## Topic Area Requirements Engineering: Content

<table>
<thead>
<tr>
<th>VL 6</th>
<th>Introduction</th>
<th>Requirements Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Desired Properties</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kinds of Requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Analysis Techniques</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Documents</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dictionary, Specification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specification Languages</td>
</tr>
<tr>
<td>VL 7</td>
<td></td>
<td>Natural Language</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decision Tables</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Syntax, Semantics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Completeness, Consistency, ...</td>
</tr>
<tr>
<td>VL 8</td>
<td></td>
<td>Scenarios</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User Stories, Use Cases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Live Sequence Charts</td>
</tr>
<tr>
<td>VL 9</td>
<td></td>
<td>Syntax, Semantics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Working Definition: Software</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Discussion</td>
</tr>
</tbody>
</table>

- VL 6 - VL 9 are placeholders for specific sections or topics within the requirements engineering content.
Structure of Topic Areas

**Example**: Requirements Engineering

- **Vocabulary**
  - e.g. consistent, complete, tacit, etc.

- **Techniques**
  - informal
  - semi-formal
  - formal
  - e.g. “Whenever a crash…”
  - e.g. “Always, if \( \langle \text{crash} \rangle \) at \( t \)…”
  - e.g. “\( \forall t, t' \in \text{Time} \)…”

- **In the course:**
  - Use Cases
  - Pattern Language
  - Decision Tables
  - Live Sequence Charts
Content

- User Stories
- Use Cases
  - Use Case Diagrams
- Sequence Diagrams
  - A Brief History
  - Live Sequence Charts
    - Syntax:
      - Elements, Locations,
    - Towards Semantics:
      - Cuts
      - Firedsets
Scenarios
One quite effective approach:

try to **approximate** the requirements with positive and negative **scenarios**.

- Dear customer, please describe example usages of the desired system.
  Customer intuition: *"If the system is not at all able to do this, then it’s not what I want."*

- Dear customer, please describe behaviour that the desired system must not show.
  Customer intuition: *"If the system does this, then it’s not what I want."*

- From there on, refine and generalise: what about exceptional cases? what about corner-cases? etc.

- Prominent early advocate: **OOSE** *(Jacobson, 1992).*
Example: Vending Machine

- **Positive scenario**: Buy a Softdrink
  1. Insert one 1 euro coin.
  2. Press the ‘softdrink’ button.
  3. Get a softdrink.

- **Positive scenario**: Get Change
  1. Insert one 50 cent and one 1 euro coin.
  2. Press the ‘softdrink’ button.
  3. Get a softdrink.
  4. Get 50 cent change.

- **Negative scenario**: A Drink for Free
  1. Insert one 1 euro coin.
  2. Press the ‘softdrink’ button.
  3. Do not insert any more money.
  4. Get two softdrinks.
Notations for Scenarios

- The idea of scenarios (sometimes without negative or anti-scenarios) (re-)occurs in many process models or software development approaches.

- In the following, we will discuss two-and-a-half notations (in increasing formality):
  - **User Stories** (part of Extreme Programming)
  - **Use Cases** and Use Case Diagrams (OOSE)
  - **Sequence Diagrams** (here: Live Sequence Charts (Damm and Harel, 2001))
User Stories
User Stories (Beck, 1999)

“A User Story is a concise, written description of a piece of functionality that will be valuable to a user (or owner) of the software.”

Per user story, use one file card with the user story, e.g. following the pattern:

As a [role] I want [something] so that [benefit].

and in addition:
- unique identifier (e.g. unique number),
- priority (from 1 (highest) to 10 (lowest)) assigned by customer,
- effort, estimated by developers,
- back side of file card: (acceptance) test case(s),
  i.e., how to tell whether the user story has been realised.

Proposed card layout (front side):

<table>
<thead>
<tr>
<th>priority</th>
<th>unique identifier, name</th>
<th>estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>As a [role] I want [something] so that [benefit].</td>
<td></td>
</tr>
<tr>
<td>risk</td>
<td>real effort</td>
<td></td>
</tr>
</tbody>
</table>
Natural Language Patterns

Natural language requirements can be (tried to be) written as an instance of the pattern “⟨A⟩ ⟨B⟩ ⟨C⟩ ⟨D⟩ ⟨E⟩ ⟨F⟩.” (German grammar) where

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>clarifies when and under what conditions the activity takes place</td>
</tr>
<tr>
<td>B</td>
<td>is MUST (obligation), SHOULD (wish), or WILL (intention); also: MUST NOT (forbidden)</td>
</tr>
<tr>
<td>C</td>
<td>is either “the system” or the concrete name of a (sub-)system</td>
</tr>
</tbody>
</table>
| D | one of three possibilities:  
  - “does”, description of a system activity,  
  - “offers”, description of a function offered by the system to somebody,  
  - “is able if”, usage of a function offered by a third party, under certain conditions |
| E | extensions, in particular an object |
| F | the actual process word (what happens) |

Example:

After office hours (= A), the system (= C) should (= B) offer to the operator (= D) a backup (= F) of all new registrations to an external medium (= E).
✔ easy to create, small units
✔ close contact to customer
✔ objective / testable: by fixing test cases early

✘ may get difficult to keep overview over whole system to be developed → maybe best suited for changes / extensions (after first iteration).
✘ not designed to cover non-functional requirements and restrictions
✘ agile spirit: strong dependency on competent developers
✘ estimation of effort may be difficult

(Balzert, 2009)
Use Cases
**Use Case: Definition**

*use case* – A **sequence of interactions** between an actor (or actors) and a system triggered by a specific actor, which **produces a result** for an actor. *(Jacobson, 1992)*

More precisely:

- A use case has **participants**: the **system** and at least one **actor**.

  - **Actor**: an actor represents what interacts with the system.

    - An actor is a **role**, which a **user** or an **external system** may assume when interacting with the system under design.

    - Actors are not part of the system, thus they are **not described in detail**.

    - Actions of actors are **non-deterministic** (possibly constrained by domain model).

- A use case is triggered by a **stimulus** as input by the **main actor**.

- A use case is **goal oriented**, i.e. the main actor wants to reach a particular goal.

- A use case describes **all interactions** between the system and the participating actors that are needed to achieve the goal (or fail to achieve the goal for reasons).

- A use case **ends** when the desired goal is achieved, or when it is clear that the desired goal cannot be achieved.
## Use Case Example: ATM Authentication

<table>
<thead>
<tr>
<th>name</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>goal</td>
<td>the client wants access to the ATM</td>
</tr>
<tr>
<td>pre-condition</td>
<td>the ATM is operational, the welcome screen is displayed, card and PIN of client are available</td>
</tr>
<tr>
<td>post-condition</td>
<td>client accepted, services of ATM are offered</td>
</tr>
<tr>
<td>post-cond. in exceptional case</td>
<td>access denied, card returned or withheld, welcome screen displayed</td>
</tr>
<tr>
<td>actors</td>
<td>client (main actor), bank system</td>
</tr>
<tr>
<td>open questions</td>
<td>none</td>
</tr>
</tbody>
</table>

### Normal Case

1. client inserts card
2. ATM reads card, sends data to bank system
3. bank system checks validity
4. ATM shows PIN screen
5. client enters PIN
6. ATM reads PIN, sends to bank system
7. bank system checks PIN
8. ATM accepts and shows main menu

### Exception Case 2a

<table>
<thead>
<tr>
<th>exc. case 2a</th>
<th>card not readable</th>
</tr>
</thead>
<tbody>
<tr>
<td>2a.1 ATM displays “card not readable”</td>
<td></td>
</tr>
<tr>
<td>2a.2 ATM returns card</td>
<td></td>
</tr>
<tr>
<td>2a.3 ATM shows welcome screen</td>
<td></td>
</tr>
</tbody>
</table>

### Exception Cases

<table>
<thead>
<tr>
<th>exc. case 2b</th>
<th>card readable, but not ATM card</th>
</tr>
</thead>
<tbody>
<tr>
<td>exc. case 2c</td>
<td>no connection to bank system</td>
</tr>
<tr>
<td>exc. case 3a</td>
<td>card not valid or disabled</td>
</tr>
<tr>
<td>exc. case 5a</td>
<td>client cancels</td>
</tr>
<tr>
<td>exc. case 5b</td>
<td>client doesn’t react within 5 s</td>
</tr>
<tr>
<td>exc. case 6a</td>
<td>no connection to bank system</td>
</tr>
<tr>
<td>exc. case 7a</td>
<td>first or second PIN wrong</td>
</tr>
<tr>
<td>exc. case 7b</td>
<td>third PIN wrong</td>
</tr>
</tbody>
</table>

(Ludewig and Lichter, 2013)
Use Case Diagrams
Use Case Diagrams: Basic Building Blocks

- Actor: \[\langle \text{actor name} \rangle\]
- Use Case: \[\langle \text{use case name} \rangle\]

or:

- Use Case: \[\langle \text{use case name} \rangle\]
## Use Case Example: ATM Authentication

<table>
<thead>
<tr>
<th>name</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>goal</td>
<td>the client wants access to the ATM</td>
</tr>
<tr>
<td>pre-condition</td>
<td>the ATM is operational, the welcome screen is displayed, card and PIN of client are available</td>
</tr>
<tr>
<td>post-condition</td>
<td>client accepted, services of ATM are offered</td>
</tr>
<tr>
<td>post-cond. in exceptional case</td>
<td>access denied, card returned or withheld, welcome screen displayed</td>
</tr>
<tr>
<td>actors</td>
<td>client (main actor), bank system</td>
</tr>
<tr>
<td>open questions</td>
<td>none</td>
</tr>
</tbody>
</table>

### Normal Case
1. client inserts card
2. ATM reads card, sends data to bank system
3. bank system checks validity
4. ATM shows PIN screen
5. client enters PIN
6. ATM reads PIN, sends to bank system
7. bank system checks PIN
8. ATM accepts and shows main menu

### Exception Case 2a
- card not readable
  1. ATM displays “card not readable”
  2. ATM returns card
  3. ATM shows welcome screen

### Exception Cases
- **exc. case 2b**: card readable, but not ATM card
- **exc. case 2c**: no connection to bank system
- **exc. case 3a**: card not valid or disabled
- **exc. case 5a**: client cancels
- **exc. case 5b**: client doesn’t react within 5 s
- **exc. case 6a**: no connection to bank system
- **exc. case 7a**: first or second PIN wrong
- **exc. case 7b**: third PIN wrong

---

*Image source: [commons.wikimedia.org](http://commons.wikimedia.org) (CC-by-sa 4.0, Dirk Ingo Franke)*

*Ludewig and Lichter, 2013*
**Example: Use Case Diagram of the ATM Use Case**

### Use Case Example: ATM Authentication

<table>
<thead>
<tr>
<th>name</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>goal</td>
<td>the client wants access to the ATM</td>
</tr>
<tr>
<td>pre-condition</td>
<td>the ATM is operational, the welcome screen is displayed, card and PIN of client are available</td>
</tr>
<tr>
<td>post-condition</td>
<td>client accepted, services of ATM are offered</td>
</tr>
<tr>
<td>post-cond. in exceptional case</td>
<td>access denied, card returned or withheld, welcome screen displayed</td>
</tr>
<tr>
<td>actors</td>
<td>client (main actor), bank system</td>
</tr>
<tr>
<td>open questions</td>
<td>none</td>
</tr>
</tbody>
</table>

#### Normal Case
1. client inserts card
2. ATM reads card, sends data to bank system
3. bank system checks validity
4. ATM shows PIN screen
5. client enters PIN
6. ATM reads PIN, sends to bank system
7. bank system checks PIN
8. ATM accepts and shows main menu

**Exc. case 2a**
- card not readable
  2a.1 ATM displays "card not readable"
  2a.2 ATM returns card
  2a.3 ATM shows welcome screen

**Exc. case 2b**
- card readable, but not ATM card

**Exc. case 2c**
- no connection to bank system

**Exc. case 3a**
- card not valid or disabled

**Exc. case 5a**
- client cancels

**Exc. case 5b**
- client doesn't react within 5 s

**Exc. case 6a**
- no connection to bank system

**Exc. case 7a**
- first or second PIN wrong

**Exc. case 7b**
- third PIN wrong

(Ludewig and Lichter, 2013)
Use Case Diagrams: More Building Blocks

More notation:

- Extend: use case A \(\langle\text{extends}\rangle\) use case B
- Use or Include: use case A \(\langle\text{uses}\rangle\) or \(\langle\text{include}\rangle\) use case B
Use Case Diagram: Bigger Examples

ATM

info services

query balance
[print statement]

print statement
[not auth.]

transactions

get cash

define standing order

basic services

authentication

 ⟨ ⟨ extend ⟩ ⟩

 ⟨ ⟨ include ⟩ ⟩

 ⟨ ⟨ include ⟩ ⟩

 ⟨ ⟨ include ⟩ ⟩

(Ludewig and Lichter, 2013)
Use Case Example: ATM Authentication

<table>
<thead>
<tr>
<th>name</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td>goal</td>
<td>the client wants access to the ATM</td>
</tr>
<tr>
<td>pre-condition</td>
<td>the ATM is operational, the welcome screen is displayed, card and PIN of client are available</td>
</tr>
<tr>
<td>post-condition</td>
<td>client accepted, services of ATM are offered</td>
</tr>
<tr>
<td>post-cond. in exceptional case</td>
<td>access denied, card returned or withheld, welcome screen displayed</td>
</tr>
<tr>
<td>actors</td>
<td>client (main actor), bank system</td>
</tr>
<tr>
<td>open questions</td>
<td>none</td>
</tr>
<tr>
<td>normal case</td>
<td>1. client inserts card 2. ATM read card, sends data to bank system 3. bank system checks validity 4. ATM shows PIN screen 5. client enters PIN 6. ATM reads PIN, sends to bank system 7. bank system checks PIN 8. ATM accepts and shows main menu</td>
</tr>
<tr>
<td>exception case 2a</td>
<td>card not readable 2a.1 ATM displays “card not readable” 2a.2 ATM returns card 2a.3 ATM shows welcome screen</td>
</tr>
<tr>
<td>exception case 2b</td>
<td>card readable, but not ATM card</td>
</tr>
<tr>
<td>exception case 2c</td>
<td>no connection to bank system</td>
</tr>
<tr>
<td>exception case 3a</td>
<td>card not valid or disabled</td>
</tr>
<tr>
<td>exception case 5a</td>
<td>client cancels</td>
</tr>
<tr>
<td>exception case 5b</td>
<td>client doesn’t react within 5 s</td>
</tr>
<tr>
<td>exception case 6a</td>
<td>no connection to bank system</td>
</tr>
<tr>
<td>exception case 7a</td>
<td>first or second PIN wrong</td>
</tr>
<tr>
<td>exception case 7b</td>
<td>third PIN wrong</td>
</tr>
</tbody>
</table>

(1.) Observables:
- event **insert_card**
- condition **card_rdbl**
- event **send_data**
- event **data_valid**
- event **pin_screen**

(2.) Finite Automaton:

```
q1  insert_card \∧\neg card_rdbl
q2  send_data
q3  data_valid
q4  pin_screen
```

(Ludewig and Lichter, 2013)
Content

- User Stories
- Use Cases
  - Use Case Diagrams
- Sequence Diagrams
  - A Brief History
  - Live Sequence Charts
    - Syntax:
      - Elements, Locations,
    - Towards Semantics:
      - Cuts
      - Firedsets
Sequence Diagrams
A Brief History of Sequence Diagrams

- **Message Sequence Charts**, ITU standardized in different versions (ITU Z.120, 1st edition: 1993); often accused of lacking a formal semantics.

- **Sequence Diagrams** of UML 1.x (one of three main authors: I. Jacobson)

- **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). LSCs have a common fragment with UML 2.x SDs: (Harel and Maoz, 2007).
Live Sequence Charts: Syntax (Body)
LSC Body Building Blocks
instance line head

I_1

I_2

I_3

(hot) line segment

(cold) line segment

instance line/
life line
LSC Body Building Blocks

instance line head

simultaneous region

(cold) line segment

(hot) instantaneous message

(hot) asynchronous message

(hot) line segment

instance line/
life line
LSC Body Building Blocks

instance line head

simultaneous region

exclusive

(cold) local invariant

(inclusive)

(hot) line segment

instance line/
life line

(cold) line segment

(hot) instantaneous message

(hot) condition

(hot) asynchronous message

instance line head

I_1

A

I_2

B

c_1

I_3

C

c_4

D

E
The Plan: A Formal Semantics for a Visual Formalism

concrete syntax
(diagram)

abstract syntax

\[((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)\]

semantics
(Büchi automaton)
Definition. [LSC Body]

Let $E$ be a set of events and $C$ a set of atomic propositions, $E \cap C = \emptyset$.

An LSC body over $E$ and $C$ is a tuple

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

where

- $\mathcal{L}$ is a finite, non-empty set of locations with
  - a partial order $\preceq \subseteq \mathcal{L} \times \mathcal{L}$,
  - a symmetric simultaneity relation $\sim \subseteq \mathcal{L} \times \mathcal{L}$ disjoint with $\preceq$, i.e. $\preceq \cap \sim = \emptyset$,
- $\mathcal{I} = \{I_1, \ldots, I_n\}$ is a partitioning of $\mathcal{L}$; elements of $\mathcal{I}$ are called instance line,
- $\text{Msg} \subseteq \mathcal{L} \times E \times \mathcal{L}$ is a set of messages with $(l, E, l') \in \text{Msg}$ only if $(l, l') \in \prec \cup \sim$; message $(l, E, l')$ is called instantaneous iff $l \sim l'$ and asynchronous otherwise,
- $\text{Cond} \subseteq (2^\mathcal{L} \setminus \emptyset) \times \Phi(C)$ is a set of conditions with $(L, \phi) \in \text{Cond}$ only if $l \sim l'$ for all $l \neq l' \in L$,
- $\text{LocInv} \subseteq \mathcal{L} \times \{\circ, \bullet\} \times \Phi(C) \times \mathcal{L} \times \{\circ, \bullet\}$ is a set of local invariants with $(l, \iota, \phi, l', \iota') \in \text{LocInv}$ only if $l \prec l'$, $\circ$: exclusive, $\bullet$: inclusive,
- $\Theta : \mathcal{L} \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot, cold}\}$ assigns to each location and each element a temperature.
• locations $\mathcal{L}$,
• $\preceq \subseteq \mathcal{L} \times \mathcal{L}$, $\sim \subseteq \mathcal{L} \times \mathcal{L}$
• $\mathcal{I} = \{I_1, \ldots, I_n\}$,
• $\text{Msg} \subseteq \mathcal{L} \times \mathcal{E} \times \mathcal{L}$,
• $\text{Cond} \subseteq (2^{\mathcal{L}} \setminus \emptyset) \times \Phi(\mathcal{C})$
• $\text{LocInv} \subseteq \mathcal{L} \times \{\bullet, \circ\} \times \Phi(\mathcal{C}) \times \mathcal{L} \times \{\circ, \bullet\}$,
• $\Theta : \mathcal{L} \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot, cold}\}$. 
• locations $L$,
• $\preceq \subseteq L \times L$, $\sim \subseteq L \times L$
• $I = \{I_1, \ldots, I_n\}$,
• $\text{Msg} \subseteq L \times E \times L$,
• $\text{Cond} \subseteq (2^L \setminus \emptyset) \times \Phi(C)$
• $\text{LocInv} \subseteq L \times \{\circ, \bullet\} \times \Phi(C) \times L \times \{\circ, \bullet\}$,
• $\Theta : L \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot}, \text{cold}\}$.

$L = \{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}, l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}, l_{3,0}, l_{3,1}, l_{3,2}, l_{3,3}\}$
Definition. [LSC Body]
Let $\mathcal{E}$ be a set of events and $\mathcal{C}$ a set of atomic propositions, $\mathcal{E} \cap \mathcal{C} = \emptyset$.

An **LSC body** over $\mathcal{E}$ and $\mathcal{C}$ is a tuple

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

where

- $\mathcal{L}$ is a finite, non-empty set of locations with
  - a **partial order** $\preceq \subseteq \mathcal{L} \times \mathcal{L}$,
  - a symmetric **simultaneity relation** $\sim \subseteq \mathcal{L} \times \mathcal{L}$ disjoint with $\preceq$, i.e. $\preceq \cap \sim = \emptyset$,
- $\mathcal{I} = \{I_1, \ldots, I_n\}$ is a partitioning of $\mathcal{L}$; elements of $\mathcal{I}$ are called **instance line**,
- $\text{Msg} \subseteq \mathcal{L} \times \mathcal{E} \times \mathcal{L}$ is a set of **messages** with $(l, E, l') \in \text{Msg}$ only if $(l, l') \in \prec \cup \sim$; message $(l, E, l')$ is called **instantaneous** iff $l \sim l'$ and **asynchronous** otherwise,
- $\text{Cond} \subseteq (2^\mathcal{L} \setminus \emptyset) \times \Phi(\mathcal{C})$ is a set of **conditions** with $(L, \phi) \in \text{Cond}$ only if $l \sim l'$ for all $l \neq l' \in L$,
- $\text{LocInv} \subseteq \mathcal{L} \times \{\circ, \bullet\} \times \Phi(\mathcal{C}) \times \mathcal{L} \times \{\circ, \bullet\}$ is a set of **local invariants** with $(l, \iota, \phi, l', \iota') \in \text{LocInv}$ only if $l \prec l'$, $\circ$: exclusive, $\bullet$: inclusive,
- $\Theta : \mathcal{L} \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot, cold}\}$ assigns to each location and each element a **temperature**.
locations \( \mathcal{L} \),
- \( \leq \subseteq \mathcal{L} \times \mathcal{L} \), \( \sim \subseteq \mathcal{L} \times \mathcal{L} \)
- \( \mathcal{I} = \{I_1, \ldots, I_n\} \),
- \( \text{Msg} \subseteq \mathcal{L} \times \mathcal{E} \times \mathcal{L} \)
- \( \text{Cond} \subseteq (2^\mathcal{L} \setminus \emptyset) \times \Phi(\mathcal{C}) \)
- \( \text{LocInv} \subseteq \mathcal{L} \times \{\circ, \bullet\} \times \Phi(\mathcal{C}) \times \mathcal{L} \times \{\circ, \bullet\} \),
- \( \Theta : \mathcal{L} \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot, cold}\} \)

\[ \mathcal{L} = \{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}, l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}, l_{3,0}, l_{3,1}, l_{3,2}, l_{3,3}\} \]

\[ l_{1,0} < l_{1,1} < l_{1,2} < l_{1,3}, \quad l_{1,2} < l_{1,4}, \quad l_{2,0} < l_{2,1} < l_{2,2} < l_{2,3}, \quad l_{3,0} < l_{3,1} < l_{3,2} < l_{3,3}, \quad l_{1,1} < l_{2,1}, \quad l_{2,2} < l_{1,2}, \quad l_{2,3} < l_{1,3}, \quad l_{3,2} < l_{1,4}, \quad l_{2,1} \sim l_{3,1}, \quad l_{2,2} \sim l_{3,2} \]

\[ \mathcal{I} = \{\{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}\}, \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}\}, \{l_{3,0}, l_{3,1}, l_{3,2}\}\} \]
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. \( \exists (L, \phi) \in \text{Cond} : l \in L \), or
  - a **local invariant**, i.e. \( \exists (l_1, \nu_1, \phi, l_2, \nu_2) \in \text{LocInv} : l \in \{ l_1, l_2 \} \),

then there is a location \( l' \) **simultaneous** 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.

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

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

- $\mathcal{L} = \{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,2}, l_{1,4}, l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}, l_{3,0}, l_{3,1}, l_{3,2}, l_{3,3}\}$
- $l_{1,0} \prec l_{1,1} \prec l_{1,2} \prec l_{1,3}, l_{1,2} \prec l_{1,4}, l_{2,0} \prec l_{2,1} \prec l_{2,2} \prec l_{2,3}, l_{3,0} \prec l_{3,1} \prec l_{3,2} \prec l_{3,3}$
- $l_{1,1} \prec l_{2,1}, l_{2,2} \prec l_{1,2}, l_{2,3} \prec l_{1,3}, l_{3,2} \prec l_{1,4}, l_{2,1} \sim l_{3,1}, l_{2,2} \sim l_{3,2}$
- $\mathcal{I} = \{\{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}\}, \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}\}, \{l_{3,0}, l_{3,1}, l_{3,2}\}\}$
- $\text{Msg} = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,2}), (l_{2,3}, D, l_{1,3}), (l_{3,3}, E, l_{1,4})\}$
- $\text{Cond} = \{\{l_{2,1}, l_{3,1}\}, c_4\}, \{\{l_{2,2}\}, c_2 \land c_3\}\}$
- $\text{LocInv} = \{(l_{1,1}, \ast, c_1, l_{1,2}, \bullet)\}$
Content

- **User Stories**
- **Use Cases**
  - Use Case Diagrams
- **Sequence Diagrams**
  - A Brief History
  - **Live Sequence Charts**
    - Syntax:
      - Elements, Locations,
    - Towards Semantics:
      - Cuts
      - Firedsets
LSC Semantics: Towards Automaton Construction
The Plan: A Formal Semantics for a Visual Formalism

concrete syntax
(diagram)

abstract syntax

\(((L, \leq, \sim), I, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)\)

semantics
(Büchi automaton)
**Definition.** Let $$((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)$$ be an LSC body.

A non-empty set $$\emptyset \neq C \subseteq \mathcal{L}$$ is called a cut of the LSC body iff $$C$$

- is **downward closed**, i.e.
  $$\forall l, l' \in \mathcal{L} \cdot l' \in C \land l \preceq l' \implies l \in C,$$

- is **closed** under simultaneity, i.e.
  $$\forall l, l' \in \mathcal{L} \cdot l' \in C \land l \sim l' \implies l \in C,$$
  and

- comprises at least **one location per instance line**, i.e.
  $$\forall I \in \mathcal{I} \cdot C \cap I \neq \emptyset.$$

The temperature function is extended to cuts as follows:

$$\Theta(C) = \begin{cases} 
\text{hot} & , \text{if } \exists l \in C \cdot (\nexists l' \in C \cdot l \prec l') \land \Theta(l) = \text{hot} \\
\text{cold} & , \text{otherwise}
\end{cases}$$

that is, $$C$$ is **hot** if and only if at least one of its maximal elements is hot.
∅ ≠ C ⊆ L – downward closed – simultaneity closed – at least one loc. per instance line
$\emptyset \neq C \subseteq L$ – downward closed – simultaneity closed – at least one loc. per instance line
\[ \emptyset \neq C \subseteq L \] – downward closed – simultaneity closed – at least one loc. per instance line
Cut Examples

ellite \[ C \subseteq \mathcal{L} \] – downward closed – simultaneity closed – at least one loc. per instance line
∅ ≠ C ⊆ L – downward closed – simultaneity closed – at least one loc. per instance line
\[ \emptyset \neq C \subseteq \mathcal{L} \] – downward closed – simultaneity closed – at least one loc. per instance line
$\emptyset \neq C \subseteq \mathcal{L}$ – downward closed – simultaneity closed – at least one loc. per instance line
∅ ≠ C ⊆ L – downward closed – simultaneity closed – at least one loc. per instance line
A Successor Relation on Cuts

The partial order “≤” and the simultaneity relation “∼” of locations induce a **direct successor relation** on cuts of an LSC body as follows:

**Definition.**

Let \( C \subseteq \mathcal{L} \) be a cut of LSC body \(((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta)\).

A set \( \emptyset \neq \mathcal{F} \subseteq \mathcal{L} \) of locations is called **fired-set** \( \mathcal{F} \) of cut \( C \) if and only if

- \( C \cap \mathcal{F} = \emptyset \) and \( C \cup \mathcal{F} \) is a cut, i.e. \( \mathcal{F} \) is closed under simultaneity,
- all locations in \( \mathcal{F} \) are direct \( \prec \)-successors of the front of \( C \), i.e.
  \[
  \forall l \in \mathcal{F} \exists l' \in C \bullet l' \prec l \land (\not\exists l'' \in C \bullet l' \prec l'' \prec l),
  \]
- locations in \( \mathcal{F} \), that lie on the same instance line, are **pairwise unordered**, i.e.
  \[
  \forall l \neq l' \in \mathcal{F} \bullet (\exists I \in \mathcal{I} \bullet \{l, l'\} \subseteq I) \implies l \not\sim l' \land l' \not\sim l,
  \]
- for each asynchronous message reception in \( \mathcal{F} \), the corresponding **sending is already in** \( C \),
  \[
  \forall (l, E, l') \in \text{Msg} \bullet l' \in \mathcal{F} \implies l \in C.
  \]

The cut \( C' \equiv C \cup \mathcal{F} \) is called **direct successor of** \( C \) **via** \( \mathcal{F} \), denoted by \( C \sim \mathcal{F} \rightarrow C' \).
\( C \cap \mathcal{F} = \emptyset \) — \( C \cup \mathcal{F} \) is a cut — only direct \( \prec \)-successors — same instance line on front pairwise unordered —

\( \phi \) — sending of asynchronous reception already in
$C \cap F = \emptyset$, so $C \cup F$ is a cut. Only direct \textless\textless-successors are considered, same as in previous instance line on the front. Pairwise unordered sending of asynchronous reception already in the model.
The TBA $B(L)$ of LSC $L$ over $C$ and $E$ is $(C_B, Q, q_{ini}, \rightarrow, Q_F)$ with

- $C_B = C \cup E!\cup$, where $E!\cup = \{E!, E? \mid E \in E\}$,

- $Q$ is the set of cuts of $L$, $q_{ini}$ is the instance heads cut,

- $\rightarrow$ consists of loops, progress transitions (from $\nsucceq_F$), and legal exits (cold cond./local inv.),

- $Q_F = \{C \in Q \mid \Theta(C) = \text{cold} \lor C = L\}$ is the set of cold cuts and the maximal cut.
**Recall:** The TBA $B(L)$ of LSC $L$ is $(C, Q, q_{ini}, \rightarrow, Q_F)$ with

- $Q$ is the set of cuts of $L$, $q_{ini}$ is the instance heads cut,
- $C_B = C \cup E!?$,
- $\rightarrow$ consists of loops, progress transitions (from $\sim \mathcal{F}$), and legal exits (cold cond./local inv.),
- $\mathcal{F} = \{C \in Q \mid \Theta(C) = \text{cold} \lor C = L\}$ is the set of cold cuts.

So in the following, we “only” need to construct the transitions’ labels:

$$\rightarrow = \{(q, \psi_{\text{loop}}(q), q) \mid q \in Q\} \cup \{(q, \psi_{\text{prog}}(q, q'), q') \mid q \sim \mathcal{F} q'\} \cup \{(q, \psi_{\text{exit}}(q), L) \mid q \in Q\}$$

- $\psi_{\text{loop}}(q)$: “what allows us to stay at cut $q$”
- $\psi_{\text{prog}}(q, q')$: “characterisation of firedset $\mathcal{F}_n$”
- $\psi_{\text{exit}}(q)$: “what allows us to legally exit”
Tell Them What You’ve Told Them…

- **User Stories**: simple example of scenarios
  - **strong point**: naming tests is necessary,
  - **weak point**: hard to keep overview; global restrictions.

- **Use-Cases**:
  - interactions between system and actors,
  - be sure to elaborate exceptions and corner cases,
  - in particular effective with customers lacking technical background.

- **Use-Case Diagrams**:
  - visualise which participants are relevant for which use-case,
  - are rather **useless** without the underlying use-case.

- **Sequence Diagrams**:
  - a **visual formalism** for interactions, i.e.,
    - precisely defined syntax,
    - precisely defined semantics (→ next lecture).

- Can be used to precisely describe the interactions of a **use-case**.
References
References


