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”)

- **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

- **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
**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. "if t, t' ∈ Time..."

**Use Cases**
- Pattern Language

**Decision Tables**
- Live Sequence Charts

---

**Topic Area Requirements Engineering: Content**

- **VL 6**
  - 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
Scenarios

- 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)
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).
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:

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

A **Use Case** is 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 actors</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
   1. ATM reads card, sends data to bank system
   2. bank system checks validity
   3. ATM shows PIN screen
   4. client enters PIN
   5. ATM reads PIN, sends to bank system
   6. bank system checks PIN
   7. 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

| 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>
Example: Use Case Diagram of the ATM Use Case

Use Case Example: ATM Authentication

<table>
<thead>
<tr>
<th>Use Case: ATM Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Actor:</strong> Client (main actor), Bank System</td>
</tr>
<tr>
<td><strong>Actor Name:</strong> Client, Bank System</td>
</tr>
<tr>
<td><strong>Goal:</strong> The client wants access to the ATM</td>
</tr>
<tr>
<td><strong>Pre-conditions:</strong> The ATM is operational, the welcome screen is displayed, card and PIN of client are available</td>
</tr>
<tr>
<td><strong>Post-conditions:</strong> Client accepted, services of ATM are offered</td>
</tr>
<tr>
<td><strong>Post-cond. in exceptional case:</strong> Access denied, card returned or withheld, welcome screen displayed</td>
</tr>
<tr>
<td><strong>Actors:</strong> Client (main actor), Bank System</td>
</tr>
<tr>
<td><strong>Open Questions:</strong> 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

2b. Card readable, but not ATM card

2c. No connection to bank system

3a. Card not valid or disabled

5a. Client cancels

5b. Client doesn't react within 5 s

6a. No connection to bank system

7a. First or second PIN wrong

7b. Third PIN wrong

(Adapted from Ludwig and Lichter, 2013)
Use Case Example: ATM Authentication

- **Goal**: Authentication
- **Pre-condition**: The client wants access to the ATM.
- **Post-condition**: The ATM is operational, the welcome screen is displayed, and card and PIN of the client are available.
- **Post-condition in exceptional case**: Access denied, card returned or withheld, welcome screen displayed.
- **Actors**: Client (main actor), Bank system.
- **Open Questions**: None.
- **Normal case**: 1. Client inserts card. 2. ATM reads card, sends data to bank system. 3. Bank system checks validity. 4. ATM displays 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 case 2b**: Card readable, but not ATM card. 3 Exception case 2c**: No connection to bank system.
- **Exception case 3a**: Card not valid or disabled.
- **Exception case 5a**: Client cancels.
- **Exception case 5b**: Client doesn't react within 5 s.
- **Exception case 6a**: No connection to bank system.
- **Exception case 7a**: First or second PIN wrong.
- **Exception case 7b**: Third PIN wrong.

*Source: Ludewig and Lichter, 2013*
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
      - Automation Construction
    - Excursion: Symbolic Büchi Automata

Informatik III
(Automata Theory)
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>As a [role] I want [something] so that [benefit].</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: 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>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>risk</th>
<th>real effort</th>
</tr>
</thead>
<tbody>
<tr>
<td></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>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>clarifies when and under what conditions the activity takes place</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>is MUST (obligation), SHOULD (wish), or WILL (intention); also: MUST NOT (forbidden)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>is either “the system” or the concrete name of a (sub-)system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>one of three possibilities:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• “does”, description of a system activity</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• “offers”, description of a function offered by the system to somebody</td>
<td></td>
<td></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

**goal**

Authentication

**pre-condition**

The bank wants to allow the client to access the ATM.

**post-condition**

The client is accepted, and the services of the ATM are offered.

**post-cond. in exceptional case**

Access denied, card returned or withheld, welcome screen displayed.

**actors**

Client (main actor), bank system

**open questions**

None

**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 case 2b**

Card readable, but not ATM card

**exception case 2c**

No connection to bank system

**exception case 3a**

Card not valid or disabled

**exception case 5a**

Client cancels

**exception case 5b**

Client doesn’t react within 5 s

**exception case 6a**

No connection to bank system

**exception case 7a**

First or second PIN wrong

**exception case 7b**

Third PIN wrong

(Ludewig and Lichter, 2013)
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 Marely, 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

LSC Body Syntax
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

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

where

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

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

\[
\mathcal{L} = \{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}, l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}, l_{3,0}, l_{3,1}, l_{3,2}, l_{3,3}\}
\]
\[
l_{1,0} \prec l_{1,1} \prec l_{1,2} \prec l_{1,3}, \quad l_{1,2} \prec l_{1,4}, \quad l_{2,0} \prec l_{2,1} \prec l_{2,2} \prec l_{2,3}, \quad l_{3,0} \prec l_{3,1} \prec l_{3,2} \prec l_{3,3},
\]
\[
l_{1,1} \prec l_{2,1}, \quad l_{2,2} \prec l_{2,1}, \quad l_{2,3} \prec l_{3,3}, \quad l_{3,2} \prec l_{4,1}, \quad l_{2,1} \approx l_{3,1}, \quad l_{2,2} \approx l_{3,2}.
\]
\[
\mathcal{I} = \{\{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, I_1\}, \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}\}, \{l_{3,0}, l_{3,1}, l_{3,2}, l_{3,3}\}\}
\]
\[
\text{Msg} = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,2}), (l_{2,3}, D, l_{1,3}), (l_{3,3}, E, l_{1,4})\}
\]
\[
\text{Cond} = \{((l_{2,1}, l_{3,1}), c_4), ((l_{2,3}), c_1)\}.
\]
\[
\text{LocInv} = \{(l_{1,1}, \text{c}, c_2 \wedge c_3, l_{1,2}, \text{c})\}
\]

Concrete vs. Abstract Syntax

\[
\mathcal{L} = \{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, l_{1,4}, l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}, l_{3,0}, l_{3,1}, l_{3,2}, l_{3,3}\}
\]
\[
l_{1,0} \prec l_{1,1} \prec l_{1,2} \prec l_{1,3}, \quad l_{1,2} \prec l_{1,4}, \quad l_{2,0} \prec l_{2,1} \prec l_{2,2} \prec l_{2,3}, \quad l_{3,0} \prec l_{3,1} \prec l_{3,2} \prec l_{3,3},
\]
\[
l_{1,1} \prec l_{2,1}, \quad l_{2,2} \prec l_{2,1}, \quad l_{2,3} \prec l_{3,3}, \quad l_{3,2} \prec l_{4,1}, \quad l_{2,1} \approx l_{3,1}, \quad l_{2,2} \approx l_{3,2}.
\]
\[
\mathcal{I} = \{\{l_{1,0}, l_{1,1}, l_{1,2}, l_{1,3}, I_1\}, \{l_{2,0}, l_{2,1}, l_{2,2}, l_{2,3}\}, \{l_{3,0}, l_{3,1}, l_{3,2}, l_{3,3}\}\}
\]
\[
\text{Msg} = \{(l_{1,1}, A, l_{2,1}), (l_{2,2}, B, l_{1,2}), (l_{2,2}, C, l_{3,2}), (l_{2,3}, D, l_{1,3}), (l_{3,3}, E, l_{1,4})\}
\]
\[
\text{Cond} = \{((l_{2,1}, l_{3,1}), c_4), ((l_{2,3}), c_1)\}.
\]
\[
\text{LocInv} = \{(l_{1,1}, \text{c}, c_2 \wedge c_3, l_{1,2}, \text{c})\}
\]
Well-Formedness

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

- For each location \( l \in L \), 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. \( \exists (l_1, \iota_1, \phi, l_2, \iota_2) \in \text{LocInv} : l \in \{l_1, l_2\} \).

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
      - Automation Construction
    - **Excursion:** Symbolic Büchi Automata
**Definition.** Let \( (L, \preceq, \sim), I, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta) \) 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.
  \[ \forall l, l' \in L \cdot l' \in C \land l \preceq l' \implies l \in C, \]

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

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

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.
\( \emptyset \neq C \subseteq \mathcal{L} \) — downward closed — simultaneity closed — at least one loc. per instance line
**Cut Examples**

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

[Diagram]

**Cut Examples**

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

[Diagram]
Cut Examples

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

Cut Examples

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

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

\[ \begin{array}{c}
\text{LSC: none} \\
\text{API: invariant I strict}
\end{array} \]
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 \( ((\mathcal{L}, \preceq, \sim), I, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta) \).

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. \( \forall l \in \mathcal{F} \exists l' \in C \cup \mathcal{F} \land l' \prec l \land (\exists l'' \in C \cup \mathcal{F} \land l' \prec l'' < l) \),
- locations in \( \mathcal{F} \) that lie on the same instance line, are pairwise unordered, i.e. \( \forall l \neq l' \in \mathcal{F} \exists I \in I \cup \{l, l'\} \subseteq I \Rightarrow l \not\sim l' \),
- for each asynchronous message reception in \( \mathcal{F} \), the corresponding sending is already in \( C \), \( \forall (l, E, l') \in \text{Msg} \land l' \in \mathcal{F} \Rightarrow l \in C \).

The cut \( C' = C \cup \mathcal{F} \) is called **direct successor of \( C \) via \( \mathcal{F} \)**, denoted by \( C \leadsto \mathcal{F} C' \).
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
The TBA $B(L)$ of LSC $L$ over $C$ and $E$ is $(C_B, Q, q_{ini}, \rightarrow, Q_F)$ with

- $C_B = C \cup E_{C_B}$, where $E_{C_B} = \{ E_{i,j}^C, E_{i,j}^E | E \in E, i, j \in I \}$,
- $Q$ is the set of cuts of $L$, $q_{ini}$ is the instance heads cut,
- $\rightarrow$ consists of loops, progress transitions (from $\rightarrow_F$), and legal exits (cold cond./local inv.),
- $Q_F = \{ C \in Q | \Theta(C) = \text{cold} \lor C = L \}$ is the set of cold cuts and the maximal cut.
• **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


