Cost-effective development of flexible self-* applications

Radu Calinescu

Computing Laboratory
University of Oxford
Outline

• Motivation
• Generic self-* framework
• Self-* application development
Motivation

- **cost-effective**
- **flexible**

- no general-purpose architecture & generic development approaches for self-* applications

- ✔ cost-effective
- ✔ flexible
Framework objectives

Significant reduction in development effort/expertise/cost
  – component reuse
  – off-the-shelf technologies & tools

Major improvement in application flexibility/robustness
  – model-based development
  – automated code generation

New powerful capabilities
  – new classes of policies
  – support for system-of-systems development
Typical drawbacks of self-* architectures
Typical drawbacks of self-* architectures

Application-specific metadata hard-coded within the autonomic manager.
Typical drawbacks of self-* architectures

- Resource-specific code
- Application-specific metadata hard-coded within the autonomic manager

Diagram:

- Autonomic manager
  - Analyse
  - Plan
  - Monitor
  - Execute

- Managed resources
  - Sensors
  - Effectors

Policies flow into the autonomic manager.
Typical drawbacks of self-* architectures

- Resource-specific code
- Application-specific metadata hard-coded within the autonomic manager
- Manually-implemented interfaces
Typical drawbacks of self-* architectures

- statically-defined high-level interfaces
- resource-specific code
- application-specific metadata hard-coded within the autonomic manager
- manually-implemented interfaces
Generic self-* architecture

- Statically-defined high-level interfaces
- Resource-specific code
- Supply the “knowledge” at runtime to application-obliviuous autonomic manager
- Manually-implemented interfaces
Generic self-* architecture

- Statically-defined high-level interfaces
- Resource-oblivious code
- Supply the "knowledge" at runtime to application-oblivious autonomic manager
- Manually-implemented interfaces
Generic self-* architecture

- Statically-defined high-level interfaces
- Resource-oblivious code
- Supply the “knowledge” at runtime to application-oblivious autonomic manager
- Automatic generation of autonomic manager interfaces & computer-aided development of resource interfaces

Oxford University Computing Laboratory
Generic self-* architecture

- policy-driven, runtime generation of high-level interfaces
- resource-oblivious code
- supply the “knowledge” at runtime to application-oblivious autonomic manager
- automatic generation of autonomic manager interfaces & computer-aided development of resource interfaces

Oxford University Computing Laboratory
Implementation

Policy engine = (.NET/C#) web service
- platform independence
- standards-based support for security
- loose coupling
Implementation

Policy engine = (.NET/C#) web service

Knowledge = XML-encoded model of resource parameters & behaviour
• off-the-shelf tools to process model (XSLT engines, XSD code generators)
Implementation

Policy engine = (.NET/C#) web service

Knowledge = XML-encoded model of resource parameters & behaviour

Manageability adaptors = web services that subclass abstract Adaptor class
- Adaptor class implements the bulk of the functionality
- several other components generated automatically from the system model

Policy engine interface = generated online from model using OO reflection
Implementation

Policy engine = (.NET/C#) web service

Knowledge = XML-encoded model of resource parameters & behaviour

Manageability adaptors = web services that subclass abstract Adaptor class

Policy engine interface = generated online from model using OO reflection

High-level interfaces = policy-driven, automatically generated web services
• expose system to its environment
• specified by “resource definition” policies
• enable system integration into self-* systems of systems
Policy engine = (.NET/C#) web service

Knowledge = XML-encoded model of resource parameters & behaviour

Manageability adaptors = web services that subclass abstract Adaptor class

Policy engine interface = generated online from model using OO reflection

High-level interfaces = policy-driven, automatically generated web services
Implementation – system-of-systems federation

Policy engine = (.NET/C#) web service

Knowledge = XML-encoded model of resource parameters & behaviour

Manageability adaptors = web services that subclass abstract Adaptor class

Policy engine interface = generated online from model using OO reflection

High-level interfaces = policy-driven, automatically generated web services
Application development

**Generation**
- generate/implement application-specific components
  - (system developer)

**Deployment**
- configure policy engine & deploy new components
  - (system administrator)

**Exploitation**
- specify/select self-* policies
  - (system user)
Sample self-* application

Allocate data-centre servers to clusters of different priorities & variable workloads such that they achieve user-defined levels of availability in the presence of cluster component failures.
- Build XML model describing the system resources, their parameters and behaviour
- Instance of pre-defined XML schema (*meta-model*)
XML system model

Generation

```xml
<?xml version="1.0" encoding="UTF-8"?>
<system xmlns="http://www.xcc.com/system"
<name>datacentre</name>

<!-- Cluster of workstations -->
<resource>
  <!-- Unique cluster ID -->
  <property> [12 lines] 
  <!-- Priority of this cluster -->
  <property> [14 lines] 
  <!-- Number of operational servers -->
  <property> [14 lines] 
  <!-- Number of servers allocated for -->
  <property> [14 lines] 
  <!-- CTMC model -->
  <property> [12 lines] 
  <!-- Expected availability (as the -->
  <property> [12 lines]
</resource>

<!-- Pool of servers to be organised i -->
<resource>
  <!-- Unique server pool ID -->
</resource>
```
Oxford University Computing Laboratory

(XML) system model

Generation
Oxford University Computing Laboratory

G2: XSL transformation

System (XML) schema

Generation

```
<xml version="1.0" encoding="UTF-8">
<xsd:schema xmlns="http://www.rcc.com/system" xmlns:xmns:x="">
  <xsd:element name="cluster" type="cluster"/>
  <xsd:complexType name="cluster">
    <xsd:sequence>
      <xsd:element name="id" type="clusterId" nillable="">
      <xsd:element name="priority" type="clusterPriority">
      <xsd:element name="requiredServers" type="cluster"><xsd:element name="allocatedServers" type="cluster">
      <xsd:element name="behaviouralModel" type="cluster">
      <xsd:element name="availability" type="clusterAvail"></xsd:element name="availabilityDefinition" type="x">
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="serverPool" type="serverPool"/>
  <xsd:complexType name="serverPool">
    <xsd:sequence>
      <xsd:element name="id" type="serverPoolId" nillable="">
      <xsd:element name="nservers" type="serverPoolNserv">
    </xsd:sequence>
  </xsd:complexType>

  <xsd:simpleType name="clusterId">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
</xsd:schema>
```
Oxford University Computing Laboratory

Generation

( XML ) system model

G2: XSL transformation

System ( XML ) schema

G3: XSD code generation

System data types (classes)
Oxford University Computing Laboratory

G2: XSL transformation

G3: XSD code generation

G4: generic service subclassing

Generation
Use web client to supply model to the policy engine.
Use web client to supply model to the policy engine

D1: policy engine configuration

Configured policy engine

Deployment (XML) system model

G2: XSL transformation

System (XML) schema

G3: XSD code generation

System data types (classes)

G4: generic service subclassing

Manageability adaptor

Generation

System model

datacentre
  ..cluster
    ....id
    ....priority
    ....requiredServers
    ....allocatedServers
    ....behaviouralModel
    ....availability
    ..serverPool

Model file: Browse...

Polling period

10 seconds

New Period: 
(XML) system model

G2: XSL transformation

System (XML) schema

G3: XSD code generation

System data types (classes)

G4: generic service subclassing

Manageability adaptor

D1: policy engine configuration

Configured policy engine

D2: adaptor deployment

Manageable resources

Generation

Deployment
D1: policy engine configuration

D2: adaptor deployment

Manageable resources

E2: resource discovery

Self-* system

Generation

Deployment

Exploitation

(XML) system model

System (XML) schema

System data types (classes)

Manageability adaptor

Manageable resources

Configured policy engine

G2: XSL transformation

G3: XSD code generation

G4: generic service subclassing

Manageability adaptor

(E2) Self-* system

(Oxford University Computing Laboratory)
D1: policy engine configuration

D2: adaptor deployment

Manageable resources

E1: policy specification

E2: resource discovery

Self-* system

Generation

Deployment

Exploitation

(XML) system model

System (XML) schema

System data types (classes)

Manageability adaptor

Configured policy engine

Manageable resources

Oxford University Computing Laboratory
Utility-function policy

\[
\text{MAXIMIZE} \left( \sum_{i=1}^{N} \text{priority}_i \text{GOAL}(\text{availability}_i \geq \text{targetAvailability}_i) \right. \\
\left. - \epsilon \sum_{i=1}^{N} \text{allocatedServers}_i \right)
\]
Utility-function policy

\[
\text{MAXIMIZE } \left( \sum_{i=1}^{N} \text{priority}_i \text{GOAL}(\text{availability}_i \geq \text{targetAvailability}_i) - \epsilon \sum_{i=1}^{N} \text{allocatedServers}_i \right)
\]
Utility-function policy

\[
\text{MAXIMIZE} \left( \sum_{i=1}^{N} \text{priority}_i \text{GOAL}(\text{availability}_i \geq \text{targetAvailability}_i) \\
- \epsilon \sum_{i=1}^{N} \text{allocatedServers}_i \right)
\]
Utility-function policy

\[ \text{MAXIMIZE} \left( \sum_{i=1}^{N} \text{priority}_i \text{GOAL} (\text{availability}_i \geq \text{targetAvailability}_i) \right) \]

\[ - \epsilon \sum_{i=1}^{N} \text{allocatedServers}_i \]
Online quantitative analysis in self-* systems

Sample self-* application: summary

The allocation of data-centre servers to clusters is managed automatically, based on high-level system objectives specified by data-centre administrators.
## Case study summary

<table>
<thead>
<tr>
<th>Resource type</th>
<th>self-* areas &amp; policy type</th>
<th>application domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>software</td>
<td>action</td>
<td>CPU capacity allocation</td>
</tr>
<tr>
<td>hardware</td>
<td>goal</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td>utility-function</td>
<td></td>
</tr>
<tr>
<td>legacy</td>
<td>resource-definition</td>
<td></td>
</tr>
<tr>
<td>autonomic-enabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>main self-* functional areas</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation of CPU capacity</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPU capacity allocation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Goal-driven CPU sched.</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disk drive DPM</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dynamic power management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ctrl. of cluster availability</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>availability management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic gen. of web content</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td>√</td>
<td>online report generation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>