Formal Verification of Device State Chart Models

Fulvio Corno, Muhammad Sanaullah
Outline

- Design process
- Formalisms
- Verification Methodology
- Results
- Conclusions
Design process for IE

- Intelligent environments gaining acceptance
  - More installations
  - Standard solutions
- Need more structured design process
  - Less “art”
  - More “engineering”
Reference model

- **Wall switch**
- **Tangible**
- **PC**
- **Smartphone**

**User Interface**

- **Agents**
- **Fuzzy**
- **Rules**
- **Algorithm**

**Intelligence**

- **Access point**
- **Protocols**
- **Gateway**
- **Model**
- **Framework**

**Middleware**

- **Sensor**
- **Meter**
- **Actuator**
- **Bus**
- **Wearable**
- **Wireless**

**Devices**
General Goals

- Adopt **formal representations** to allow a sound design process
- Enable validation and verification **throughout** the design process
- Integrate the solution in the Dog2.1gateway toolset
## Adopted formalisms

<table>
<thead>
<tr>
<th>Level</th>
<th>Design artifact</th>
<th>Technique</th>
<th>Formalism</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>System requirements</td>
<td>Temporal Logics</td>
<td>UCTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intelligence</td>
<td>Intelligent algorithms</td>
<td>State machines</td>
<td>UML Statecharts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Middleware</td>
<td>Device categories</td>
<td>Ontology</td>
<td>DogOnt classes</td>
</tr>
<tr>
<td></td>
<td>System configuration</td>
<td>Ontology</td>
<td>DogOnt instances</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Devices</td>
<td>Device models</td>
<td>State machines</td>
<td>UML Statecharts</td>
</tr>
<tr>
<td></td>
<td>Whole system behavior</td>
<td>Parallel state machines</td>
<td>UML Statecharts</td>
</tr>
</tbody>
</table>
The DogOnt ontology

Formalism
- UCTL
- UML Statecharts
- DogOnt classes
- DogOnt instances
- UML Statecharts
DogOnt instances: DimmerLamp

Inherited from Lamp

- OffStateValue
- OnStateValue

Inherited from Controllable

- LightIntensityState
- LightIntensityStateValue
- StateChangeNotificationFunctionality
- QueryFunctionality
- OnOffFunctionality

Room

SetCommand

LightRegulationFunctionality

stepUpCommand

stepDownCommand

Formalism

- UCTL
- UML Statecharts
- DogOnt classes
- DogOnt instances
- UML Statecharts
Overall system components

…to be continued…
Device modeling

- Ontologies are declarative formalisms: device properties
- For device behavior we need an operational formalism
  - Statecharts (Harel, 1987, now in UML 2.0)
Use cases

- Ontologies are declarative formalisms: device properties
- For device behavior we need an operational formalism
  - Statecharts (Harel, 1987, now in UML 2.0)
- We use Statecharts for
  - Modeling the behavior of each *device* type
  - Implementing the *Intelligent Algorithms* within the gateway
  - Building a *whole-system model* allowing simulation and emulation
- Statecharts have a formal semantics: formal verification is possible
Overall system components

…to be continued…
Overall system components

System Configuration
  Load model
    Gateway
      Sense & Control
        Real devices
      Run
        DogOnt
        Intelligent Algorithms

Device Statechart
  Composition
    Whole Environment Model
      Simulation
    Emulation
      Composition
        Whole System Model
          Simulation

…to be continued…
Temporal logic

- UCTL logic
  - Branching-time
  - State-based and action-based
- Operators
  - Next (X,N)
  - Future (F)
  - Globally (G)
  - All (A)
  - Exists (E)
  - Until (U)
- UMC Model Checker
  - Supports Statecharts as a model

Examples

\[
AG(openRequest(T1))
A[\top \{\neg openRequest(T1)\} U tsDone(T1) \top]
\]

\[
AG(daDoorOpen(DAE.ext))
A[\top \{\neg daDoorOpen(DA.inner)\} U extDoorClosed() \top]
\]
Overall system components

- System requirements
- DogOnt
- Intelligent Algorithms
- Load model
- Sense & Control
- Real devices
- Device Statechart
- Whole Environment Model
- Whole System Model
- Formal Verification Simulation
- Formal Verification Simulation
- Formal Verification
But… (goal of this paper)

- Formal verification relies on the composition of device state charts
- Environment control relies on information in DogOnt device properties
- How to ensure their consistency?
- Solution: use formal verification, too
The problem
The problem

- Naming consistency for states
- Naming consistency for commands
- Naming consistency for notifications
- Acceptance of commands
- Reachability of declared states
- Generation of declared notification
- Range of numeric status variables
Approach

- From DogOnt, extract UCTL properties
- From DogOnt, build a synthetic environment for the device
- Integrate Device State Chart in the synthetic environment
- For every property
  - Run Model checker
Approach

- From DogOnt, extract UCTL properties
- From DogOnt, build a synthetic environment for the device
- Integrate Device Statechart in the synthetic environment
- For every property, run Model checker

Building a closed system model, ready for verification

Diagram:
- Environment
  - Environment Generate Commands (EGC)
  - Environment Receive Notifications (ERN)
- Device
- Model Checking
  - OK
  - ERR
- Device Statechart
- Closed system model
Approach

Example: DimmerLamp generated & verified properties

--- Action Properties
--- the acceptance of all the commands in DSC
  EF {sending(stepDown)} true
  EF {sending(stepUp)} true
  EF {sending(set)} true
  EF {sending(off)} true
  EF {sending(on)} true
---
  EF {accepting (stepDown)} true
  EF {accepting (stepUp)} true
  EF {accepting (set)} true
  EF {accepting (off)} true
  EF {accepting (on)} true

--- the generation of all the notifications in DSC
  EF {sending(stateChanged)} true
  EF {accepting(stateChanged)} true

--- State Properties
--- the reachability of all the states in DSC
  EF (offState)
  EF (onState)
  EF (LightIntensityState)
Experimental Results

- UCTL Model Checker
- Dog2.1 standard device classes
- Device classes verified: 11
- Number of verifies properties: 114
  - Some design errors found and corrected
- CPU time: < 1 sec / property

- Formally validated device statechart library in Dog2.1
Conclusions

- Engineering the Design Process for Intelligent Environments
- Formalisms and tools are needed
- Ontologies, Statecharts, Temporal Logics

http://elite.polito.it
http://domoticdog.sourceforge.net
fulvio.corno@polito.it