Oracle-based Mode-change Propagation in Hierarchical Components.

Tomas Pop, Frantisek Plasil, Matej Outly, Michal Malohlava, Tomas Bures

Tomas.Pop@d3s.mff.cuni.cz

http://d3s.mff.cuni.cz
• Civilization search
Motivation Example

- Civilization search
- **Phases of execution**
  - *Init, Search, TDisc...*
Motivation Example

- Civilization search
- Several *phases of execution*
  - INIT, Search, TDisc...

Corresponding architecture

Architecture variants
Motivation Example

- Civilization search
- Several *phases of execution*
  - INIT, Search, TDisc...

**Application**

<table>
<thead>
<tr>
<th>Wheel drivers (actuators)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left wheel driver</td>
</tr>
<tr>
<td>Right wheel driver</td>
</tr>
</tbody>
</table>

**Brain**

<table>
<thead>
<tr>
<th>Search strategy</th>
<th>Word map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Civilization search library</td>
<td></td>
</tr>
<tr>
<td>Road movement library</td>
<td></td>
</tr>
<tr>
<td>Town discovery library</td>
<td></td>
</tr>
<tr>
<td>Init code</td>
<td></td>
</tr>
</tbody>
</table>

**Sensors**

<table>
<thead>
<tr>
<th>Bumper driver</th>
<th>Simple camera driver</th>
</tr>
</thead>
</table>

- Corresponding architecture
- Architecture variants
Motivation

- Mass production of hardware
  - Cents matter
- Optimization for minimal software resource demands
  - Use only what is needed
  - Allows for deployment on low-cost hardware
  - *Mode switching* at run-time
Problem Statement

- Switching of architecture variants (modes) in component based control systems is challenging
  - Must not be demanding at run-time
  - Mode relevant information is spread among multiple components
    - E.g., state of multiple sensors and other components
  - Handwritten architecture modifications are complex and error prone
Challenges

- Relation specification in respect to composability
- Generation of mode-transition automata for composed system
What Will Be the Seminar About?

- Formal model allowing for an efficient mode switching in hierarchical component system
  - Composability of the mechanism
  - Not polluting existing interfaces
  - Feasible and predictable time and space complexity
• Identify relevant part of state of component
  ▪ Set of *properties*
• Model relations between these
  ▪ Make it composable in the hierarchy
• Determine active sub-components based on mode (property of the parent component)
• Generate mode transition automata for the whole application (at deployment time)
Proposed Approach - Property networks

- Component is associated with a set of properties

Diagram:

Component $C_1$ is associated with a set of properties. The component $C_3$ is further associated with properties $p_1$, $p_2$, $p_3$, and $p_4$. The component $C_2$ is associated with property $p_5$. The propagation of properties is indicated by the dotted lines $b_1$ and $b_2$.
Components are associated with a set of properties.

Property values are bounded by relations:
- Directed graph
- Propagation

Local change has an impact “where relevant”
Each component has a *mode* property

*Component mode* is

- Set of *child* subcomponents used
- Component attributes
  - e.g. real-time scheduling info

Local change of property value

- Is propagated by the network
- Influence valuation of properties where relevant
  - Also the ones representing mode
- Determines new application architecture

Change propagation possibly demanding

- But can be pre-computed to an Oracle at deployment time
Proposed Approach - Property networks

Interpretation
- $C_1$ - cApplication
- $C_2$ – cBrain
- $C_3$ – cSearchStrategy
- $p_4$ – mode$_{Brain}$
- $p_5$ – groundType
- $p_6$ – mode$_{SearchStrategy}$

$\begin{align*}
f_3 : p_4 &= \begin{cases} 
X & p_3 = B \lor p_2 = 1 \\
Y & p_3 = C \land p_2 = 0 
\end{cases}
\end{align*}$
Run is started

- **Until the network stabilize**
  - No further changes...

After a Change...

```
Tomas Pop, CBSE 2012
Oracle-based Mode-change Propagation in Hierarchical Components
```
Network Run Stabilization

\[ p_1 \xrightarrow{f_1 : Id} p_3 \]
\[ p_2 \xrightarrow{f_2 : Id} p_1 \]

\[ D(p_i) = \{0, 1\} \]
Id : Identity
Neg : negation

\[ p_1 \xrightarrow{f_1} p_2 \]
\[ p_2 \xrightarrow{f_2} p_1 \]

a) will stabilize
b) will stabilize
c) will not stabilize

\[ p_1 \xrightarrow{f_1} p_2 \]
\[ p_2 \xrightarrow{f_2} p_1 \]
\[ f_1 \]
\[ f_2 \]

a)

\[ p_1 \xrightarrow{f_1} p_2 \]
\[ p_2 \xrightarrow{f_2} p_1 \]
\[ f_1 \]
\[ f_2 \]

b) \( k = 2, j = 4, l \in \{0, 1, 2\} \)

\[ p_1 \xrightarrow{f_1} p_2 \]
\[ p_2 \xrightarrow{f_2} p_1 \]
\[ f_1 \]
\[ f_2 \]

\[ p_1 \xrightarrow{f_1} p_2 \]
\[ p_2 \xrightarrow{f_2} p_1 \]
\[ f_1 \]
\[ f_2 \]

\[ p_1 \xrightarrow{f_1} p_2 \]
\[ p_2 \xrightarrow{f_2} p_1 \]
\[ f_1 \]
\[ f_2 \]

\[ p_1 \xrightarrow{f_1} p_2 \]
\[ p_2 \xrightarrow{f_2} p_1 \]
\[ f_1 \]
\[ f_2 \]
Triggering Event and Network Run

- Event Property value assignment from component code
  - E.g., color seen by a camera changes
- The network propagates the event until it stabilizes (*network run*)
  - Relation with changed inputs is reevaluated
    - Similar to shifting tokens in Petri Nets
  - Stabilization - no further changes of property valuations
  - If not, bad relation design (*oscillation*)
    - Checked by development tools
Worst Case Property Network Run Length

- Exponential in the number of properties
  - Possibly all the possible valuations
- Knowing initial state and all the possible events, all the possible runs can pre-computed at deployment time
  - Oracle (mode transition automata)
  - At run-time, the oracle is queried in a constant time
Oracle-based Mode-change Propagation in Hierarchical Components

Events

- $e_1: p_5 \leftarrow A$
- $e_2: p_5 \leftarrow B$
- $e_3: p_5 \leftarrow C$

Functions

- $f_1: p_2 = \begin{cases} 0 & p_4 \neq Y \\ 1 & p_1 = 1 \land p_4 = Y \end{cases}$
- $f_3: p_4 = \begin{cases} X & p_3 = B \lor p_2 = 1 \\ Y & p_3 = C \land p_2 = 0 \\ f_2, f_4, f_5 : Identity \end{cases}$
At Run-time - Overview

- Component can change value of a property that is associated with it via API

- After each change
  - The oracle is queried for new valuation of all properties
  - The valuation is changed
  - Runtime framework takes responsibility for modifications of the architecture according to oracle hints
Implementation and Experience

- Introduction to existing SOFA-HI component system
  - takes a feasible effort
- Tools for network relations design necessary
  - Checking for well-formed property networks
    - No oscillation
- Model seems to be intuitive
  - In the case study, 13 properties
Overview

- Property networks allow for highly deterministic architectural modifications
  - Pre-computed oracle
  - New variant in constant time
  - Oscillation is not possible at runtime
- Time and space complexity is feasible
  - Constant time and 2kB for presented case-study
- Does not pollute interfaces and business logic
  - It is orthogonal to them
Thank You, Questions