Softwaretechnik / Software-Engineering

Lecture 9: Scenarios & Use Cases

2018-06-04

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
**Risks Implied by Bad Requirements Specifications**

**design and implementation,**

- without specification, programmers may just “ask around” when in doubt, possibly yielding different interpretations 
  → difficult integration

**negotiation** (with customer, marketing department, or …)

**documentation,** e.g., the user’s manual.

- without specification, the user’s manual author can only describe what the system does, not what it should do (“every observation is a feature”)

- later re-implementations.
  - the new software may need to adhere to requirements of the old software; if not properly specified, the new software needs to be a 1:1 re-implementation of the old → additional effort

**preparation of tests,**

- without a description of allowed outcomes, tests are randomly searching for generic errors (like crashes) 
  → systematic testing hardly possible

**acceptance** by customer, resolving later objections or regress claims,

- without specification, it is unclear at delivery time whether behaviour is an error (developer needs to fix) or correct (customer needs to accept and pay) → nasty disputes, additional effort

**re-use,**

- without specification, re-use needs to be based on re-reading the code → risk of unexpected changes
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 crash at t…”
- e.g. “∀ t, t' ∈ Time •…”
Introduction

Requirements Specification
  - Desired Properties
  - Kinds of Requirements
  - Analysis Techniques

Documents
  - Dictionary, Specification

Specification Languages
  - Natural Language
  - Decision Tables
    - Syntax, Semantics
    - Completeness, Consistency, ...
  - Scenarios
    - User Stories, Use Cases
    - Live Sequence Charts
      - Syntax, Semantics

Definition: Software & SW Specification

Wrap-Up
Content

- Scenarios: The Idea
- Use Cases
  - Use Case Diagrams
- User Stories
- Sequence Diagrams
  - A Brief History
  - Live Sequence Charts
    - LSC Body Syntax:
      - LSC Model Elements, Locations
      - Well-Formedness
    - Towards Semantics:
      - Cuts, Firedsets
      - Automaton Construction
    - Excursion: Symbolic Büchi Automata

Informatik III
(Automata Theory)
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.

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:

- **Use Cases** and Use Case Diagrams (**OOSE**)
- **User Stories** (part of **Extreme Programming**)
- **Sequence Diagrams** (here: **Live Sequence Charts** (Damm and Harel, 2001))
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)
**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**
1. ATM displays “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)
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.

<table>
<thead>
<tr>
<th>name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>goal</td>
<td>...</td>
</tr>
<tr>
<td>pre-condition</td>
<td>...</td>
</tr>
<tr>
<td>post-condition</td>
<td>...</td>
</tr>
<tr>
<td>post-cond. in exceptional case</td>
<td>...</td>
</tr>
<tr>
<td>actors</td>
<td>...</td>
</tr>
<tr>
<td>open questions</td>
<td>...</td>
</tr>
<tr>
<td>normal case</td>
<td>1. ...</td>
</tr>
</tbody>
</table>
Use Case Diagrams
Use Case Diagrams: Basic Building Blocks

 ⟨actor name⟩  ⟩  ⟨use case name⟩

 or:  ⟨use case name⟩
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
  2a.1 ATM displays "card not readable"
  2a.2 ATM returns card
  2a.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

(From 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 to access 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</td>
<td>access denied, card returned or withheld, welcome screen displayed</td>
</tr>
<tr>
<td>exceptional case</td>
<td>none</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**
1. ATM displays “card not readable”
2. card not readable
2a.1 ATM returns card
2a.2 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 seconds
- 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:

- Use case A
  - ⟨⟨extends⟩⟩
  - use case B

- Use case A
  - ⟨⟨uses⟩⟩ or ⟨⟨include⟩⟩
  - use case B
Use Case Diagram: Bigger Examples

(Ludewig and Lichter, 2013)
Content

• Scenarios: The Idea
• Use Cases
  • Use Case Diagrams
• User Stories
• Sequence Diagrams
  • A Brief History
  • Live Sequence Charts
    • LSC Body Syntax:
      • LSC Model Elements, Locations
      • Well-Formedness
    • Towards Semantics:
      • Cuts, Firedsets
      • Automaton Construction
  • Excursion: Symbolic Büchi Automata

Informatik III (Automata Theory)
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** – **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></td>
<td></td>
</tr>
<tr>
<td><strong>As a [role] I want [something] so that [benefit]</strong>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>risk</td>
<td>real effort</td>
<td></td>
</tr>
</tbody>
</table>

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.
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** – **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></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>As a [role] I want [something] so that [benefit].</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>
<tr>
<td>D</td>
<td>one of three possibilities:</td>
</tr>
<tr>
<td></td>
<td>• “does”, description of a system activity,</td>
</tr>
<tr>
<td></td>
<td>• “offers”, description of a function offered by the system to somebody,</td>
</tr>
</tbody>
</table>
User Stories: Discussion

✔ 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 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 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

#### 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

(From Ludwig and Lichter, 2013)
Content

- Scenarios: The Idea
- Use Cases
  - Use Case Diagrams
- User Stories
- Sequence Diagrams
  - A Brief History
  - Live Sequence Charts
    - LSC Body Syntax:
      - LSC Model Elements, Locations
      - Well-Formedness
    - Towards Semantics:
      - Cuts, Firedsets
      - Automaton Construction
    - Excursion: Symbolic Büchi Automata

Informatik III
(Automata Theory)
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
(2018 Edition)
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))

read out relevant information

apply construction procedure

semantics
(Büchi automaton)

software

does the software satisfy the LSC?

does the software read out relevant information

apply construction procedure

semantics
(Büchi automaton)

software

does the software satisfy the LSC?

read out relevant information

apply construction procedure

semantics
(Büchi automaton)

software

does the software satisfy the LSC?
LSC Body Syntax
LSC Body Building Blocks

- instance line head
- simultaneous region
- exclusive
- (cold) local invariant
- inclusive
- (hot) line segment
- coregion
- (cold) asynchronous message
- instance line/life line
- (cold) line segment
- (cold) condition
- (hot) instantaneous message
- (hot) condition
- (hot) condition
- not a location
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 $l \neq l' \in L$,
- $\text{LocInv} \subseteq \mathcal{L} \times \{\circlearrowleft, \bullet\} \times \Phi(\mathcal{C}) \times \mathcal{L} \times \{\circlearrowleft, \bullet\}$ is a set of local invariants
  with $(l, \iota, \phi, l', \iota') \in \text{LocInv}$ only if $l \prec l'$, $\circlearrowleft$: 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 \( \mathcal{L} \),
• \( \preceq \subseteq \mathcal{L} \times \mathcal{L} \), \( \sim \subseteq \mathcal{L} \times \mathcal{L} \)
• \( \mathcal{I} = \{I_1, \ldots, I_n\} \),
• \( \text{Msg} \subseteq \mathcal{L} \times \mathcal{E} \times \mathcal{L} \),
• \( \text{Cond} \subseteq (2^{\mathcal{L}} \setminus \emptyset) \times \Phi(\mathcal{C}) \)
• \( \text{LocInv} \subseteq \mathcal{L} \times \{\circ, \bullet\} \times \Phi(\mathcal{C}) \times \mathcal{L} \times \{\circ, \bullet\} \),
• \( \Theta : \mathcal{L} \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot}, \text{cold}\} \).

\[ \Theta(l_{1,0}) = \text{hot} \]
\[ \Theta(l_{3,0}) = \text{cold} \]

\( \mathcal{L} = \{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,2}, l_{1,4}, l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}, l_{3,0}, l_{3,1}, l_{3,2}, l_{3,3} \} \)
**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 $(l, l') \in \preceq \cup \sim$; 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 $(\mathcal{L}, \phi) \in \text{Cond}$ only if $l \sim l'$ for all $l \neq l' \in \mathcal{L}$,
- $\text{LocInv} \subseteq \mathcal{L} \times \{\circ, \bullet\} \times \Phi(\mathcal{C}) \times \mathcal{L} \times \{\circ, \bullet\}$ 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**.
From Concrete to Abstract Syntax

- Locations $\mathcal{L}$,
- $\preceq \subseteq \mathcal{L} \times \mathcal{L}$, $\sim \subseteq \mathcal{L} \times \mathcal{L}$
- $\mathcal{I} = \{I_1, \ldots, I_n\}$
- $\text{Msg} \subseteq \mathcal{L} \times \mathcal{E} \times \mathcal{L}$
- $\text{Cond} \subseteq (2^\mathcal{L} \setminus \emptyset) \times \Phi(C)$
- $\text{LocInv} \subseteq \mathcal{L} \times \{\circ, \bullet\} \times \Phi(C) \times \mathcal{L} \times \{\circ, \bullet\}$
- $\Theta : \mathcal{L} \cup \text{Msg} \cup \text{Cond} \cup \text{LocInv} \rightarrow \{\text{hot, cold}\}$

- $\mathcal{L} = \{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}, l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}, l_{3,0}, l_{3,1}, l_{3,2}, l_{3,3}\}$
- $l_{1,0} \prec l_{1,1} \prec l_{1,2} \prec l_{1,3}$, $l_{1,2} \prec l_{1,4}$, $l_{2,0} \prec l_{2,1} \prec l_{2,2} \prec l_{2,3}$, $l_{3,0} \prec l_{3,1} \prec l_{3,2} < l_{3,3}$, $l_{1,1} \prec l_{2,1}$, $l_{2,2} \prec l_{1,2}$, $l_{2,3} \prec l_{1,3}$, $l_{3,2} \prec l_{1,4}$.
- $\mathcal{I} = \{\{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}\}, \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}\}, \{l_{3,0}, l_{3,1}, l_{3,2}, l_{3,3}\}\}$.
- $\text{Msg} = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,2}), (l_{2,3}, D, l_{1,3}), (l_{3,3}, E, l_{1,4})\}$
- $\text{Cond} = \{\{l_{2,1}, l_{3,1}\}, c_4\}, \{\{l_{2,2}\}, c_1\}\}$
- $\text{LocInv} = \{(l_{1,1}, \circ, c_2 \land c_3, l_{1,2}, \bullet)\}$
**Concrete vs. Abstract Syntax**

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

- Scenarios: The Idea
- Use Cases
  - Use Case Diagrams
- User Stories
- Sequence Diagrams
  - A Brief History
  - Live Sequence Charts
    - LSC Body Syntax:
      - LSC Model Elements, Locations
      - Well-Formedness
    - Towards Semantics:
      - Cuts, Firedsets
      - Automaton Construction
    - Excursion: Symbolic Büchi Automata

Informatik III
(Automata Theory)
LSC Semantics: Towards Automaton Construction
**Definition.** Let \(((L, \preceq, \sim), \mathcal{I}, \operatorname{Msg}, \operatorname{Cond}, \operatorname{LocInv}, \Theta)\) be an LSC body.

A non-empty set \(\emptyset \neq C \subseteq L\) is called a **cut** of the LSC body iff \(C\)

- **is downward closed**, i.e.
  \[\forall l, l' \in L \bullet l' \in C \land l \preceq l' \implies l \in C,\]

- **is closed under simultaneity**, i.e.
  \[\forall l, l' \in L \bullet l' \in C \land l \sim l' \implies l \in C,\] and

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

The temperature function is extended to cuts as follows:

\[\Theta(C) = \begin{cases} 
  \text{hot} & \text{if } \exists l \in C \bullet (\nexists l' \in C \bullet l < l') \land \Theta(l) = \text{hot} \\
  \text{cold} & \text{otherwise}
\end{cases}\]

that is, \(C\) is **hot** if and only if at least one of its maximal elements is hot.
Cut Examples

\[ \emptyset \neq \mathcal{C} \subseteq \mathcal{L} \] – downward closed – simultaneity closed – at least one loc. per instance line

\[ LSC: \text{none} \quad AM: \text{invariant} \quad I: \text{strict} \]
$\emptyset \neq C \subseteq \mathcal{L}$ – downward closed – simultaneity closed – at least one loc. per instance line
Cut Examples

\[ \emptyset \neq \mathcal{C} \subseteq \mathcal{L} \] – downward closed – simultaneity closed – at least one loc. per instance line

\[ \text{LSC: none} \quad \text{AM: invariant} \quad \text{L: strict} \]
Cut Examples

\[ \emptyset \neq \mathcal{C} \subseteq \mathcal{L} \] – 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
\[ \emptyset \neq C \subseteq \mathcal{L} \text{ – downward closed – simultaneity closed – at least one loc. per instance line} \]
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** $\preceq$-successors of the front of $C$, i.e.
  \[ \forall l \in \mathcal{F} \exists l' \in C \bullet l' \preceq l \land (\exists l'' \in C \bullet l' \preceq l'' \preceq l), \]
- locations in $\mathcal{F}$, that lie on the same instance line, are **pairwise unordered**, i.e.
  \[ \forall l \neq l' \in \mathcal{F} \bullet (\exists I \in \mathcal{I} \bullet \{l, l'\} \subseteq I) \implies l \not\preceq l' \land l' \not\preceq l, \]
- 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 $C' = C \cup \mathcal{F}$ is called **direct successor of $C$ via $\mathcal{F}$**, denoted by $C \leadsto_{\mathcal{F}} C'$. 
\[ C \cap F = \emptyset \quad - C \cup F \text{ is a cut} - \text{only direct} \prec \text{-successors} - \text{same instance line on front pairwise unordered} - \text{sending of asynchronous reception already in} \]

\[ C \cap F = \emptyset \quad - C \cup F \text{ is a cut} - \text{only direct} \prec \text{-successors} - \text{same instance line on front pairwise unordered} - \text{sending of asynchronous reception already in} \]

\[ LSC: \text{ none} \quad - AM: \text{ invariant} \quad - I: \text{ strict} \]

\[ I_1 \quad I_2 \quad I_3 \]

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

\[ l_{1,1} \quad l_{2,1} \quad l_{3,1} \]

\[ l_{1,2} \quad l_{2,3} \]

\[ C \]

\[ E \quad F \quad G \]
$C \cap \mathcal{F} = \emptyset - C \cup \mathcal{F}$ is a cut – only direct $\prec$-successors – same instance line on front pairwise unordered – sending of asynchronous reception already in

LSC: none
AM: invariant I: strict
Content

- Scenarios: The Idea
- Use Cases
  - Use Case Diagrams
- User Stories
- Sequence Diagrams
  - A Brief History
  - Live Sequence Charts
    - LSC Body Syntax:
      - LSC Model Elements, Locations
      - Well-Formedness
    - Towards Semantics:
      - Cuts, Firedsets
      - Automaton Construction
    - Excursion: Symbolic Büchi Automata
LSC Semantics: TBA Construction
The TBA $B(\mathcal{L})$ of LSC $\mathcal{L}$ over $C$ and $E$ is $(C_B, Q, q_{ini}, \to, Q_F)$ with

- $C_B = C \cup E_I^{T}$, where $E_I^{T} = \{E_{i,j}^{I}, E_{i,j}^? | E \in E, i,j \in I\}$,
- $Q$ is the set of cuts of $\mathcal{L}$, $q_{ini}$ is the instance heads cut,
- $\to$ consists of loops, progress transitions (from $\leadsto \mathcal{P}$), and legal exits (cold cond./local inv.),
- $Q_F = \{C \in Q | \Theta(C) = \text{cold} \lor C = \mathcal{L}\}$ is the set of cold cuts and the maximal cut.
Tell Them What You’ve Told Them...

- **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,
  - pretty *useless* without the underlying use-case.

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

- **Sequence Diagrams:**
  - a *visual formalism* for interactions, i.e.,
    - precisely defined syntax,
    - precisely defined semantics
      (construct automaton from abstract syntax)
  - Can be used to precisely describe the interactions of a *use-case*. 
References
References


