Autonomic Monitoring and Management of Component-based Services

Cristian RUZ

Thèse dirigée par Françoise BAUDE,
au sein de l’équipe OASIS

Jury

Pr. Mireille BLAY-FORRARINO  Président du Jury
Pr. Philippe LALANDA  Rapporteur
Pr. Lionel SEINTURIER  Rapporteur
Dr. Romain ROUVOY  Co-Rapporteur
Dr. Luc BELLISSARD  Examinateur
Pr. Françoise BAUDE  Directeur de Thèse

23 Juin 2011
Introduction

Context

Goals and Contribution

State of the Art

Design

Implementation

Validation

Conclusions
1. Introduction

2. Context

3. Goals and Contribution

4. State of the Art

5. Design

6. Implementation

7. Validation

8. Conclusions
Motivation

Evolution in software construction

- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly
Motivation

Evolution in software construction

- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly
Motivation

Evolution in software construction

- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly
Motivation

Evolution in software construction

- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly
1. Introduction

Motivation

Evolution in software construction

- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly
Motivation

Evolution in software construction

- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly
1. Introduction

Motivation

Dynamic environment

- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation
- Heterogeneity and distribution
  - Transfer autonomic adaptation task to each element

Provider
Motivation

Dynamic environment

- Decentralized, distributed, dynamic applications
  - External conditions may change
  - Software needs to dynamically react and adapt to changes
    - Complexity not easy for a human manager
    - Autonomic adaptation
  - Heterogeneity and distribution
    - Transfer autonomic adaptation task to each element
1. Introduction

Motivation

Dynamic environment

- Decentralized, distributed, dynamic applications
  - External conditions may change
  - Software needs to dynamically react and adapt to changes
    - Complexity not easy for a human manager
    - Autonomic adaptation
  - Heterogeneity and distribution
    - Transfer autonomic adaptation task to each element
Motivation

Dynamic environment

- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation
- Heterogeneity and distribution
  - Transfer autonomic adaptation task to each element
Motivation

Dynamic environment

- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation
- Heterogeneity and distribution
  - Transfer autonomic adaptation task to each element
1. Introduction

Motivation

Dynamic environment

- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation
- Heterogeneity and distribution
  - Transfer autonomic adaptation task to each element
Motivation

Dynamic environment

- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
- Autonomic adaptation
- Heterogeneity and distribution
  - Transfer autonomic adaptation task to each element
Motivation

Dynamic environment

- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation

Heterogeneity and distribution

- Transfer autonomic adaptation task to each element
Motivation

Dynamic environment
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation
- Heterogeneity and distribution
  - Transfer autonomic adaptation task to each element
Motivation

Dynamic environment

- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation
- Heterogeneity and distribution
  - Transfer autonomic adaptation task to each element
Table of Contents

1. Introduction

2. Context

3. Goals and Contribution

4. State of the Art

5. Design

6. Implementation

7. Validation

8. Conclusions
Developing dynamic adaptable software

Component-based Software Development
- Development of independent pieces of code
- Encapsulated, reusable units
- Better adaptation to changing requirements

Service-orientation
- Providers offers specific functionalities as a service
- Services are composable using standard means
- Facilitate the construction of new added-value applications
- Loosely coupled compositions of heterogeneous services
Developing dynamic adaptable software

Component-based Software Development
- Development of independent pieces of code
- Encapsulated, reusable units
- Better adaptation to changing requirements

Service-orientation
- Providers offers specific functionalities as a service
- Services are composable using standard means
- Facilitate the construction of new added-value applications
- Loosely coupled compositions of heterogeneous services
Developing dynamic adaptable software

Component-based Software Development
- Development of independent pieces of code
- Encapsulated, reusable units
- Better adaptation to changing requirements

Service-orientation
- Providers offer specific functionalities as a service
- Services are composable using standard means
- Facilitate the construction of new added-value applications

Loosely coupled compositions of heterogeneous services
Developing dynamic adaptable software

Component-based Software Development
- Development of independent pieces of code
- Encapsulated, reusable units
- Better adaptation to changing requirements

Service-orientation
- Providers offers specific functionalities *as a service*
- Services are composable using standard means
- Facilitate the construction of new added-value applications
- Loosely coupled compositions of heterogeneous services
Service-Orientation

Basic model for Service Oriented Architecture

Various levels in a Service Oriented Architecture

(Papazoglou, Traverso, Dustdar, Leymann, 2007)
Service Component Architecture (SCA)

Designing services using a component-based approach

- Design-time model for building service-based systems
- Technologically agnostic
- Multiple runtime implementations: IBM Websphere App Server, Fabric3, Apache Tuscany, Paremus, FraSCAti
- Specification does not consider dynamic evolution
Advantages … and challenges

Advantages in software development

- Growing ecosystem of services and compositions
- Easier to modify an application dynamically and quickly adapt

Challenges

- Proper management of complex compositions
- Maintenance depends on different providers
- Several characteristics are less controllable (QoS)
- Need to timely react to unforeseen conditions, and with minimal perturbation
Advantages ... and challenges

Advantages in software development

- Growing ecosystem of services and compositions
- Easier to modify an application dynamically and quickly adapt

Challenges

- Proper management of complex compositions
- Maintenance depends on different providers
- Several characteristics are less controllable (QoS)
- Need to timely react to unforeseen conditions, and with minimal perturbation
Advantages . . . and challenges

Advantages in software development

- Growing ecosystem of services and compositions
- Easier to modify an application dynamically and quickly adapt

Challenges

- Proper management of complex compositions
- Maintenance depends on different providers
- Several characteristics are less controllable (QoS)
- Need to timely react to unforeseen conditions, and with minimal perturbation
Advantages . . . and challenges

Advantages in software development

- Growing ecosystem of services and compositions
- Easier to modify an application dynamically and quickly adapt

Challenges

- Proper management of complex compositions
- Maintenance depends on different providers
- Several characteristics are less controllable (QoS)
- Need to timely react to unforeseen conditions, and with minimal perturbation
Autonomic Computing

Response to the increasing complexity in the maintenance of systems, exceeding the capacity of human beings

- Based on the idea of self-governing systems
- Context-awareness, and self-* properties
- Activities represented in a *feedback control loop*

- Phases in the *MAPE* autonomic control loop
Dr. Cristian Ruz (INRIA Oasis Team)  

Autonomic M&M of Comp-based Services  

23/06/2011 11 / 52

2. Context

**Autonomic Computing**

Response to the increasing complexity in the maintenance of systems, exceeding the capacity of human beings

- Based on the idea of self-governing systems
- Context-awareness, and self-* properties
- Activities represented in a *feedback control loop*
- Phases in the *MAPE* autonomic control loop
# Table of Contents

1. Introduction
2. Context
3. Goals and Contribution
4. State of the Art
5. Design
6. Implementation
7. Validation
8. Conclusions
Situation

“Everything can change”

- Lack of uniformity and flexibility
- Impossibility of foreseeing all situations
- Complexity of developing effective autonomic tasks

Need for adaption. And for dynamic adaptation.
3. Goals and Contribution

Situation

“Everything can change”

- Lack of uniformity and flexibility
- Impossibility of foreseeing all situations
- Complexity of developing effective autonomic tasks

Need for adaption. And for dynamic adaptation.
Goals

Improve the adaptability of service-based applications

- Providing a common means to monitor and manage services
- Adapting to changing management requirements
3. Goals and Contribution

How to achieve this goal?

- Structured way to manage the composition
- Consider distribution and heterogeneity of providers
- Facilitate the insertion of autonomic tasks

Requirements

- Flexibility
  - Allow to modify the solution at runtime
- Extensibility
  - Allow to incorporate custom elements to the solution
- Heterogeneity
  - Retrieve information and execute actions over different technologies
- Efficiency
  - Avoid unnecessary communication and deliver timely responses
- Autonomicity
  - Allow to take autonomic decisions
How to achieve this goal?

- Structured way to manage the composition
- Consider distribution and heterogeneity of providers
- Facilitate the insertion of autonomic tasks

Requirements

- **Flexibility**
  - Allow to modify the solution at runtime
- **Extensibility**
  - Allow to incorporate custom elements to the solution
- **Heterogeneity**
  - Retrieve information and execute actions over different technologies
- **Efficiency**
  - Avoid unnecessary communication and deliver timely responses
- **Autonomicity**
  - Allow to take autonomic decisions
3. Goals and Contribution

How to achieve this goal?

- Structured way to manage the composition
- Consider distribution and heterogeneity of providers
- Facilitate the insertion of autonomic tasks

Requirements

- **Flexibility**
  - Allow to modify the solution at runtime
- **Extensibility**
  - Allow to incorporate custom elements to the solution
- **Heterogeneity**
  - Retrieve information and execute actions over different technologies
- **Efficiency**
  - Avoid unnecessary communication and deliver timely responses
- **Autonomicity**
  - Allow to take autonomic decisions
3. Goals and Contribution

How to achieve this goal?

- Structured way to manage the composition
- Consider distribution and heterogeneity of providers
- Facilitate the insertion of autonomic tasks

Requirements

- **Flexibility**
  - Allow to modify the solution at runtime

- **Extensibility**
  - Allow to incorporate custom elements to the solution

- **Heterogeneity**
  - Retrieve information and execute actions over different technologies

- **Efficiency**
  - Avoid unnecessary communication and deliver timely responses

- **Autonomicity**
  - Allow to take autonomic decisions
3. Goals and Contribution

How to achieve this goal?

- Structured way to manage the composition
- Consider distribution and heterogeneity of providers
- Facilitate the insertion of autonomic tasks

Requirements

- **Flexibility**
  - Allow to modify the solution at runtime
- **Extensibility**
  - Allow to incorporate custom elements to the solution
- **Heterogeneity**
  - Retrieve information and execute actions over different technologies
- **Efficiency**
  - Avoid unnecessary communication and deliver timley responses
- **Autonomicity**
  - Allow to take autonomic decisions
How to achieve this goal?

- Structured way to manage the composition
- Consider distribution and heterogeneity of providers
- Facilitate the insertion of autonomic tasks

Requirements

- **Flexibility**
  - Allow to modify the solution at runtime
- **Extensibility**
  - Allow to incorporate custom elements to the solution
- **Heterogeneity**
  - Retrieve information and execute actions over different technologies
- **Efficiency**
  - Avoid unnecessary communication and deliver timely responses
- **Autonomicity**
  - Allow to take autonomic decisions
3. Goals and Contribution

Solution Overview

Flexible Monitoring and Management framework

- Common and efficient means to monitor and manage service-based applications.
- Allows to introduce monitoring and management concerns and autonomic behaviour at runtime.
- Allows to modify the adaptability features, and support evolving management requirements.
3. Goals and Contribution

How?

- **Implementing an autonomic control loop**
  - Support for autonomicity
  - Encapsulating each phase of the MAPE loop as a component
    - Leverage the technology of services to a common ground (heterogeneity)
    - Use components to extend the behaviour of the control loop
  - Attaching the autonomic control loops to services
    - Define interfaces for the MAPE loops to interact and collaborate
    - Take timely decisions, close to the involved services (efficiency)
  - Allowing to dynamically reconfigure the autonomic control loop
    - Add/remove components at runtime as needed (flexibility)
3. Goals and Contribution

How?

- **Implementing an autonomic control loop**
  - Support for *autonomicity*
- Encapsulating each phase of the MAPE loop as a component
  - Leverage the technology of services to a common ground (heterogeneity)
  - Use components to extend the behaviour of the control loop
- Attaching the autonomic control loops to services
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (efficiency)
- Allowing to dynamically reconfigure the autonomic control loop
  - Add/remove components at runtime as needed (flexibility)
3. Goals and Contribution

How?

- Implementing an autonomic control loop
  - Support for autonomicity
- Encapsulating each phase of the MAPE loop as a component
  - Leverage the technology of services to a common ground (heterogeneity)
  - Use components to extend the behaviour of the control loop
- Attaching the autonomic control loops to services
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (efficiency)
- Allowing to dynamically reconfigure the autonomic control loop
  - Add/remove components at runtime as needed (flexibility)
### How?

- **Implementing an autonomic control loop**
  - Support for **autonomicity**
- **Encapsulating each phase of the MAPE loop as a component**
  - Leverage the technology of services to a common ground (**heterogeneity**)
  - Use components to **extend** the behaviour of the control loop
- **Attaching the autonomic control loops to services**
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (**efficiency**)
- **Allowing to dynamically reconfigure the autonomic control loop**
  - Add/remove components at runtime as needed (**flexibility**)

---

**Diagram:**

- Analyze
  - Monitor
    - Plan
      - Execute

---

Cristian Ruz (INRIA Oasis Team)  |  Autonomic M&M of Comp-based Services  |  23/06/2011  |  17 / 52
3. Goals and Contribution

How?

- **Implementing an autonomic control loop**
  - Support for **autonomicity**

- **Encapsulating each phase of the MAPE loop as a component**
  - Leverage the technology of services to a common ground (heterogeneity)
  - Use components to **extend** the behaviour of the control loop

- **Attaching the autonomic control loops to services**
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (**efficiency**)
  - Allowing to dynamically reconfigure the autonomic control loop
  - Add/remove components at runtime as needed (**flexibility**)

![Diagram showing the MAPE loop for Service A, Service B, and Service C](attachment:image.png)
3. Goals and Contribution

How?

- Implementing an autonomic control loop
  - Support for autonomicity
- Encapsulating each phase of the MAPE loop as a component
  - Leverage the technology of services to a common ground (heterogeneity)
  - Use components to extend the behaviour of the control loop
- Attaching the autonomic control loops to services
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (efficiency)
- Allowing to dynamically reconfigure the autonomic control loop
  - Add/remove components at runtime as needed (flexibility)
3. Goals and Contribution

How?

- **Implementing an autonomic control loop**
  - Support for *autonomicity*

- **Encapsulating each phase of the MAPE loop as a component**
  - Leverage the technology of services to a common ground (*heterogeneity*)
  - Use components to **extend** the behaviour of the control loop

- **Attaching the autonomic control loops to services**
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (**efficiency**)

- **Allowing to dynamically reconfigure the autonomic control loop**
  - Add/remove components at runtime as needed (**flexibility**)

![Diagram of services](attachment://diagram.png)
3. Goals and Contribution

How?

- Implementing an autonomic control loop
  - Support for autonomicity
- Encapsulating each phase of the MAPE loop as a component
  - Leverage the technology of services to a common ground (heterogeneity)
  - Use components to extend the behaviour of the control loop
- Attaching the autonomic control loops to services
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (efficiency)
- Allowing to dynamically reconfigure the autonomic control loop
  - Add/remove components at runtime as needed (flexibility)
How?

- Implementing an autonomic control loop
  - Support for autonomicity
- Encapsulating each phase of the MAPE loop as a component
  - Leverage the technology of services to a common ground (heterogeneity)
  - Use components to extend the behaviour of the control loop
- Attaching the autonomic control loops to services
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (efficiency)
- Allowing to dynamically reconfigure the autonomic control loop
  - Add/remove components at runtime as needed (flexibility)
3. Goals and Contribution

How?

- **Implementing an autonomic control loop**
  - Support for autonomicity
- **Encapsulating each phase of the MAPE loop as a component**
  - Leverage the technology of services to a common ground (heterogeneity)
  - Use components to extend the behaviour of the control loop
- **Attaching the autonomic control loops to services**
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (efficiency)
- **Allowing to dynamically reconfigure the autonomic control loop**
  - Add/remove components at runtime as needed (flexibility)
3. Goals and Contribution

**How?**

- **Implementing an autonomic control loop**
  - Support for **autonomicity**

- **Encapsulating each phase of the MAPE loop as a component**
  - Leverage the technology of services to a common ground (**heterogeneity**)
  - Use components to **extend** the behaviour of the control loop

- **Attaching the autonomic control loops to services**
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (**efficiency**)

- **Allowing to dynamically reconfigure the autonomic control loop**
  - Add/remove components at runtime as needed (**flexibility**)

![Diagram of MAPE loop components](image)
How?

- Implementing an autonomic control loop
  - Support for autonomicity
- Encapsulating each phase of the MAPE loop as a component
  - Leverage the technology of services to a common ground (heterogeneity)
  - Use components to extend the behaviour of the control loop
- Attaching the autonomic control loops to services
  - Define interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to the involved services (efficiency)
- Allowing to dynamically reconfigure the autonomic control loop
  - Add/remove components at runtime as needed (flexibility)
# Table of Contents

1. **Introduction**

2. **Context**

3. **Goals and Contribution**

4. **State of the Art**

5. **Design**

6. **Implementation**

7. **Validation**

8. **Conclusions**
4. State of the Art

Landscape of tools
4. State of the Art

Landscape of tools

**Analysis**
- WSLA
- WS-Policy
- WS-Agreement
- Everest+
- QoS-aware Selection
- K-Means clustering
- QoS-aware re-binding
- Genetic Alg.

**Monitoring**
- Autopilot
- ARM
- Magpie
- Ganglia
- WSDM
- Cilia
- Dynamo
- Cremona
- GMA
- Lattice
- Cloud Status
- FScript/FPath
- iPOJO
- SAFRAN
- FraSCAti

**Planning**

**Execution**
Landscape of tools

Analysis
- Kieker (2009)
- VRESCo (2007)
- WildCat (2005)

Monitoring
- PADRES (2010)
- RESERVOIR (2009)

Planning

Execution
4. State of the Art

Landscape of tools

- Analysis
  - CAPPUCINO (2010)
  - Ceylon (2010)
  - StarMX (2009)
  - Entropy (2009)
  - Dynaco (2006)
  - SAFDIS (2010)

- Monitoring

- Planning

- Execution
4. State of the Art

Autonomic Loops for Services

- **CAPPUCINO**
  - Context-aware adaptation for services
  - Control loop distributed in ubiquitous environments
  - Dynamically reconfiguration of communication protocols and collectors

- **Ceylon**
  - Development of autonomic managers
  - Composition from smaller autonomic tasks
  - Dynamic reconfiguration of the autonomic manager according to the goals described

- **SAFDIS**
  - Service-based adaptation service
  - Distributed collaboration for adaptation planning
  - Migration of services as adaptation actions
## Summary

<table>
<thead>
<tr>
<th></th>
<th>Autopilot</th>
<th>Rainbow</th>
<th>StarMX</th>
<th>Entropy</th>
<th>Dynaco</th>
<th>Cappucino</th>
<th>Ceylon</th>
<th>SAFDIS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Monitoring</strong></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td><strong>Analysis</strong></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td><strong>Planning</strong></td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td><strong>Execution</strong></td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td><strong>Scope</strong></td>
<td>grids</td>
<td>generic</td>
<td>java apps.</td>
<td>virtualised resources</td>
<td>components</td>
<td>ubiquitous services</td>
<td>development of autonomous applications</td>
<td>services</td>
</tr>
<tr>
<td><strong>Extensibility</strong></td>
<td>-</td>
<td>++</td>
<td>++</td>
<td>-</td>
<td>++</td>
<td>-</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td><strong>Flexibility</strong></td>
<td>runtime on/off sensors</td>
<td>design</td>
<td>design</td>
<td>-</td>
<td>design</td>
<td>runtime</td>
<td>design</td>
<td></td>
</tr>
<tr>
<td><strong>Comm.</strong></td>
<td>mediation middleware</td>
<td>?</td>
<td>JMX</td>
<td>?</td>
<td>?</td>
<td>SCA REST</td>
<td>middleware for events</td>
<td>?</td>
</tr>
</tbody>
</table>
5. Design

Table of Contents

1. Introduction
2. Context
3. Goals and Contribution
4. State of the Art
5. Design
6. Implementation
7. Validation
8. Conclusions
The Big Picture

Implementation of each phase of the MAPE autonomic control loop as a component.

- Monitoring, Analysis, Planning, Execution
- Attach the MAPE components to the service that they manage
- Made that capabilities accessible through pre-defined interfaces

Regular services turn into a managed service

- The service is improved with additional interfaces
- The interfaces allow to interact with the monitoring and management capabilities

Cristian Ruz (INRIA Oasis Team)
The Big Picture

Implementation of each phase of the MAPE autonomic control loop as a component.

- Monitoring, Analysis, Planning, Execution
- Attach the MAPE components to the service that they manage
- Made that capabilities accessible through pre-defined interfaces

Regular services turns into a managed service

- The service is improved with additional interfaces
- The interfaces allows to interact with the monitoring and management capabilities
The Big Picture

Implementation of each phase of the MAPE autonomic control loop as a component.

- Monitoring, Analysis, Planning, Execution
- Attach the MAPE components to the service that they manage
- Made that capabilities accessible through pre-defined interfaces

Regular services turn into a managed service

- The service is improved with additional interfaces
- The interfaces allow to interact with the monitoring and management capabilities

Service

Managed Service

Service A

actions

metrics

SLOs

actions

design time addition

metrics

actions
Inside the managed service

The framework is itself a component-based application
Inside the managed service

The framework is itself a component-based application
Inside the managed service

The framework is itself a component-based application

Managed Service A

Analysis

Monitored

Monitoring

Service A

metrics

SLOs

actions

metrics

monitoring data

actions
Inside the managed service

The framework is itself a component-based application
Inside the managed service

The framework is itself a component-based application
Monitoring components connected through the application

- Interacting through their interfaces
- Adapted to the monitoring needs/requirements of each service

<table>
<thead>
<tr>
<th>metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>getMetricList()</td>
</tr>
<tr>
<td>getMetric(metricName)</td>
</tr>
<tr>
<td>subscribe(metricName)</td>
</tr>
<tr>
<td>unsubscribe(metricName)</td>
</tr>
<tr>
<td>insertMetric(metric, metricName)</td>
</tr>
<tr>
<td>removeMetric(metricName)</td>
</tr>
</tbody>
</table>

management of sensors (polling/listening) to collect information

computation of metrics from obtained values

(optional) storage of obtained values or metrics

(optional) communication with other monitoring components

metrics-service
metrics-reference1
metrics-reference2
metrics-referencei
...
Monitoring Example

Interaction between monitoring components to compute metrics

- Monitoring backbone following the composition
- Monitoring components collaborate to compute a metric
- The actual way to compute the metric may be different for each service
Monitoring Example

Interaction between monitoring components to compute metrics

- Monitoring backbone following the composition
- Monitoring components collaborate to compute a metric
- The actual way to compute the metric may be different for each service
Monitoring Example

Interaction between monitoring components to compute metrics

- **Monitoring backbone** following the composition
- Monitoring components collaborate to compute a metric
- The actual way to compute the metric may be different for each service

\[
e(c) = e(a) + e(b) + e(d) + e(e)
\]

- \( e(a) = 6 \text{ kW} + e(b) \)
- \( e(b) = 2 \text{ kW} \)
- \( e(d) = 8 \text{ kW} \)
- \( e(e) = 2 \text{ kW} \)
Analysis

Checking of compliance to SLA requirements

- Expressed as SLOs (Service Level Objectives)
- Computed from metrics obtained from the monitoring component

### Table: SLOs and Operations

<table>
<thead>
<tr>
<th>SLOs</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>addSLO(SLO, sloName)</td>
<td></td>
</tr>
<tr>
<td>removeSLO(metricName)</td>
<td></td>
</tr>
<tr>
<td>enableSLO(metricName)</td>
<td></td>
</tr>
<tr>
<td>disableSLO(metricName)</td>
<td></td>
</tr>
</tbody>
</table>

### Alarm

<table>
<thead>
<tr>
<th>alarm</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>notify(alarmType, condition)</td>
<td></td>
</tr>
</tbody>
</table>
Analysis Example

Analysis components use the *monitoring backbone* to obtain the metrics they need to perform SLO checking

- Different Analyzers may check different conditions without interfering with others
Analysis Example

Analysis components use the *monitoring backbone* to obtain the metrics they need to perform SLO checking.

- Different Analyzers may check different conditions without interfering with others.

```
SLO:
cost(A) < 30

Metric:
cost(A) = cost(B) + cost(C)
```

```
Service B
Mb

Service A
Ma

Service C
Mc
```

```
Metrics:
cost(B) = ...
```

```
Metrics:
cost(C) = ...
```

```
cost(A) = 28
```

```
cost(B) = 18
```

```
cost(C) = 10
```
Analysis Example

Analysis components use the *monitoring backbone* to obtain the metrics they need to perform SLO checking.

- Different Analyzers may check different conditions without interfering with others.
Planning

Implementation of strategies or decision algorithms

- Activated upon an alarm from the Analysis phase
- Generates a set of actions to apply on the system

<table>
<thead>
<tr>
<th>Planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarm-service [alarm]</td>
</tr>
<tr>
<td>metrics-reference [metrics]</td>
</tr>
<tr>
<td>actions-reference [actions]</td>
</tr>
</tbody>
</table>

| actions |
| sendActions(actionList) |
| sendAction(action) |

association of faulting condition and severity level to an appropriate strategy

support for one or more strategies

generation of (list of) actions in a defined format
Planning Example

Uses the Monitoring components to get the information it may need
Planning Example

Uses the Monitoring components to get the information it may need

Strategy:
getMetric(cost,B)=?
getMetric(cost,C1)=?
getMetric(cost,C2)=?
output: replace(C1,C2);

Alarm:
cost(A) >= 30

SLO:
cost(A) < 30

cost(B)=18

cost(C1)=20

cost(C2)=5
Planning Example

Uses the Monitoring components to get the information it may need

- **SLO:** cost(A) < 30
- **Alarm:** cost(A) >= 30
- **Strategy:**
  - getMetric(cost,B)=?
  - getMetric(cost,C1)=?
  - getMetric(cost,C2)=?
  - output: replace(C1,C2);
- **Action:** replace(C1,C2);
- **Costs:**
  - cost(A) = 18
  - cost(B) = 18
  - cost(C1) = 20
  - cost(C2) = 5
Execute actions on the service according to the specific means allowed

- Connected to support localized actions
- Must translate the commands to concrete actions
Execution Example

Actions can be propagated to the appropriate service

- Specific ways to execute actions depend on the service
- Encapsulated in the execution components

Actions:
- replace(C1,C2);
- unbind(B,b1);
- set(C2,threads,10);

Cristian Ruz (INRIA Oasis Team)
Execution Example

Actions can be propagated to the appropriate service

- Specific ways to execute actions depend on the service
- Encapsulated in the execution components

Actions:
- replace(C1, C2);
- unbind(B, b1);
- set(C2, threads, 10);

---

Cristian Ruz (INRIA Oasis Team)  Autonomic M&M of Comp-based Services  23/06/2011  32 / 52
5. Design

Execution Example

Actions can be propagated to the appropriate service

- Specific ways to execute actions depend on the service
- Encapsulated in the execution components

Actions:
- replace(C1,C2);
- unbind(B,b1);
- set(C2,threads,10);

Cristian Ruz (INRIA Oasis Team)  Autonomic M&M of Comp-based Services
5. Design

Execution Example

Actions can be propagated to the appropriate service

- Specific ways to execute actions depend on the service
- Encapsulated in the execution componentes

```
Actions: replace(C1,C2);
unbind(B,b1);
set(C2,threads,10);
```
Execution Example

Actions can be propagated to the appropriate service

- Specific ways to execute actions depend on the service
- Encapsulated in the execution componentes
Execution Example

Actions can be propagated to the appropriate service

- Specific ways to execute actions depend on the service
- Encapsulated in the execution components
Summary

Design presented in a generic way using SCA
- Implementable in an SCA runtime
- Basic interfaces may be extended as needed
Table of Contents

1. Introduction
2. Context
3. Goals and Contribution
4. State of the Art
5. Design
6. Implementation
7. Validation
8. Conclusions
Implementation

Goals

- Provide a concrete instantiation of the framework
- Taking into account the generic design
  - But considering the features of the runtime
- Other implementations can be carried on
Technical Background

- Grid Component Model (GCM)
  - Extension of the Fractal Component Model
    - Support for distributed deployment
    - Support for collective communications
    - Separation between F and NF concerns (Naoumenko, 2010)
  - Using the GCM/ProActive reference implementation
    - Based on asynchronous active objects, and futures
    - JMX-based instrumentation

![Diagram of Grid Component Model (GCM)]
Technical Background

- **Grid Component Model (GCM)**
  - Extension of the Fractal Component Model
  - Support for distributed deployment
  - Support for collective communications
  - Separation between F and NF concerns (Naoumenko, 2010)
- Using the GCM/ProActive reference implementation
  - Based on asynchronous active objects, and futures
  - JMX-based instrumentation
Technical Background

- Grid Component Model (GCM)
  - Extension of the Fractal Component Model
  - Support for distributed deployment
  - Support for collective communications
  - Separation between F and NF concerns (Naoumenko, 2010)

- Using the GCM/ProActive reference implementation
  - Based on asynchronous active objects and futures
  - JMX-based instrumentation
Technical Background

- Grid Component Model (GCM)
  - Extension of the Fractal Component Model
  - Support for distributed deployment
  - Support for collective communications
  - Separation between F and NF concerns (Naoumenko, 2010)
- Using the GCM/ProActive reference implementation
  - Based on asynchronous active objects and futures
  - JMX-based instrumentation
Technical Background

- **Grid Component Model (GCM)**
  - Extension of the Fractal Component Model
  - Support for distributed deployment
  - Support for collective communications
  - Separation between F and NF concerns (Naoumenko, 2010)

- Using the GCM/ProActive reference implementation
  - Based on asynchronous active objects, and futures
  - JMX-based instrumentation
6. Implementation

Technical Background

- Grid Component Model (GCM)
  - Extension of the Fractal Component Model
  - Support for distributed deployment
  - Support for collective communications
  - Separation between F and NF concerns (Naoumenko, 2010)
- Using the GCM/ProActive reference implementation
  - Based on asynchronous active objects, and futures
  - JMX-based instrumentation
Technical Background

- **Grid Component Model (GCM)**
  - Extension of the Fractal Component Model
  - Support for distributed deployment
  - Support for collective communications
  - Separation between F and NF concerns (Naoumenko, 2010)
- **Using the GCM/ProActive reference implementation**
  - Based on asynchronous active objects, and futures
  - JMX-based instrumentation
6. Implementation

Mapping from SCA to GCM

- Following the SCA design
  - MAPE Components attached to GCM membranes
  - Using NF server and client interfaces

- Technical contributions
  - Implementation of each MAPE component
  - Definition of an API to manipulate MAPE components

---

**Diagram:**

- Managed Service A
- Analysis
- Monitoring
- Planning
- Execution
- Service A

**Axes:**

- Metrics
- SLOs
- Actions

**Mentions:**

- Cristian Ruz (INRIA Oasis Team)
- Autonomic M&M of Comp-based Services
6. Implementation

Mapping from SCA to GCM

- Following the SCA design
  - MAPE Components attached to GCM membranes
  - Using NF server and client interfaces

- Technical contributions
  - Implementation of each MAPE component
  - Definition of an API to manipulate MAPE components
6. Implementation

Mapping from SCA to GCM

- Following the SCA design
  - MAPE Components attached to GCM membranes
  - Using NF server and client interfaces

- Technical contributions
  - Implementation of each MAPE component
  - Definition of an API to manipulate MAPE components
Mapping from SCA to GCM

- Following the SCA design
  - MAPE Components attached to GCM membranes
  - Using NF server and client interfaces
- Technical contributions
  - Implementation of each MAPE component
  - Definition of an API to manipulate MAPE components
6. Implementation

Monitoring Component

Collection, storage, computation of metrics
- Collecting JMX events from GCM/ProActive
- Supports insertion/removal of metrics
- Allows access to metrics via push/pull methods

- Improved instrumentation of GCM/ProActive
6. Implementation

Analysis Component

Checking of conditions and generation of alarms

- Subscribes or query to the Monitoring Component
- Sends an Alarm object if necessary
- SLO Representation: \( \langle \text{metric, cond, threshold} \rangle \)

```
[SLO]
    ... "<", 30> )

➀
➁
➂
➃
➄ ➅
➆
➇
```

```
subscribe("responseTime")
responseTime = 40s.
alarm(FAULT, <responseTime,"<",30>)
```

```
<table>
<thead>
<tr>
<th>SLO</th>
<th>Enabled</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;responseTime,&quot;&lt;&quot;,30&gt;</td>
<td>true</td>
<td>subscribe</td>
</tr>
<tr>
<td>&lt;freeSpace,&quot;&gt;&quot;,0.1&gt;</td>
<td>true</td>
<td>pull, 10m</td>
</tr>
<tr>
<td>&lt;cost,&quot;&lt;&quot;,50&gt;</td>
<td>false</td>
<td>subscribe</td>
</tr>
</tbody>
</table>
```

Cristian Ruz (INRIA Oasis Team)  Autonomic M&M of Comp-based Services  23/06/2011  39 / 52
6. Implementation

Planning Component

Execution of planning algorithms (strategies)

- Associates an Alarm to one or more strategies
- Support for multiple strategies using multicast interfaces
  - Selection, parallel execution of strategies
- Information obtained from the Monitoring layer

<table>
<thead>
<tr>
<th>metric</th>
<th>level</th>
<th>planner</th>
</tr>
</thead>
<tbody>
<tr>
<td>responseTime</td>
<td>faulting</td>
<td>planner1</td>
</tr>
<tr>
<td>cost</td>
<td>faulting</td>
<td>planner2</td>
</tr>
<tr>
<td>freeSpace</td>
<td>preventive</td>
<td>planner3</td>
</tr>
</tbody>
</table>
Execution Component

Execution of Actions over the component/service

- Support to execute reconfiguration actions on other components
- Support for start/stop, bind/unbind, deploy/undeploy, migrate, ...

- Scripting language PAGCMScript (extension of FScript)
Table of Contents

1. Introduction
2. Context
3. Goals and Contribution
4. State of the Art
5. Design
6. Implementation
7. Validation
8. Conclusions
Micro-benchmark

Execution of an application that generates computations and updates in the MAPE components

- Local and distributed execution
- 14% overhead. Worst-case situation
- Actual value depends on strategies implemented
Use Case

Use case exemplified: Tourism Service application

- Local and remote deployments, possibly in different infrastructures
- Setting up the insertion of the control loop
- Autonomic migration
- Distinct control loops
7. Validation

Use Case: Setting up the system

- Inserting of MAPE components is handled by the API
- Creation of the required NF Bindings, using the GCM controllers
- NF Bindings follow the functional architecture
7. Validation

Use Case: Setting up the system

- Inserting of MAPE components is handled by the API
  - Creation of the required NF Bindings, using the GCM controllers
  - NF Bindings follow the functional architecture
Use Case: Setting up the system

- Inserting of MAPE components is handled by the API
  - Creation of the required NF Bindings, using the GCM controllers
  - NF Bindings follow the functional architecture
Use Case: Autonomicity following the architecture

Autonomic action is propagated inside the composite
Use Case: Autonomicity following the architecture

Autonomic action is propagated inside the composite

Alarm: avgRT > 15

getMetric ("requestPath")

migrate("SMS", "VN-2")

migrate("SMS", "VN-2")

migrate("SMS", "VN-2")

Manager

SLO Service

Execution Service

Monitoring Service

Planning

Analysis

Monitoring

Execution

Events DB

Composer

Email

SMS

Tourism Service

getMetric ("requestPath")

migrate("SMS", "VN-2")
Use Case: Autonomicity following the architecture

Autonomic action is propagated inside the composite
Use Case: Autonomicity on external components

Inner autonomic loop

- The modification decision is taken internally
- Actions can affect external components (horizontal level)
7. Validation

Use Case: Autonomicity on external components

Inner autonomic loop

- The modification decision is taken internally
- Actions can affect external components (horizontal level)
Use Case: Autonomicity on external components

Inner autonomic loop

- The modification decision is taken internally
- Actions can affect external components (horizontal level)
7. Validation

Use Case: Autonomicity on external components

Inner autonomic loop

- The modification decision is taken internally
- Actions can affect external components (horizontal level)
Use Case: Autonomicity on external components

Inner autonomic loop

- The modification decision is taken internally
- Actions can affect external components (horizontal level)
Conclusions

Framework to provide adaptation capabilities to component-based services

- MAPE phases separated in components
- Autonomic control loops attached to each component
- Components can collaborate to implement the autonomic task
- Design presented in a generic way, and exemplified in a concrete implementation
- Flexibility to add the required management capabilities
8. Conclusions

Perspectives

Challenges in autonomic computing
- Base for experimenting with the implementation of collaborative strategies
  - Partition high level goals into subgoals
  - Hierarchical planning
- Determine safe non-conflicting planning strategies and reconfigurations

Challenges in service-oriented development
- Adaptable interfaces
  - Make more dynamic the insertion of MAPE components
- Manage multiple levels of a service-based application
  - Covering from SaaS level to infrastructure level
  - Coordinating multi-cloud environments
Autonomic Monitoring and Management of Component-based Services

Cristian RUZ

Thèse dirigée par Françoise BAUDE, au sein de l’équipe OASIS

Jury
Pr. Mireille BLAY-FORNARİNO Président du Jury
Pr. Philippe LALANDA Rapporteur
Pr. Lionel SEINTURIER Rapporteur
Dr. Romain ROUVOY Co-Rapporteur
Dr. Luc BELLISSARD Examinateur
Pr. Françoise BAUDE Directeur de Thèse

23 Juin 2011
Publications


4. C. Ruz, F. Baude, B. Sauvan *Component-based generic approach for reconfigurable management of component-based SOA applications*. In MONA 2010 - 3rd Workshop on Service Monitoring, Adaptation, and Beyond (MONA+), in conjunction with ECOWS 2010 - The 8th IEEE European Conference on Web Services, ACM Digital library, Ayia-Napa, Cyprus. December 2010
