Execution and Development of Adaptive Component-based Applications

Verteilte Systeme Kolloquium
Universität Kassel

Dipl.-Inf. Andreas Rasche
Operating Systems & Middleware Group
Prof. Dr. Andreas Polze
Hasso-Plattner-Institute
University of Potsdam
About the speaker

- Aug. 2006: Research visit at Blekinge Techniska Högskola (3 month)
- Since Oct 2002: research assistance at the „Operating Systems and Middleware“ group of Prof. Dr. Andreas Polze
- 1997-2002: Study of Computer Science, Humboldt University, Berlin
- Teaching activities:
  - Operating Systems for Embedded Computing
  - Programming Embedded Systems at BTH
  - Komponentenprogrammierung und Middleware
  - Seminar Nicht-funktionale Eigenschaften Eingebetteter Systeme
  - Seminar Aspektorientierte Programmierung
  - Seminar Fehlertolerante Systeme
- http://www.dcl.hpi.uni-potsdam.de/people/rasche/
- andreas.rasche@hpi.uni-potsdam.de
Outline

- Adaptive applications using alternative application configurations
- Dynamic reconfiguration in component-platforms (Java/.NET)
  - Reaching a reconfigurable state
  - Migration & dynamic deployment
  - Dynamic update and state transfer
  - Performance evaluation & demo
- AOP tools for generating (re-)configuration specific logic
- Case study: adaptive control applications in a remote lab
- Conclusions
Adaptive Component-based Applications

- Varying resources and context parameters demand adaptation
- Requirement: keep application properties (app.-level QoS) in user-desired range
- Components are units of deployment that can be composed by a third party
- Same interfaces can be implemented by multiple components having different properties
- Different combinations of components (configuration) can fulfill functional requirements of an application
- Applications can be composed for different usage situations
- **Solution**: Selection and activation of appropriate configuration for given environmental properties allow for adaptation
- **Challenge**: Integrate dynamic reconfiguration in component platforms
Capsules – Components at Runtime

- A capsule logically groups a set of objects
- Each object has a type
- Each type is defined in a component
- Each component has a version

- Configuration specific logic

- Root objects
- Capsule objects
- Primitive types (string, int, byte)
- Internal references
- External references

- Application threads

- Other capsules
The Adapt.Net Configuration Infrastructure

Monitoring

Adaptation Engine

Configuration Manager

XML-Adaptation Profile

XML-Configuration Description

IConfigure

parameter
Application configurations

```
<configuration configurationname="cl">
    <capsule name="Viewer" args="" loadStrategy="OBJECT"
        assembly="MessageView.dll" assemblyVersion="1.2.17.42"
        mainType="AdaptNet.ConfiguredObjectProxy.MessageView" location="localhost">
        <port name="m_source" type="OUT" vartype="IMessageSource"/>
        <port name="default" type="IN" vartype="System.Object"/>
    </capsule>
    <capsule name="Source" args="" loadStrategy="CorbaCapsule"
        assembly="MessageView.dll" assemblyVersion="1.2.17.42"
        mainType="AdaptNet.ConfiguredObjectProxy.MessageSource" location="localhost">
        <port name="default" type="IN" vartype="System.Object"/>
        <parameter name="encoding" value="UTF8" type="string"/>
    </capsule>
    <connector sourcecapsule="Viewer" sourceattribute="m_source"
        sinkcapsule="Source" sinkinterface="default" type="IIOP"/>
</configuration>
```
Adaptation through dynamic reconfiguration

- Dynamic reconfiguration includes:
  - Addition/removal of capsules
  - Changing capsule parameter
  - Migration(new location)/ updating (new logic) capsules
  - Changing connections between capsules

- Reconfiguration actions must remain consistency
  - No method execution during updates
  - No execution at capsules with unconnected sink capsules

![Diagram of capsule connections]

(1) A → C
(2) A → B → C
(3) A → B
(4) A → C
Reaching a reconfigurable state

- A capsule is reconfigurable if there is **no on-going method execution of capsules’ objects on any threads’ stack**!

- A reconfigurable state can be reached by:
  - Blocking new method calls from threads and other capsules
  - Waiting for all ongoing method calls to complete

- Acyclic graphs: connections can be blocked orderly

- Cyclic graphs: single threads must be blocked

- Reader-Writer-Locks for synchronization
  - Read-Lock is acquired for each normal method call
  - Write-Lock is acquired by the update logic
  - Usage of recursive locks for recursive calls
Reconfiguration of Distributed Applications

- RW-Locks in .NET- and Java-platform do not work distributed

- **Problem**: When blocking a thread it must not have on-going method calls on involved capsules

- **Solution**: logical thread-IDs and counters
  - Counter per capsule with on-going methods per thread
  - Update counter when entering/leaving a capsule via a root-object
  - During blocking: threads with no on-going method on involved capsules (counter in all capsule is zero) can be blocked
Reconfiguration of Distributed Applications

- RW-Locks in .NET- and Java-platform do not work distributed

- **Problem:** When blocking a thread it must not have on-going method calls on involved capsules

- **Solution:** logical thread-IDs and counters
  - Counter per capsule with on-going methods per thread
  - Update counter when entering/leaving a capsule via a root-object
  - During blocking: threads with no on-going method on involved capsules (counter in all capsule is zero) can be blocked
Application-specific synchronization

- In case of synchronization among application threads the algorithm must be extended.
- All capsules on a path between involved capsules (the block-set) are added to the block-set.

**Diagram:**

1. ID 1
2. lock(R)
3. continue with R locked
4. context switch
5. ID 2
6. try to lock R
7. K1
8. K2 update

- K1
- K2 update
- K3 update
- R
Component Migration & Dynamic Deployment
C from H1 to H2

1. Read Configuration
2. Analyze File Dependencies
3. Transfer Files
4. Create C
5. Block Connections

Configuration XML-Description
Component Migration & Dynamic Deployment
C from H1 to H2

Host 1

A

Save State

Configuration Manager

Host 2

B

Restore State

Configuration Manager

Transfer State

6

Save State

9

Reconnect C

7

Transfer State

8

Restore State

Configuration Manager

10

Restart Processing

11

Remove old C

1
Dynamic Updates

- Complex reconfiguration operation
  - activation of new code (and data layout)

- Capsules have to be updated dynamically to:
  - Activate more appropriate algorithms at runtime
  - Integrate bug-fixed versions (remove security vulnerabilities)
  - Change graphical representation of adapted architecture

- Classes cannot be exchanged directly (in Java/.NET)
  - New versions of objects must be created
  - State must be transferred from old to new version

- Algorithm for reaching reconfigurable state used to apply update atomically
Traversing the Object Graph

- Start from all root objects
- For each field of all objects traverse all references
- In case of an update:
  - Create an instance of the new version
  - Copy the state by transferring all fields from the old to the new instance
  - For reference fields: traverse target first and install potential new version afterwards
- Usage of Reflection (GetFields, Set-/GetValue)

```java
MyObject temp = oldObj.GetValue("Weight");
newObj.SetValue("Weight", temp);
```
Traversing the Object Graph II

- Cycle recognition (visited nodes)
- Creation of new types (no constructor execution)
- Dynamic assembly loading (shadow copies)
- Arrays (update type and content)
- Delegates (update target and method)
- Generics (update bound types)
- Type and assembly objects
- Activation/deactivation/update of aspects
- State transformation for changed data layout
AOP tools and (re-)configuration specific logic

- Synchronization logic for dynamic reconfiguration
  - Management of capsules’ counters
  - Blocking of threads

- Implementation of component’s configuration interface
  - Set-up of communication connections
  - Parameterization
  - Initiation of blocking for dynamic reconfiguration
  - State transfer for migration and dynamic updates

- New programming model for marking connection endpoints and parameters

```java
public class Filter{
    [Parameter]
    int compression;
    [Connection]
    IStream sink;
}
Tool Support for Adaptive Applications
Tool Support for Adaptive Applications
Adaptation Profiles

![Image of Adaptation Profile Creator 1.0 software interface]
DEMO: PictureShow

- Small picture viewer implemented in C#
- Update statistics (trace aspect activation):

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Update time</td>
<td>110 ± 1 ms</td>
</tr>
<tr>
<td>Traversed Nodes</td>
<td>ca. 596</td>
</tr>
<tr>
<td>Updated objects</td>
<td>1</td>
</tr>
</tbody>
</table>

**Configuration 1** (high bandwidth)

**Configuration 2** (low bandwidth)
Evaluation: Lumisoft Mail-Server

- Freeware implemented in C#
- Ca. 61.000 lines of code
- Update statistics (trace aspect activation):

<table>
<thead>
<tr>
<th></th>
<th>213 ± 7,5 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traversed Nodes</td>
<td>ca. 1.326</td>
</tr>
<tr>
<td>Updated objects</td>
<td>ca. 3</td>
</tr>
</tbody>
</table>

- Overhead Evaluation: IMAP-Throughput (Fetchmail: 10kByte)
  - Without synch. aspect: 45,10 ± 3,69 fetches in 10 seconds
  - With synch. aspect: 45,24 ± 4,49 fetches in 10 seconds

Xeon 2,8 GHz 2 CPUs, 2 GB RAM, Windows XP Sp2, Microsoft .NET 2.0, MailServer Version 0.88
http://www.lumisoft.ee/lswww/download/downloads/MailServer/Devel/
Evaluation: PaintDotNet

- Free open source C# image editor
- >133,000 lines of code
- Update statistics (fixing a small bug)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Update time</td>
<td>3,52 ± 0,20 sec.</td>
</tr>
<tr>
<td>Traversed Nodes</td>
<td>ca. 28,000</td>
</tr>
<tr>
<td>Updated objects</td>
<td>ca. 200</td>
</tr>
<tr>
<td>Handled leaf nodes</td>
<td>ca. 700,000</td>
</tr>
</tbody>
</table>

- Changed lines of code: ca. 30
- Synchronization overhead: not noticeable

Xeon 2,8 GHz 2 CPUs, 2 GB RAM, Windows XP Sp2, Microsoft .NET 2.0, PaintDotNet 3.0
Case Study: Adaptive Control Applications in the Distributed Control Lab

SOAP

experiment manager

result management

job/experiment management

Web-Browser

Mobile Access

Visual Studio Integration

Foucault’s Pendulum

Industrial Control

“Higher Striker” real-time control
Fault Tolerance and Security with dynamic reconfiguration

- Problem: malicious code submitted via the Internet
- Solution: execute an adaptive control application
  - Verified safety controller
- Observed parameters
  - Pendulums amplitude
  - Duration of job execution
  - State of user capsule (abnormal termination)
Performance Evaluation - Foucault's Pendulum

Xeon 2.8 GHz 2 CPUs, 2 GB RAM, Windows XP Sp2, Microsoft .NET 2.0, 50 Measurements
Adaptive Real-time Control Applications

- Predictable execution of .NET applications (Real-time.Net)
  - Pre-compilation of target binaries
  - Library extensions for real-time programming (periodic threads, priority inversion avoidance, priority control ...)
  - Mechanisms for hardware-near programming (direct memory access, port-based I/O, interrupt handling)

- Predictable reconfiguration times possible with Adapt.Net

Diagram showing the compilation and reconfiguration process involving .NET Assembly, CIL, RTL, and GCC with backends for C++, C#, and VB.NET, leading to target binaries.
Heterogeneous Industrial Control Applications

Steuerung Fertigungstrasse

- Speicher Programmierbare Steuerung (SPS)
- Ce.NET Embedded PC
- Soft-SPS Laufzeitsystem

Roboterarm-Steuerung

- Windows XP x86 PC
- Serial Interface Controller
- Java-basierter Zugriff auf Intelligent Interface

Feldgerät

- CAN Bus
- Ethernet Echtzeitkommunikation

Konfigurationsmanager

- Windows XP
- .NET/Java

Monitoring

- Windows XP
- .NET GUI Anwendung

Alerting

- J2ME Mobile Phone

ADS TCP/IP Kommunikation (Hersteller-spezifisch)

GSM TCP/IP

.NET Remoting

CORBA/IIOP

Ethernet Echtzeitkommunikation
Conclusions

- Configurations can be composed/developed independently
  - Non-functional app.-properties can be tested for aimed situation
  - New configurations can be added (by a separate planner/...)

- Algorithm for dynamic reconfiguration of distributed multithreaded applications with cyclic dependencies
  - Low overhead for normal method execution

- Dynamic updates for activating alternative algorithms/ hot-fixes
  - Without manipulation of the virtual machine

- AOP capable of generating (re-)configuration specific logic

- Adaptive applications can be used for protecting experiment hardware in a remote laboratory environment
Further Reading - Dynamic Reconfiguration

- ReDAC - Dynamic Reconfiguration of distributed component-based applications with cyclic dependencies. Rasche, Andreas; Polze, Andreas: accepted at 11th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 5-7 Mai 2008, Orlando, Florida.


Further Reading - Distributed Control Lab


- **Real-time robotics and process control experiments in the Distributed Control Lab.** Andreas Rasche, Bernhard Rabe, Martin von Löwis, Jan Möller, and Andreas Polze. in IEE Software, Special Edition on Microsoft Research Embedded Systems RFP, ISSN 1462-5970, S. 229– 235, Oktober 2005


http://www.dcl.hpi.uni-potsdam.de
Evaluation: Method call overhead

<table>
<thead>
<tr>
<th>Method Call Type</th>
<th>Time (µs ± Error)</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-process .NET remoting call</td>
<td>474610,7±5872,8µs</td>
</tr>
<tr>
<td>normal .NET reflection invoke call</td>
<td>7709,7±1996 µs</td>
</tr>
<tr>
<td>Rapier-Loom aspect for synch. of recursive comps. (rw-lock)</td>
<td>578,8±24,3µs</td>
</tr>
<tr>
<td>Rapier-Loom aspect for synch. of non-recursive components</td>
<td>226,3±7,9µs</td>
</tr>
<tr>
<td>normal interface call</td>
<td>9,0±0,1µs</td>
</tr>
</tbody>
</table>

Time for 1000 method invocations of int Count(int c), 100 measurements, 1st. skipped
It's not that bad ...

- Only a few method invocations are synchronized / on interwoven (root-) objects