MoDES 2007

Live Sequence Charts as Test Purposes for Real-Time Systems

Shuhao Li

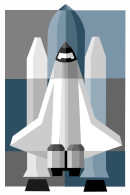
Center for Embedded Software Systems
Aalborg University

Model-Based Testing (MBT)

- Model-based testing of embedded real-time systems

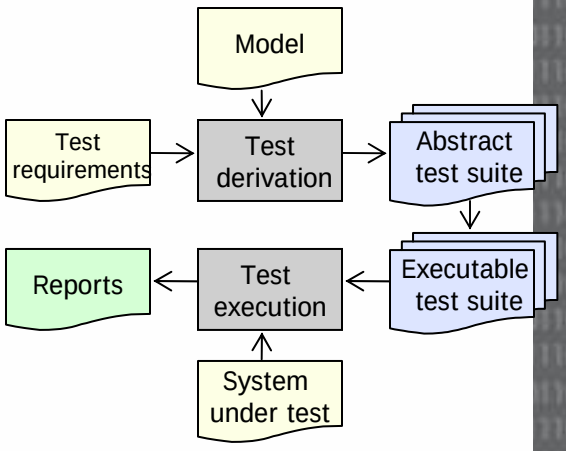
- ”model-based”: timed automata (TA), TIOA, TIOSM, etc.

- ”testing”: a realistic and practical approach (finite size test suite)



- ”embedded”: reactive systems (complex interaction with external environments), widely deployed, sometimes safety-critical

- ”real-time”: non-trivial timing constraints (hard real-time, dense real-time)

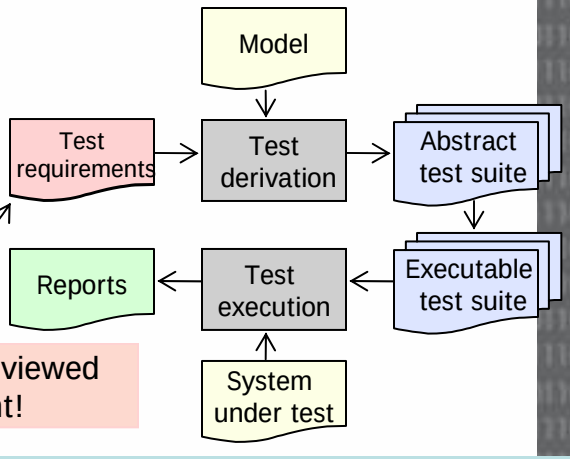


the information flow of MBT

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

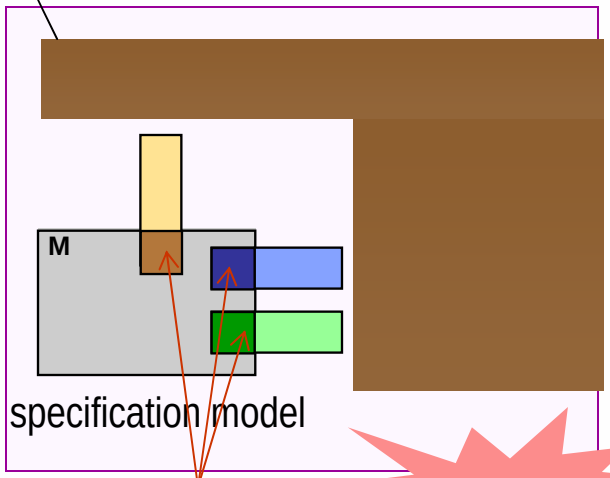
Motivation

- MBT: strengths
  - ✓ A rigorous, systematic testing method
  - ✓ Enabling early testing
  - ✓ Enabling automation
- Problem (not only of MBT)
  - ✗ Unable to conduct comprehensive and in-depth testing
    - Enormous (and usually infinite) state space of the system behavior
    - Limited project resources
    - Tight release time
- "Test purpose" may be a remedy!
  - Why bother?
    - "Testing on-demand", "Targeted testing"
  - Definition
    - A **test purpose** defines a [redacted] of the system that one particularly wants to check on the IUT.



test purpose can be viewed as a test requirement!

what is a **desired** (allowed) behavior,
what is a **prohibited** behavior,
...



behavior to be tested

using LSCs!

What can be this kind of properties???

How can we specify this kind of properties???

Existing Work on Test Purpose

- Informal descriptions (in natural language or pseudo-code)
- Temporal logics
  - Uppaal query language [Hessel 03 FATES], a subset of TCTL
- Timed Automaton and its variants
  - TIOA (Timed Input-Output Automaton) [Bouaziz 06 TestCom] [Fouchal 00 RTCSA]
  - TIOSM (Timed Input Output State Machine) [Castanet 98 ICCCN] [Kone 01 CompJ]
- Message Sequence Charts (MSC) [Grabowski 93 SDL-forum] and its variants
  - MSC2000 [En-Nouaary 04 WITUL]
  - UML Sequence Diagrams [Binder 99 book]

Test Purpose by Informal Description

- A Test Purpose is a prose description of a well-defined testing objective focusing on a specific (set of) requirements.
- Example:

TP/AP/ACF/MA/CA-000 is the 000-th **test purpose** for the capability testing (CA) of the Mac Id assignment (MA) procedures of the association control function (ACF) implemented at Access Point (AP) side.

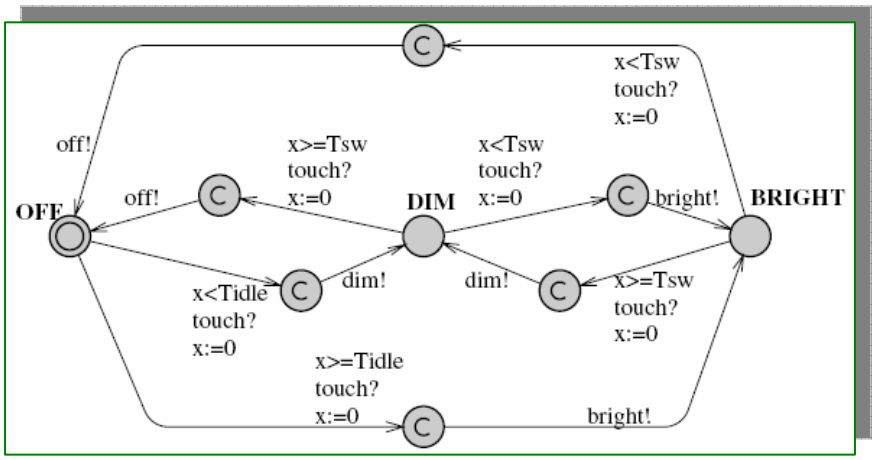
5.2.1.2 Mac Id assignment

TP/AP/ACF/MA/CA-000	Reference: TS 101 761-2 [1], clause 5.1.1.2 Initial condition: MT_disassociated_from_AP. Check, that: after receiving the RLC_MAC_ID_ASSIGN message the IUT replies with a RLC_MAC_ID_ASSIGN_ACK message containing the assigned Mac Id.

Test Purpose in Temporal Logics

- Test purpose in the form of
  - Uppaal query language (a subset of TCTL) [Hessel 03 FATES]
- Example:

Test Purpose: Check that the light can become bright.
E<> LightController.bright



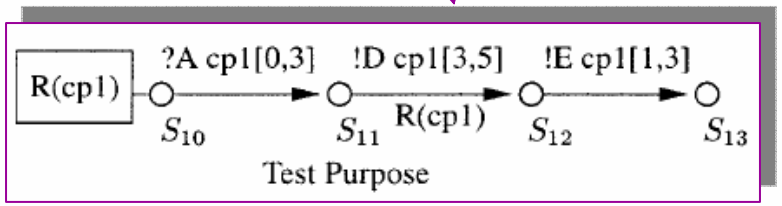
[Hessel 03 FATES]

a light controller

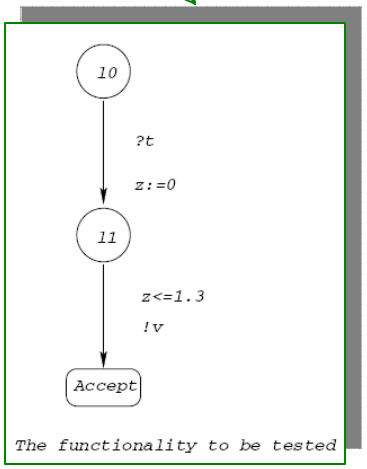
- How to derive test suite
  - Resort to model checking! To generate a diagnostic trace (witness or counter-example)

Test Purpose in Timed Automata

- Test purpose in the forms of
  - Timed Input Output State Machine (TIOSM) [Castanet 98 ICCCN] [Kone 01 CompJ]
  - Timed Input-Output Automaton (TIOA) [Bouaziz 06 TestCom] [Fouchal 00 RTCSA]
- Examples



[Castanet 98 ICCCN]



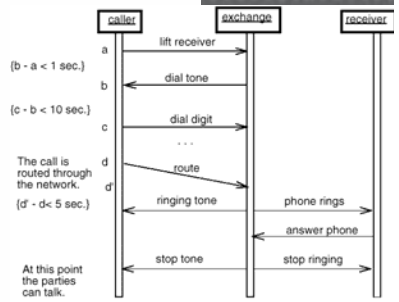
[Bouaziz 06 TestCom]

- How to derive test suite

  (typical approach:)

  1. Transform the test purposes into the same formalisms as the specification models
  2. Make synchronous products
  3. Derive test suites from the synchronous products

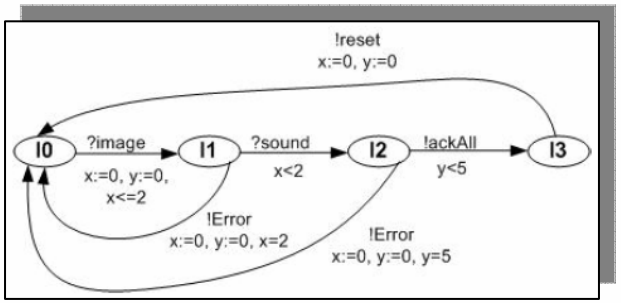
Test Purpose in Message Sequence Charts



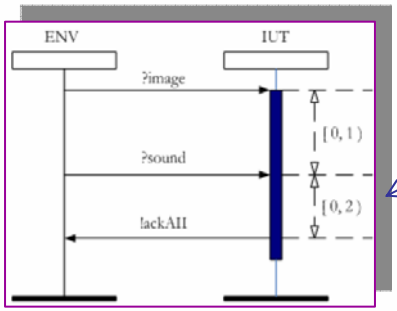
- Message Sequence Charts (MSCs)
  - Describing the scenarios of multi-object interactions
  - Initially used as test purpose by Grabowski et al. [Grabowski 93 SDL-forum]

Similar in **UML Sequence Diagrams**

- MSC2000
  - Can express both timers and the time constraints between any pair of events appearing in MSC diagrams
- Example



specification of multimedia system



a test purpose

[En-Nouaary 04 WITUL]

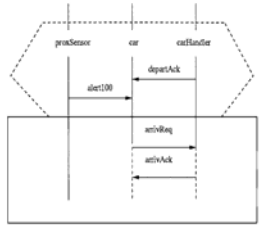
- Test derivation: based on synchronous product

Remarks on Existing Work

- Informal descriptions
  - Imprecise, ambiguious semantics
  - Problem with automated tool support
- Temporal Logics
  - Difficult to formulate complex scenarios for inexperienced testers
- Timed Automaton
  - Not scenario-oriented
- Message Sequence Charts
  - Support only possible scenarios of system behavior
    - They can state what might possibly occur, not what must occur
    - In the extreme, an empty system can trivially satisfies all the MSCs
  - Insufficient expressive power

They are not aimed to specify **scenario-based** behavior!

Live Sequnece Charts (LSCs)



- What is [redacted] Sequence Chart? --- a smooth extension of MSC

the ability to specify **liveness** (i.e., things that **must** occur)

- Why LSC?
  - Differentiate between necessity and possibility
    - [redacted] (prechart, main charts): not violated
    - [redacted]: should be satisfied by at least one satisfying run
    - Temperature (cold, hot)
  - Anti-scenarios
  - Multiple scenarios (combined with each other, or even with themselves)
  - Generic scenarios (to be instantiated by different objects of the same class)

"**how** to do something" (specifying behavior)

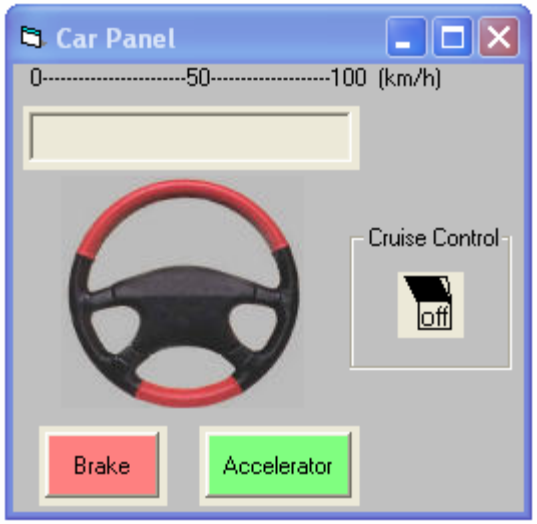
"can do **what**" (specifying tests)

LSCs can tell **vivid stories** of the complex objects interactions across the system

CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

ms

Test Purposes in LSC



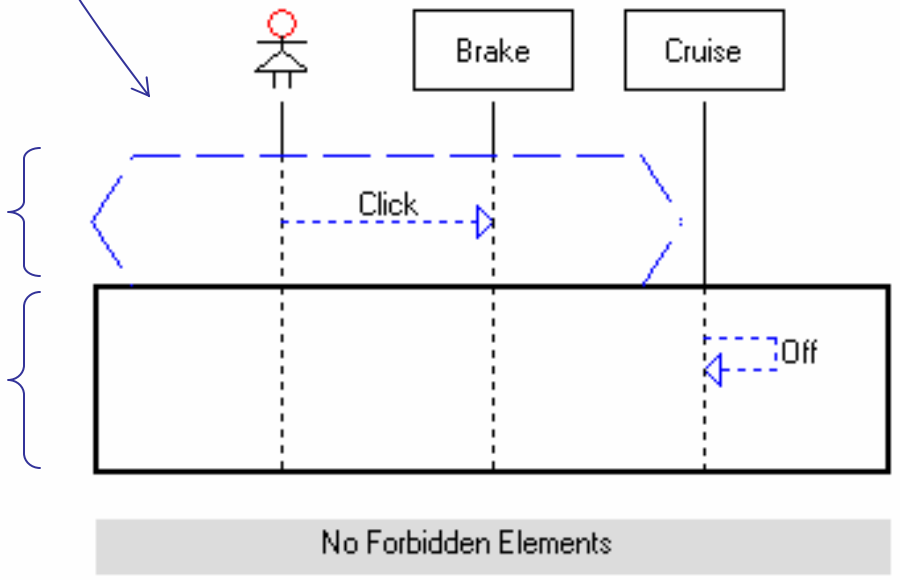
Car Panel

[Harel 04 Play-Engine 4.1.0]

To specify test purpose, one can use **universal** LSCs or **existential** LSCs

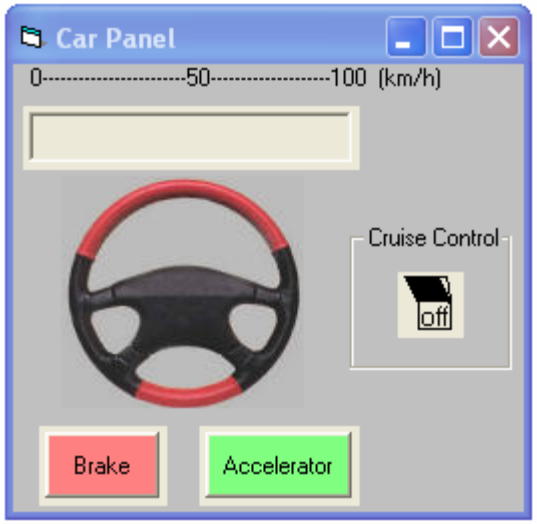
prechart:
"the external stimulus"

main chart:
"the system response"



A "Brake Cruiser" Scenario

Test Purpose in LSC – Describing Timing Constraints

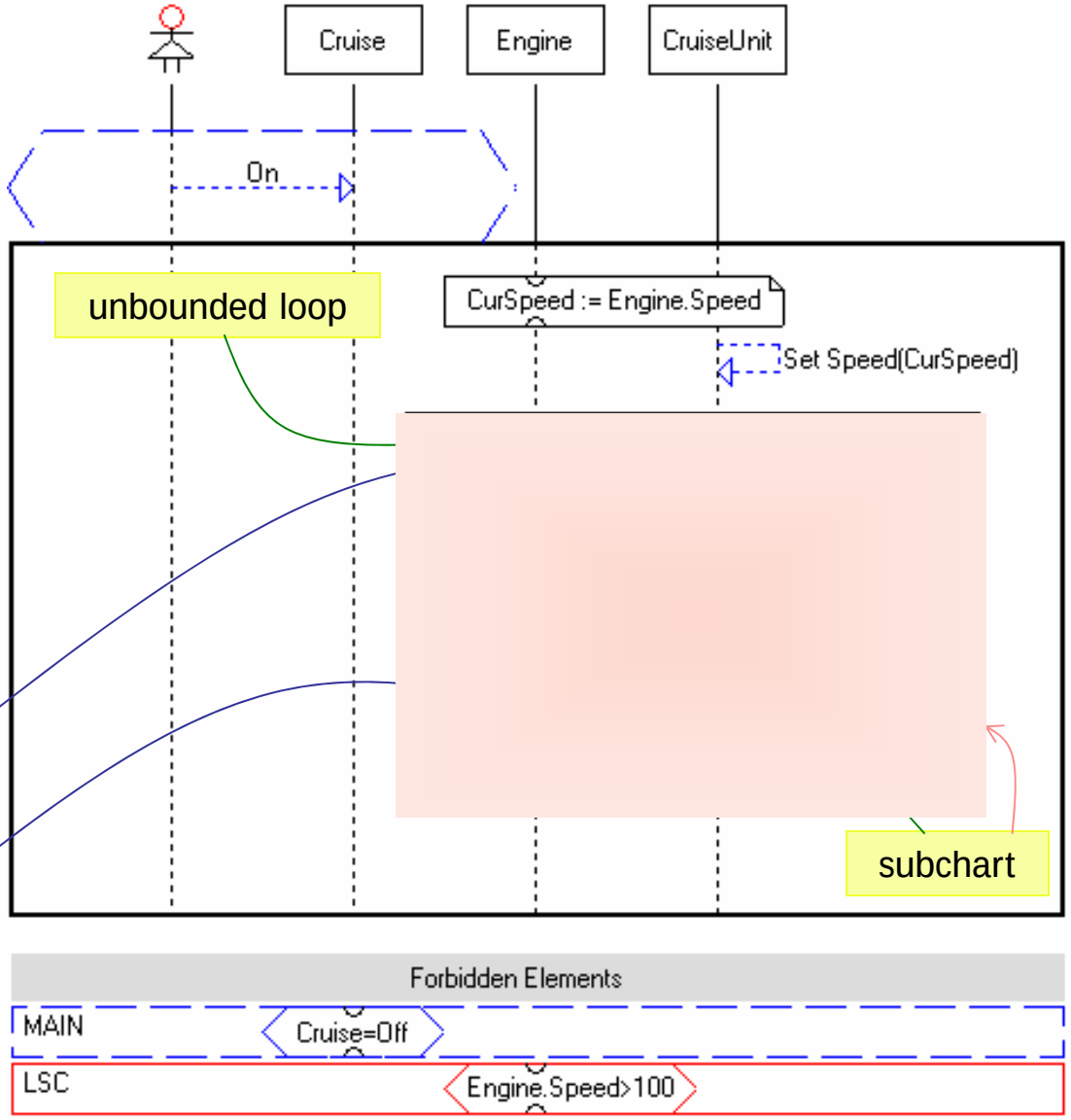


Car Panel

[Harel 04 Play-Engine 4.1.0]

get current time

hot timing constraints



A "Cruise On" Scenario

LSCs as Test Purposes: Benefits (1/2)

- LSCs can specify possible, mandatory and forbidden behavior
  - For example, LSCs can specify “unacceptable behavior pattern” (anti-scenario) which is difficult to describe in terms of MSCs
- Rich language constructs
  - variable, assignment, function, condition, branching, loop, symbolic mechanism, timing constraint, etc.
  - can specify multiple scenarios that combine with each other, or even with themselves (to execute multiple chart copies)

LSCs as Test Purposes: Benefits (2/2)

the **driving** LSCs

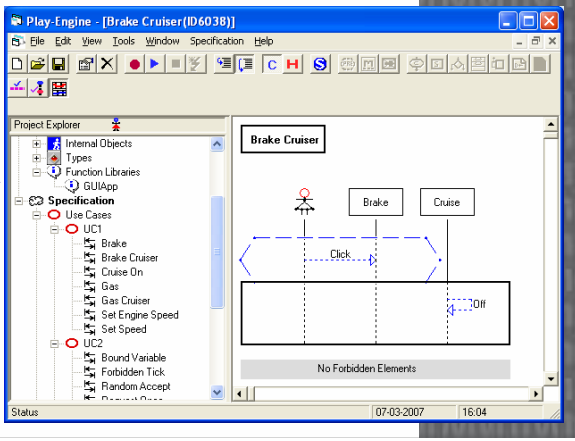
- Early stage requirement validation
  - LSCs can serve as (universal LSCs) as well as (universal and existential LSCs, as test purposes)
  - LSCs lead to synergy between executing (exhibiting) and testing (monitoring) **executable** software specifications

the **monitored** LSCs

- Regression testing
  - The runs that satisfy the existential LSCs (part of the test purpose) are recorded
  - Later, user actions and environment actions of these runs are replayed to verify that existential LSCs are still fulfilled and universal charts not violated

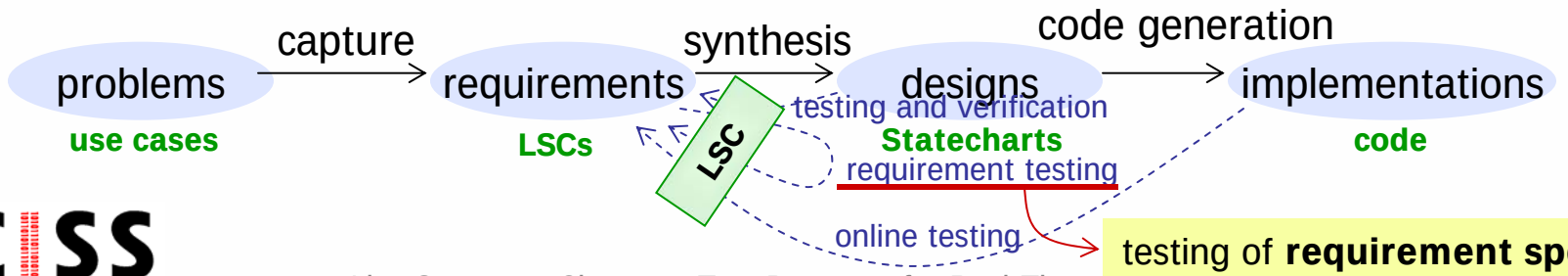
- On-line testing
  - Events from the system under test (SUT) would be received by the Play-Engine through a TCP/IP connection and injected into the exec mechanism
  - The Play-Engine monitor all the LSCs
  - The Play-Engine sends generated events back to SUT

Use the Play-Engine!



Perspectives and Visions (1/3)

- LSCs: inter-object testing and intra-object testing "under the same roof"
  - LSCs can specify properties in coarse grained requirement (scenario) models and fine grained design (behavioral) models
  - Enable iterative, gradual and smooth refinements from specifications to designs
  - ? How to effectively maintain the LSCs during the development and testing process
- Inter-object, scenario-oriented, targeted testing (property-oriented testing at higher level)
  - How to formulate and organize these properties (test purposes) in LSCs (reuse, refactoring, …)
  - Test coverage and selection criteria (considering different chart types and chart modes)
  - Traceability: when the LSCs are transformed into TA, how to trace back

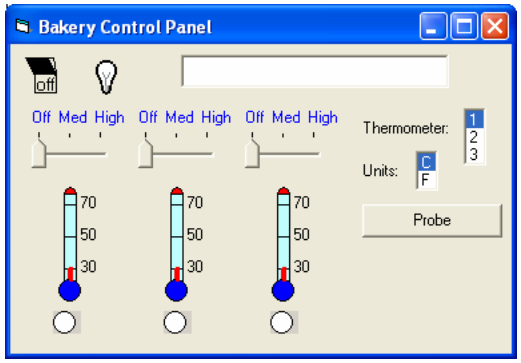


Perspectives and Visions (2/3)

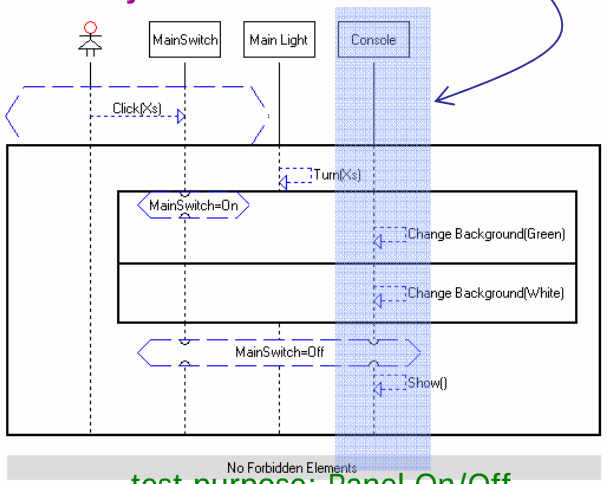
Many details to be considered!

- LSCs projection
  - Project to the system (as a black box), the user, and the environment
    - at early stage: distribution of responsibilities (to determine the boundaries of the system requirements)
    - at integration testing stage: to validate the system behavior from the perspectives of the user, the system, and the environment, respectively
  - Project to each participanting object to test them separately. (during transition from requirements to designs)
  - Project to an arbitrary number of relevant objects to test the interactions among them

- **intra-object** testing



[Harel 04 Play-Engine]



test purpose: Panel On/Off

Perspectives and Visions (3/3)

- Test derivation based on LSCs test purposes
  - Solution #1:
    - LSC-driven state space search of the TA network
    - Existential LSCs should be satisfied by at least one satisfying run
    - Universal LSCs should not be violated
  - Solution #2:
    1. Translate test purpose LSCs into TA (based on Saulius's work)
    2. Make a synchronous product of these TA and the specification (a network of TA)
    3. Identify those accepting states and violating states in the product automaton
    4. Generate (finite length) test cases from the product automaton according to certain selection criteria
    5. Monitoring the execution of the implementation or the specification

conformance testing under the given test purpose