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

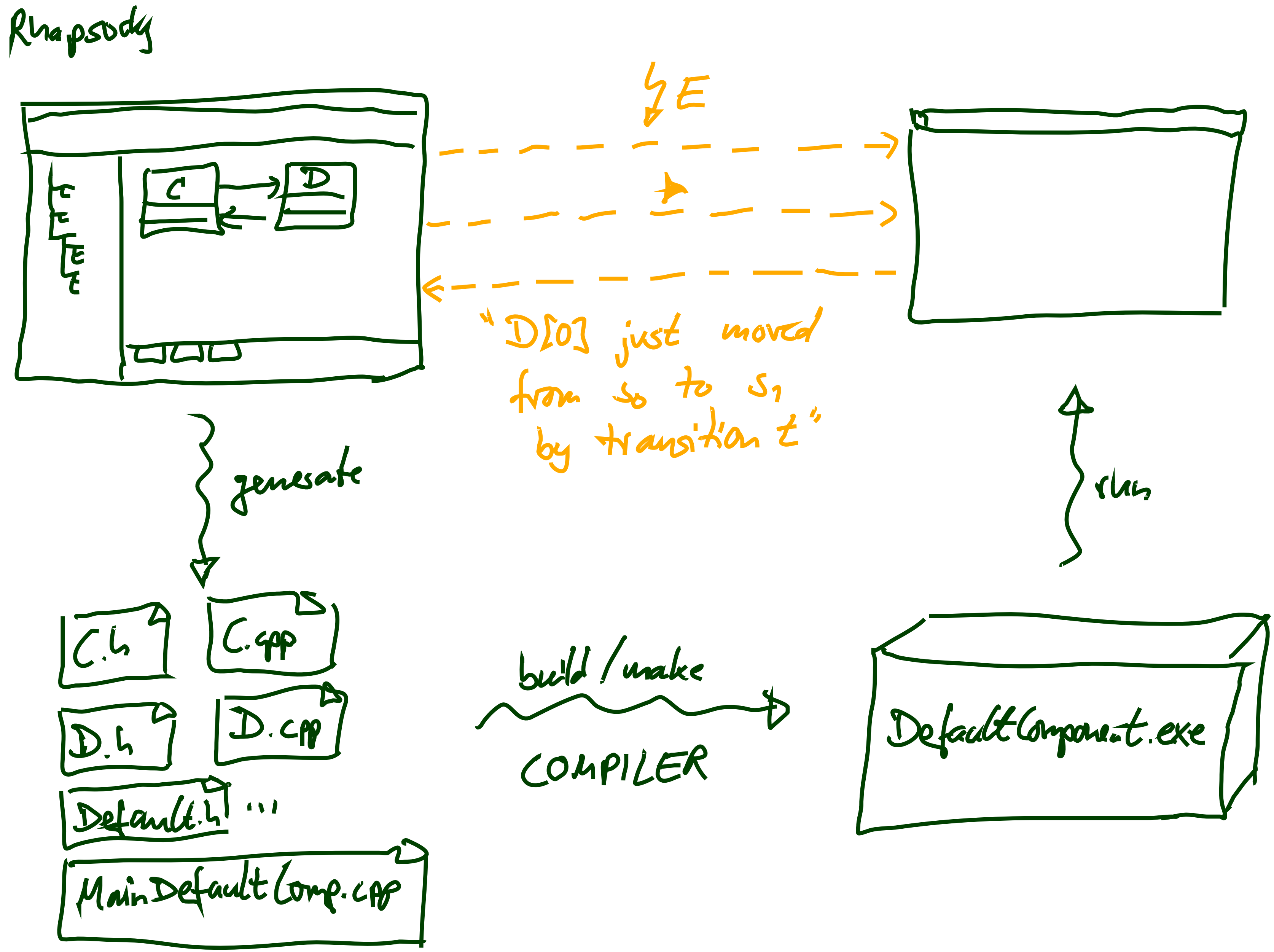
Last Lecture:

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

This Lecture:

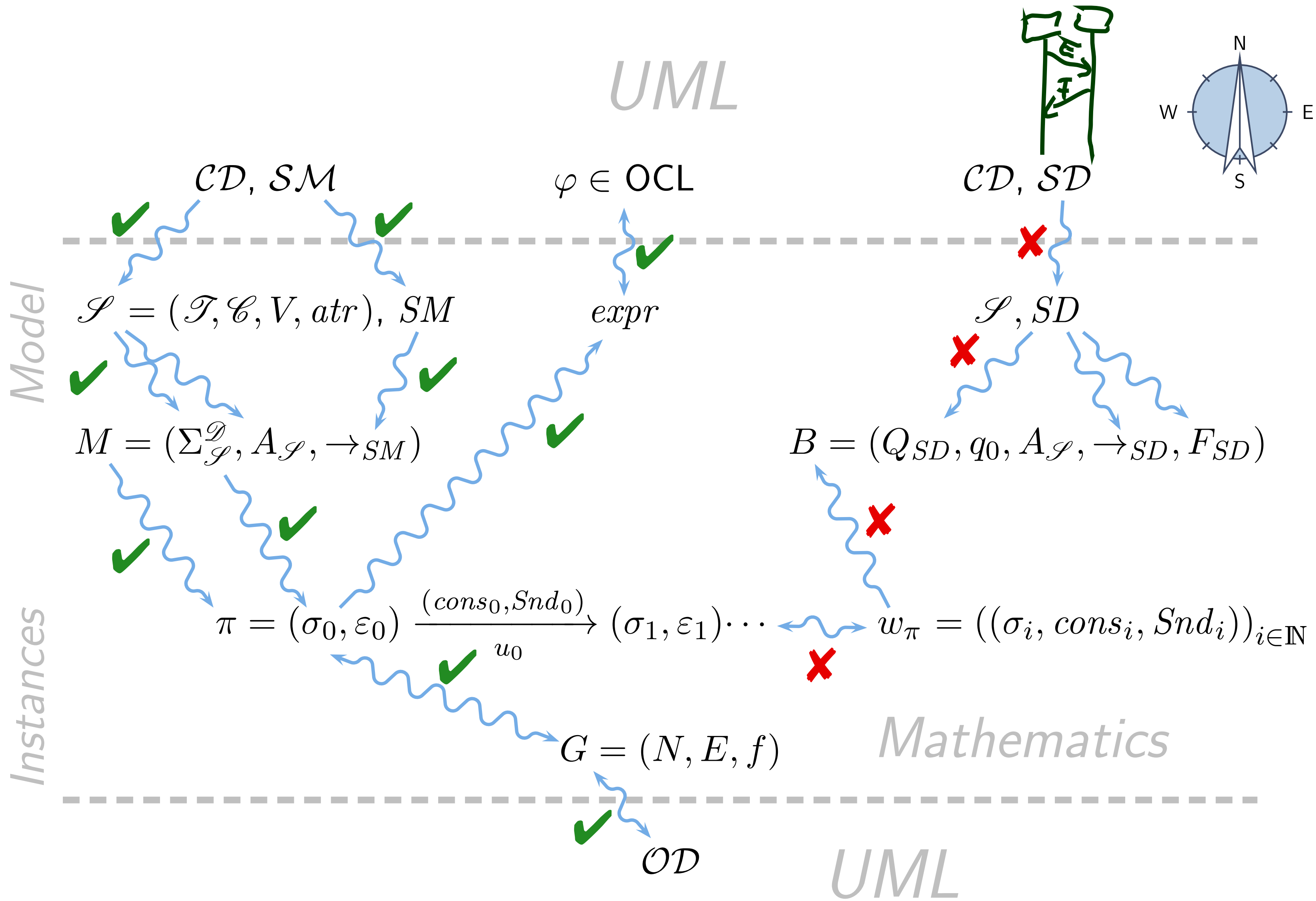
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What are constructive and reflective descriptions of behaviour?
  - What are UML Interactions?
  - What is the abstract syntax of this LSC?
  - How is the semantics of LSCs constructed?
  - What is a cut, fired-set, etc.?
- **Content:**
  - Rhapsody code generation
  - Interactions: Live Sequence Charts
  - LSC syntax
  - Towards semantics

A Closer Look to Rhapsody Code Generation


Rhapsody
C
D
E
"D[0] just moved from s_0 to s_1 by transition t"
generate
C.h
C.cpp
D.h
D.cpp
Default.h ...
MainDefaultComp.cpp
build / make
COMPILER
DefaultComponent.exe
run

You are here.

Course Map


UML
$\mathcal{CD}$, $\mathcal{SM}$
$\varphi \in$ OCL
$\mathcal{CD}$, $\mathcal{SD}$
N
W
E
S
Model
$\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$, SM
$expr$
$\mathscr{S}$, SD
$M = (\Sigma_{\mathscr{S}}^{\mathscr{D}}, A_{\mathscr{S}}, \rightarrow_{SM})$
$B = (Q_{SD}, q_0, A_{\mathscr{S}}, \rightarrow_{SD}, F_{SD})$
Instances
$\pi = (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \cdots$
$w_\pi = ((\sigma_i, cons_i, Snd_i))_{i \in \mathbb{N}}$
$G = (N, E, f)$
Mathematics
$\mathcal{OD}$
UML

Reflective Descriptions of Behaviour

Requirements

Recall:

- The **semantics** of the **UML model** $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD})$ 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** ϑ is a property of computations;
something which is either satisfied or not satisfied by a computation

$$\pi = (\sigma_0, \varepsilon_0) \xrightarrow[u_1]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \xrightarrow[u_2]{(cons_1, Snd_1)} \cdots \in [\![\mathcal{M}]\!],$$

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"?

Constructive vs. Reflective Descriptions

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.)

Interactions as Reflective Description

- In UML, reflective (temporal) descriptions are subsumed by **interactions**.

  A UML model $\mathcal{M} = (\mathscr{CD}, \mathscr{SM}, \mathscr{OD}, \mathscr{I})$ has a set of interactions $\mathscr{I}$.
- An interaction $\mathcal{I} \in \mathscr{I}$ can be (OMG claim: equivalently) **diagrammed** as
  - **communication diagram** (formerly known as collaboration diagram),
  - **timing diagram**, or
  - **sequence diagram**.

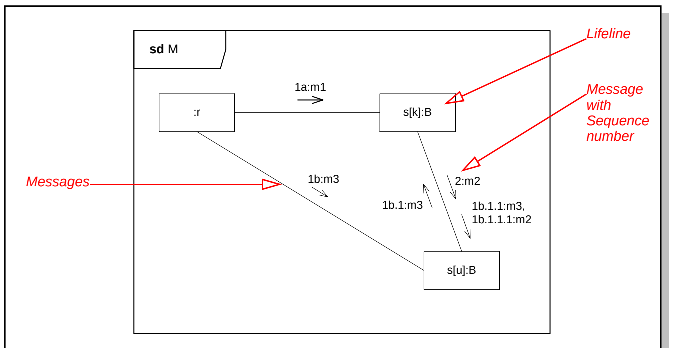


Figure 14.27 - Communication diagram (OMG, 2007, 515)

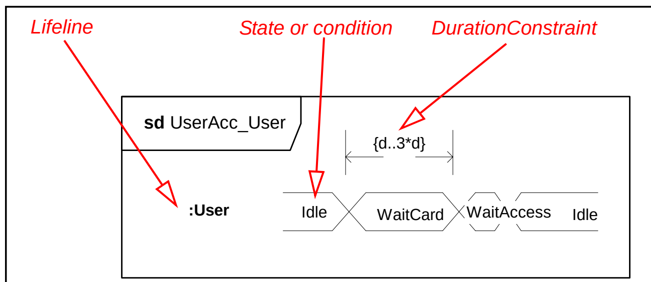


Figure 14.30 - Compact Lifeline with States (OMG, 2007, 522)

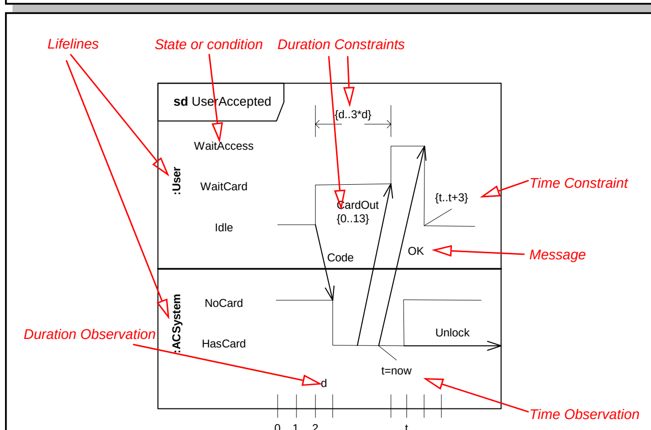


Figure 14.31 - Timing Diagram with more than one Lifeline and with Messages (OMG, 2007, 522)

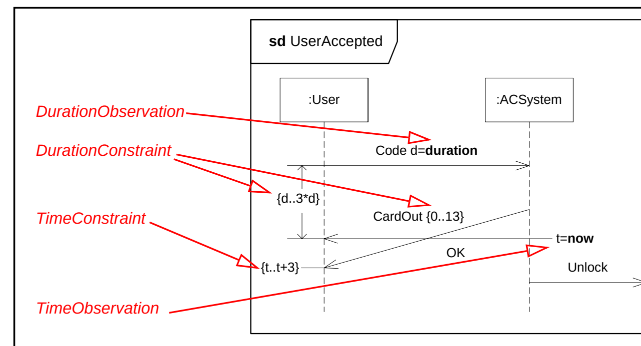


Figure 14.26 - Sequence Diagram with time and timing concepts (OMG, 2007, 513)

Interactions as Reflective Description

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

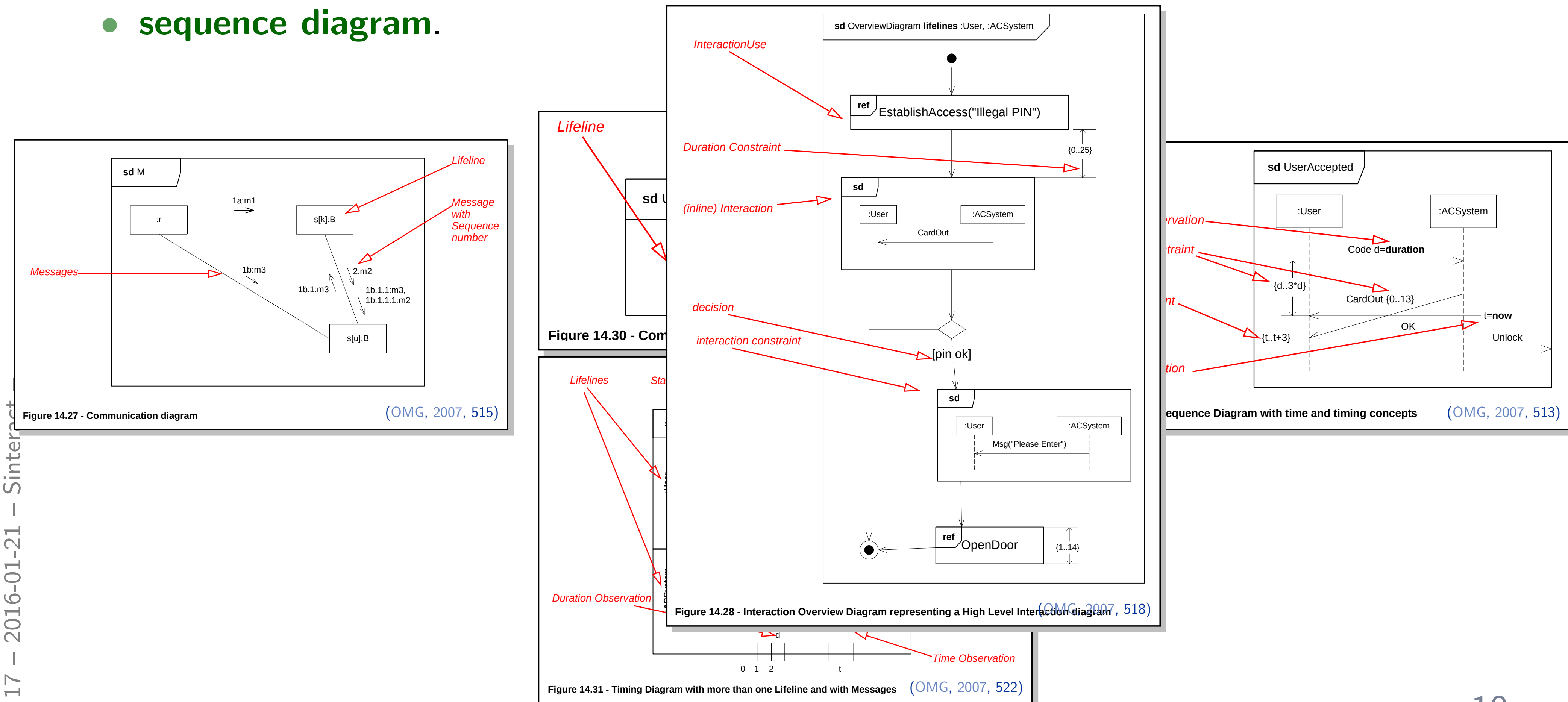


Figure 14.27 - Communication diagram (OMG, 2007, 515)

Figure 14.30 - Com…

Figure 14.28 - Interaction Overview Diagram representing a High Level Interaction diagram (OMG, 2007, 518)

…equence Diagram with time and timing concepts (OMG, 2007, 513)

Figure 14.31 - Timing Diagram with more than one Lifeline and with Messages (OMG, 2007, 522)

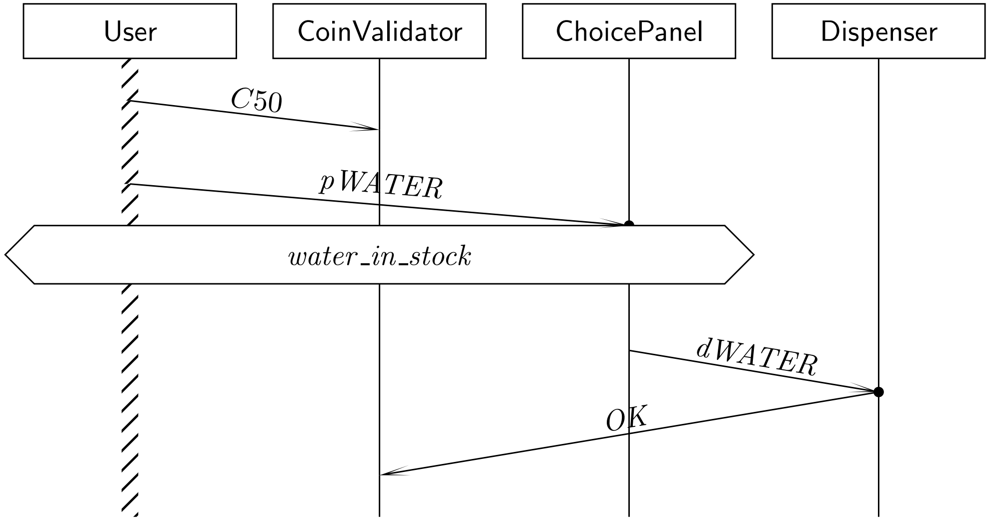
Why Sequence Diagrams?

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

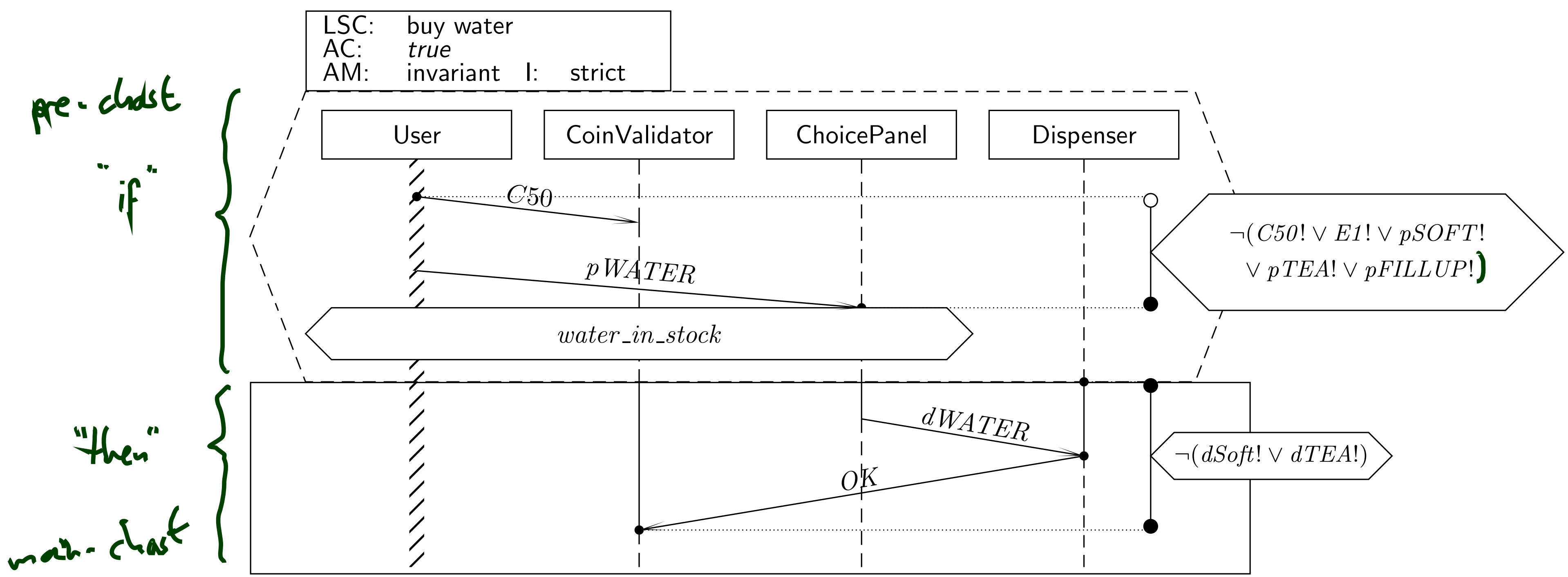
Most severe **drawbacks** of these formalisms:

- 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.



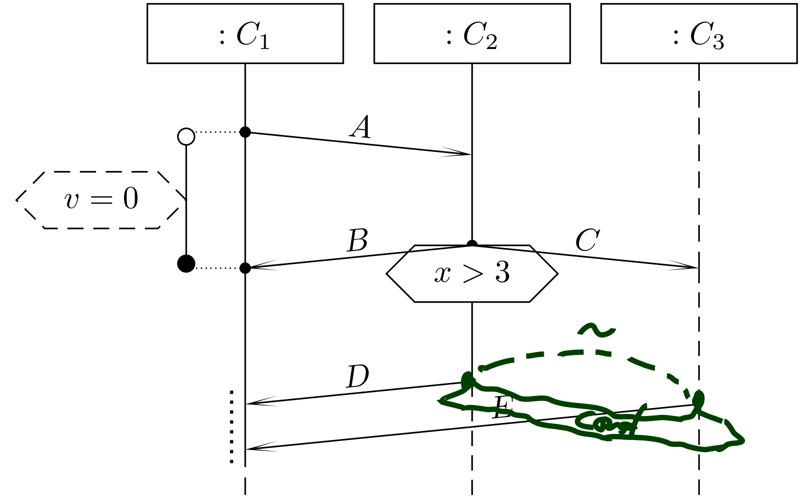
Live Sequence Charts — Syntax

LSC Body: Abstract Syntax

Let $\Theta = \{\text{hot}, \text{cold}\}$. An **LSC body** is a tuple

$$(I, (\mathscr{L}, \preceq), \sim, \mathscr{S}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv})$$

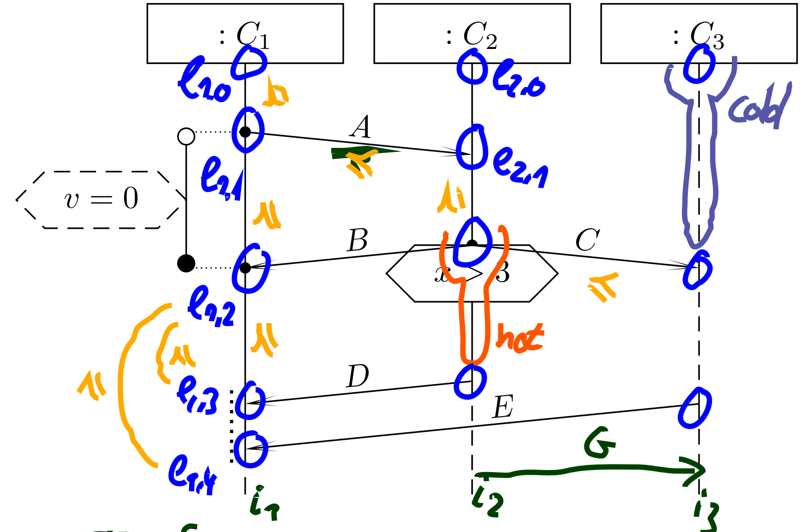
- I is a finite set of **instance lines**,
- $(\mathscr{L}, \preceq)$ is a finite, non-empty, **partially ordered** set of **locations**; each $l \in \mathscr{L}$ is associated with a temperature $\theta(l) \in \Theta$ and an instance line $i_l \in I$,
- $\sim \subseteq \mathscr{L} \times \mathscr{L}$ is an **equivalence relation** on locations, the **simultaneity** relation,



LSC Body: Abstract Syntax

Let $\Theta = \{\text{hot}, \text{cold}\}$. An **LSC body** is a tuple

$$(I, (\mathscr{L}, \preceq), \sim, \mathscr{S}, \mathsf{Msg}, \mathsf{Cond}, \mathsf{LocInv})$$



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

$I = \{i_1, i_2, i_3\}$

$\mathscr{L} = \{l_{1,0}, l_{1,1}, \dots, l_{2,0}, l_{2,1}, \dots\}$

$l_{1,0} \preceq l_{1,1} \preceq l_{1,2} \dots$

$l_{1,1} \preceq l_{2,1}, \dots$

$\mathsf{Msg} = \{(l_{1,1}, A, l_{2,1}), \dots\}$

$\mathsf{Cond} = \{(\{l_{2,2}\}, x > 3, hot), \dots\}$

$\mathsf{LocInv} = \{(l_{1,1}, \circ, v = 0, cold, l_{1,2}, \bullet), \dots\}$

Well-Formedness

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

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

$$\exists\,(l_1, b, l_2) \in \mathsf{Msg} : l \in \{l_1, l_2\}.$$

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

Damm, W. and Harel, D. (2001). LSCs: Breathing life into Message Sequence Charts. *Formal Methods in System Design*, 19(1):45–80.

Harel, D. (1997). Some thoughts on statecharts, 13 years later. In Grumberg, O., editor, *CAV*, volume 1254 of *LNCS*, pages 226–231. Springer-Verlag.

Harel, D. and Maoz, S. (2007). Assert and negate revisited: Modal semantics for UML sequence diagrams. *Software and System Modeling (SoSyM)*. To appear. (Early version in SCESM'06, 2006, pp. 13-20).

Harel, D. and Marelly, R. (2003). *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer-Verlag.

Klose, J. (2003). *LSCs: A Graphical Formalism for the Specification of Communication Behavior*. PhD thesis, Carl von Ossietzky Universität Oldenburg.

OMG (2007). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.