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-FORNARINO	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

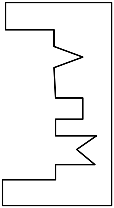
Structure

Table of Contents

Motivation

Evolution in software construction

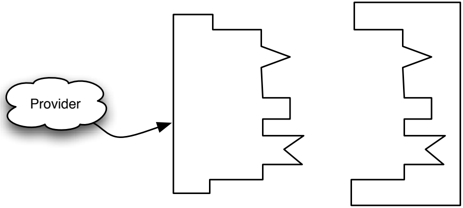
- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly



Motivation

Evolution in software construction

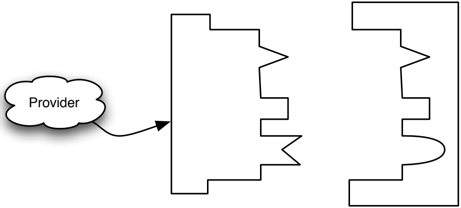
- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly



Motivation

Evolution in software construction

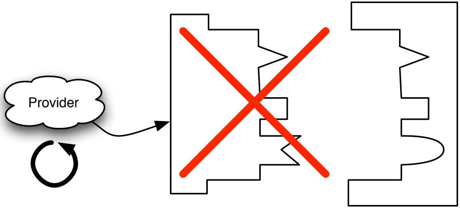
- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly



Motivation

Evolution in software construction

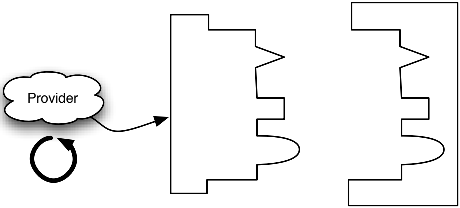
- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly



Motivation

Evolution in software construction

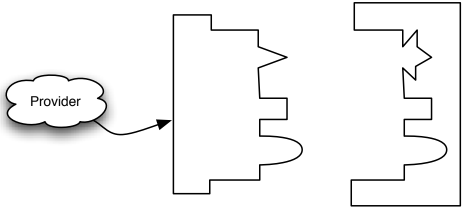
- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly



Motivation

Evolution in software construction

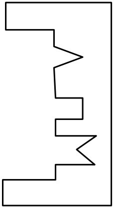
- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly



Motivation

Dynamic environment

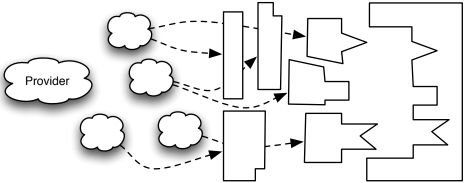
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation
- Heterogeneity and distribution



Motivation

Dynamic environment

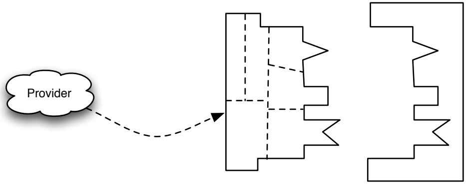
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation
- Heterogeneity and distribution
  - Transfer autonomic adaptation task to each element



Motivation

Dynamic environment

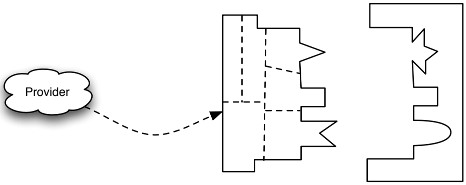
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation
- Heterogeneity and distribution
  - Transfer autonomic adaptation task to each element



Motivation

Dynamic environment

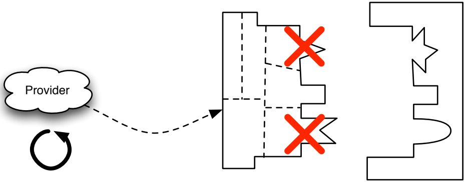
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
    - Complexity not easy for a human manager
    - Autonomic adaptation
- Heterogeneity and distribution
    - Transfer autonomic adaptation task to each element



Motivation

Dynamic environment

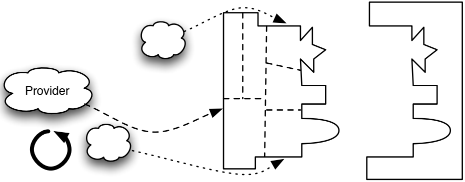
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
    - Complexity not easy for a human manager
    - Autonomic adaptation
- Heterogeneity and distribution
    - Transfer autonomic adaptation task to each element



Motivation

Dynamic environment

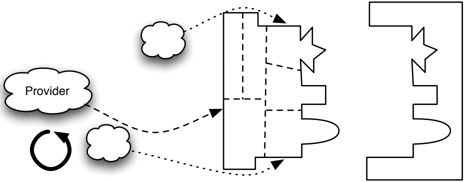
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
    - Complexity not easy for a human manager
    - Autonomic adaptation
- Heterogeneity and distribution
    - Transfer autonomic adaptation task to each element



Motivation

Dynamic environment

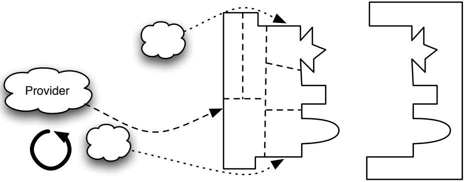
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
    - Complexity not easy for a human manager
    - Autonomic adaptation
- Heterogeneity and distribution
    - Transfer autonomic adaptation task to each element



Motivation

Dynamic environment

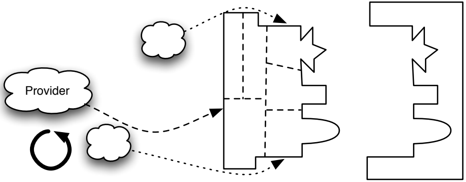
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
    - Complexity not easy for a human manager
    - Autonomic adaptation
- Heterogeneity and distribution
    - Transfer autonomic adaptation task to each element



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**

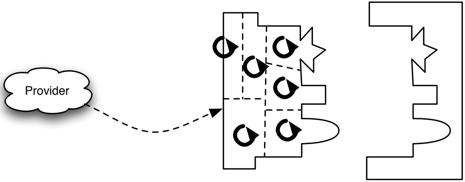


Table of Contents

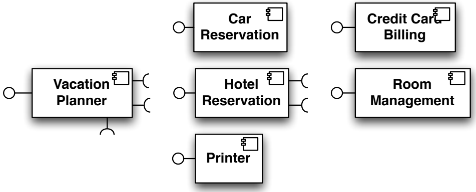
Developing dynamic adaptable software

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



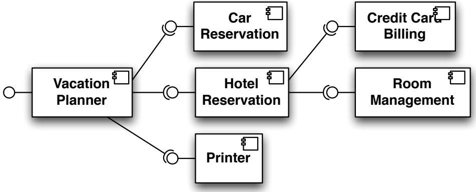
Developing dynamic adaptable software

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



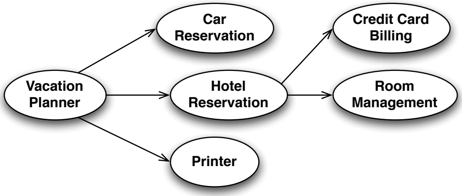
Developing dynamic adaptable software

Component-based Software Development

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

Service-orientation

- Providers 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



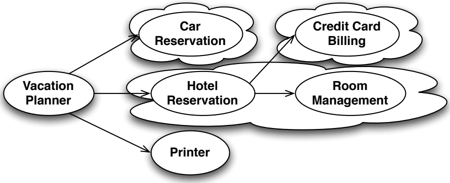
Developing dynamic adaptable software

Component-based Software Development

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

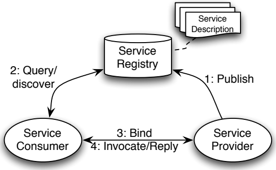
Service-orientation

- 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

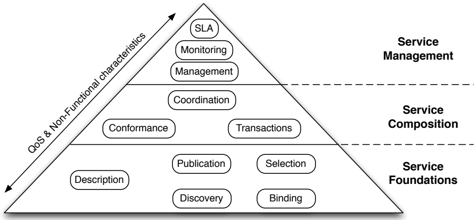


Service-Orientation

Basic model for Service Oriented Architecture



Various levels in a Service Oriented Architecture

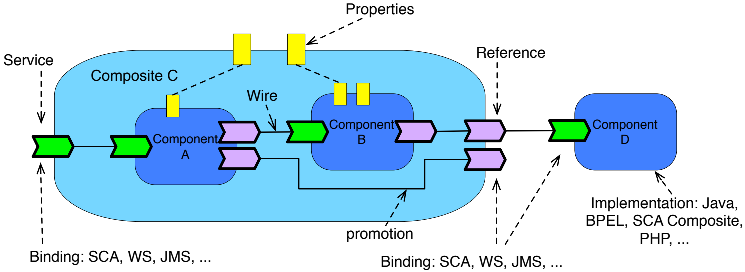


(Papazoglou, Traverso, Dustdar, Leymann, 2007)

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 back from 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

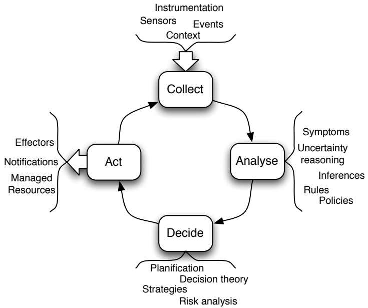
Challenges

- 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

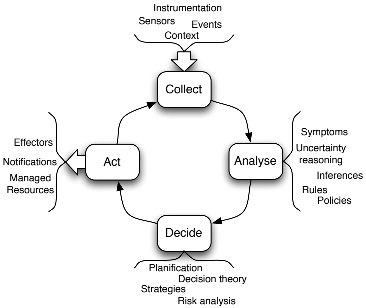
- Based on the idea of self-governing systems
- Context-awareness, and self-* properties
- Activities represented in a *feedback control loop*
- Phases in the *MAPE* autonomic control loop



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



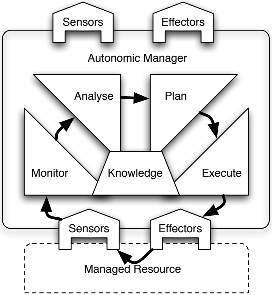


Table of Contents

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.

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

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**
- **Heterogeneity**
- **Efficiency**
- **Autonomicity**

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**
- **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 timley responses
- **Autonomicity**

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

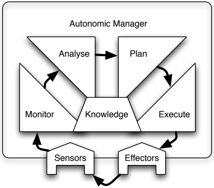
Solution Overview

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.

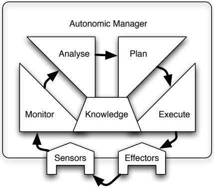
How?

- **Implementing an autonomic control loop**
  - Support for **autonomicity**
- **Encapsulating each phase of the MAPE loop as a component**
- **Attaching the autonomic control loops to services**
- **Allowing to dynamically reconfigure the autonomic control loop**



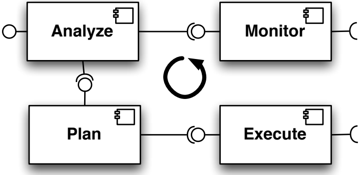
How?

- **Implementing an autonomic control loop**
  - Support for **autonomicity**
- **Encapsulating each phase of the MAPE loop as a component**
- **Attaching the autonomic control loops to services**
- **Allowing to dynamically reconfigure the autonomic control loop**



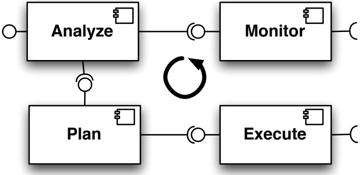
How?

- **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)



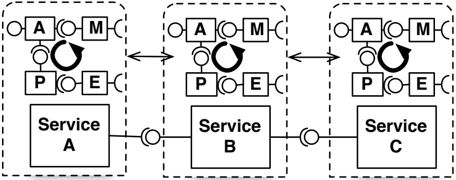
How?

- **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**
- **Allowing to dynamically reconfigure the autonomic control loop**



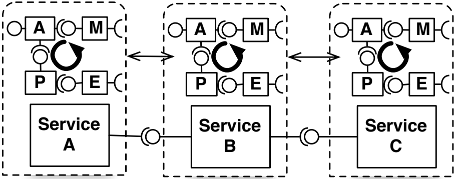
How?

- **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**)



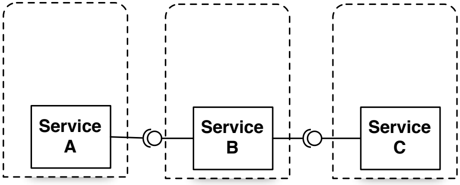
How?

- **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**)



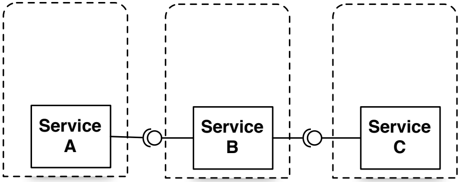
How?

- **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**)



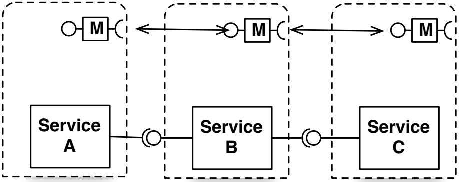
How?

- **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**)



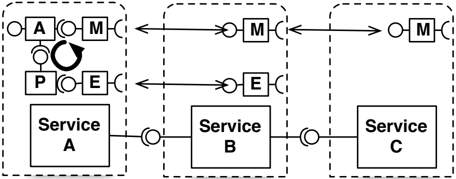
How?

- **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**)



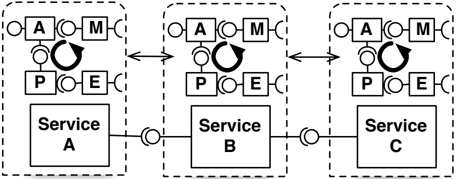
How?

- **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**)



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**)

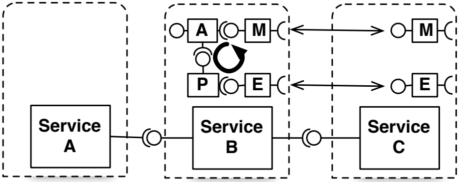
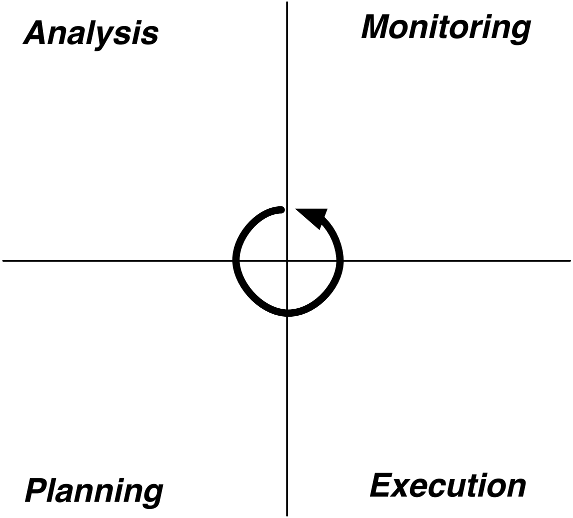
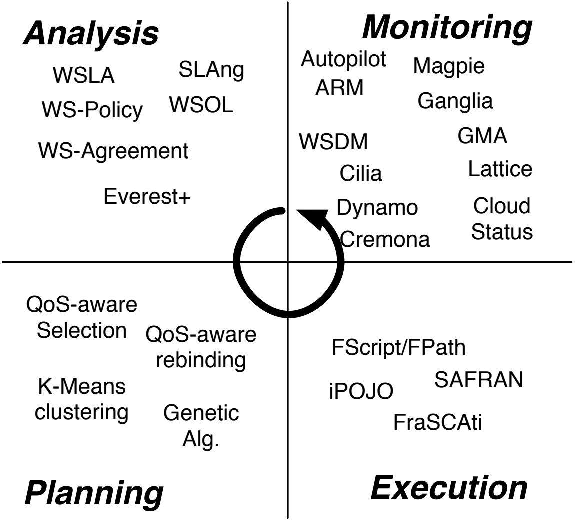


Table of Contents

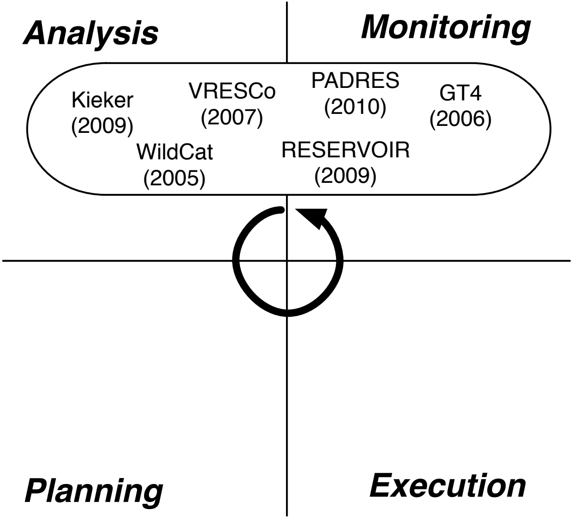
Landscape of tools



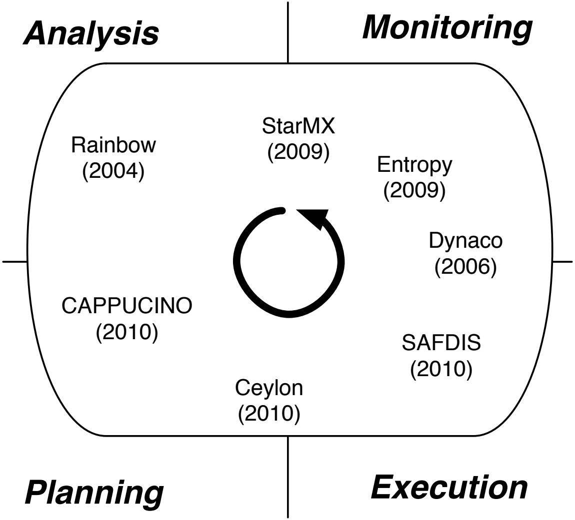
Landscape of tools



Landscape of tools



Landscape of tools



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

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

Table of Contents

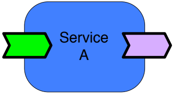
The Big Picture

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 accesible 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



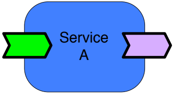
The Big Picture

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 accesible 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



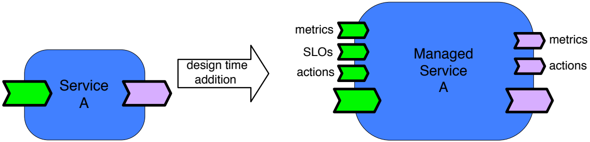
The Big Picture

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 accesible through pre-defined interfaces

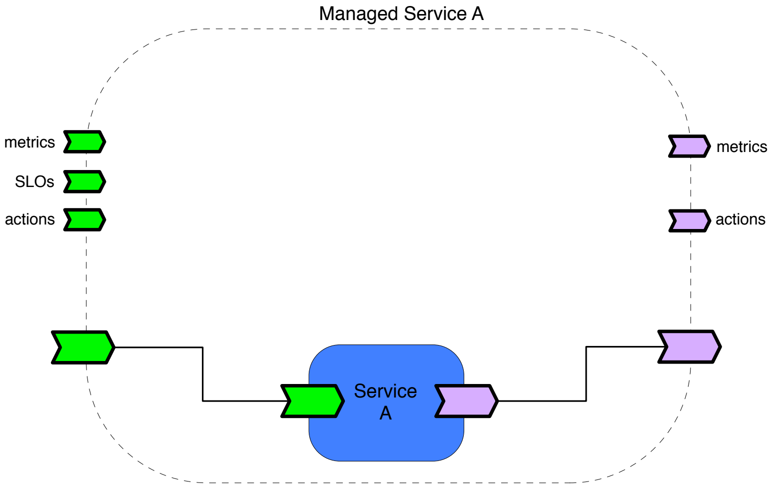
Regular services turns into a managed service

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



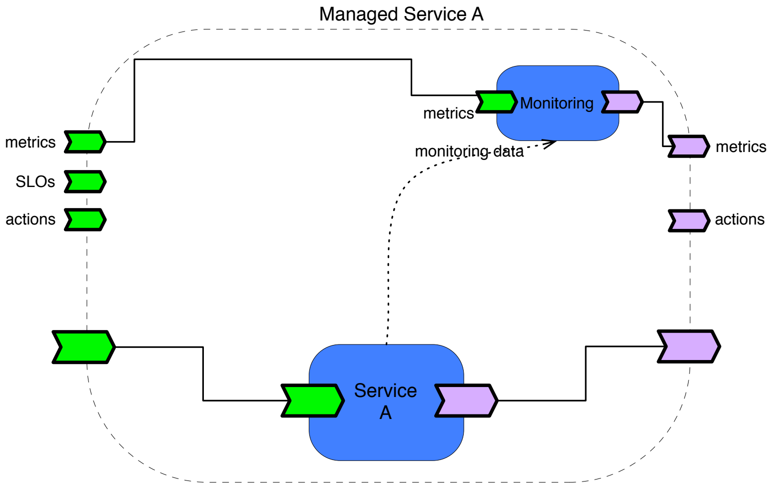
Inside the managed service

The framework is itself a component-based application



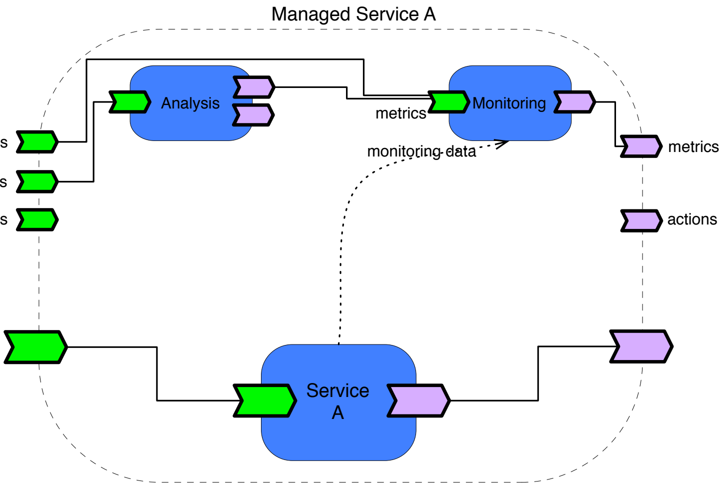
Inside the managed service

The framework is itself a component-based application



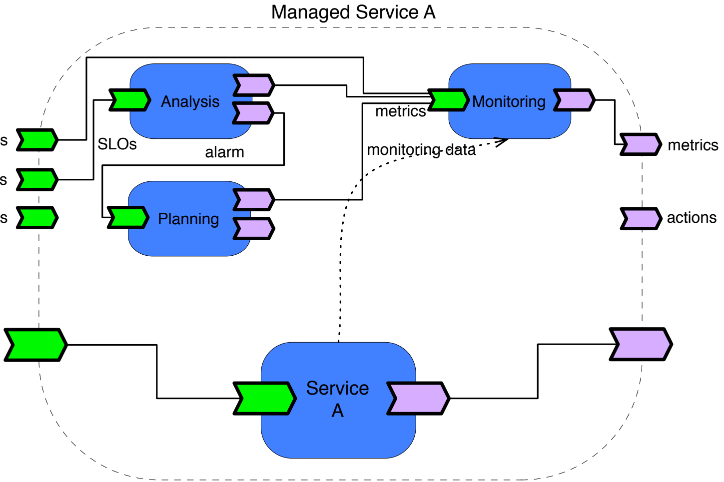
Inside the managed service

The framework is itself a component-based application



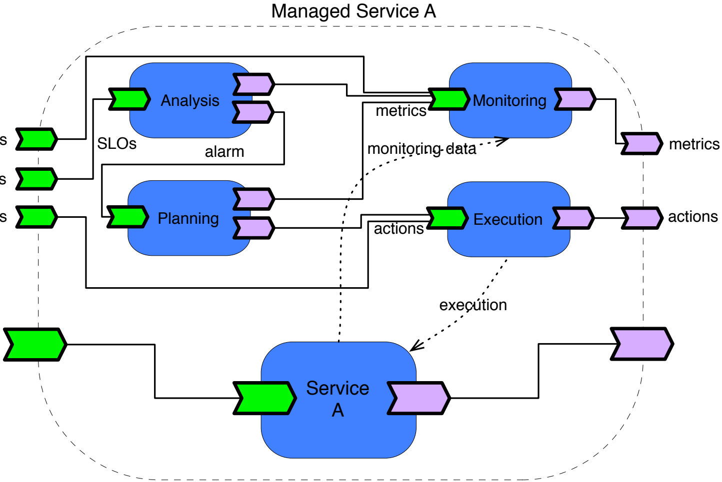
Inside the managed service

The framework is itself a component-based application



Inside the managed service

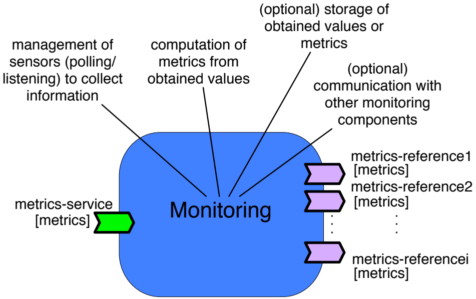
The framework is itself a component-based application



Monitoring

Monitoring components connected through the application

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

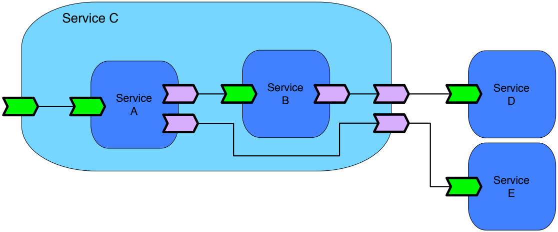


metrics
getMetricList()
getMetric(metricName)
subscribe(metricName)
unsubscribe(metricName)
insertMetric(metric, metricName)
removeMetric(metricName)

Monitoring Example

Interaction between monitoring components to compute metrics

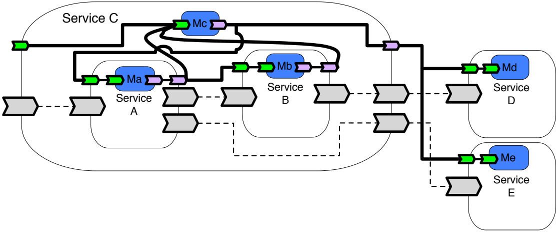
- *Monitoring backbone* following the composition
- Monitoring components collaborate to compute a metric
- The actual way to compute the metric may be different for each service



Monitoring Example

Interaction between monitoring components to compute metrics

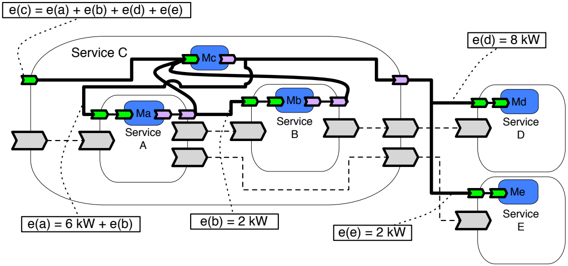
- *Monitoring backbone* following the composition
- Monitoring components collaborate to compute a metric
- The actual way to compute the metric may be different for each service



Monitoring Example

Interaction between monitoring components to compute metrics

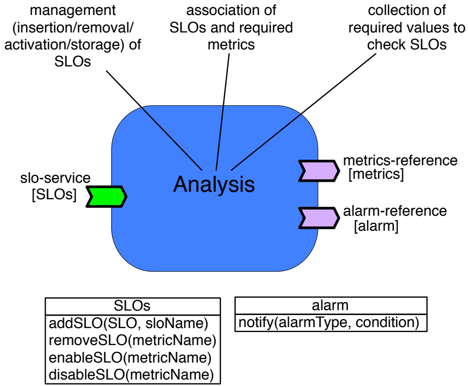
- *Monitoring backbone* following the composition
- Monitoring components collaborate to compute a metric
- The actual way to compute the metric may be different for each service



Analysis

Checking of compliance to SLA requirements

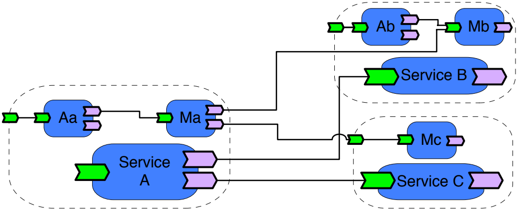
- Expressed as SLOs (Service Level Objectives)
- Computed from metrics obtained from the monitoring component



Analysis Example

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

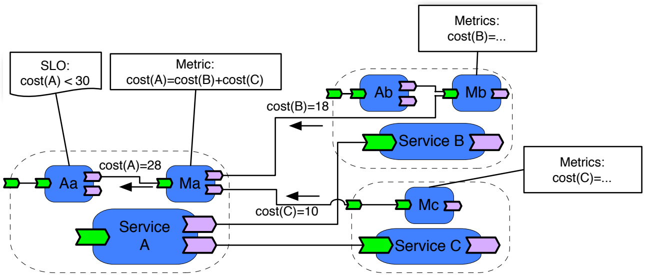
- Different Analyzers may check different conditions without interferring with others



Analysis Example

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

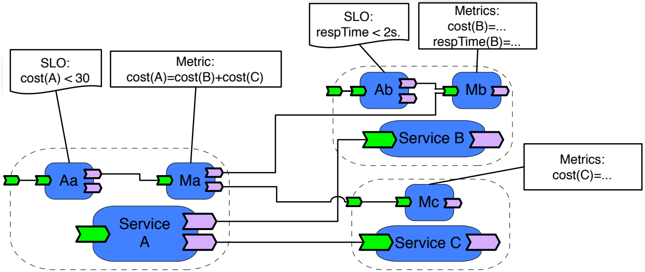
- Different Analyzers may check different conditions without interferring with others



Analysis Example

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

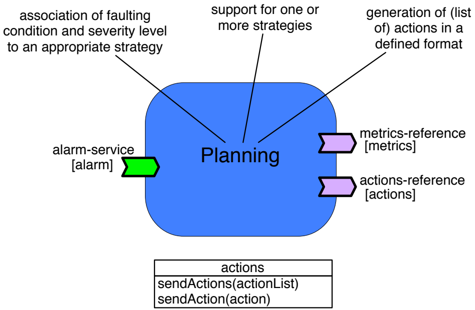
- Different Analyzers may check different conditions without interferring with others



Planning

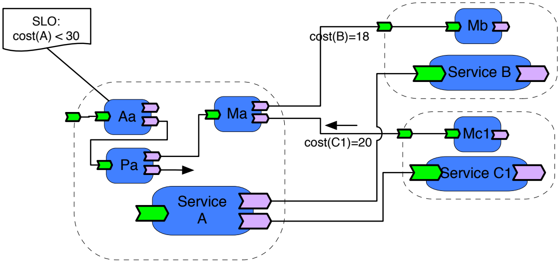
Implementation of strategies or decision algorithms

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



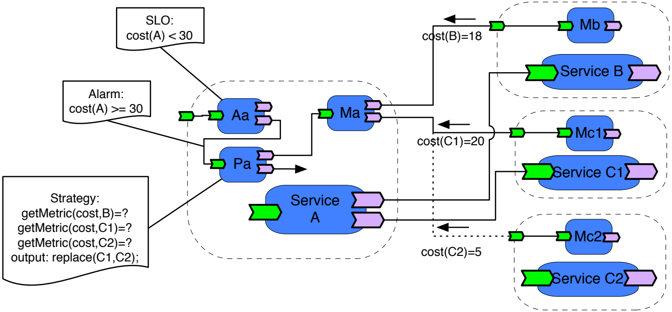
Planning Example

Uses the Monitoring components to get the information it may need



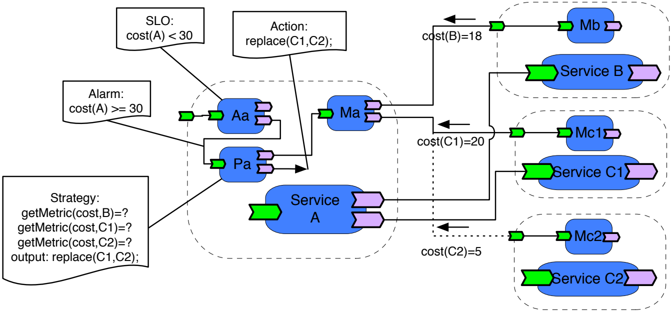
Planning Example

Uses the Monitoring components to get the information it may need



Planning Example

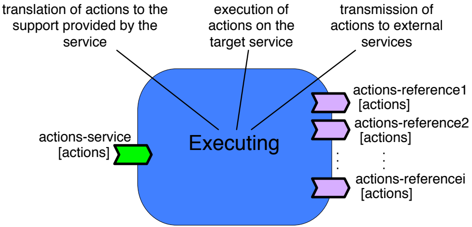
Uses the Monitoring components to get the information it may need



Execution

Execute actions on the service according to the specific means allowed

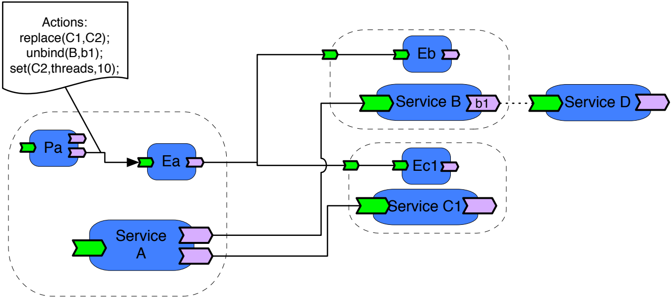
- Connected to support localized actions
- Must translate the commands to concrete actions



Execution Example

Actions can be propagated to the appropriate service

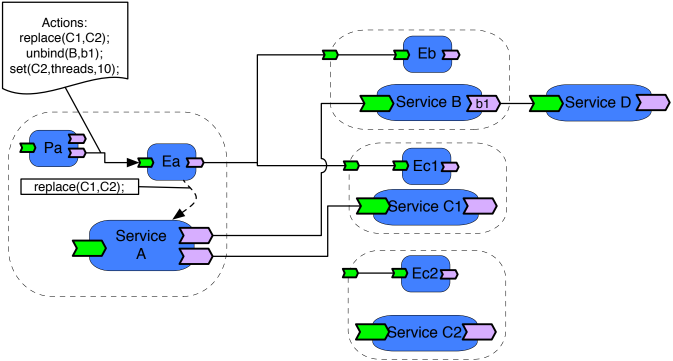
- Specific ways to execute actions depend on the service
- Encapsulated in the execution componentes



Execution Example

Actions can be propagated to the appropriate service

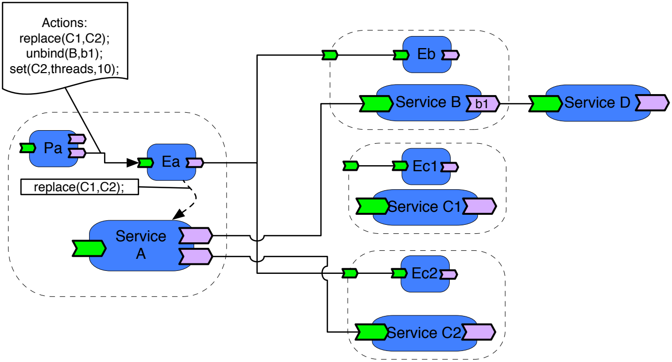
- Specific ways to execute actions depend on the service
- Encapsulated in the execution componentes



Execution Example

Actions can be propagated to the appropriate service

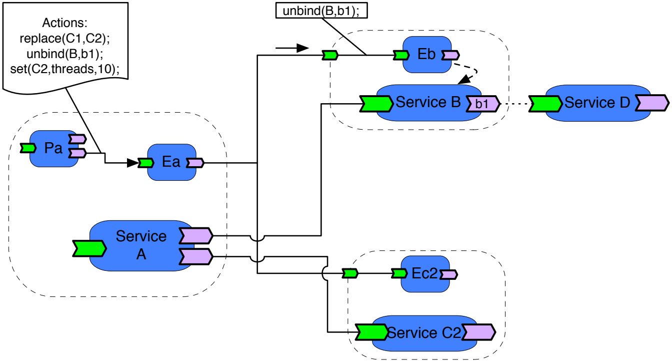
- Specific ways to execute actions depend on the service
- Encapsulated in the execution componentes



Execution Example

Actions can be propagated to the appropriate service

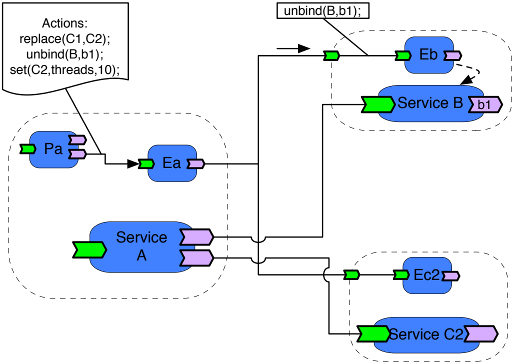
- Specific ways to execute actions depend on the service
- Encapsulated in the execution componentes



Execution Example

Actions can be propagated to the appropriate service

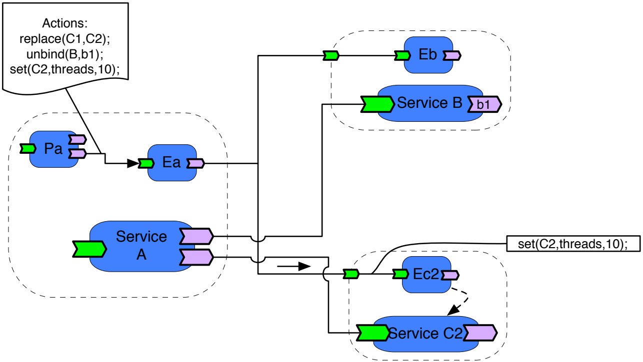
- Specific ways to execute actions depend on the service
- Encapsulated in the execution componentes



Execution Example

Actions can be propagated to the appropriate service

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



Summary

Design presented in a generic way using SCA

- Implementable in an SCA runtime
- Basic interfaces may be extended as needed

Table of Contents

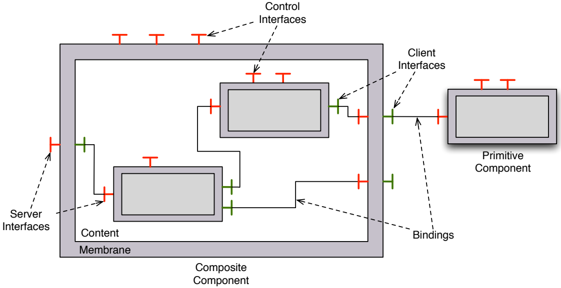
Implementation

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

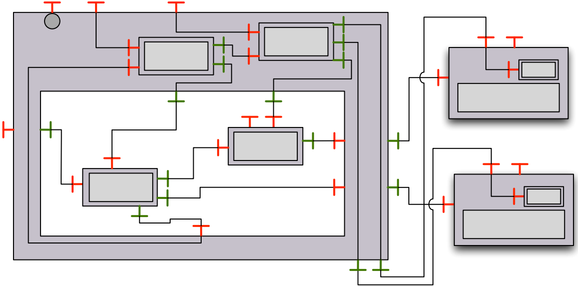
Technical Background

- 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



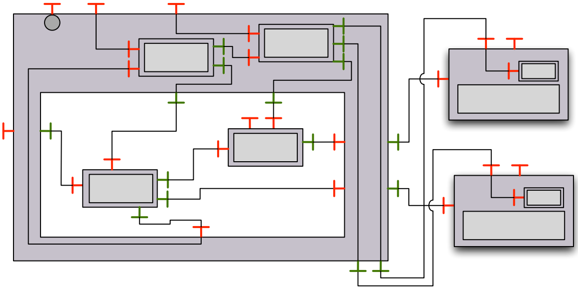
Technical Background

- 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



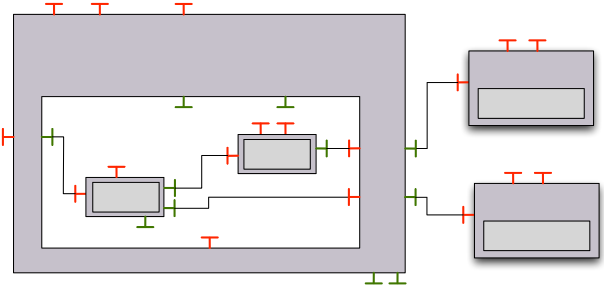
Technical Background

- 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



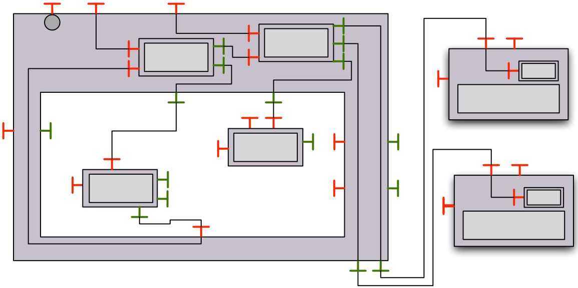
Technical Background

- 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
  - JMX-based instrumentation



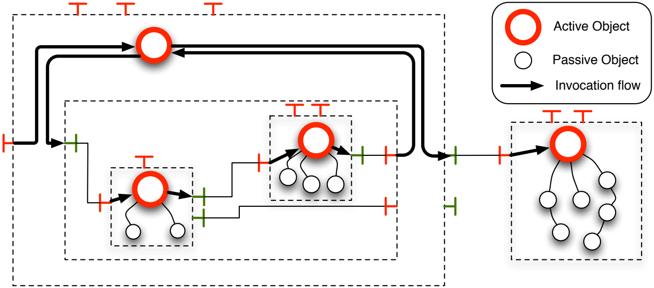
Technical Background

- 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



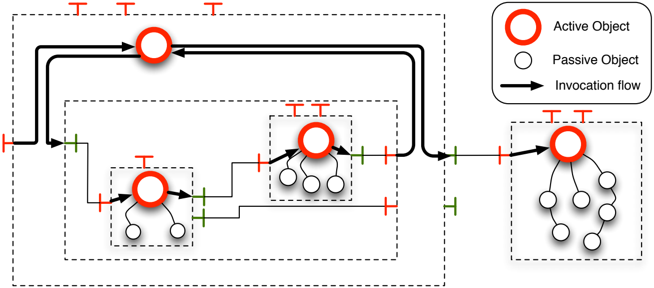
Technical Background

- 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



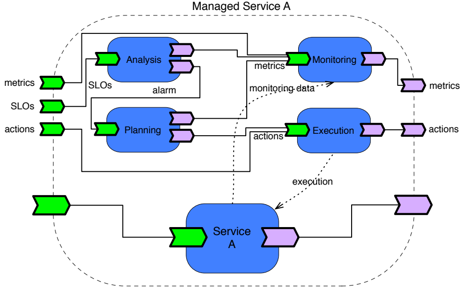
Technical Background

- 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



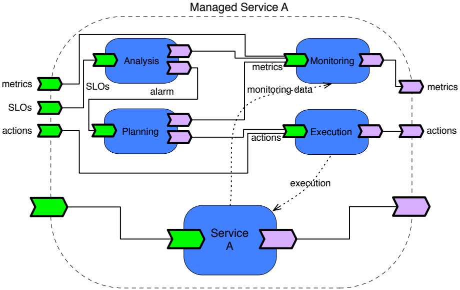
Mapping from SCA to GCM

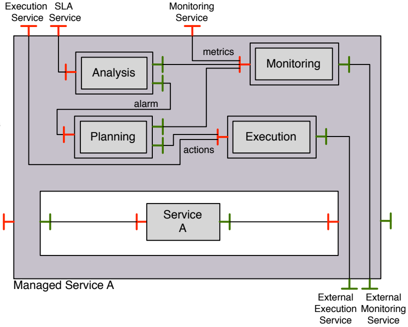
- Following the SCA design
  - MAPE Components attached to GCM membranes
  - Using NF server and client interfaces
- Technical contributions
  - Implementation of each MAPE component
  - Definition of an API to manipulate MAPE components



Mapping from SCA to GCM

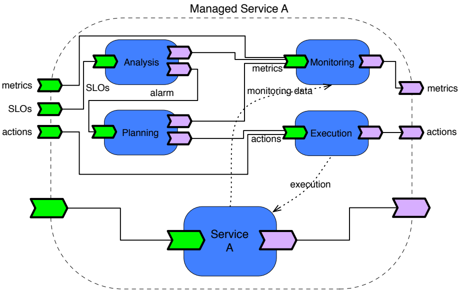
- Following the SCA design
  - MAPE Components attached to GCM membranes
  - Using NF server and client interfaces
- Technical contributions
  - Implementation of each MAPE component
  - Definition of an API to manipulate MAPE components

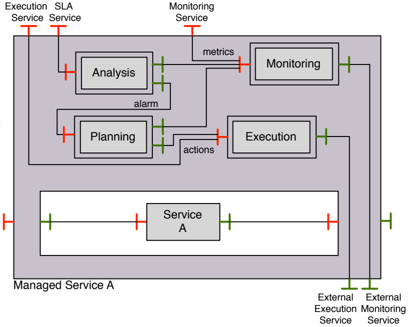




Mapping from SCA to GCM

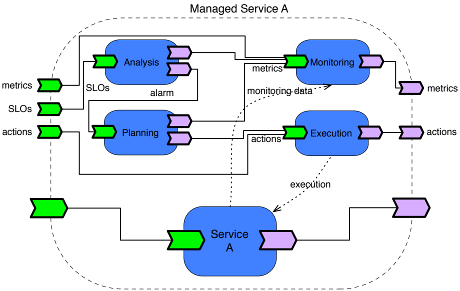
- Following the SCA design
  - MAPE Components attached to GCM membranes
  - Using NF server and client interfaces
- Technical contributions
  - Implementation of each MAPE component
  - Definition of an API to manipulate MAPE components

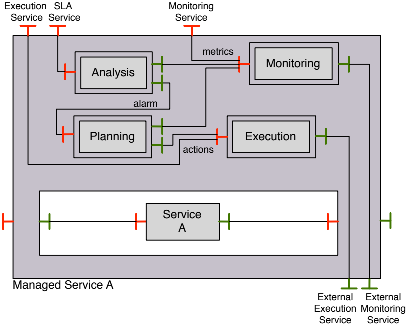




Mapping from SCA to GCM

- 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

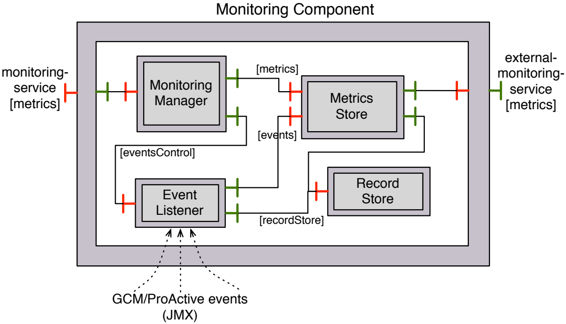




Monitoring Component

Collection, storage, computation of metrics

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

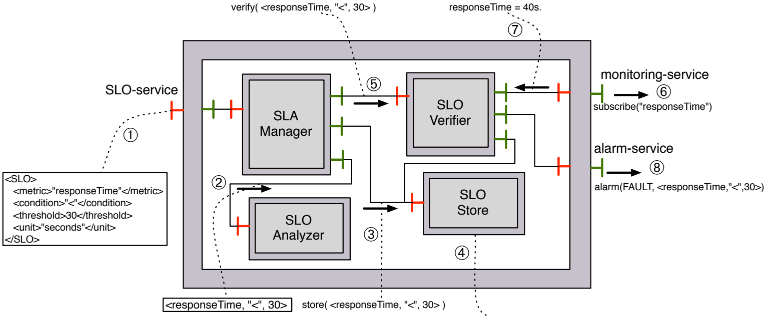


- Improved instrumentation of GCM/ProActive

Analysis Component

Checking of conditions and generation of alarms

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

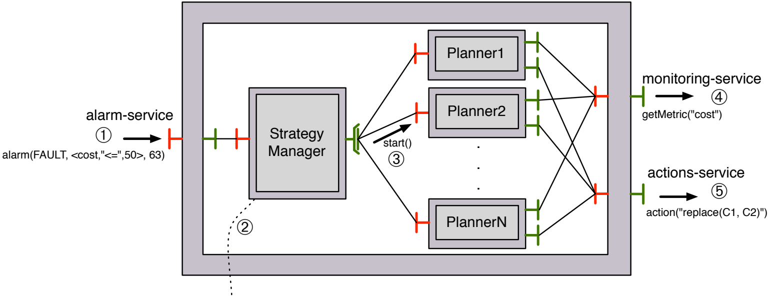


SLO	Enabled	Mode
<responseTime, "<", 30>	true	subscribe
<freeSpace, ">", 0.1>	true	pull, 10m
<cost, "<=", 50>	false	subscribe

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

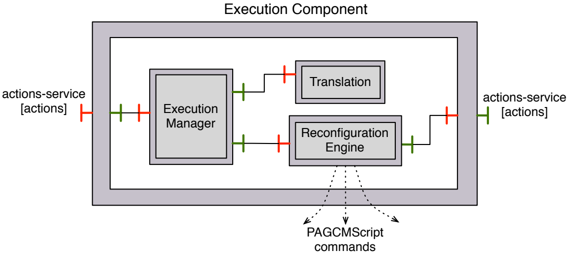


metric	level	planner
responseTime	faulting	planner1
cost	faulting	planner2
freeSpace	preventive	planner3

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

Micro-benchmark

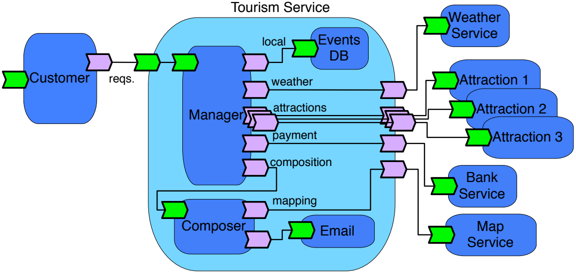
Execution of an application that generates computations and updates in the MAPE components

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

Use Case

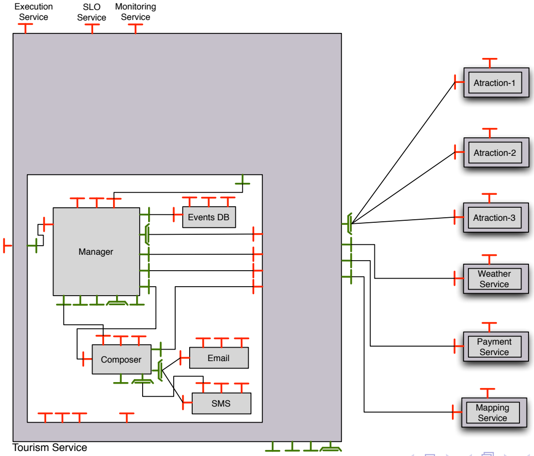
Use case exemplified: Tourism Service application

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



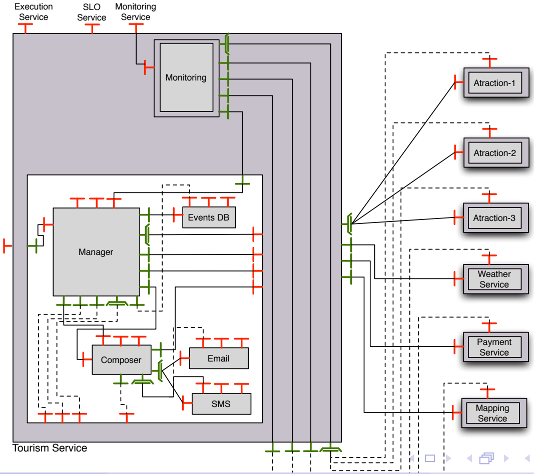
Use Case: Setting up the system

- 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



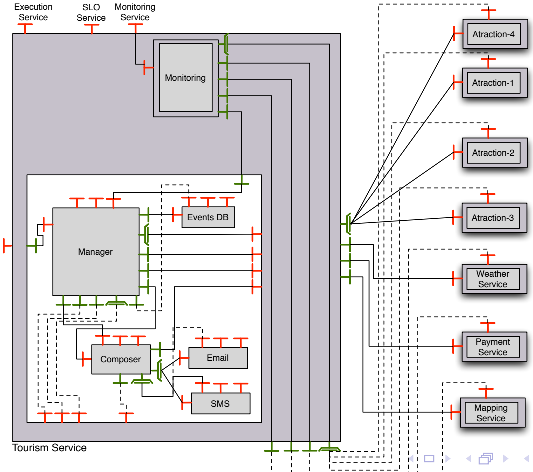
Use Case: Setting up the system

- 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



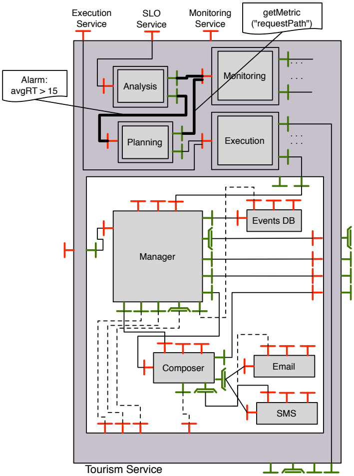
Use Case: Setting up the system

- 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



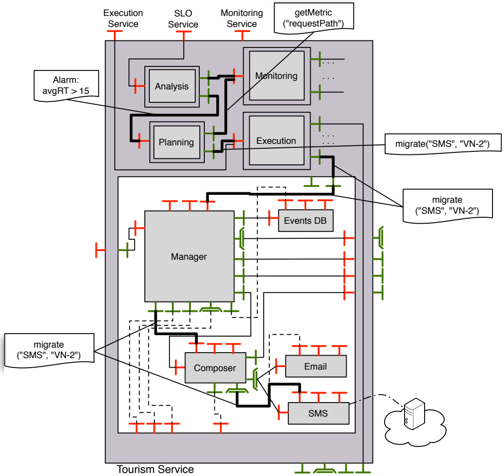
Use Case: Autonomicity following the architecture

Autonomic action is propagated inside the composite



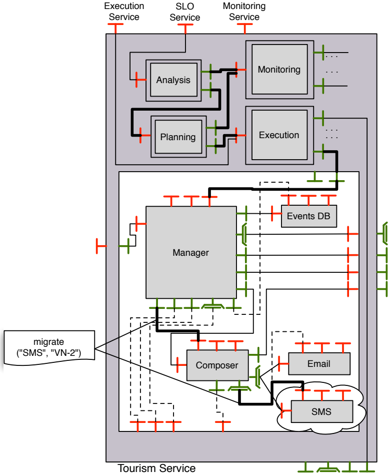
Use Case: Autonomicity following the architecture

Autonomic action is propagated inside the composite



Use Case: Autonomicity following the architecture

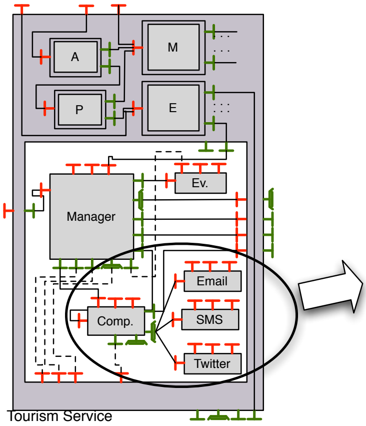
Autonomic action is propagated inside the composite



Use Case: Autonomicity on external components

Inner autonomic loop

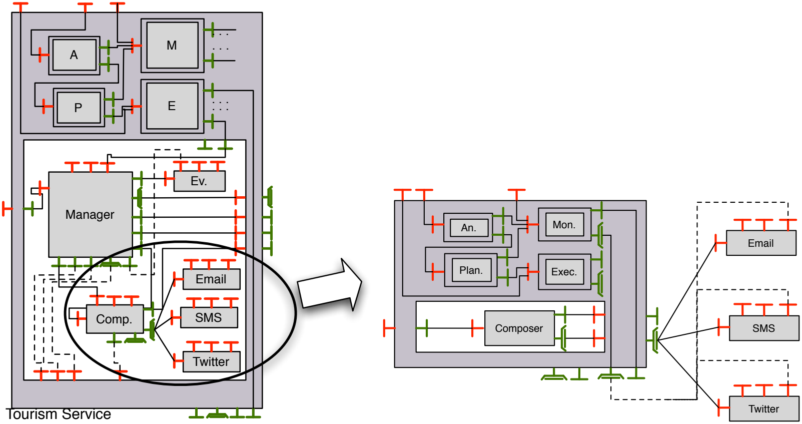
- The modification decision is taken internally
- Actions can affect external components (horizontal level)



Use Case: Autonomicity on external components

Inner autonomic loop

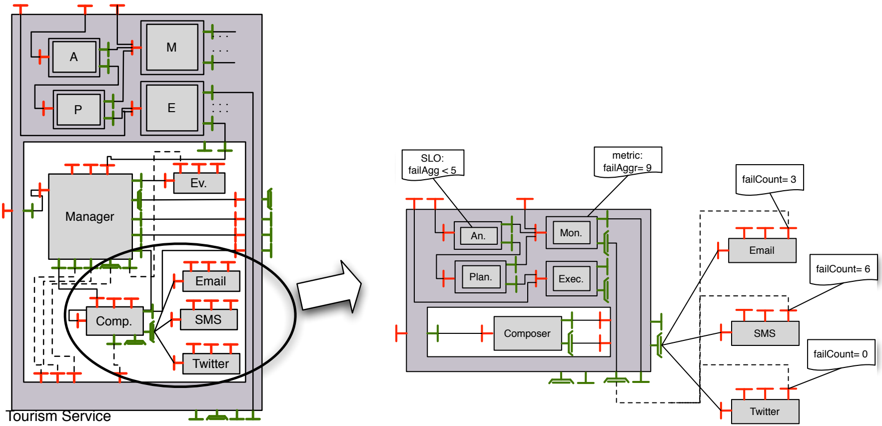
- The modification decision is taken internally
- Actions can affect external components (horizontal level)



Use Case: Autonomicity on external components

Inner autonomic loop

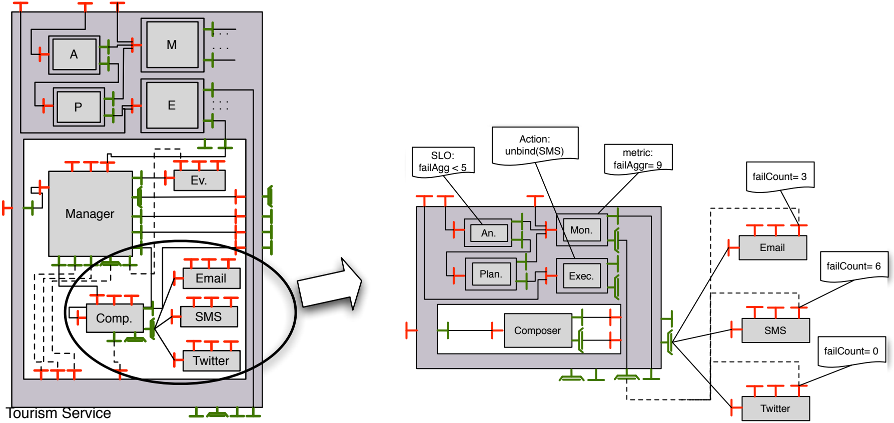
- The modification decision is taken internally
- Actions can affect external components (horizontal level)



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)

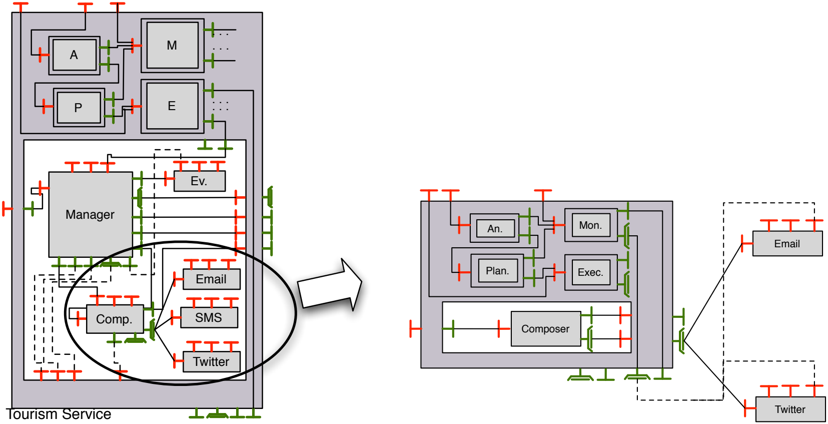


Table of Contents

Conclusions

Framework to provide adaptation capabilities to component-based services

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

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-FORNARINO	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

1. C. Ruz, F. Baude, B. Sauvan. *Flexible adaptation loop for component-based SOA applications*. In 7th International Conference on Autonomic and Autonomous Systems (ICAS 2011), Venice, Italy. May 2011. IEEE Computer Society, ISBN 978-1-61208-006-2
2. I. Filali, F. Huet, V. Legrand, E. Mathias, P. Merle, C. Ruz, R. Krummenacher, E. Simperl, C. Hammerling and J.P. Lorré. *ESB Federation for Large-Scale SOA*. In 25th ACM Int. Symposium on Applied Computing, pp 2459-2466, Sierre, Switzerland. March 2010
3. C. Ruz, F. Baude, B. Sauvan, A. Mos, A. Boulze *Flexible SOA Lifecycle on the Cloud using SCA* In 3rd International Workshop on Service-oriented Enterprise Architecture for Enterprise Engineering (SoEA4EE 2011), Helsinki, Finland. August 2011. To appear, IEEE Computer Society
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
5. C. Ruz, F. Baude, B. Sauvan. *Enabling SLA Monitoring for Component-Based SOA Applications – A Component-Based Approach*. In 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA) – Work in Progress Session. ISBN 978-3-902457-27-1. Lille, France. September 2010
6. C. Ruz, F. Baude. *Enabling SLA monitoring for component-based SOA applications*. In XXVIII International Conference of the Chilean Computer Society (SCCC'09), Santiago, Chile. November 2009.