Using dynamic component switching for application-based Self-Adaptation

ECE750-T11 Component-Based Software Systems – Project Presentation
Professor Ladan Tahvildari

University of Waterloo - Faculty of Engineering
Dept. of Electrical & Computer Engineering

Manbeen Kohli
Yan Li
Outline

- Self-Adaptive Systems
- Proposed Project
- Approach
- Expected Results
- Related Research
- References
Outline

- Self-Adaptive Systems
- Proposed Project
- Approach
- Expected Results
- Related Research
- References
Self-Adaptive Systems

Introduction

- Software complexity crisis
  - Variant concerns (availability, performance, security, etc)
  - Dynamics of the environment and unexpected conditions
- Internet and pervasive computing
  - Dissolves boundaries for how, when and where humans and computers interact
- Need better mechanism for software management
- Self-Adaptive or Autonomic systems can manage **themselves** given high-level objective from administrators
- First introduced by Paul Horn of IBM in March 2001
  - Term derived from Biology – the autonomous nervous system
Self-Adaptive Systems

An Example

The essence of autonomic systems is self-management
Self-Adaptive Systems
Aspects of Self-Management

Increase Responsiveness
Adapt to dynamically changing environments

Operational Efficiency
Tune resources, balance workloads to best use IT resources

Business Resiliency
Discover, diagnose, act to prevent disruptions

Secure Information & Resources
Anticipate, detect, identify, deter attacks

These aspects heavily influence the architecture of a self-adaptive system
Self-Adaptive Systems
Architecture – The Autonomic Element

- AEs are the basic atoms of autonomic systems
- An AE contains
  - Exactly one autonomic manager
  - Zero or more managed element(s)
- AE is responsible for
  - Managing own behavior in accordance with policies
  - Interacting with other autonomic elements to provide or consume computational services
Self-Adaptive Systems
Types of Adaptation

- **External Adaptation**
  - External resources monitor the system and determine the adaptation required

- **Internal Adaptation**
  - Adaptation mechanisms are **wired into** the application itself

- Our proposed project focuses on **internal adaptation**
Outline

- Self-Adaptive Systems
- Proposed Project
- Approach
- Expected Results
- Related Research
- References
Proposed Project

Problem Statement

This project focuses on how dynamic component switching can be used by a system to reconfigure itself as a reaction to a run-time change.
Proposed Project
Motivation

- Systems can go from steady state …

Few minutes later…

- to overloaded without warning
Proposed Project
Prototype

- We will build a **news website system** providing services like news, stocks, weather, streaming media (time permitting) etc.
- Under normal workload - all services will be provided
- Under heavy workload – **gradually degrade services**
- Gradual degradation implemented by **dynamically switching components** optimized for a pre-determined workload
Proposed Project
High-level system architecture

- Streaming media
- News items EJB
- Stocks EJB
- Weather EJB
- News DB
- Stocks DB
- Weather DB
- Servlet/Controllers
- Client (web pages)
Proposed Project

Goals

- High Service Availability
  - Maximum users should be able to access web-site
  - Under heavy workload system can degrade to a text-only website

- Reasonable Response time
  - At no point should system render itself not responsive

We have taken the following approach to accomplish the aforementioned tasks …
Outline

- Self-Adaptive Systems
- Proposed Project
- Approach
- Expected Results
- Related Research
- References
Approach Overview

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predecessors</th>
<th>Resource Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>4 days</td>
<td>Wed 6/11/07</td>
<td>Mon 6/18/07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design News website</td>
<td>3 days</td>
<td>Wed 6/11/07</td>
<td>Fri 6/15/07</td>
<td></td>
<td>Manbeen</td>
</tr>
<tr>
<td>Design data input subsystem</td>
<td>0.5 days</td>
<td>Mon 6/18/07</td>
<td>Mon 6/26/07</td>
<td>2</td>
<td>Manbeen</td>
</tr>
<tr>
<td>Design test suite</td>
<td>0.5 days</td>
<td>Mon 6/13/07</td>
<td>Mon 6/26/07</td>
<td>3</td>
<td>Manbeen</td>
</tr>
<tr>
<td>Research Technologies</td>
<td>4 days</td>
<td>Wed 6/11/07</td>
<td>Mon 6/18/07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research J2EE 1.4</td>
<td>2 days</td>
<td>Wed 6/13/07</td>
<td>Thu 6/14/07</td>
<td>6</td>
<td>Yan</td>
</tr>
<tr>
<td>Research J2EE 1.5</td>
<td>2 days</td>
<td>Fri 6/15/07</td>
<td>Mon 6/18/07</td>
<td>6</td>
<td>Yan</td>
</tr>
<tr>
<td>Development</td>
<td>20 days</td>
<td>Tue 5/19/07</td>
<td>Mon 7/16/07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>News item component</td>
<td>10 days</td>
<td>Tue 6/13/07</td>
<td>Mon 7/7/07</td>
<td>4</td>
<td>Manbeen</td>
</tr>
<tr>
<td>JSPs and action classes</td>
<td>6 days</td>
<td>Tue 7/9/07</td>
<td>Thu 7/10/07</td>
<td>9</td>
<td>Manbeen</td>
</tr>
<tr>
<td>Weather conditions component</td>
<td>4 days</td>
<td>Wed 7/11/07</td>
<td>Mon 7/16/07</td>
<td>10</td>
<td>Manbeen</td>
</tr>
<tr>
<td>Stocks component</td>
<td>11 days</td>
<td>Tue 6/25/07</td>
<td>Thu 7/10/07</td>
<td>13</td>
<td>Yan</td>
</tr>
<tr>
<td>Data input subsystem</td>
<td>8 days</td>
<td>Tue 6/13/07</td>
<td>Fri 6/26/07</td>
<td>7</td>
<td>Yan</td>
</tr>
<tr>
<td>Determine System Capacity</td>
<td>7 days</td>
<td>Wed 7/11/07</td>
<td>Thu 7/19/07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>determine capacity for large and medium images</td>
<td>3 days</td>
<td>Wed 7/11/07</td>
<td>Fri 7/13/07</td>
<td>12</td>
<td>Yan</td>
</tr>
<tr>
<td>Determine system capacity for small images and text only site</td>
<td>3 days</td>
<td>Tue 7/17/07</td>
<td>Thu 7/19/07</td>
<td>11</td>
<td>Manbeen</td>
</tr>
<tr>
<td>Change Adaptability Rules</td>
<td>3 days</td>
<td>Fri 7/20/07</td>
<td>Tue 7/24/07</td>
<td>16</td>
<td>Manbeen</td>
</tr>
<tr>
<td>Retest system and fix bugs</td>
<td>10 days</td>
<td>Wed 7/25/07</td>
<td>Tue 8/7/07</td>
<td>17</td>
<td>All</td>
</tr>
</tbody>
</table>

Feasibility study

System design and implementation

Determine system capacity by emulating workload

Revisit adaptation rules
Approach

Feasibility study

- Research different technologies usable for implementation
  - J2EE, .NET etc
- Study differences between J2EE 1.4 & 1.5
- Investigate different J2EE 1.5 compliant application servers
  - Sun Application Server, BEA Weblogic etc
- Explore performance measuring tools
  - Apache JMeter, Probekit etc
Approach:
System Design
Approach
News Component Design

NewsItem
- newsId
- title
- category
- isMainPage
- isHeadline
- isCategoryHeadline
- newsTimestamp
- newsMediaItems
- newsTextItems

NewsMediaItem
- newsId
- mediaId
- newsItem

NewsTextItems
- newsId
- ordinal
- newsText
- newsItem

MediatItem
- mediaId
- category
- mediaTypeltems
- selectedMediaTypeltems
- selectedMediaTypes

MediaTypeItem
- mediaId
- mediaTypeId
- location
- mediaItem
- mediaTypeAttributes

MediaTypeAttribute
- mediaTypeId
- attributeId
- attributeName
- attributeValue
Approach
Adaptability implementation details

- Each news item contains associated media (images and/or streaming media) of varying quality
  - the current server workload and bandwidth requirements of the media determine the component to be displayed on the JSP
- Display of stock and weather is determined by similar factors
Approach
System capacity determination

- Apache JMeter is a Java application used to load test functional behavior and measure performance
- JMeter can be used to determine system capacity by running the system in different modes
  - System displays highest quality components
  - System displays medium quality components
  - System displays low quality components
  - System displays only text
Approach
Revisit adaptation

- When servlet receives client request
  - Determine **number of active requests**
  - **Pick component** of quality that can be handled by system based on pre-determined system capacity (but avoid constant thrashing)
  - Display component on client page
**Approach**

**Current Status**

- Completed
- In Progress
- Will start soon

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
<th>Predicators</th>
<th>Assumptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Design</td>
<td>4 days</td>
<td>Wed 6/13/07</td>
<td>Mon 6/18/07</td>
<td>Marbeen</td>
</tr>
<tr>
<td>2</td>
<td>Design/News server</td>
<td>3 days</td>
<td>Wed 6/13/07</td>
<td>Fri 6/15/07</td>
<td>Marbeen</td>
</tr>
<tr>
<td>3</td>
<td>Design/custom subsystem</td>
<td>0.5 days</td>
<td>Mon 6/18/07</td>
<td>Mon 6/18/07</td>
<td>Yan</td>
</tr>
<tr>
<td>4</td>
<td>Design/full sub</td>
<td>0.5 days</td>
<td>Mon 6/18/07</td>
<td>Mon 6/18/07</td>
<td>Marbeen</td>
</tr>
<tr>
<td>5</td>
<td>Research Technologies</td>
<td>4 days</td>
<td>Wed 6/13/07</td>
<td>Mon 6/19/07</td>
<td>Yan</td>
</tr>
<tr>
<td>6</td>
<td>Research/CEE 1.4</td>
<td>2 days</td>
<td>Wed 6/13/07</td>
<td>Thu 6/14/07</td>
<td>Yan</td>
</tr>
<tr>
<td>7</td>
<td>Research/CEE 1.5</td>
<td>2 days</td>
<td>Fri 6/15/07</td>
<td>Mon 6/18/07</td>
<td>Yan</td>
</tr>
<tr>
<td>8</td>
<td>Development</td>
<td>20 days</td>
<td>Tue 5/15/07</td>
<td>Mon 7/6/07</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>News item component</td>
<td>10 days</td>
<td>Tue 5/15/07</td>
<td>Mon 7/6/07</td>
<td>Yan</td>
</tr>
<tr>
<td>10</td>
<td>JSDPs and action codes</td>
<td>6 days</td>
<td>Tue 7/5/07</td>
<td>Tue 7/10/07</td>
<td>Marbeen</td>
</tr>
<tr>
<td>11</td>
<td>Writable component</td>
<td>4 days</td>
<td>Wed 7/11/07</td>
<td>Mon 7/18/07</td>
<td>Marbeen</td>
</tr>
<tr>
<td>12</td>
<td>Stock component</td>
<td>4 days</td>
<td>Tue 7/10/07</td>
<td>Mon 7/18/07</td>
<td>Yan</td>
</tr>
<tr>
<td>13</td>
<td>Data input subversion</td>
<td>5 days</td>
<td>Tue 5/15/07</td>
<td>Fri 6/22/07</td>
<td>Yan</td>
</tr>
<tr>
<td>14</td>
<td>Determine System Capacity</td>
<td>7 days</td>
<td>Wed 7/11/07</td>
<td>Thu 7/15/07</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Determine capacity for large and medium images</td>
<td>3 days</td>
<td>Wed 7/11/07</td>
<td>Fri 7/13/07</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Determine system capacity for small images and test only site</td>
<td>3 days</td>
<td>Tue 7/10/07</td>
<td>Thu 7/10/07</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Change Adaptability Rules</td>
<td>3 days</td>
<td>Wed 7/5/07</td>
<td>Thu 7/5/07</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Refine system need for bugs</td>
<td>10 days</td>
<td>Wed 7/5/07</td>
<td>Tue 7/15/07</td>
<td>Marbeen</td>
</tr>
</tbody>
</table>

- Feasibility study
- System design and implementation
- Determine system capacity by emulating workload
- Revisit adaptation rules
Approach
Current Accomplishment

DEMO
Outline

- Self-Adaptive Systems
- Proposed Project
- Approach
- Expected Results
- Related Research
- References
Expected Results

- The system must be able to **reconfigure** itself in response to the system changes by using **service degradation**
- **High Availability**
  - Under **normal workload** the system is expected to provide **high quality components**
  - Under **heavy workload** the system is supposed to degrade itself to a **text only website**
- **Reasonable Response Time**
  - At **no point** in time should the system render itself completely **non-responsive**
  - The system must respond within a reasonable time
Outline

- Self-Adaptive Systems
- Proposed Project
- Approach
- Expected Results
- Related Research
- References
Related Research

Rainbow: architecture-based self-adaptation with reusable infrastructure (Garlan et. all [3])

- *Rainbow uses an abstract architectural model to monitor an executing system’s runtime properties, evaluates the model for constraint violation, and—if a problem occurs—performs global- and module-level adaptations on the running system [3]*

- Advantages
  - Can be applied to legacy systems
  - Adaptation techniques based on system architecture

- Disadvantages
  - External adaptation has an associated latency
  - Can only be applied to systems providing access hooks for monitoring and adaptation
  - Monitoring and Adaptation is performed in a single rainbow framework – leads to scalability and single point of failure concerns
Related Research

Architecture-based self-adaptation in the presence of multiple objectives (Garlan et al [4])

- Development of an adaptation language based on **utility theory**
- Considers a fictitious news website with objectives – reasonable response time and budget server pool operation cost
- System employs **architecture based self-adaptation** using selection of **strategies** based on the associated utility value
- System **does not** gradually degrade offered services

```
strategy SwitchToTextualContent() {
  t0: (responseTime() > Resp_Time_Threshold) -> switchToTextualContent(m);
  do {
    tl: (responseTime() < Resp_Time_Threshold) -> done;
  }
}

strategy AdjustServerPoolSize(int kdelta) {
  t0: (responseTime() > Resp_Time_Threshold) -> adjustServerPoolSize(kdelta);
  do {
    tl: (responseTime() < Resp_Time_Threshold) -> done;
  }
}
```
Outline

- Self-Adaptive Systems
- Problem Statement
- Proposed Project
- Approach
- Expected Results
- Related Research
- References
References


References


14. The J2EE 1.5 Tutorial, Sun Microsystems, Feb 2007

Using dynamic component switching for application-based Self-Adaptation