an **instance head**, i.e. \( l' \) is minimal wrt. \( \preceq \), or
- a **message**, i.e.

\[
\exists (l_1, b, l_2) \in \text{Msg} : l \in \{l_1, l_2\}.
\]

**Note:** if messages in a chart are **cyclic**, then there doesn’t exist a partial order (so such charts **don’t even have** an abstract syntax).
References
References


