Component-based generic approach for reconfigurable management of component-based SOA applications

Cristian Ruz, Françoise Baude, Bastien Sauvan

{Cristian.Ruz, Francoise.Baude, Bastien.Sauvan}@inria.fr

INRIA, I3S, CNRS, UNSA

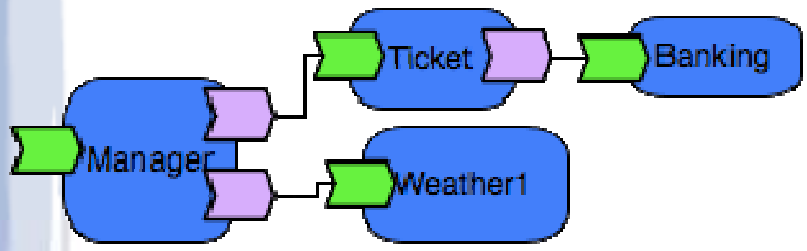
3rd Workshop MONA+, December 1st 2010, Ayia Napa, Cyprus

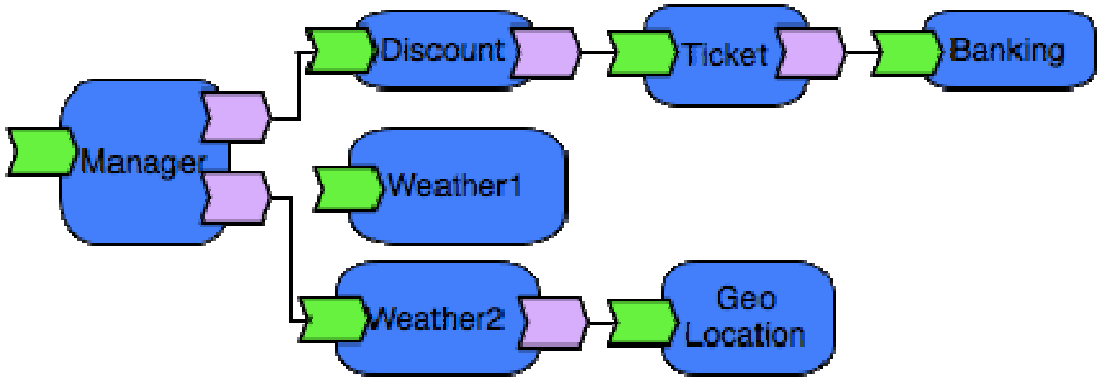
Context

- SOA – Service Oriented Architectures
  - Loosely coupled heteregoneous services and providers
  - Providers also act as consumers → prosumers
  - Dynamic nature of SOA
- SLA – Service Level Agreements
  - Contracts established between consumers and providers
  - Necessity to check the compliance at runtime
  - Necessity to monitor the required metrics at runtime

Problem

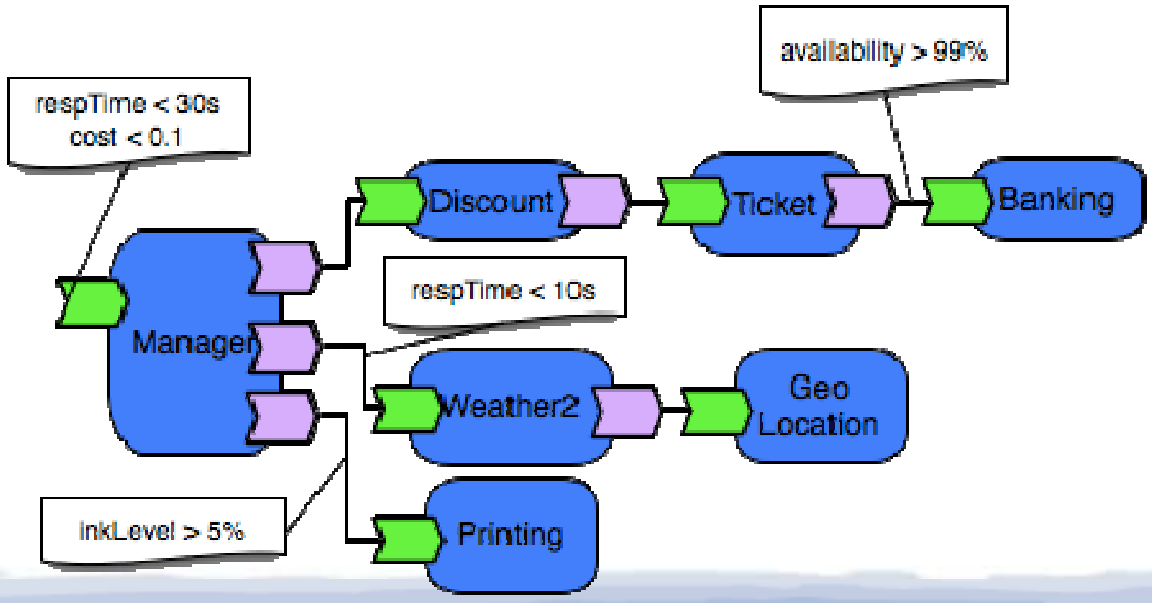
- Dynamic nature of applications
  - Relationships change → Monitor other services
  - SLA terms change → Different metrics are required
  - Different services require different metrics





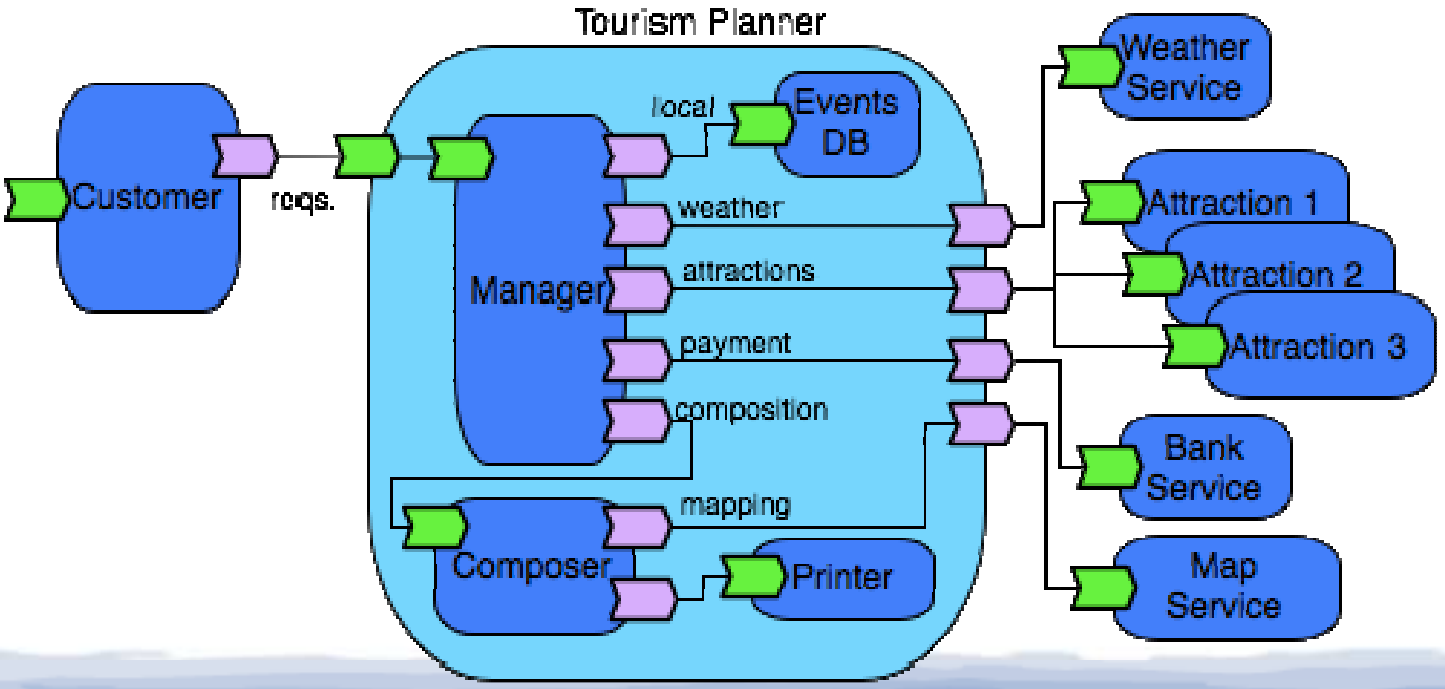
Problem

- We cannot foresee all the monitoring requirements
  - Need for a flexible approach
  - Monitor what you need to monitor
  - Check the conditions you need to check



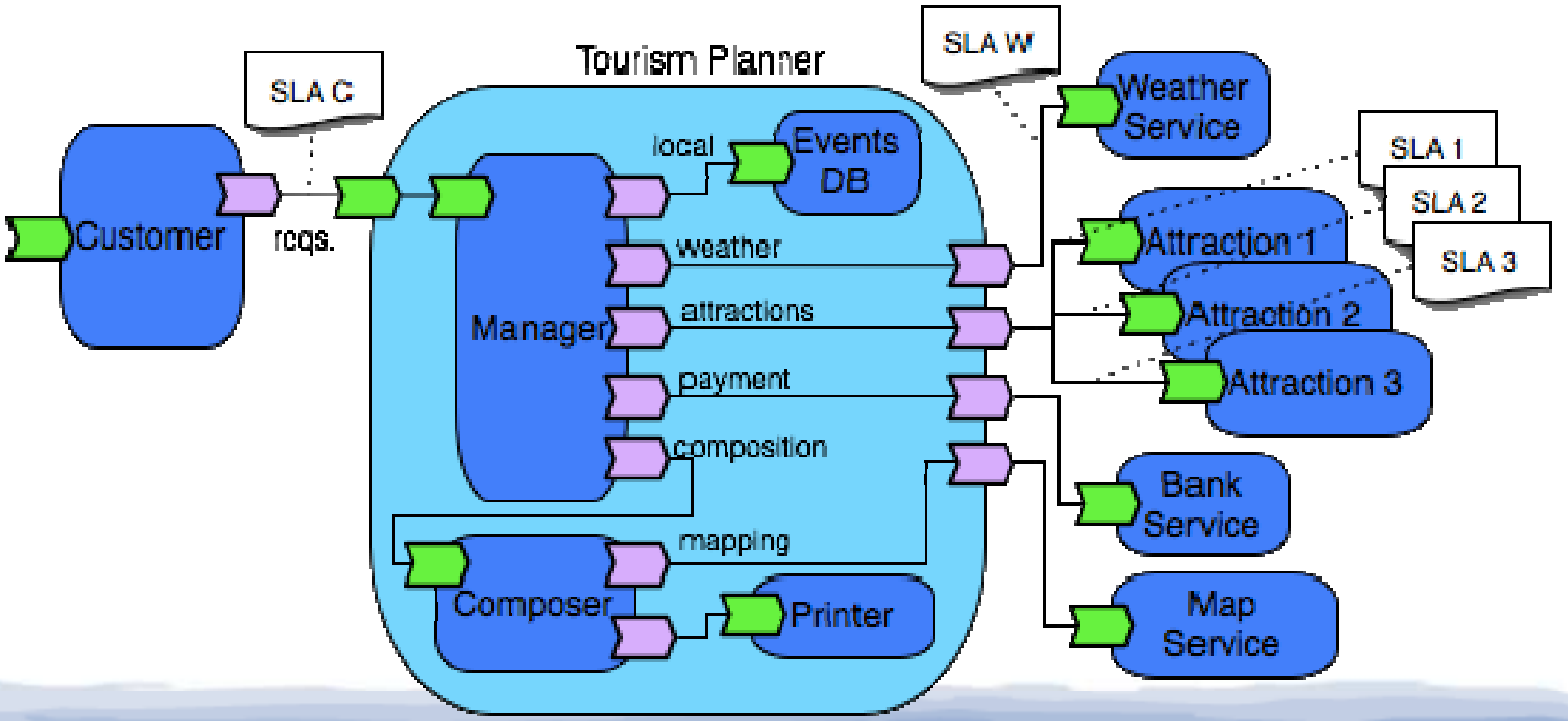
How to handle flexibility?

- Components meet services: SCA
  - Service Component Architecture
  - Industry supported + technologically agnostic specification
  - Allows to handle dynamicity, complexity, heterogeneity
  - Monitoring and management are a task of the platform



Need for adaptation

- SLAs can be established at multiple levels
  - Different SLAs require to check different metrics
  - SLAs can change at runtime
- Reconfigurations can take place at different levels
  - Different services allows different actions



Summarizing ...

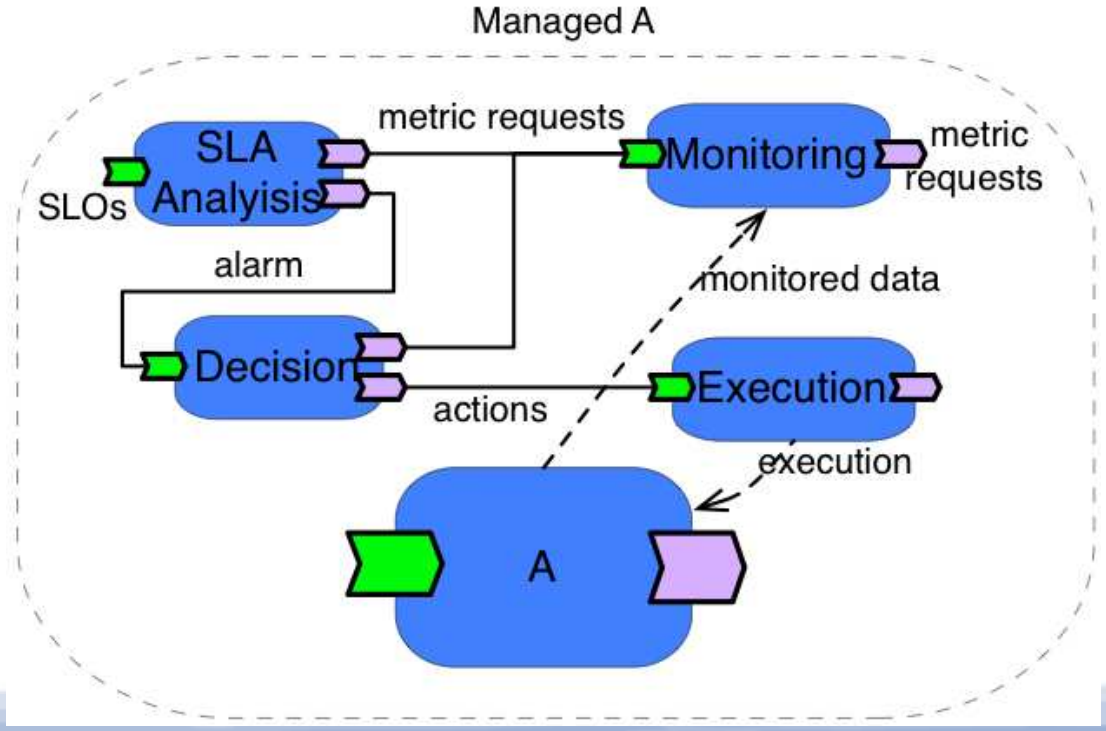
- The evolution of SLAs and metrics cannot be completely foreseen at design time
  - It is not feasible to design a monitoring system where all the metrics and conditions are ready to be monitored
  - A flexible system where the required metrics and conditions can be inserted or removed would be beneficial
- A component-based approach can provide this level of flexibility

Proposal

- Monitoring and management framework
  - Component-based design
  - Metrics and conditions can be introduced/removed at runtime
  - Different metrics can be taken at different services
  - Minimize intrusion
  - Separation of concerns
- Objective: adapt at runtime to monitoring and management requirements of the application
  - Provide a common ground for monitoring and actions
  - Connected to the management features of each service

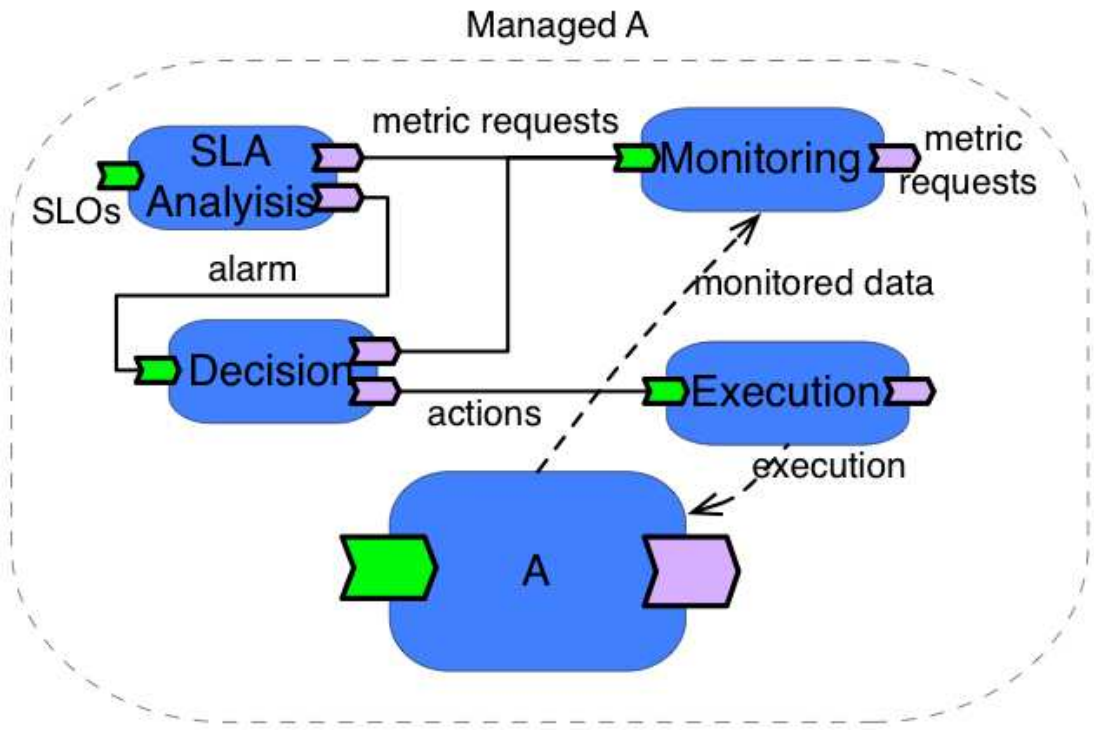
Big Picture

- Componentized MAPE autonomic control loop
  - Monitoring, Analysis, Planning, Execution
  - Set of “non-functional” components attached to each “business component”
  - Component turns into a “managed component”

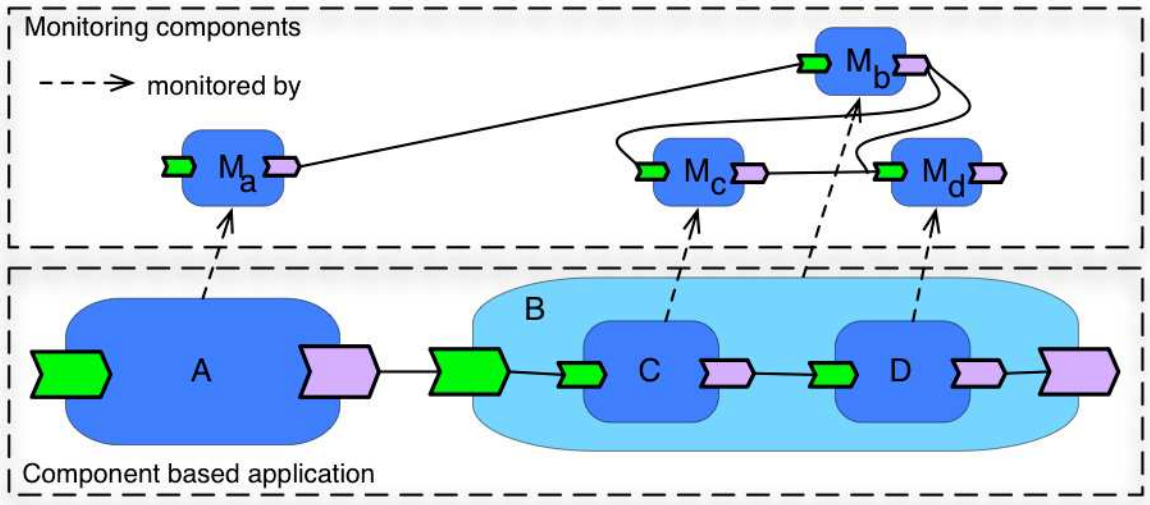


Big Picture

- Each component in charge of a step
  - Implementation can vary according to the application
  - Ability to adapt the framework at runtime
  - Not all components require all the MAPE steps

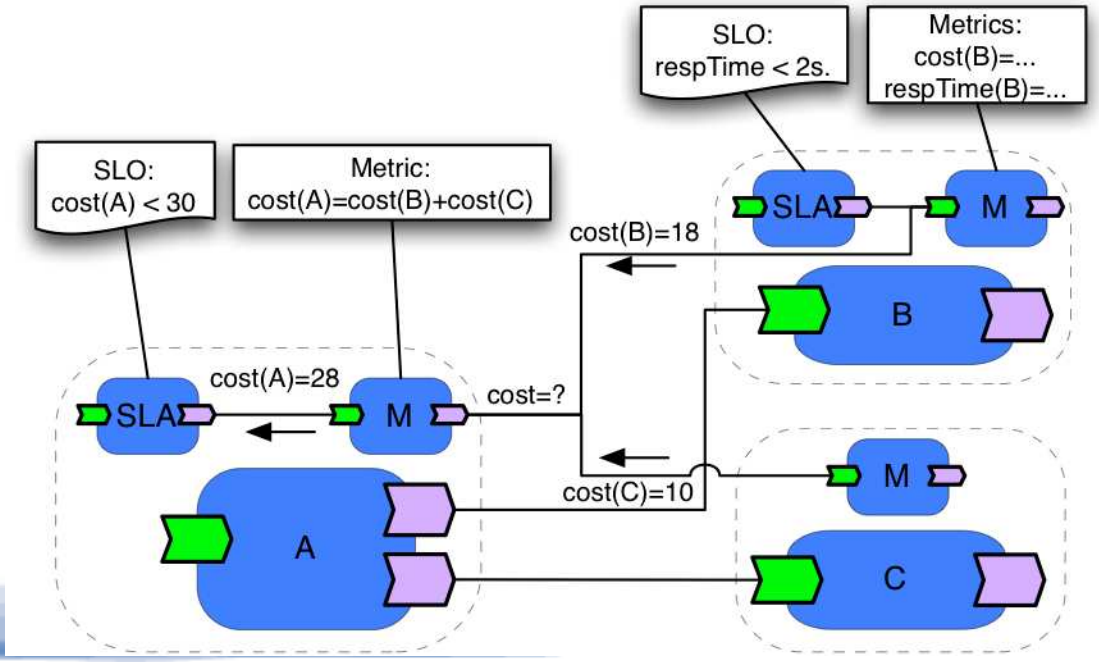


Monitoring

- Monitoring layer distributed along the application
  - Monitoring components interact through monitoring interfaces
  - Monitoring components are interconnected to form a monitoring layer
  - Each monitoring component can be adapted to monitor the metrics required from each component
  - 

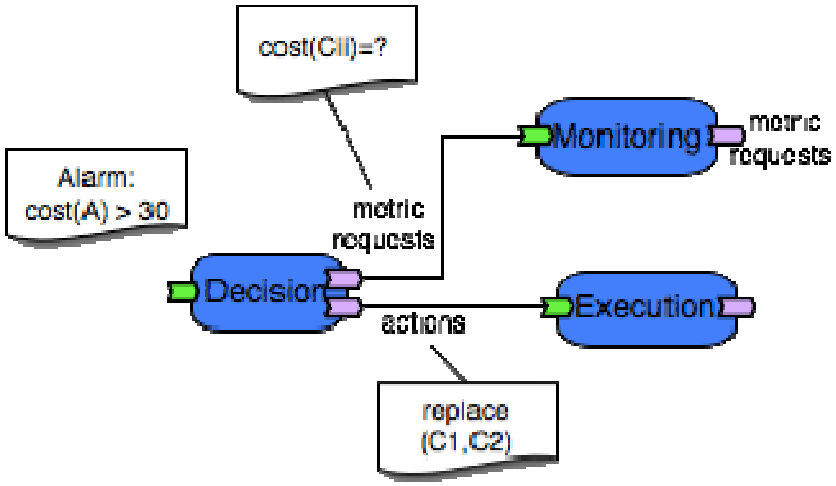
SLA Analysis

- SLA Analyzers check compliance to SLA
  - SLA expressed as a set of Service Level Objectives (SLO)
  - SLOs require metrics from the Monitoring Component
  - SLA Analyzer subscribes to the Monitoring Component for the metrics it needs
  - Upon a violation of the SLO, a notification is sent



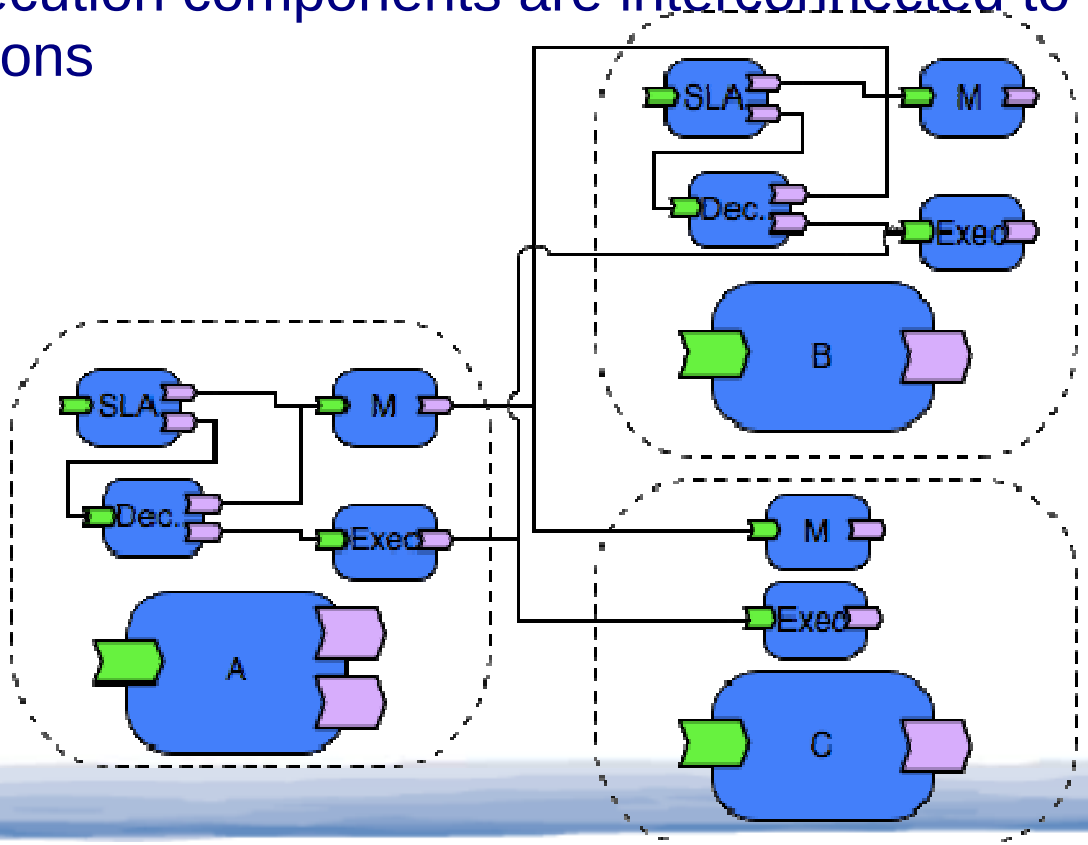
Decision

- Strategy to implement upon a violation
  - Decision component requires metrics from the Monitoring component
  - Implement and strategy oriented to return the system to a desired objective state
  - Output is a sequence of actions to modify the system



Execution

- Executes actions over the system
  - Must be able to execute actions over the system, in the way provided by the service or middleware
  - Execution may also involve actions over other components
  - Execution components are interconnected to propagate their actions

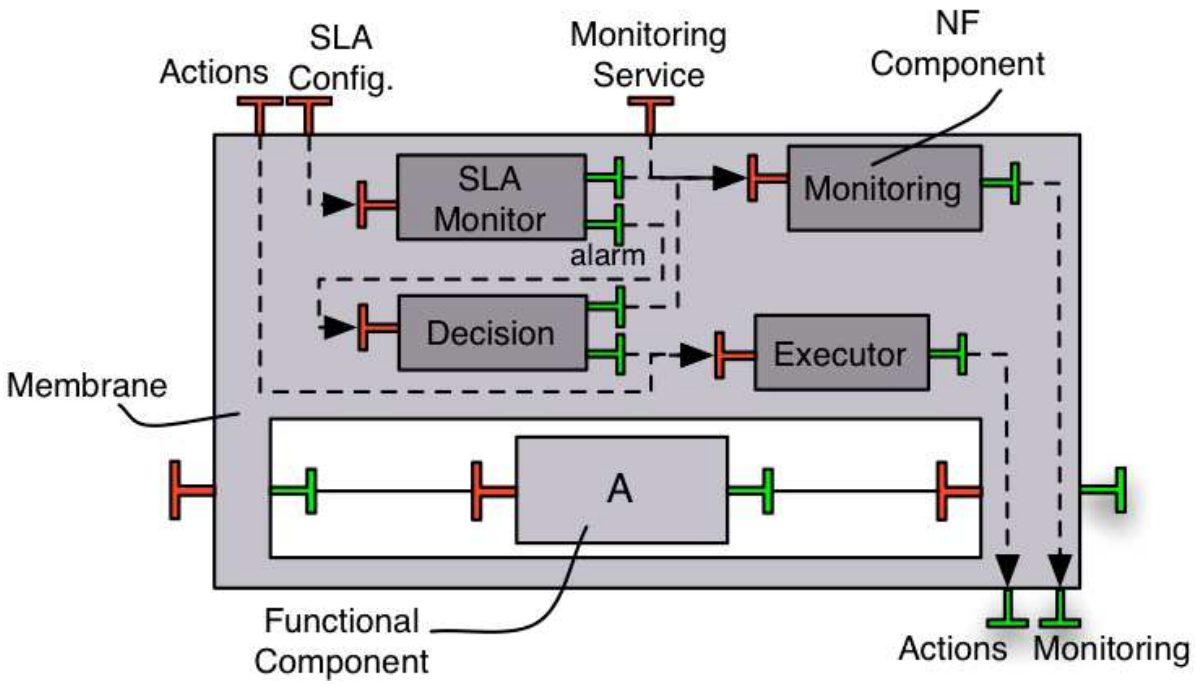


Implementation

- Implemented over the GCM/ProActive middleware
  - ProActive Grid middleware.
    - Java middleware for parallel and distributed computing
    - Active Object programming model with asynchronous communication
  - GCM: Grid Component Model
    - Component model for grid applications, based on Fractal
    - Support for collective communications
    - Large scale deployment through desktop/grid/cloud
  - GCM/ProActive middleware as an SCA compatible platform

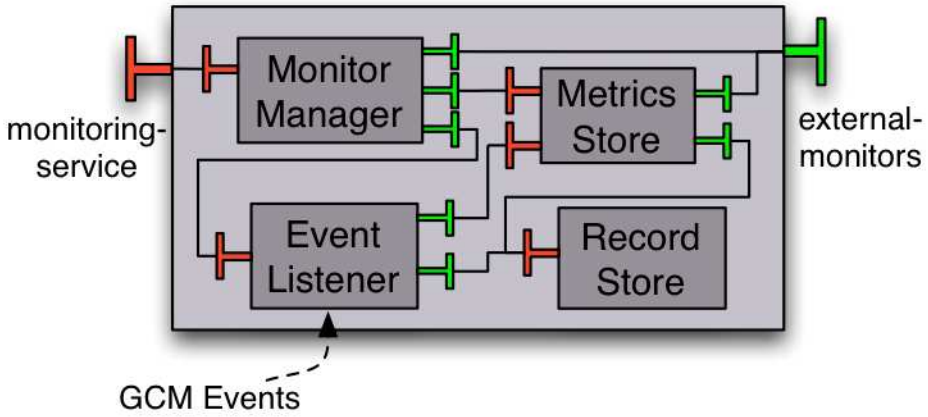
Framework Implementation

- Implemented as a set of NF components
  - NF components are weaved in the membrane of GCM/SCA components
  - Access to the framework features through NF interfaces
  - Separation of concerns between functional and NF content



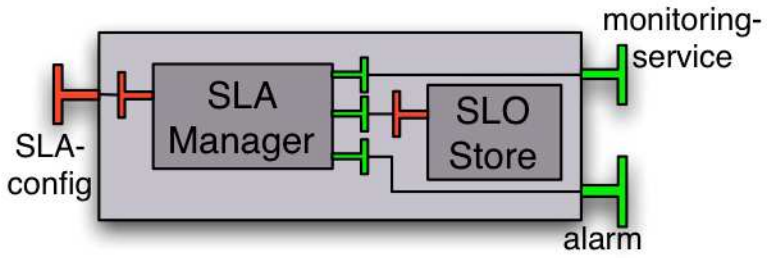
Monitoring Component

- Collects, stores, and computes metrics
  - Event Listener receives events from the ProActive/GCM middleware
  - Record Store keeps records of local events
  - Metrics Store stores and computes the metrics available for the monitored component
  - Interface allows to add/remove and retrieve values



SLA Monitor

- Provides SLA storage and checking
  - SLO Store stores conditions in a simple form:
    - <metricName, condition, threshold>
  - SLA Managers subscribe to required metrics and checks the SLOs.
    - Triggers an alarm if an SLO is not achieved



Decision component

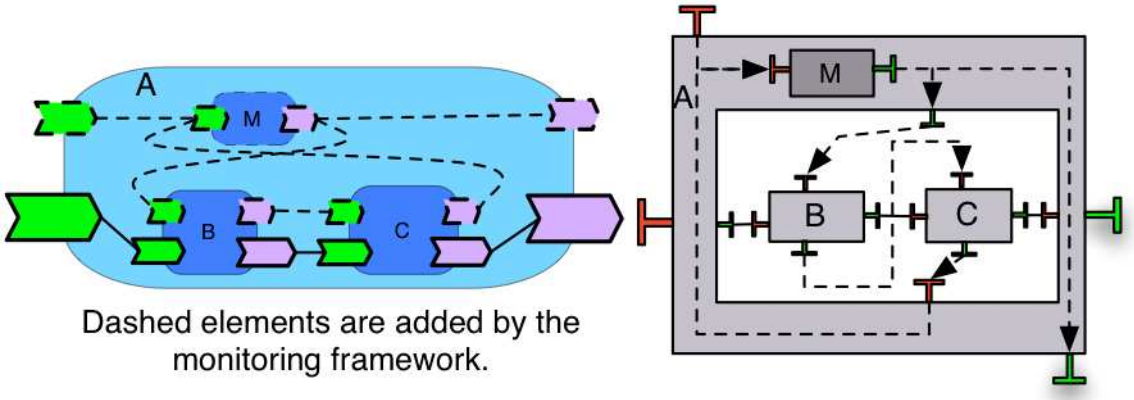
- Strategy associated to a faulting SLO
  - Strategy oriented to take the system to an objective state.
  - May include: architectural rebindings, parameters modifications, service selection, service migration, service replacing, ...
  - Examples:
    - Replace the slowest component (provided there is another).
    - Migrate the component to a less expensive infrastructure
    - Select a more energy-efficient service
  - Queries the Monitoring Component to obtaine the required input for the strategy
  - The component-based approach allows to host several strategies, or replace them at runtime

Execution component

- Executes a set of actions over the components
  - Set of actions must be described in a defined language
    - We have used PAGCMScript, a DSL extending FScript
  - List of actions can involve executing actions on other components
  - Examples:
    - unbind($manager/interface::”weather”);
    - bind($manager/interface::”weather”, $weather2/interface::”service”);
    - gcm-migrate($manager, $ec2-node);
  - Execution component must be able to execute actions over what is provided by the runtime support

Generalization

- Framework expressed in terms of SCA model
  - GCM NF Components are a set of SCA Components automatically added to the the design.
  - Separation between business code, from monitoring and management concerns.
  - The wrapping of the target services is performed by the framework.



Dashed elements are added by the monitoring framework.

Future Work

- Experimentation plans
  - Experiment with different planning strategies
  - Experiment with more complex compositions
  - Deeper evaluation of performance
- Validation with other implementations
- Extend SLA Analysis to predictive checkings

Conclusions

- Generic and flexible monitoring and management framework
  - Component based solution
  - Separation of concerns between business and monitoring and management tasks
  - Metrics, SLOs, strategies, can be added or removed at runtime
  - Adaptable to the monitoring requirements of the application
    - Limited by the information provided by the platform
  - Bring the monitoring and management tasks over heterogenous services to a common ground
  - Implemented over an SCA compatible platform

Component-based generic approach for reconfigurable management of component-based SOA applications

Cristian Ruz, Françoise Baude, Bastien Sauvan

INRIA, I3S, CNRS

Thank you !!

3rd Workshop MONA+, December 1st 2010, Ayia Napa, Cyprus