Configuration and Dynamic Reconfiguration of Component-based Applications with Microsoft .NET

ISORC 2003
Hakodate - Japan
Andreas Rasche and Andreas Polze
Outline

- Motivation
- Configuration framework for component-based applications
- Algorithm for dynamic reconfiguration
- Making components configurable using Aspect-Oriented Programming (AOP)
- Measurements
- Current Research: Distributed Control Lab
- Conclusions
Motivation

- Predictable end-to-end availability of services
- Mobile devices require application adaptability
- **Dynamic reconfiguration** provides a powerful mechanism to adapt **component-based distributed applications** to changing environmental conditions
- Evaluation of reconfiguration times in .NET
Our Approach: Adaptive Software using Dynamic Reconfiguration

- Mapping of profiles to application configurations based on environmental conditions
- Selection of application configuration according to conditions provides best service for a given situation
- Definition of
  - observer: monitoring of environmental settings
  - profiles: mapping of environmental conditions to application configurations
  - configurations of component-based applications
- Online monitoring of environment
- Change of application configuration using dynamic reconfiguration if required (changed conditions)
Description of configurations of component-based applications

- "A Configuration of a component-based application denotes the set of its parameterized components and the connections among them."
- XML-based description language
- Configuration Description: List of components, their attributes, and connectors
- Support for a variety of component connectors
Configuration Framework

- **Configuration Manager**
  - Selects matching app configuration based on observed conditions and corresponding XML-configuration description
  - Instantiates/queries defined observers
  - Realizes distributed object activation
  - Enables adaptation of distributed applications using dynamic reconfiguration if required

- **Standard reusable Observer-components**
  - Network Bandwidth, CPU Power, Memory Consumption

- **Components provide hooks for configuration management**
  - Interface `IConfigure` must be implemented – can be automated
Architecture for Adaptive Systems
Our Reconfiguration Algorithm

- M. Wermelinger, J. Magee / J. Kramer
- Applications follow Actor Execution Model by G. Agha
  - Application consists of interconnected components
  - State of components changes only through interactions with other components
- Transaction Concept
  - Sequence of message exchanges over one connection
  - Initiator of a transaction is informed about its completion
  - Finishes in finite time
- Model matches wide range of typical applications
  - Including Client/Server-style applications
Dynamic Reconfiguration - Steps

- Start, Parameterization of new components
- Turn application into reconfigurable state
  - No pending requests
  - Block all connections involved in reconfiguration
    - Prohibit new transactions over identified connections
    - Wait for all ongoing transactions to complete
      - Blocking has to be ordered because of dependent transactions
- Parameterization of changed components
- Reconnect/Start all components
- Remove old components
Configuration – a cross-cutting concern (AOP)

- Additional configuration-specific code has to be added to involved components
  - Handling-/Blocking Transactions
  - Start / Stop of component processing
  - Connection handling
  - Implementation of the IConfigure interface

- This code cross-cuts functional component code!
- We use aspect-oriented programming for automatic addition of non-functional configuration specific code

- **Usage of LOOM.Net – Aspect Weaver for .NET**
  - based on (binary) components
Making a Component Configurable

- Automatic implementation of configuration hooks
- Component programmer only has to mark transactions and provide access to connection references
Evaluation : Reconfiguration in the .NET Environment

- Standard PCs : 1GHz PIII – 256 MB RAM
- 100 Mbit/s LAN
- .NET Remoting communication using binary channels
- .NET Framework 1.0 SP1 / Windows 2000 SP3
- reconfiguration of a component attribute
- distributed application
- pure .Net Remoting components
- no separate process
addition of a component with separate process

- **blackout time**
- **reconfiguration time**

**distribution of measured values**

**duration in ms**

18 19 20 21 22 23 540 560 580 600 620 640
Using dynamic Reconfiguration for Fault-tolerance / Security

- Current Research: Distributed Control Lab (DCL)
  - Online lab for distributed robotics and control experiments
- Problem: malicious code can damage hardware
- Solution: dynamic reconfiguration of component-based control application to replace user code
- Configuration framework as safeguard mechanism
- Experiment: Control of Foucault’s Pendulum
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:

```java
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: [Browse] [Upload]

Start Job
### State Flow

**Measurement result**

- **Ground Speed** (Min: 0.4933106, Max: 1.025359)
- **Used Energy** (Min: 0.322648, Max: 54.33134)

You can use the download link for viewing or saving the source data of this diagram.
Pendulum Experiment
Control Configurations

Configuration 1: safety controller
- Safety Controller

Configuration 2: user program (cold standby)
- Safety Controller
- User Program

Configuration 3: user program (warm standby)
- Safety Controller
- Proxy
- User Program
Distributed Control Lab
Ongoing Work

- Detailed publication about pendulum and DCL architecture follows
- Control of Lego Mindstorm Robots
- Cooperation with University of Pisa / Italy
- High Striker / Real-time and Windows CE
- Model Railway Control Application

- ADAPT.NET - Adaptation framework for distributed component-based .NET applications including dynamic reconfiguration and object migration
Related Work

- Original work by M. Wermelinger provides theoretical foundation
- Some systems handle adaptation especially for mobile devices
  - DACIA: relocation, replication and replacement of components
  - Odyssey: application aware adaptation
  - Oreizy: architecture based application adaptation
  - K. Nahrstedt et al.: middleware extension for adaptation based on fuzzy logic
Conclusions

- We have implemented and evaluated our Dynamic Reconfiguration Framework
- Reconfiguration times are highly acceptable for adaptation in mobile systems
- .NET environment provides sound basis for dynamic reconfiguration
- Applicable to a wide range of scenarios
  - Current focus on secure control systems in unsafe environments
About the authors - Additional Information

- **Prof. Dr. rer. nat. Andreas Polze**
  - Professor at the Operating System & Middleware group at Hasso-Plattner-Institute for Software Systems Engineering: University of Potsdam / Germany

- **Dipl-Inf. Andreas Rasche**
  - Diploma Humboldt University of Berlin
  - Since 2002 Research Assistant Operating System & Middleware chair at HPI

- Additional Information at: [www.dcl.hpi.uni-potsdam.de](http://www.dcl.hpi.uni-potsdam.de)
  - Download LOOM.Net
  - Distributed Control Lab