Domain-Specific Language Architecture for Automation Systems: An Industrial Case Study

Christopher Preschern
Andrea Leitner
Christian Kreiner
Agenda

• My field of Research
• Motivation of the Paper
• What is PISCAS?
• PISCAS Demo
• PISCAS Architecture
• Results
• Conclusion
My field of research

• Embedded System Architectures
  – Safety
  – Security

• Architectural Patterns
  – Effect on Safety/Security Certification

→ this paper is based on results on results of my master’s thesis
Motivation

• Suggest design decisions on how to easily develop DSLs for automation systems
• Show the application and the consequences of these design decisions
What is PISCAS?

- **PISCiculture Automation System**
  - www.piscas.eu

- Product Line approach to develop fish farm automation systems
  - MetaEdit+ for DSL development
  - B&R automation devices
What is PISCAS?

- PISCAS Model
- Automation Software
  - Hardware mapping
  - IEC 61131 source code
  - Visualization
- Configuration Files
  - Web portal SQL files
  - Network device configuration
- Documentation
  - Graphical overview
  - Wiring plan
  - List of parts
  - Labels for wiring closet
PISCAS Demo

- Documentation
- Automation Software
- Web Portal
Fish farm automation domain description

• Elements: Feeders, oxygen level control, water level supervision + alarm system

• Fish farms mostly differ in their arrangement/amount of devices

• Device types change rather often

• Main functionality stays the same
## PISCAS Architecture

<table>
<thead>
<tr>
<th>Physical system</th>
<th>GOPPRRR concepts</th>
<th>Automation software</th>
</tr>
</thead>
<tbody>
<tr>
<td>automation plant</td>
<td><em>Graph</em></td>
<td>overall software</td>
</tr>
<tr>
<td>device</td>
<td><em>Object</em></td>
<td>function block</td>
</tr>
<tr>
<td>wire</td>
<td><em>Relationship</em></td>
<td>connecting function block interface variables</td>
</tr>
<tr>
<td>-</td>
<td><em>Role</em></td>
<td>-</td>
</tr>
<tr>
<td>wire connection</td>
<td><em>Port</em></td>
<td>function block interface variables</td>
</tr>
<tr>
<td>device attribute</td>
<td><em>Property</em></td>
<td>function block parameters</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Graph**
  - **Properties**
    - Name: Switch1
    - Voltage: 230V
  - **Object**
  - **Port**
  - **Relationship**
  - **Role**

*Christopher Preschern* 03.07.2012
PISCAS Architecture

Automation Source Code

Switch-Component
Switch[1].output := Switch[1].input;

Mapping
Light[1].input := Switch[1].output

Light-Component
IF Light[1].input = TRUE THEN
Light[1].output := TRUE;
END_IF

Domain Specific Language

Output Port

Hardware Devices

Input Port
PISCAS Architecture
PISCAS Design Decisions

• Abstract Object / Abstract Relationship
• Elements connected in Pipes&Filters style
• Direct representation of physical automation devices as Objects
• Explicit modeling of I/O modules
Benefits of the PISCAS Architecture

• Easy generation of documentation, visualization and automation software mapping

• Intuitive modeling for automation domain expert and fish farm expert
Final Remarks

• Provide guidelines for automation DSL development
  – benefits for visualization/documentation generation
  – benefits for application modeling
  – benefits for automation system maintenance

• Two PISCAS systems in operation
  – hardware installation plan was very useful
  – low software maintenance effort
Thank you very much for your attention

Any questions?
## PISCAS Application Modeling

<table>
<thead>
<tr>
<th></th>
<th>Fish Farm element modeling (ponds, switches, lights, ...)</th>
<th>B&amp;R I/O module modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fish farm A</td>
<td>2h</td>
<td>3h</td>
</tr>
<tr>
<td>Fish farm B</td>
<td>1h</td>
<td>1.5h</td>
</tr>
<tr>
<td>Add new components to B (model approximately doubled)</td>
<td>1h</td>
<td>2h</td>
</tr>
</tbody>
</table>
DSL Complexity

\[ C_{\text{interface}} = n_{\text{Relationships}} + n_{\text{Roles}} + n_{\text{Constraints}} \]

\[ C'_{\text{element}} = n_{\text{Objects}} + n_{\text{Ports}} \]

\[ C_{\text{properties}} = n_{\text{Properties}} \]
Code Generator Dependence

\[ D_{\text{interface}} = \#LOC_{\text{Relationships}} + \#LOC_{\text{Roles}} + \#LOC_{\text{Constraints}} \]
\[ D_{\text{element}} = \#LOC_{\text{Objects}} + \#LOC_{\text{Ports}} \]
\[ D_{\text{properties}} = \#LOC_{\text{Properties}} \]