Software and Software Specification, formally

Definition. Software is a finite description $S$ of a (possibly infinite) set $\llbracket S \rrbracket$ of (finite or infinite) computation paths of the form $\sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots$ where

- $\sigma_i \in \Sigma$, $i \in \mathbb{N}_0$, is called state (or configuration), and
- $\alpha_i \in A$, $i \in \mathbb{N}_0$, is called action (or event).

The (possibly partial) function $\llbracket \cdot \rrbracket : S \mapsto \llbracket S \rrbracket$ is called interpretation of $S$.

Definition. A software specification is a finite description $S$ of a (possibly infinite) set $\llbracket S \rrbracket$ of softwares, i.e. $\llbracket S \rrbracket = \{ (S_1, \llbracket \cdot \rrbracket_1), (S_2, \llbracket \cdot \rrbracket_2), \ldots \}$. The (possibly partial) function $\llbracket \cdot \rrbracket : S \mapsto \llbracket S \rrbracket$ is called interpretation of $S$. 

Formal Methods in Requirements Engineering

Software and Software Specification, formally

- $\Sigma$ is a finite set of states
- $A$ is a finite set of actions
- $\llbracket \cdot \rrbracket : S \mapsto \llbracket S \rrbracket$
- $\llbracket S \rrbracket$ is a set of computation paths

The notation $S_1 \xrightarrow{\alpha} S_2$ denotes a transition from state $S_1$ to state $S_2$ by action $\alpha$. 

Syntax and Semantics

- Full LSC syntax
- Activation, Pre-Chart, Chart Mode
- Automaton Construction
- Loop/Progress/Exit Conditions
- LSCs vs. Software
- Excursion: Symbolic Büchi Automata

Methodology

- Requirements Engineering with scenarios
- Strengthening scenarios into requirements
- Requirements Engineering Wrap-Up

\[ \sigma_0^0 \alpha_0^0 \rightarrow \sigma_0^1 \alpha_0^1 \rightarrow \sigma_0^2 \cdot \cdot \cdot \]

\[ \{ (S_0^0, [\cdot]) \} \]

\[ \sigma_1^0 \alpha_1^1 \rightarrow \sigma_1^1 \alpha_1^2 \rightarrow \sigma_1^2 \cdot \cdot \cdot \]

\[ \{ (S_1^0, [\cdot]) \} \]

\[ \sigma_2^0 \alpha_2^1 \rightarrow \sigma_2^1 \alpha_2^2 \rightarrow \sigma_2^2 \cdot \cdot \cdot \]

\[ \{ (S_2^0, [\cdot]) \} \]

\[ I \]

\[ M \]

\[ S_1 \text{ implements } S \text{ via } I \text{ and } M \]

\[ I \]

\[ M' \]

\[ S_2 \text{ implements } S \text{ via } I \text{ and } M \]

• Formal Methods in Requirements Engineering
• Software & Software Specification, formally
• Requirements Engineering, formally
• Examples:
  • Decision Tables
  • Use Cases
  • Live Sequence Charts
    • LSC Semantics:
      • Full LSC syntax
      • Activation, Pre-Chart, Chart Mode
      • Automaton Construction
      • Loop / Progress / Exit Conditions
    • LSCs vs. Software
  • Excursion: Symbolic Büchi Automata
• Methodology
  • Requirements Engineering with scenarios
  • Strengthening scenarios into requirements
• Requirements Engineering Wrap-Up
activation interactions.

- anti-scenarios ("this must not happen"),
- Pre-charts
- LSC's TBA.

Sinduced by a computation path of there is a word satisfied is existential

A

if and only if

Sinduced by the computation paths of

•

LSC's TBA.

Accepted

- •

References

Hanser, 6th edition.

dpunkt.verlag, 3rd edition.

Software Engineering

Live Sequence Charts

Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine