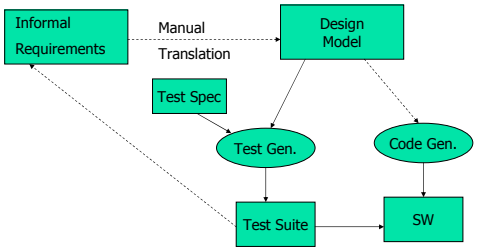
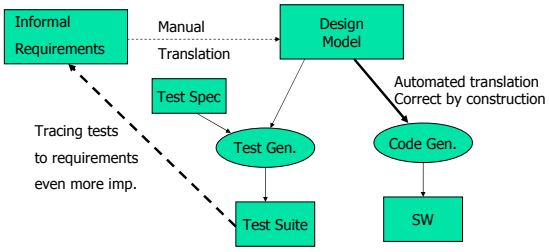
Model Synthesis with MSC based models

Ankit Goel
Abhik Roychoudhury
National University of Singapore

Model-driven SW Development



"Golden" Models



The three layers

- Informal Requirements
  - In Natural Language
- Design Models (Behavioral)
  - Often UML-based in industrial practice
- Software implementation

- Traceability across the layers?

Traceability across the layers

- Debugging of model-driven SW
  - Relate SW lines of code to model elements

- "Debugging" of design models
  - Relating model elements to individual requirements in the requirements doc.

Systematic Model Synthesis

- Not automated
  - Classification of requirements documents
  - Suitability of diff doc. to different modeling formalisms.
  - Towards a guided translation of requirements
    - Safety-critical domains e.g. avionics.

What kind of models?

- Mix of …
  - Class diagrams (Structure)
  - State diagrams (Behavior)
- Different in style from req.
  - Giving system scenarios as Seq. diagrams
  - Global properties (temporal logic)

The dichotomy

- Intra-process style design models
  - State Diagrams
- Requirements focus on inter-process communication
  - System scenarios as Seq. Diagrams
- More manual work in translation
  - Error-prone model construction!

In this talk, focus on …

- … Message Sequence Charts (MSC)
  - MSC based design models
    - Tighter connection with req.
  - Executable
    - Difficulty in test/code gen., otherwise
  - Classifying the manual activity in model construction in a systematic way.

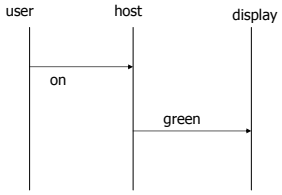
Experience gained from …

- Avionics
  - Flight controller component from NASA CTAS automation tools
- Automotive
  - MOST --- automotive infotainment protocol for devices in car network
- Various examples in rail transportation
  - Rail shuttle system - Paderborn
- Telecommunication
  - Features like call-waiting, call forwarding etc.

Kinds of Requirement Doc.

- Declarative
  - Collection of Global properties
    - "Whenever x sends m to y, y eventually responds with n1, n2, n3"
    - Formalized as temporal logic
- Operational
  - Identify processes in system.
  - Typical system scenarios as MSCs

Possible behavior via MSC



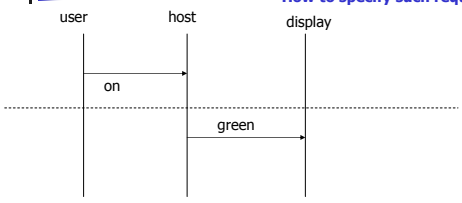
The user may switch on the host following by the host turning the display to green.

MSC-based exec. models

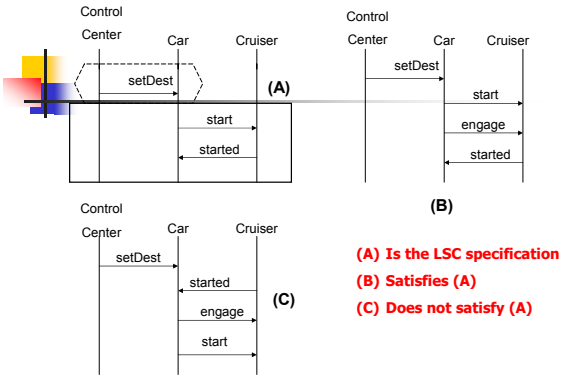
- Recent work, in last decade.
  - Live Sequence Charts (LSC)
    - Damm and Harel – FMSD 2001
    - Triggered Message Sequence Charts (TMSC)
      - Sengupta and Cleaveland – FSE 2002
  - Interacting Process Classes (IPC)
    - Hybrid model, our recent work ICSE 2006
- Match them to req. doc. types.

Illegal Behavior

How to specify such requirements ?



Whenever the user switches on the host , the host must turn the display to green.

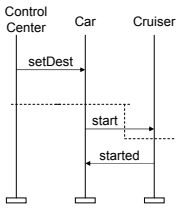


(A) Is the LSC specification

(B) Satisfies (A)

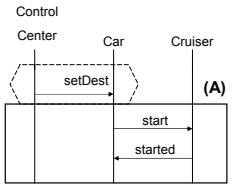
(C) Does not satisfy (A)

TMSC



1. **Trigger and action similar to LSCs.**
2. **Separation between trigger and action is on per-process basis.**
3. **Suitable for asynchronous exec.**

Suitable for Declarative Req



"*Whenever the control center sends a setDest message to a car c, the car sends a start message to the cruiser to which the cruiser acknowledges with a started message.*"

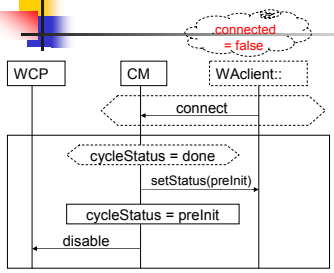
Hypothetical requirement, but this is the style in real documents for very safety critical domains

Amenable to even NLP techniques.

Requirements get converted to indiv. charts, traceability is easy.

NASA CTAS

- Automation tools for managing large volume arrival air traffic in large airports.
  - Final Approach Spacing Tool
    - Determine speed and trajectory of incoming aircrafts on their final approach.
    - Master controller updates weather info. to "clients"
      - **controllers using inputs to compute aircraft trajectories.**
    - **Modeled and simulated the Weather update subsystem from Requirements Document.**



(a) Requirement 2.6.2

Requirement 2.6.2:

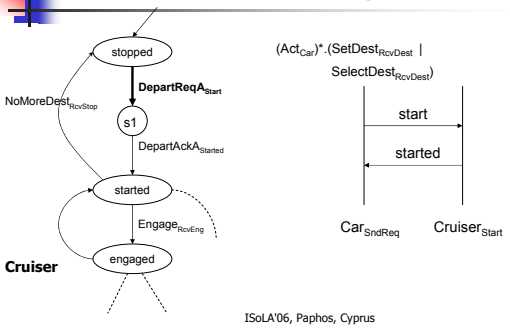
The CM should perform the following actions when ***a weather-aware client*** attempts to establish a socket connection to CM

(a) set the weather-aware client's weather status to `pre-initializing'
(b) set the weather-cycle status to `pre-initializing'
(c) disable the weather control panel

Interacting Process Classes

- Each process as high-level LTS.
- Action in LTS is a transaction
  - Guarded MSC involving several processes.
  - Executed atomically.
  - Any execution trace of the system
    - Sequence of MSCs.
- Hybrid model
  - State-based & MSC-based notations

Two-level Modeling



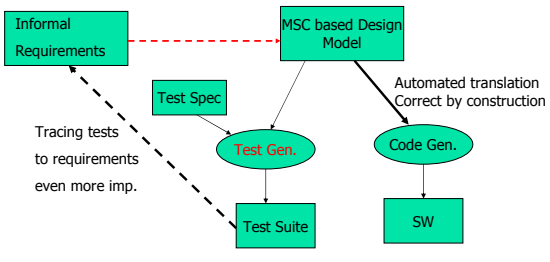
IPC models against req.

- Declarative req. documents (e.g CTAS)
  - Collection of global system properties about event orderings
  - Local flows in a process's high-level LTS needs to be constructed.
- Operational req. documents (e.g. MOST)
  - Overall scenarios as MSCs
  - Break them into MSCs of the IPC model.

IPC models against req.

- Hybrid req. documents
  - Few global system properties about event orderings.
  - Overall system scenarios as MSCs.
  - Examples: medical devices (blood pressure infusion pump)
  - Both IPC and TMSC are suitable.

Back to the overall flow



Test generation

- System Model M
  - Executable MSC based language like IPC
- Test Spec
  - Seq. of MSCs appearing in the system model
- Witness Test Case
  - Seq of MSCs containing test spec as subseq.
- Search
  - Symbolic exec. semantics of M (symbolic in process id.)
- Results --- CTAS, MOST, Rail-Shuttle
  - Another talk!

Wrapping up

- The big picture
  - Push towards scenario-based (MSC based) system models
    - Executable, yet MSC based
    - "Closer" to requirements
- Things done
  - Synthesizing these models from req.
  - Symbolic exec semantics (per class of proc.)
    - ***Efficient simulation, test generation (IPC)***
- Leading towards traceability of tests to req.