Autonomic Management of Component-based Services

Cristian Ruz, PhD

SCADA Workshop

Ingenieur Équipe OASIS

INRIA Sophia Antipolis Méditerranée
France

July 24, 2012

PLAN

MOTIVATION

CONTEXT

FRAMEWORK

- Problem
- Design of the solution
- Implementation
- Use Cases

PERSPECTIVES

1
MOTIVATION

Motivation

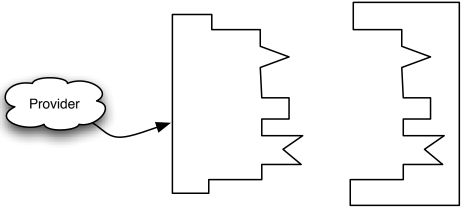
Evolution in software construction

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

Motivation

Evolution in software construction

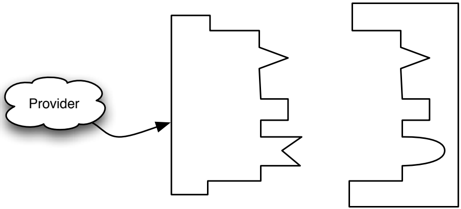
- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly



Motivation

Evolution in software construction

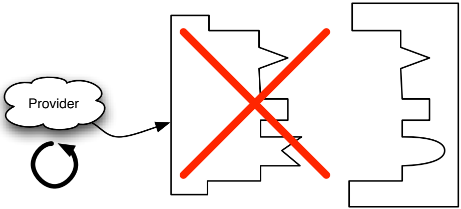
- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly



Motivation

Evolution in software construction

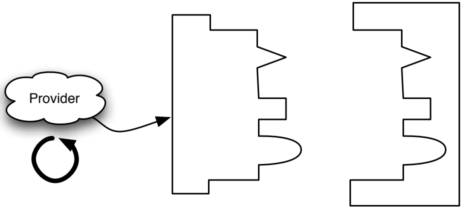
- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly



Motivation

Evolution in software construction

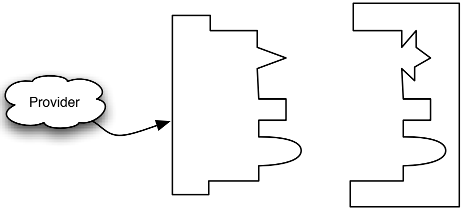
- Monolithic, centralized, stable applications
- Close world assumption
- Software changes slowly



Motivation

Evolution in software construction

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



Motivation

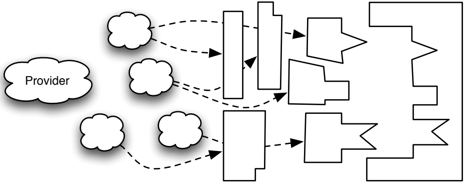
Dynamic environment

- Decentralized, distributed, dynamic applications

Motivation

Dynamic environment

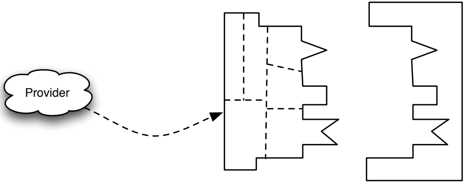
- Decentralized, distributed, dynamic applications



Motivation

Dynamic environment

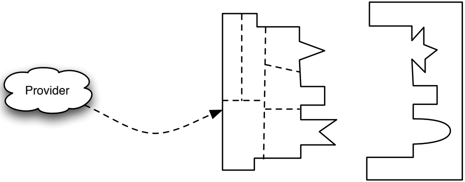
- Decentralized, distributed, dynamic applications



Motivation

Dynamic environment

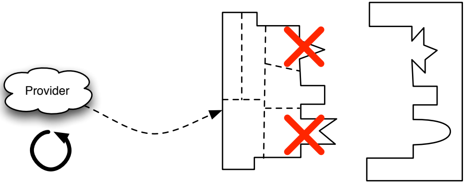
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes



Motivation

Dynamic environment

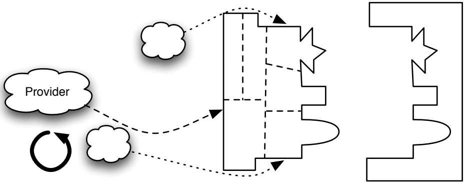
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes



Motivation

Dynamic environment

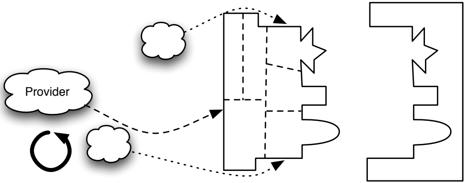
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes



Motivation

Dynamic environment

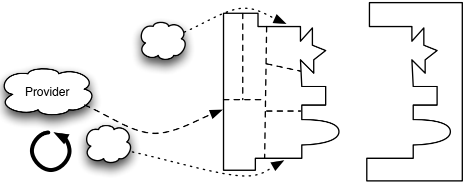
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager



Motivation

Dynamic environment

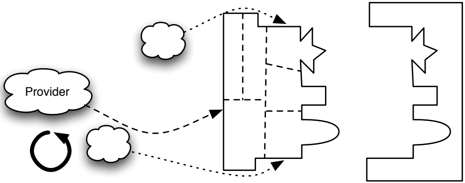
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation



Motivation

Dynamic environment

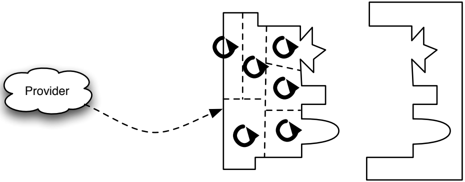
- Decentralized, distributed, dynamic applications
- External conditions may change
- Software needs to dynamically react and adapt to changes
  - Complexity not easy for a human manager
  - Autonomic adaptation
- Heterogeneity and distribution



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**

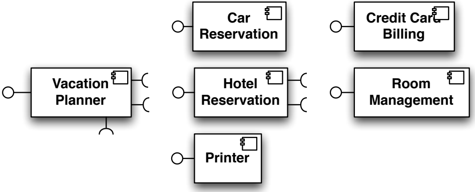


2
CONTEXT

Developing dynamic adaptable software

Component-based Software Development

Service-orientation

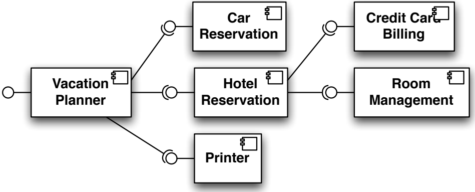


Developing dynamic adaptable software

Component-based Software Development

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

Service-orientation



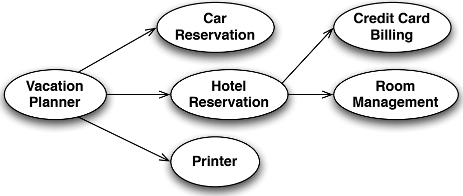
Developing dynamic adaptable software

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



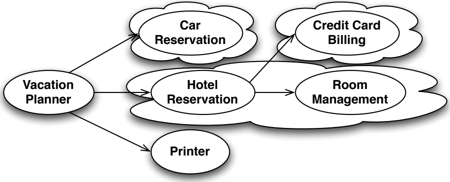
Developing dynamic adaptable software

Component-based Software Development

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

Service-orientation

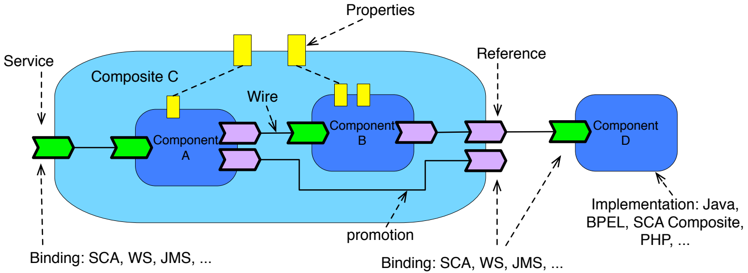
- Providers offers specific functionalities *as a service*
- Services are composable using standard means
- Facilitate the construction of new added-value applications
- Loosely coupled compositions of heterogeneous services



Service 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


Properties
Service
Composite C
Wire
Reference
Component A
Component B
Component D
promotion
Implementation: Java, BPEL, SCA Composite, PHP, ...
Binding: SCA, WS, JMS, ...
Binding: SCA, WS, JMS, ...

Advantages . . . and challenges

Advantages in software development

Advantages . . . and challenges

Advantages in software development

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

Advantages . . . and challenges

Advantages in software development

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

Challenges

Advantages . . . and challenges

Advantages in software development

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

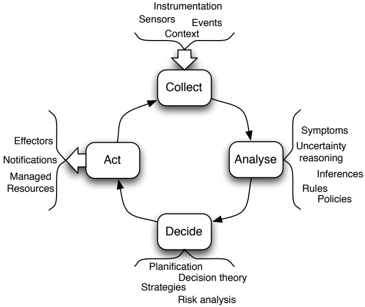
Challenges

- 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

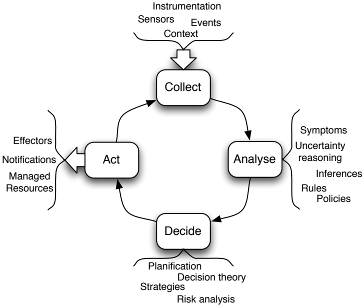
- Based on the idea of self-governing systems
- Context-awareness, and self-* properties
  - Self-{configuring, healing, optimizing, protecting, . . . }
- Activities represented in a *feedback control loop*

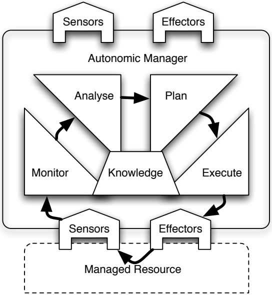


Autonomic Computing

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
  - Self-{configuring, healing, optimizing, protecting, . . . }
- Activities represented in a *feedback control loop*
- Phases in the *MAPE* autonomic control loop





3
FRAMEWORK

Problem

How to implement dynamic adaptations?

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

Problem

How to implement dynamic adaptations?

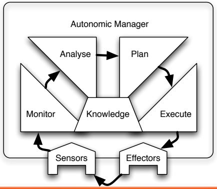
- Lack of uniformity and flexibility
- Impossibility of foreseeing all situations
- Complexity of developing effective autonomic tasks

Goal: **Improve the adaptability of service-based applications**

Solution Overview

Flexible Monitoring and Management Framework

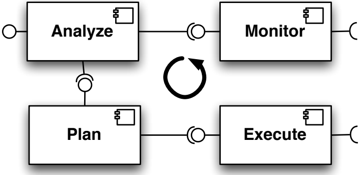
- **Implementing an *autonomic* control loop**



Solution Overview

Flexible Monitoring and Management Framework

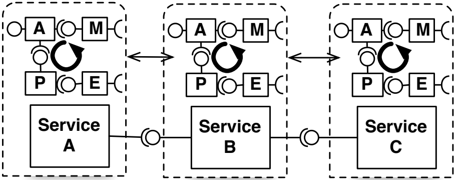
- **Implementing an *autonomic* control loop**
- **Encapsulating each MAPE phase as a component**
  - Use components to **extend** the behaviour of the control loop



Solution Overview

Flexible Monitoring and Management Framework

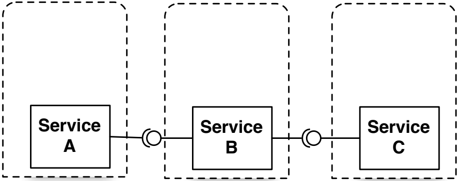
- **Implementing an *autonomic* control loop**
- **Encapsulating each MAPE phase as a component**
  - Use components to **extend** the behaviour of the control loop
- **Attaching the autonomic control loops to services**
  - Interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions, close to services (**efficiency**)



Solution Overview

Flexible Monitoring and Management Framework

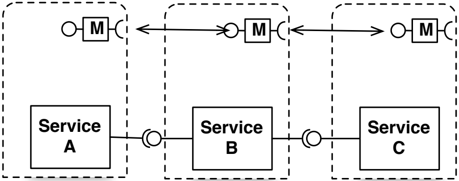
- **Implementing an *autonomic* control loop**
- **Encapsulating each MAPE phase as a component**
  - Use components to **extend** the behaviour of the control loop
- **Attaching the autonomic control loops to services**
  - Interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions,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



Solution Overview

Flexible Monitoring and Management Framework

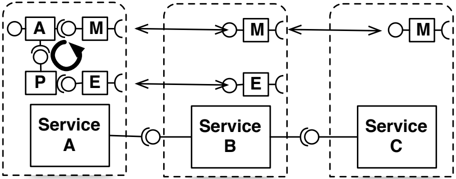
- **Implementing an *autonomic* control loop**
- **Encapsulating each MAPE phase as a component**
  - Use components to **extend** the behaviour of the control loop
- **Attaching the autonomic control loops to services**
  - Interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions,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



Solution Overview

Flexible Monitoring and Management Framework

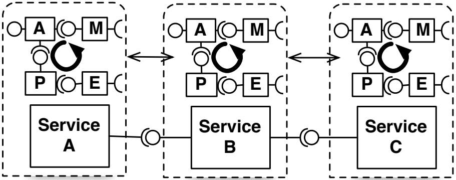
- **Implementing an *autonomic* control loop**
- **Encapsulating each MAPE phase as a component**
  - Use components to **extend** the behaviour of the control loop
- **Attaching the autonomic control loops to services**
  - Interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions,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



Solution Overview

Flexible Monitoring and Management Framework

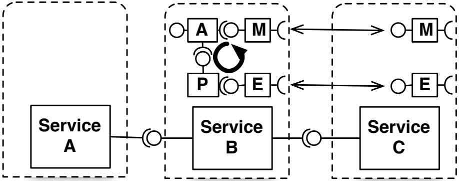
- **Implementing an *autonomic* control loop**
- **Encapsulating each MAPE phase as a component**
  - Use components to **extend** the behaviour of the control loop
- **Attaching the autonomic control loops to services**
  - Interfaces for the MAPE loops to interact and collaborate
  - Take timely decisions,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



Solution Overview

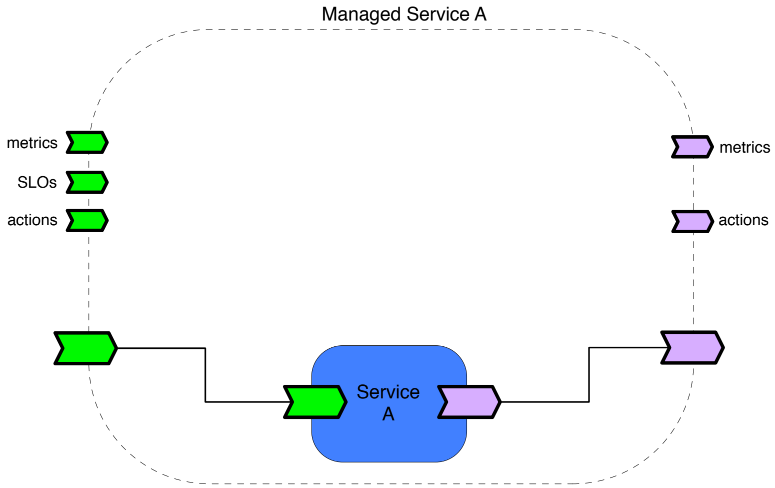
Flexible Monitoring and Management Framework

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



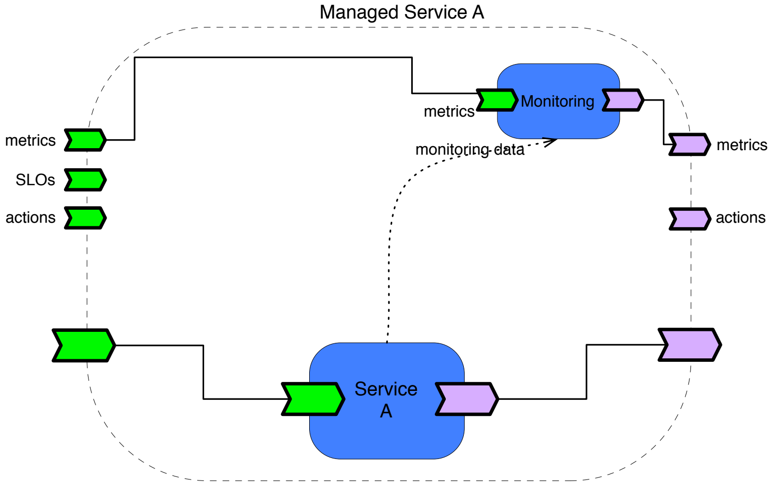
Inside a *Managed Service*

The *framework* itself is component-based application.



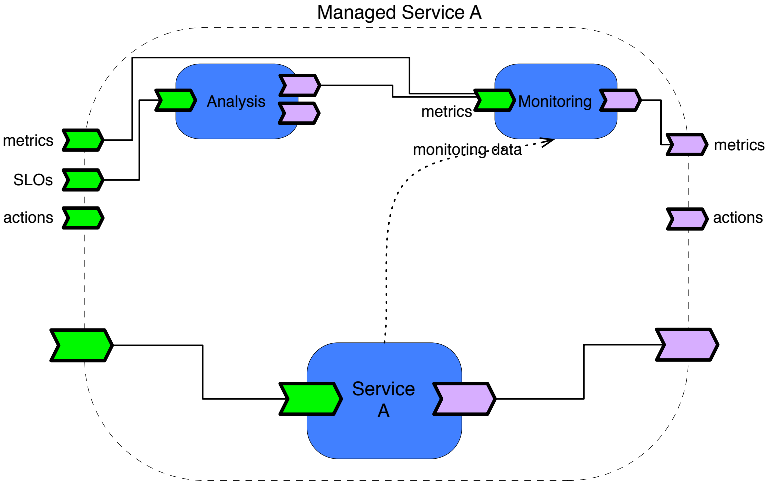
Inside a *Managed Service*

The *framework* itself is component-based application.



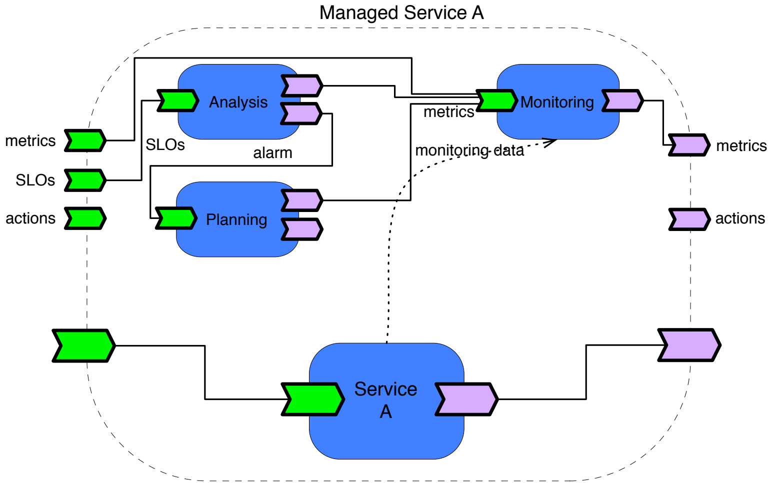
Inside a *Managed Service*

The *framework* itself is component-based application.



Inside a *Managed Service*

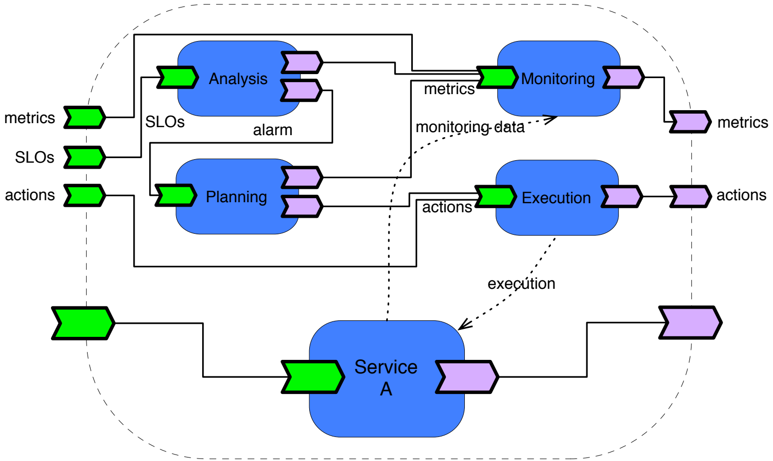
The *framework* itself is component-based application.



Inside a *Managed Service*

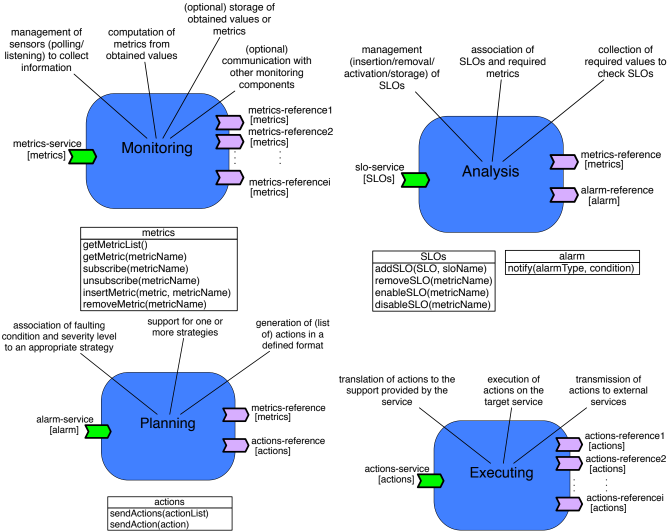
The *framework* itself is component-based application.

Managed Service A



MAPE Components

Basic API for each component



Monitoring

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]

metrics-reference1 [metrics]

metrics-reference2 [metrics]

⋮

metrics-referencei [metrics]

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

Analysis

management (insertion/removal/ activation/storage) of SLOs

association of SLOs and required metrics

collection of required values to check SLOs

slo-service [SLOs]

metrics-reference [metrics]

alarm-reference [alarm]

SLOs
addSLO(SLO, sloName)
removeSLO(metricName)
enableSLO(metricName)
disableSLO(metricName)

alarm
notify(alarmType, condition)

Planning

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

alarm-service [alarm]

metrics-reference [metrics]

actions-reference [actions]

actions
sendActions(actionList)
sendAction(action)

Executing

translation of actions to the support provided by the service

execution of actions on the target service

transmission of actions to external services

actions-service [actions]

actions-reference1 [actions]

actions-reference2 [actions]

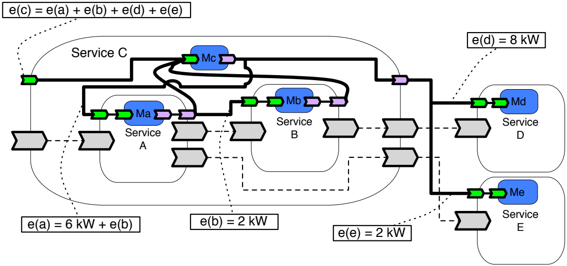
⋮

actions-referencei [actions]

Monitoring: Example

Monitoring components connected through the application

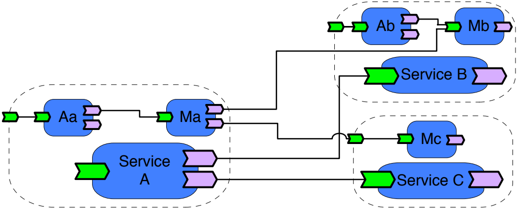
- *Monitoring backbone* through the application
- Components collaborate to compute metrics
- Each component may implement the computation logic differently



Analysis: Example

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

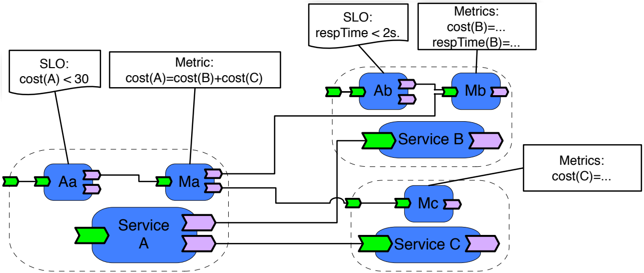
- Different Analyzers may check different conditions without interferring with others



Analysis: Example

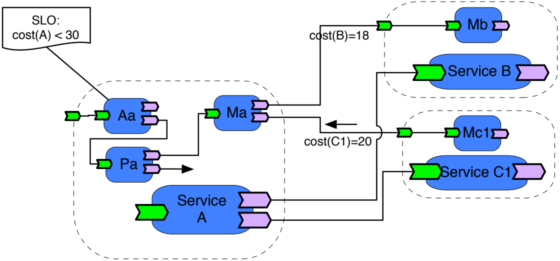
Analysis components use the *monitoring backbone* to obtain the metrics they need to perform SLO checking

- Different Analyzers may check different conditions without interferring with others



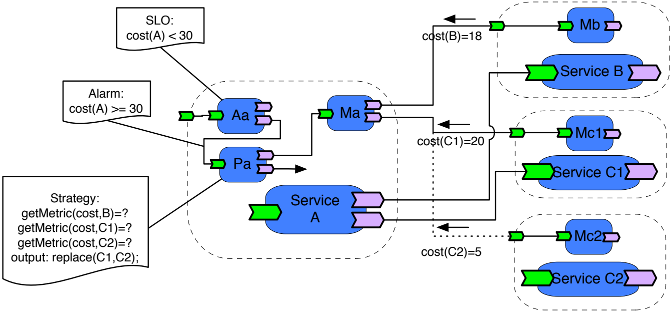
Planning: Example

Implementation of strategies of decision algorithms



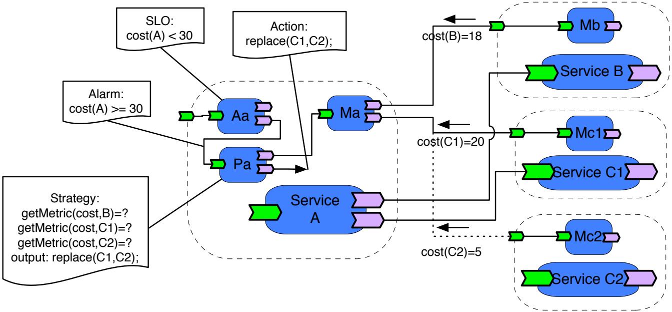
Planning: Example

Implementation of strategies of decision algorithms



Planning: Example

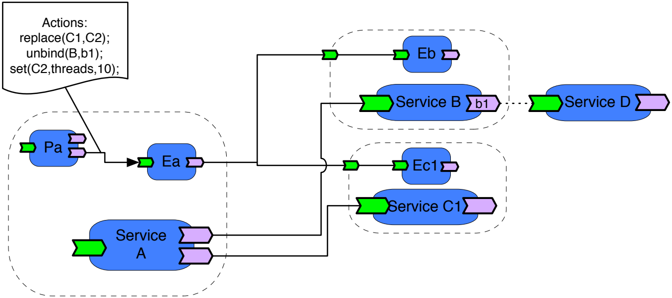
Implementation of strategies of decision algorithms



Execution: Example

Actions may be propagated to the appropiate service

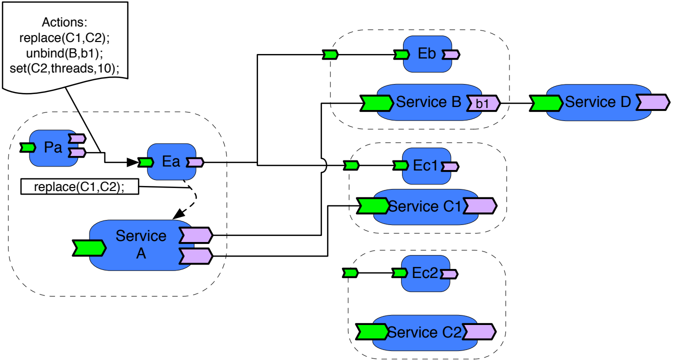
- Execute actions on the service according to the specific means allowed



Execution: Example

Actions may be propagated to the appropiate service

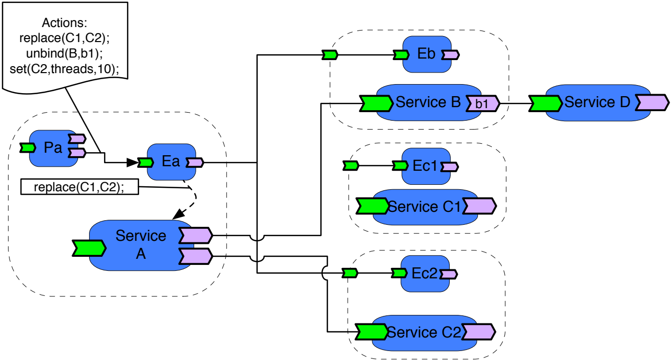
- Execute actions on the service according to the specific means allowed



Execution: Example

Actions may be propagated to the appropiate service

- Execute actions on the service according to the specific means allowed



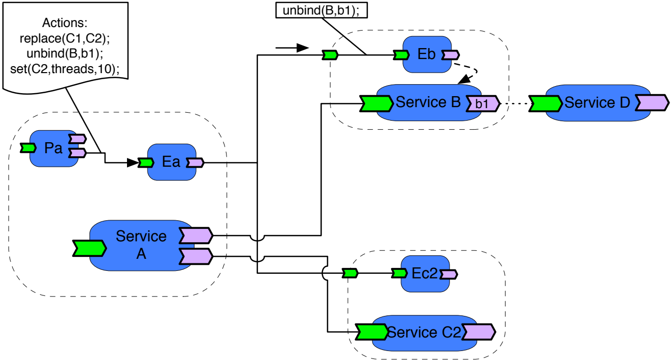
Actions:
replace(C1,C2);
unbind(B,b1);
set(C2,threads,10);

replace(C1,C2);

Execution: Example

Actions may be propagated to the appropiate service

- Execute actions on the service according to the specific means allowed



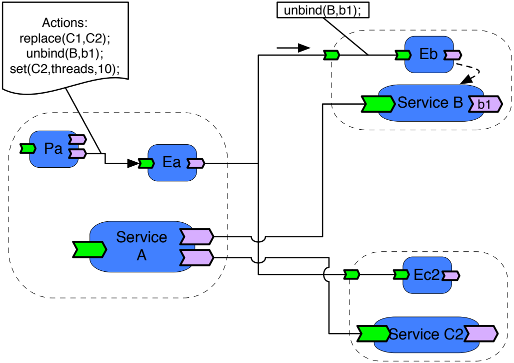
Actions:
replace(C1,C2);
unbind(B,b1);
set(C2,threads,10);

unbind(B,b1);

Execution: Example

Actions may be propagated to the appropiate service

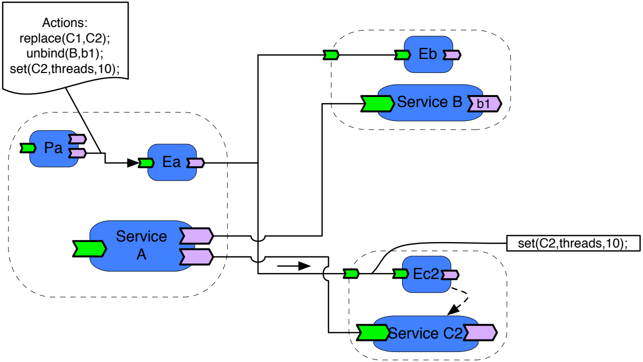
- Execute actions on the service according to the specific means allowed



Execution: Example

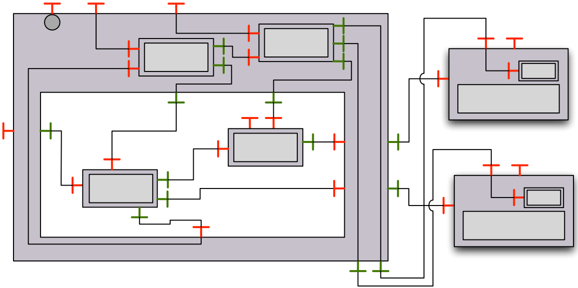
Actions may be propagated to the appropiate service

- Execute actions on the service according to the specific means allowed



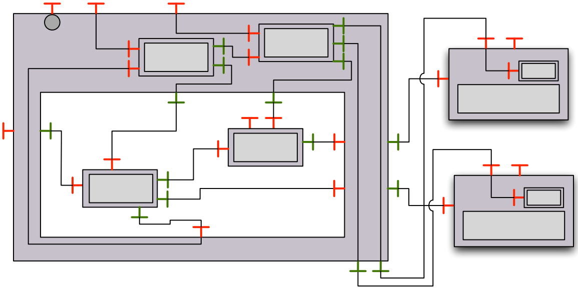
Implementation: background

- Grid Component Model (GCM)
  - Extension of the Fractal Component Model
  - Support for distributed deployment
  - Support for collective communications (*multicast*/*gathercast*)



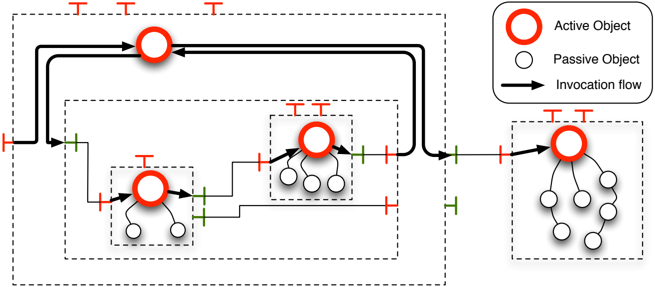
Implementation: background

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



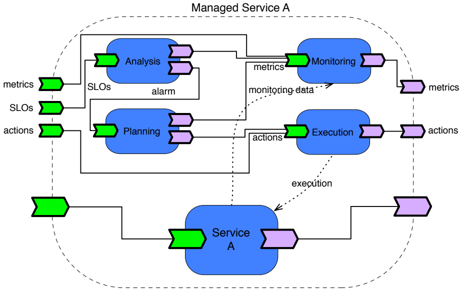
Implementation: background

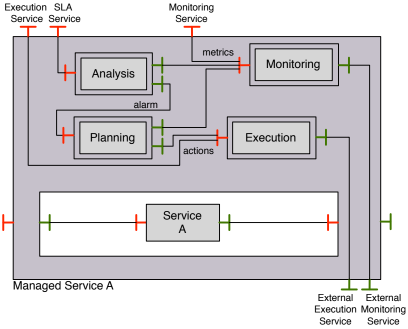
- Grid Component Model (GCM)
  - Extension of the Fractal Component Model
  - Support for distributed deployment
  - Support for collective communications (*multicast*/*gathercast*)
  - Separation between F and NF concerns
- Using the GCM/ProActive reference implementation
  - Based on asynchronous active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



Implementación: *SCA→ GCM*

- MAPE components in the membrane of GCM componentes
- NF (non-functional) interfaces
- Implementation of each MAPE component
- Definition of an API to manipular los componentes MAPE

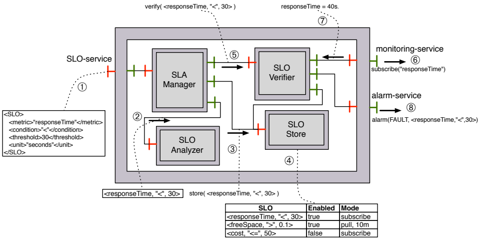




Monitoring and Analysis Components

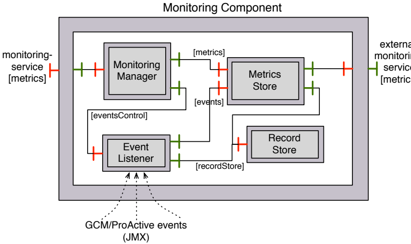
Collection, storage, computation of metrics

- *Listeners* JMX
- Insertion/removal of metrics. Push/pull access.
- Sending of *Alarm* objects
- *SLO* representation: $\langle \textit{metric}, \textit{condition}, \textit{threshold} \rangle$



```
<SLO>
  <metric>"responseTime"</metric>
  <condition>"<"</condition>
  <threshold>30</threshold>
  <unit>"seconds"</unit>
</SLO>
```

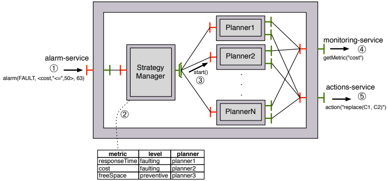
SLO	Enabled	Mode
<responseTime, "<", 30>	true	subscribe
<freeSpace, ">", 0.1>	true	pull, 10m
<cost, "<=", 50>	false	subscribe



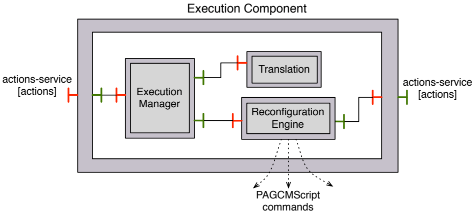
Planning and Execution Components

Execution of planning algorithms (strategies)

- *Alarm*s associated to one or more strategies
- Support for multiple strategies using multicast interfaces
    - Selection, parallel execution of strategies
- Delegation of actions to other components
    - *GCMScript* for executing reconfigurations

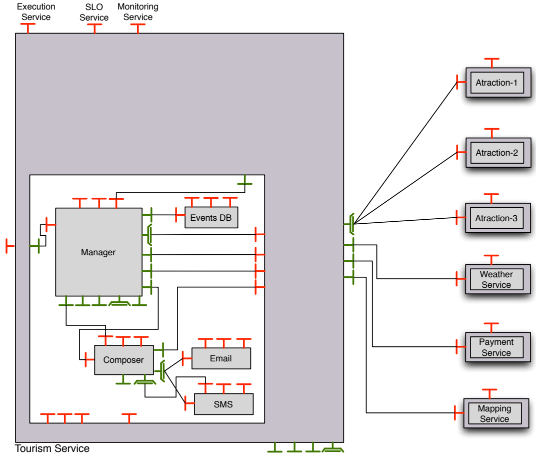


metric	level	planner
responseTime	faulting	planner1
cost	faulting	planner2
freeSpace	preventive	planner3



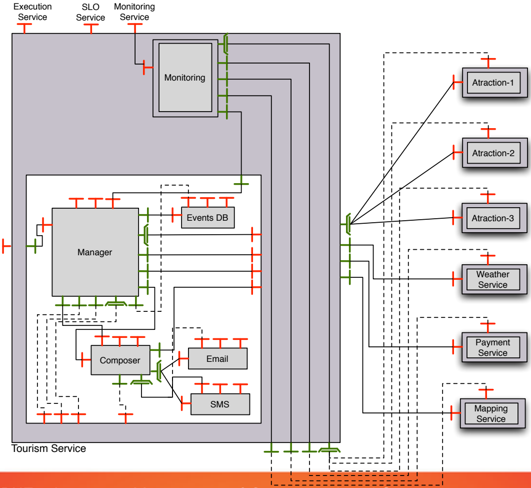
Use Case: Setting up

- Insertion of MAPE components via API
    - Automatic creation of *bindings* following the functional architecture of the system.



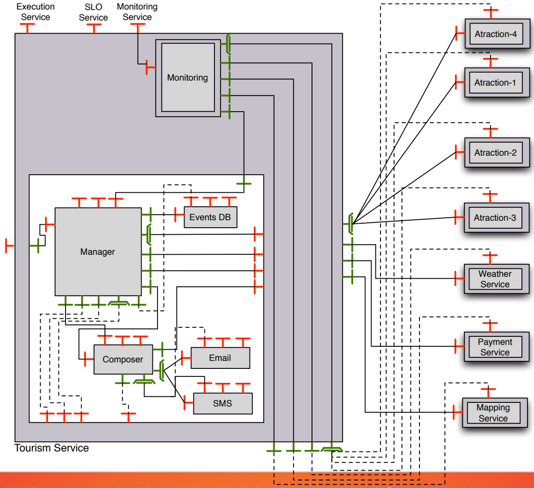
Use Case: Setting up

- Insertion of MAPE components via API
  - Automatic creation of *bindings* following the functional architecture of the system.



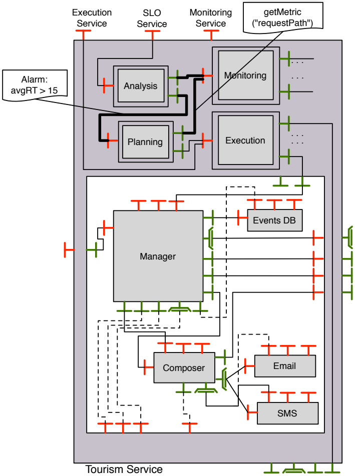
Use Case: Setting up

- Insertion of MAPE components via API
  - Automatic creation of *bindings* following the functional architecture of the system.



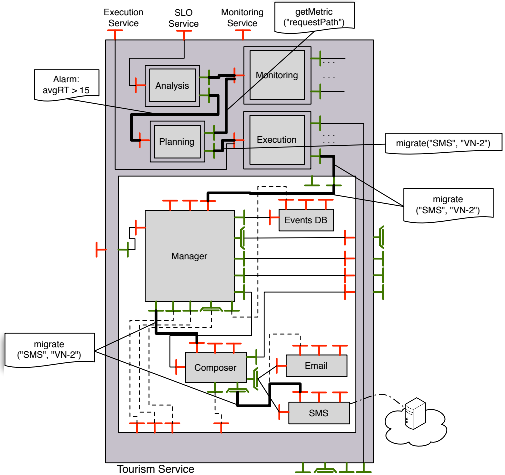
Use Case: Propagation of autonomic adaptations

The actions is propagated through the internal components



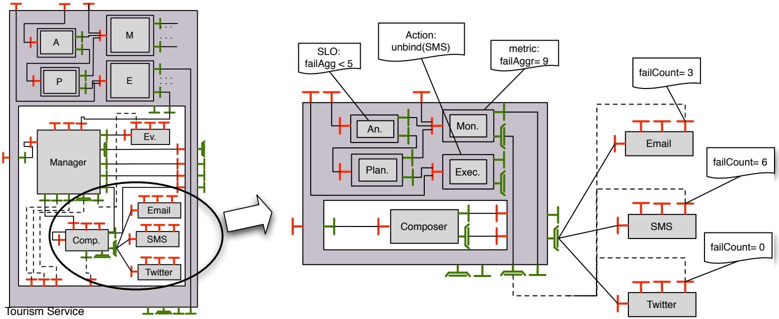
Use Case: Propagation of autonomic adaptations

The actions is propagated through the internal components



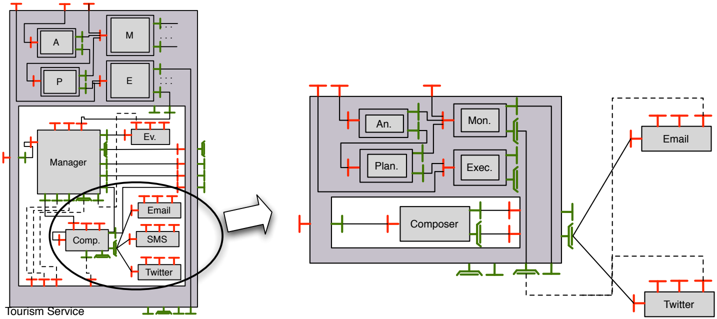
Use Case: Propagation to external components

Internal control loop

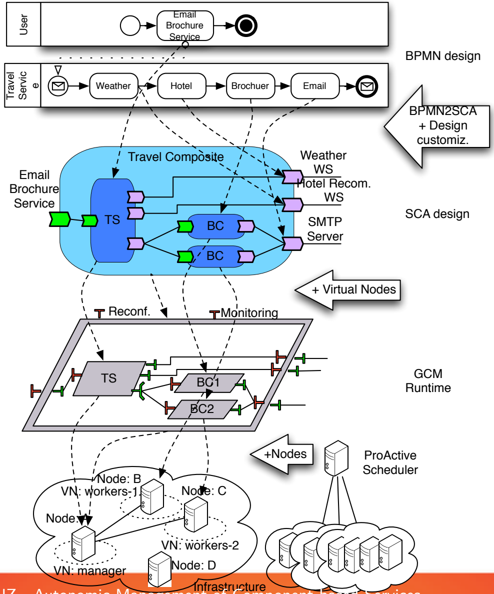


Use Case: Propagation to external components

Internal control loop



Use Case: Mapping the lifecycle of services



4
PERSPECTIVAS

Additional work

- Non-Functional ADL
- Distributed reconfiguration of compoments
- Autonomic *deployment* on *cloud* environments
- Integration of the autonomic framework with *skeletons*
- Dynamic adaptation of *workflows*

Perspectives

Challenges on autonomic computing

- Implementation and experimentation of collaborative strategies
    - Division of goals in sub-tasks
    - Hierarchical planning
- Verification of (*safety*) of reconfiguration actions
    - Avoid *livelock* of adaptations
    - Avoid inconsistencies of the applications

MERCI

INRIA Sophia Antipolis Méditerranée

http://www-sop.inria.fr/oasis/personnel/Cristian.Ruz/