Softwaretechnik / Software-Engineering

Lecture 9: Live Sequence Charts

2016-06-06

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
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

Discussion
  - Working Definition: Software
Excursion: Symbolic Büchi Automata

LSC Semantics:
- Cuts, Firedsets,
- Automaton Construction
- Full LSC (activation, chart mode)

Pre-Charts
- Requirements Engineering with scenarios
- Strengthening scenarios into requirements

Software, formally
- Software specification
- Requirements Engineering, formally
- Software implements specification

LSCs vs. Software
- Software implements LSCs
- Scenarios and tests
- Play In/Play Out

Requirements Engineering Wrap-Up
LSC Semantics
The Plan: A Formal Semantics for a Visual Formalism

concrete syntax
(diagram)

abstract syntax
((L, ≤, ~), I, Msg, Cond, LocInv, Θ)

(büchi automaton)

true
Excursion: Symbolic Büchi Automata
From Finite Automata to Symbolic Büchi Automata

\( \mathcal{L}(A) = 0(1)^* \)

\( \mathcal{L}(B) = 0(10)^\omega \)

\( \mathcal{L}(B') = [011]^* 10^\omega \)

\( \mathcal{L}(B) \leq \Sigma^\omega \)

**A:**
- States: \( q_1, q_2 \)
- Alphabet: \( \Sigma = \{0, 1\} \)
- Initial state: \( q_1 \)
- Final states:
  - Even: \( q_1 \)
  - Odd: \( q_2 \)

- **B:**
  - States: \( q_1, q_2 \)
  - Alphabet: \( \Sigma = \{0, 1\} \)
  - Initial state: \( q_1 \)
  - Final states:
    - Even: \( q_1 \)
    - Odd: \( q_2 \)

- **Büchi infinite words**

- **Symbolic Büchi**

- **A_{sym}**
  - States: \( q_1, q_2 \)
  - Alphabet: \( \Sigma = \{x \rightarrow \mathbb{N}\} \)
  - Initial state: \( q_1 \)
  - Final states:
    - Even: \( q_1 \)
    - Odd: \( q_2 \)

- **B_{sym}**
  - States: \( q_1, q_2 \)
  - Alphabet: \( \Sigma = \{x \rightarrow \mathbb{N}\} \)
  - Initial state: \( q_1 \)
  - Final states:
    - Even: \( q_1 \)
    - Odd: \( q_2 \)
Definition. A **Symbolic Büchi Automaton** (TBA) is a tuple

\[ B = (\mathcal{C}_B, Q, q_{\text{ini}}, \rightarrow, Q_F) \]

where

- \( \mathcal{C}_B \) is a set of atomic propositions,
- \( Q \) is a finite set of states,
- \( q_{\text{ini}} \in Q \) is the initial state,
- \( \rightarrow \subseteq Q \times \Phi(\mathcal{C}_B) \times Q \) is the finite transition relation.

Each transitions \((q, \psi, q') \in \rightarrow\) from state \(q\) to state \(q'\) is labelled with a formula \(\psi \in \Phi(\mathcal{C}_B)\).

- \( Q_F \subseteq Q \) is the set of fair (or accepting) states.
**Run of TBA**

**Definition.** Let $B = (C_B, Q, q_{ini}, \to, Q_F)$ be a TBA and

$$w = \sigma_1, \sigma_2, \sigma_3, \ldots \in (\Phi(C_B) \to B)^\omega$$

an infinite word, each letter is a valuation of $\Phi(C_B)$.

An infinite sequence

$$\rho = q_0, q_1, q_2, \ldots \in Q^\omega$$

of states is called **run** of $B$ over $w$ if and only if

- $q_0 = q_{ini}$,
- for each $i \in \mathbb{N}_0$ there is a transition $(q_i, \psi_i, q_{i+1}) \in \to$ s.t. $\sigma_i \models \psi_i$.

---

**Example:**

$$L = (x = 2)(x = 5)(x = 4)^\omega$$

$$R = q_A, q_B, q_A, q_B, \ldots$$

**$B_{sym}:$**

$$\Sigma = (\{x\} \to \mathbb{N})$$

- $even(x)$
- $odd(x)$
Definition.

We say TBA $B = (C_B, Q, q_{ini}, \rightarrow, Q_F)$ accepts the word

$$w = (\sigma_i)_{i \in \mathbb{N}_0} \in (\Phi(C_B) \rightarrow B)^\omega$$

if and only if $B$ has a run

$$\rho = (q_i)_{i \in \mathbb{N}_0}$$

over $w$ such that fair (or accepting) states are visited infinitely often by $\rho$, i.e., such that

$$\forall i \in \mathbb{N}_0 \exists j > i : q_j \in Q_F.$$ 

We call the set $\text{Lang}(B) \subseteq (\Phi(C_B) \rightarrow B)^\omega$ of words that are accepted by $B$ the language of $B$. 

Example:

$$\Sigma = \{x\} \rightarrow \mathbb{N}$$
LSC Semantics: TBA Construction
The Plan: A Formal Semantics for a Visual Formalism

concrete syntax
(diagram)

abstract syntax

(((L, ≤, ~), I, Msg, Cond, LocInv, Θ))

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.
  \[\forall l, l' \in \mathcal{L} \bullet l' \in C \land l \preceq l' \implies l \in C,\]

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

- 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:

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

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

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

\[
\begin{align*}
I_1 & \quad l_{1,0} & \quad \sim & \quad l_{2,0} \\
I_2 & \quad \sim & \quad l_{3,0} \\
I_3 \\
\end{align*}
\]
$\emptyset \neq C \subseteq \mathcal{L}$ – downward closed – simultaneity closed – at least one loc. per instance line
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 \( \emptyset \neq F \subseteq \mathcal{L} \) of locations is called **fired-set** \( F \) of cut \( C \) if and only if

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

The cut \( C' = C \cup F \) is called **direct successor of** \( C \) **via** \( F \), denoted by \( C \sim_F C' \).
$C \cap F = \emptyset$ – $C \cup F$ is a cut – only direct $\prec$-successors – same instance line on front pairwise unordered – sending of asynchronous reception already in
The TBA $B(\mathcal{L})$ of LSC $\mathcal{L}$ over $C$ and $\mathcal{E}$ is $(C_B, Q, q_{ini}, \rightarrow, Q_F)$ with

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

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

$$\rightarrow = \{(q, \psi_{loop}(q), q) \mid q \in Q\} \cup \{(q, \psi_{prog}(q, q'), q') \mid q \sim_{\varepsilon} q'\} \cup \{(q, \psi_{exit}(q), L) \mid q \in Q\}$$
“Only” construct the transitions’ labels:

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

\[
\psi_{\text{loop}}(q) = \psi_{\text{Msg}}(q) \land \psi_{\text{LocInv}}^\text{hot}(q) \land \psi_{\text{LocInv}}^\text{cold}(q)
\]

\[
\psi_{\text{exit}}(q) = \left(\psi_{\text{loop}}^\text{hot}(q) \land \neg \psi_{\text{LocInv}}^\text{cold}(q)\right) \lor \bigvee_{1 \leq i \leq n} \left(\psi_{\text{prog}}^\text{hot}(q, q_i) \lor \psi_{\text{LocInv}}^\text{cold}(q, q_i) \land \neg \psi_{\text{Cond}}(q, q_i)\right)
\]

\[
\psi_{\text{prog}}(q, q_n) = \left(\psi_{\text{Msg}}(q, q_n) \land \psi_{\text{Cond}}(q, q_n) \land \psi_{\text{LocInv}, \bullet}^\text{hot}(q, q_n) \land \psi_{\text{LocInv}, \bullet}^\text{cold}(q, q_n)\right)
\]
Loop Condition

\[ \psi_{\text{loop}}(q) = \psi_{\text{Msg}}(q) \land \psi_{\text{LocInv}}^{\text{hot}}(q) \land \psi_{\text{LocInv}}^{\text{cold}}(q) \]

- \( \psi_{\text{Msg}}(q) = \neg \bigvee_{1 \leq i \leq n} \psi_{\text{Msg}}(q, q_i) \land (\text{strict} \implies \bigwedge_{\psi \in E_1 ? \cap \text{Msg}(L)} \neg \psi) \)

- \( \psi_{\theta}^{\text{LocInv}}(q) = \bigwedge_{\ell = (l, \nu, \phi, l', \nu') \in \text{LocInv}, \Theta(\ell) = \theta, \ell \text{ active at } q} \)

A location \( l \) is called \textbf{front location} of cut \( C \) if and only if \( \nexists l' \in L \cdot l < l' \).

Local invariant \((l_o, \nu_0, \phi, l_1, \nu_1)\) is \textbf{active} at cut (!) \( q \) if and only if \( l_0 \preceq l < l_1 \) for some front location \( l \) of cut \( q \) or \( l = l_1 \land \nu_1 = \bullet \).

- \( \text{Msg}(\mathcal{F}) = \{ E! \mid (l, E, l') \in \text{Msg}, \ l \in \mathcal{F} \} \cup \{ E? \mid (l, E, l') \in \text{Msg}, \ l' \in \mathcal{F} \} \)

- \( \text{Msg}(\mathcal{F}_1, \ldots, \mathcal{F}_n) = \bigcup_{1 \leq i \leq n} \text{Msg}(\mathcal{F}_i) \)

\[ \begin{array}{c}
\begin{tikzpicture}
\node (I1) at (0,0) {$I_1$};
\node (I2) at (2,0) {$I_2$};
\node (I3) at (4,0) {$I_3$};
\node (O) at (-1,1) {$O$};
\node (A) at (1,2) {$A$};
\node (B) at (1,0) {$B$};
\node (C) at (3,2) {$C$};
\node (D) at (1,-2) {$D$};
\node (E) at (3,-2) {$E$};
\node (c1) at (-0.5,-1) {$c_1$};
\node (c2) at (1,0.5) {$c_2 \land c_3$};
\draw[->] (O) -- (A);
\draw[->] (A) -- (I1);
\draw[->] (B) -- (I2);
\draw[->] (C) -- (I3);
\draw[->] (D) -- (E);
\end{tikzpicture}
\end{array} \]
**Progress Condition**

\[ \psi_{\text{prog}}^{\text{hot}}(q, q_i) = \psi_{\text{Msg}}^{\text{hot}}(q, q_n) \land \psi_{\text{Cond}}^{\text{hot}}(q, q_n) \land \psi_{\text{LocInv}, \bullet}^{\text{hot}}(q_n) \]

- \[ \psi_{\text{Msg}}^{\text{hot}}(q, q_i) = \bigwedge_{\psi \in \text{Msg}(q_i \setminus q)} \psi \land \bigwedge_{j \neq i} \bigwedge_{\psi \in (\text{Msg}(q_j \setminus q) \setminus \text{Msg}(q_i \setminus q))} \neg \psi \land (\text{strict} \implies \bigwedge_{\psi \in (\mathcal{E}_1 \cap \text{Msg}(L)) \setminus \text{Msg}(F_i)} \neg \psi) \]

- \[ \psi_{\text{Cond}}^{\text{hot}}(q, q_i) = \bigwedge_{\gamma = (L, \phi) \in \text{Cond}, \Theta(\gamma) = \theta, L \cap (q_i \setminus q) \neq \emptyset} \phi \]

- \[ \psi_{\text{LocInv}, \bullet}^{\text{hot}}(q, q_i) = \bigwedge_{\lambda = (l, \iota, \phi, l', \iota') \in \text{LocInv}, \Theta(\lambda) = \theta, \lambda \bullet\text{-active at } q_i} \phi \]

Local invariant \((l_0, \iota_0, \phi, l_1, \iota_1)\) is \(\bullet\)-active at \(q\) if and only if

- \(l_0 < l < l_1\), or
- \(l = l_0 \land \iota_0 = \bullet\), or
- \(l = l_1 \land \iota_1 = \bullet\)

for some front location \(l\) of cut (!) \(q\).
Example

\[
\begin{align*}
q_1 &
\rightarrow \phi, l_2, 1, q_2
\end{align*}
\]
**Full LSCs**

A full LSC \( \mathcal{L} = (((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta), ac_0, am, \Theta_\mathcal{L}) \) consists of

- **body** \( ((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta) \),
- **activation condition** \( ac_0 \in \Phi(C) \),
- **strictness flag** \( \text{strict} \) (if false, \( \mathcal{L} \) is **permissive**)
- **activation mode** \( am \in \{\text{initial, invariant}\} \),
- **chart mode** **existential** \( (\Theta_\mathcal{L} = \text{cold}) \) or **universal** \( (\Theta_\mathcal{L} = \text{hot}) \).

Concrete syntax:
A **full LSC** \( \mathcal{L} = (((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta), ac_0, am, \Theta_\mathcal{L}) \) consists of

- **body** \( ((\mathcal{L}, \preceq, \sim), \mathcal{I}, \text{Msg}, \text{Cond}, \text{LocInv}, \Theta) \),
- **activation condition** \( ac_0 \in \Phi(C) \),
- **strictness flag** \( \text{strict} \) (if false, \( \mathcal{L} \) is **permissive**)
- **activation mode** \( am \in \{\text{initial, invariant}\} \),
- **chart mode** **existential** (\( \Theta_\mathcal{L} = \text{cold} \)) or **universal** (\( \Theta_\mathcal{L} = \text{hot} \)).

A set of words \( W \subseteq (C \rightarrow \mathbb{B})^\omega \) is **accepted** by \( \mathcal{L} \) if and only if

- **cold**
  \[
  \exists w \in W \cdot w^0 \models ac \land \\
  w^0 \models \psi_{\text{cold}}^\text{Cond}(\emptyset, C_0) \land w/1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))
  \]
- **hot**
  \[
  \forall w \in W \cdot w^0 \models ac \implies \\
  w^0 \models \psi_{\text{cold}}^\text{Cond}(\emptyset, C_0) \land w/1 \in \text{Lang}(\mathcal{B}(\mathcal{L}))
  \]

where \( ac = ac_0 \land \psi_{\text{cold}}^\text{Cond}(\emptyset, C_0) \land \psi_{\text{Msg}}^\text{Cond}(\emptyset, C_0) \); \( C_0 \) is the minimal (or **instance heads**) cut.
Full LSC Semantics: Example

LSC: $\mathcal{L}$  
AC: $c_1$  
AM: initial I: permissive

$I_1$  $I_2$  $I_3$

$E$  $\phi$  $F$  
$G$

$\neg E$  
$E!$

$E!$

$E? \land \neg \phi$

$E?$  $\neg E$  

$E? \land \phi$

$F!$

$F!$

$(F? \lor G! \lor G?)$

$(G! \land G?)$

$(G! \land G?)$

$G! \land G? \land F?$

$G! \land G? \land \neg F?$

$\neg F?$

$F?$

$G! \land G?$

$\text{true}$

true
Example: Vending Machine

- **Positive scenario**: Buy a Softdrink
  (i) Insert one 1 euro coin.
  (ii) Press the ‘softdrink’ button.
  (iii) Get a softdrink.

- **Positive scenario**: Get Change
  (i) Insert one 50 cent and one 1 euro coin.
  (ii) Press the ‘softdrink’ button.
  (iii) Get a softdrink.
  (iv) Get 50 cent change.

- **Negative scenario**: A Drink for Free
  (i) Insert one 1 euro coin.
  (ii) Press the ‘softdrink’ button.
  (iii) Do not insert any more money.
  (iv) Get two softdrinks.
Example: Buy A Softdrink

- LSC: buy softdrink
- AC: true
- AM: invariant I: permissive

Diagram:
- User
- Vend. Ma.
- E1
- pSOFT
- SOFT
Example: Get Change

LSC: get change
AC: true
AM: invariant I: permissive

User

Vend. Ma.

C50

E1

pSOFT

SOFT

chg-C50
Anti-Scenarios: Don’t Give Two Drinks

LSC: only one drink
AC: true
AM: invariant I: permissive

User  Vend. Ma.

E1

pSOFT

SOFT

SOFT

¬C50! ∧ ¬E1!

false
A full LSC $\mathcal{L} = (PC, MC, ac_0, am, \Theta, \varphi)$ actually consist of

- **pre-chart** $PC = ((L_P, \preceq_P, \sim_P), I_P, Msg_P, Cond_P, LocInv_P, \Theta_P)$ (possibly empty),

- **main-chart** $MC = ((L_M, \preceq_M, \sim_M), I_M, Msg_M, Cond_M, LocInv_M, \Theta_M)$ (non-empty),

- **activation condition** $ac_0 \in \Phi(C)$,

- **strictness flag** strict (if false, $\mathcal{L}$ is permissive)

- **activation mode** $am \in \{\text{initial, invariant}\}$,

- **chart mode** existential ($\Theta = \text{cold}$) or universal ($\Theta = \text{hot}$).
Universal LSC: Example

LSC: buy water
AC: true
AM: invariant I: strict

User [C50]
CoinValidator [pWATER]
ChoicePanel [water_in_stock]
Dispenser [dWATER, OK]

 ~(C50 ! E1 ! pSOFT! ∨ pTEA! ∨ pFILLUP!)
 ~(dSoft! ∨ dTEA!)
## Pre-Charts Semantics

<table>
<thead>
<tr>
<th>Θ L</th>
<th>( am = \text{initial} )</th>
<th>( am = \text{invariant} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \exists w \in W \exists m \in \mathbb{N}_0 \mid w^0 \models ac )</td>
<td>( \exists w \in W \exists k &lt; m \in \mathbb{N}_0 \mid w^k \models ac )</td>
<td></td>
</tr>
<tr>
<td>( &amp; w^0 \models \psi^{\text{Cond}}_{\text{hot}} (\emptyset, C^P_0) )</td>
<td>( &amp; w^k \models \psi^{\text{Cond}}_{\text{hot}} (\emptyset, C^P_0) )</td>
<td></td>
</tr>
<tr>
<td>( &amp; w/1, \ldots, w/m \in \text{Lang}(\mathcal{B}(PC)) )</td>
<td>( &amp; w/k + 1, \ldots, w/m \in \text{Lang}(\mathcal{B}(PC)) )</td>
<td></td>
</tr>
<tr>
<td>( &amp; w^{m+1} \models \psi^{\text{Cond}}_{\text{hot}} (\emptyset, C^M_0) )</td>
<td>( &amp; w^{m+1} \models \psi^{\text{Cond}}_{\text{hot}} (\emptyset, C^M_0) )</td>
<td></td>
</tr>
<tr>
<td>( &amp; w/m + 1 \in \text{Lang}(\mathcal{B}(MC)) )</td>
<td>( &amp; w/m + 1 \in \text{Lang}(\mathcal{B}(MC)) )</td>
<td></td>
</tr>
</tbody>
</table>

### Hot

\( \forall w \in W \mid w^0 \models ac \)  
\( \& w^0 \models \psi^{\text{Cond}}_{\text{cold}} (\emptyset, C^P_0) \)  
\( \& w/1, \ldots, w/m \in \text{Lang}(\mathcal{B}(PC)) \)  
\( \& w^{m+1} \models \psi^{\text{Cond}}_{\text{cold}} (\emptyset, C^M_0) \)  
\( \Rightarrow w^{m+1} \models \psi^{\text{Cond}}_{\text{cold}} (\emptyset, C^M_0) \)  
\( \& w/m + 1 \in \text{Lang}(\mathcal{B}(MC)) \)
One quite effective approach:

(i) **Approximate** the software requirements: ask for positive / negative existential scenarios.

(ii) **Refine** result into universal scenarios (and validate them with customer).

That is:

- Ask the customer to describe **example usages** of the desired system.
  
  In the sense of: “If the system is not at all able to do this, then it’s not what I want.”
  
  (→ positive use-cases, existential LSC)

- Ask the customer to describe behaviour that **must not happen** in the desired system.
  
  In the sense of: “If the system does this, then it’s not what I want.”
  
  (→ negative use-cases, LSC with pre-chart and hot-\textit{false})

- Investigate preconditions, side-conditions, exceptional cases and corner-cases.
  
  (→ extend use-cases, refine LSCs with conditions or local invariants)

- Generalise into universal requirements, e.g., **universal LSCs**.

- **Validate** with customer using new positive / negative scenarios.
Strengthening Scenarios Into Requirements

Customer announcement (Lastenheft) → Customer offer (Pflichtenheft) → Customer software contract (incl. Pflichtenheft) → Developer software delivery

Needs! Solution!
- Ask customer for (pos./neg.) scenarios, note down as existential LSCs:

- Strengthen into requirements, note down as universal LSCs:

- Re-Discuss with customer using example words of the LSCs’ language.
Tell Them What You’ve Told Them...

- **Live Sequence Charts** (if well-formed)
  - have an abstract syntax.

- From an abstract syntax, mechanically construct its **TBA**.

- A **universal LSC** is **satisfied** by a software $S$ if and only if
  - all words induced by the computation paths of $S$
  - are accepted by the LSC’s TBA.

- An **existential LSC** is **satisfied** by a software $S$ if and only if
  - there is a word induced by a computation path of $S$
  - which is accepted by the LSC’s TBA.

- **Pre-charts** allow us to specify
  - anti-scenarios (“this must not happen”),
  - activation interactions.

- **Method**:
  - discuss (anti-)scenarios with customer,
  - generalise into universal LSCs and re-validate.
References
References

