Main visual formalism used for requirements:
message sequence charts (MSCs)

But,... we need richer requirements
may/must;
can/always;
fragmental and overlapping scenarios;
anti-scenarios;
etc.
We must go far beyond MSCs
Consistency gets very difficult

Live Sequence Charts and the Play-In/Play-Out Approach

Hillel Kugler
New York University
Work by David Harel, Rami Marely and others
Software Engineering Course
7 February 2005

Requirements;
Scenario-based
Stories about the system

MSCs:
(Message Sequence Charts)
inter-object behavior
(one story for all relevant objects)
**Live sequence charts (LSC's)**

"LSC's: Breathing Life into Message Sequence Charts"

(Damm & Harel, '98)

A natural extension of classical MSCs, with modalities (universal/existential, hot/cold, etc.) and structure (subcharts, conditionals, loops, etc.)

---

**Basic form of an LSC**

![Diagram of an LSC](image)

---

**Element** | **Mandatory** | **Provisional**
---|---|---
chart | universal \(\forall\) all system runs satisfy chart | existential \(\exists\) at least one run satisfies chart
location in chart | hot | cold
message | hot if message is sent |
condition | hot if condition must be met otherwise abort |
Play-In/ Play-Out (Harel & Marelly)

- Subcharts
- Loops
- Cold conditions enable control structures
- Hot conditions enable anti-scenarios:

The Play-Engine: Play-Out

- Universal charts drive the execution
- (external event; step*; stable?) = superstep
- Play-Out activates system events
- Feeling of working with Final Implementation

Play-Engine can thus be viewed as a “Universal Reactive System”.

The Play-Engine: Play-In

- Use the GUI whenever possible; clicks, right-clicks, menus, etc.
- Reflect as much as possible in the GUI itself, including mouse-over displaying and pop-up notes.
- Allow direct chart manipulation too, and chart element selection.

- Slicker HCI definitely desirable.
LSC Language Features

- Messages
- Assignments
- Conditions
- Advanced Features:
  - Symbolic Instances
  - Time
  - Forbidden Elements

Playing with Time

Symbolic Instances

- Associate instances with classes rather than with objects.

- Useful for parameterized systems or when the number of objects may change dynamically (construction and destruction).
Playing with Time (cont)

- Play-Out mechanism extended to support timed specifications.
- The Synchrony Assumption.
- Clock can be advanced manually or by being connected to the computer clock.
- Intervals between clock-ticks are determined by the user.

Applications

- Smart Cards Machine (with Gera Weiss) (Ametist project)

Utilizing cold time elements

Forbidden messages and conditions (scoped)
Depannage Service - Telecom
France Telecom - Pierre Combes (OMEGA)

Flight Control - Voting and Monitoring
IAI - Halman, Haruvi, Lahav, Zenou (OMEGA)

LSCs and the Play/In-Play-Out approach
- Requirement Specification
- Rapid prototyping
- Testing
- Verification (Smart Play-Out)

Modeling C. elegans
(J. Hubbard, N. Kam, J. Fisher, M. Stern, D. Harel and others)
 Lots of Related Work  
(Apologies to work not mentioned)

- MSC's - Alur/Holzmann/Peled/Yannakakis, Henriksen/Mukund/Kumar/Thiagarajan, Sista/Koskimies/Tuomi/Mannisto, Brox, Kruger, Schumann /Whittle ...
- LSC's - Damm Klose Wittke ...
- SCR - Heitmeyer Kirby Labaw Bharadwaj ...
- Temporal Logic ...
- Model Checking ...

Perhaps no need for model or code at all?

References:  
http://www.wisdom.weizmann.ac.il/~harel  
http://www.cs.nyu.edu/~kugler

A book on the topic:  

Come, Let's Play:  
Scenario-Based Programming Using LSCs and the Play-Engine  
D. Harel and R. Marely  
Springer-Verlag, 2003

(includes the Play-Engine software)