Foucault's Pendulum in the Distributed Control Lab

WORDS 2003F
Capri Island - Italy
Andreas Rasche, Peter Tröger, Michael Diriska and Andreas Polze
Outline

- Motivation
- The Distributed Control Lab (DCL) Architecture
- Foucault's Pendulum Details
- Dynamic Reconfiguration as safe-guard mechanism
- Evaluation of the approach
- Experiments in the DCL
- Conclusions
Motivation

- Online access to physical experiments over the Web
- Test-bed for interconnected middleware-components and embedded systems
- Reach a predictable system behaviour in unstable environments
- Study techniques to prevent malicious code damaging physical equipment
- Foucault's Pendulum demonstrates usage of dynamic reconfiguration for online replacement of user control
Distributed Control Lab

- 2001 project start at Hasso-Plattner-Institute
- Practice of writing control algorithms for real-time control problems
- Study of system predictability, availability and security in context of middleware-based dynamic control systems
- Extensible architecture for hosting physical control experiments
- Investigation of algorithms for user code observation and replacement of control components
- Experiment: physical installation and specific control software
The Distributed Control Lab

[Diagram showing the components and interactions of the Distributed Control Lab, including Web Frontend, Windows CE Client, Command Line Client, Experiment Service, Result Management, Experiment Management, Foucault's Pendulum, Control of Lego Robots, and High Striker Realtime and Windows CE.]
The following jobs are registered for your account in the system:

<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>State</th>
<th>Queued on</th>
<th>View details</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pendulum (Owner: Andreas Rasche)</td>
<td>HPI</td>
<td>Currently executed</td>
<td>Tue, 23 Sep 2003 10:51:27 GMT</td>
<td>View details</td>
<td>Cancel</td>
</tr>
<tr>
<td>Lego_Robots (Owner: Stefan Henze)</td>
<td>HPI</td>
<td>Finished (result code: StopSignal), 3 results available</td>
<td>Fri, 19 Sep 2003 16:59:36 GMT</td>
<td>View details</td>
<td>Delete</td>
</tr>
<tr>
<td>Lego_Robots (Owner: Lego Test Account)</td>
<td>HPI</td>
<td>Finished (result code: StopSignal), 3 results available</td>
<td>Fri, 19 Sep 2003 14:33:33 GMT</td>
<td>View details</td>
<td>Delete</td>
</tr>
<tr>
<td>Lego Simulator (Owner: Andreas Rasche)</td>
<td>HPI</td>
<td>Finished (result code: Successful), 4 results available</td>
<td>Fri, 19 Sep 2003 12:45:32 GMT</td>
<td>View details</td>
<td>Delete</td>
</tr>
<tr>
<td>Lego_Robots (Owner: Stefan Henze)</td>
<td>HPI</td>
<td>Finished (result code: StopSignal), 3 results available</td>
<td>Fri, 19 Sep 2003 11:18:49 GMT</td>
<td>View details</td>
<td>Delete</td>
</tr>
<tr>
<td>Lego_Robots (Owner: Andreas Rasche)</td>
<td>HPI</td>
<td>Finished (result code: StopSignal), 3 results available</td>
<td>Thu, 18 Sep 2003 16:28:56 GMT</td>
<td>View details</td>
<td>Delete</td>
</tr>
<tr>
<td>Lego_Robots (Owner: Andreas Rasche)</td>
<td>HPI</td>
<td>FinishExperiment crashed</td>
<td>Thu, 18 Sep 2003 16:27:35 GMT</td>
<td>View details</td>
<td>Delete</td>
</tr>
<tr>
<td>Lego_Robots (Owner: Andreas Rasche)</td>
<td>HPI</td>
<td>Finished (result code: StopSignal), 3 results available</td>
<td>Thu, 18 Sep 2003 16:22:24 GMT</td>
<td>View details</td>
<td>Delete</td>
</tr>
<tr>
<td>Lego_Robots (Owner: Andreas Rasche)</td>
<td>HPI</td>
<td>Finished (result code: Failed), 2 results available</td>
<td>Thu, 18 Sep 2003 16:20:50 GMT</td>
<td>View details</td>
<td>Delete</td>
</tr>
<tr>
<td>Lego_Robots (Owner: Andreas Rasche)</td>
<td>HPI</td>
<td>Finished (result code: StopSignal), 3 results available</td>
<td>Thu, 18 Sep 2003 16:15:58 GMT</td>
<td>View details</td>
<td>Delete</td>
</tr>
</tbody>
</table>
Problem: Malicious Code

- Investigation of Solution for malicious code detection
  - Source Code Analysis
  - Language limitations / special compiler
  - Simulation before execution on physical experiment
  - **Analytic Redundancy**
    - Online observation of user programs
    - Replacement of user programs before reach of uncontrollable state
    - Dynamic Reconfiguration of component-based control application
    - Monitoring of environmental settings and component states
Foucault’s Pendulum

- First installation 1848 by Leon Foucault in the Pantheon in Paris
- Demonstrates earth rotation
- Today many installation including one in UN-building in New York
- Problem: Pendulum must be kept swinging
- Solution: electro magnet under an iron ball
- Experiment: Find best control algorithm to keep the pendulum swinging
  - Using minimum energy
  - Reaching the highest amplitude
The Pendulum Experiment

You can enter here the program to steer the magnet, which is situated under the pendulum. The necessary programming details are explained in this documentation (german).

You can use one of the following code examples:

```c
while(true)
{
    // Peak for Next Event
    se=pendel.GetNext();
    // New Event ?
    if(se!="null")
    {
        // First time at this place ?
        if(last=="null") last=se;
        // Rugel tritt ein
    }
}
```

Upload your code file:
You can use the download link for viewing or saving the source data of this diagram.
Pendulum Control – detailed

- Laser light barriers sampled into 4KByte FIFO-memories with 23.4 kHz
- USB-Controller checks half-full-Flag
- 64 Byte blocks of data transferred via USB 1.1
- Real-Time OS-Threads process incoming signals / produce out-going bit stream
Our Approach: Dynamic Reconfiguration as safe-guard mechanism

- Mapping of profiles to application configurations based on environmental conditions and component states
- XML-based description of
  - observer: monitoring of environmental settings and component states
  - profiles: mapping of environmental conditions to application configurations
  - configurations of component-based applications
- Online monitoring of environment and components
- Change of application configuration using dynamic reconfiguration if required
- Configuration Manager instantiates, sets attributes, connects and starts components — performs reconfiguration
- Dynamic reconfiguration based on XML-based configuration description using an algorithm of M. Wermelinger
  - Based on blocking of connections between components
Pendulum Experiment
Control Configurations

Configuration 1: safety controller

Configuration 2: user program (cold standby)

Configuration 3: user program (warm standby)
Measurements:
Abnormal Termination of User Program
Future Work

- “Higher Striker” – Real Time Control Experiment
  - Running on a RTOS (Ce.Net, eCos)
  - Small buffers possible / short delay
  - Parallel Port I/O
  - Sampling rate 38 kHz
- Port of .Net Environment (parts) to Lego-Robots
- Additional experiments / simulations
Related Work

- “Verbund Virtuelles Labor” project at University Reutlingen / Germany
- iLab project (WebLab) at MIT
- Virtual Lab at University of Hagen / Germany
- Tele-Laboratory at University of Pisa
- Tele-Lab / Simplex architecture
Conclusions

- DCL: environment for remote experiment access based on COTS Operating System and Middleware
- Safety against malicious code demonstrated
- Analytic Redundancy / Runtime observation of user control at Foucault’s Pendulum applicable
- Replacement of faulty control algorithms using dynamic reconfiguration
- Measured reconfiguration times highly acceptable for control applications