

Self-* systems: an architectural challenge



Matteo Miraz – miraz@elet.polimi.it

Ph.D. Student at:

Dipartimento di Elettronica e Informazione
Politecnico di Milano, Italy

Background papers

- Self-Managed Systems: an Architectural Challenge
by J. Kramer and J. Magee
Future of Software Engineer (FOSE), 2007
- Increasing Systems Dependability through Architecture-based Self-Repairing
by D. Garlan, S. Cheng and B. Schmerl
Architecting Dependable Systems, 2003

Outline

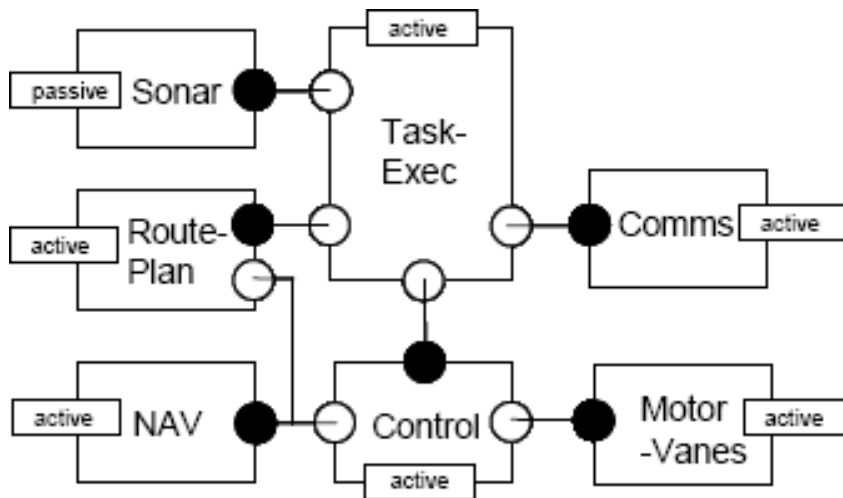
- Why an architectural point of view?
- The Garlan's approach
- Research open issues in architectural self-*

Where is used self-*something*?

- In this course: autonomous services (SOA)
- Self-something area involves:
 - telecommunications switches
 - deep space control software
 - networking
 - systems
 - services
 - ...

Why an architectural point of view?

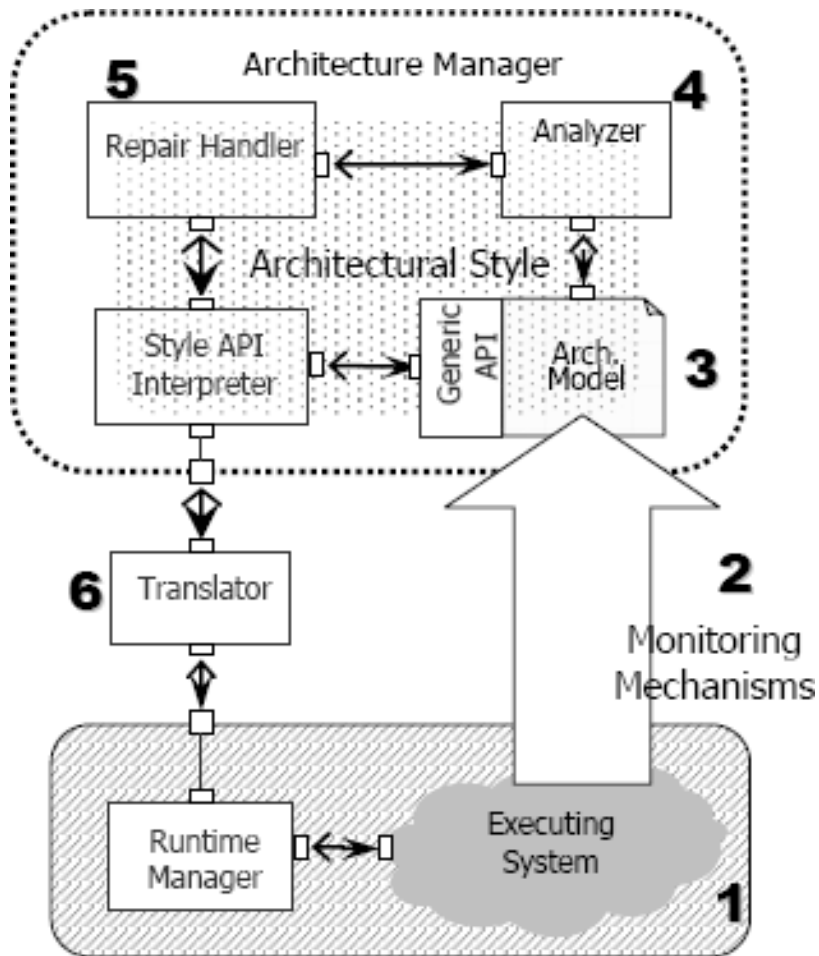
- Why *architecture* is so important?
 - Right level of abstraction and generality
 - Components (semi-)automatically manage their interactions



The Garlan's approach

- Generally:
 - use a particular architecture style (e.g. RPC)
- Garlan's idea:
 - use the system architecture model at runtime
 - provides a global perspective of the system
 - better identification of the source of the problem
 - make “integrity” constraints explicit

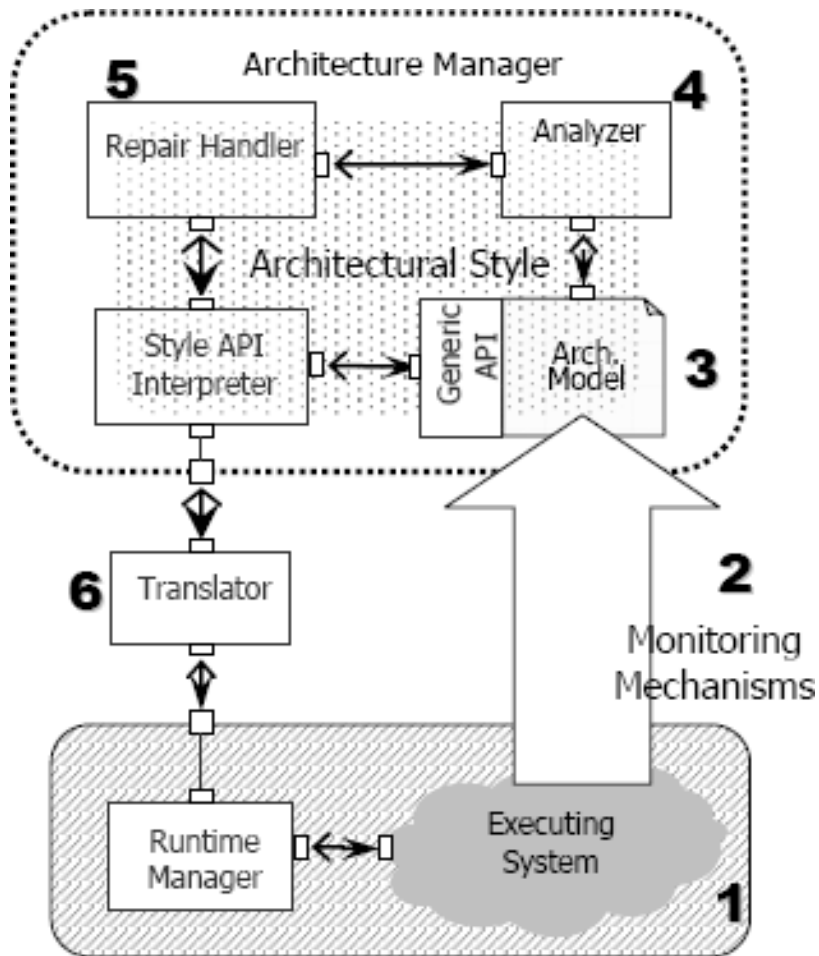
Adaptation Framework (1)



(1) the executing system is monitored to observe its run time behaviour

(2) monitored values are abstracted and related to architectural properties of an architectural model

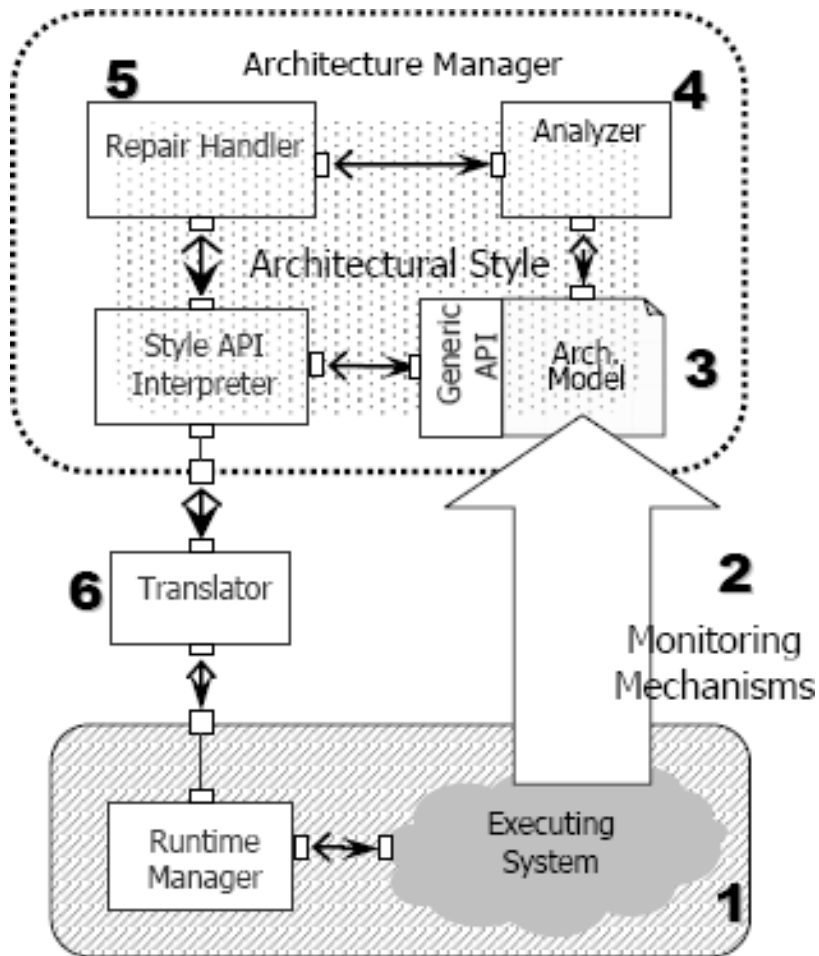
Adaptation Framework (2)



(3) changing properties trigger architectural analysis

(4) unacceptable operations causes repairs

Adaptation Framework (3)



(5) repairs adapt the architecture

(6) architectural changes are propagated to the running system

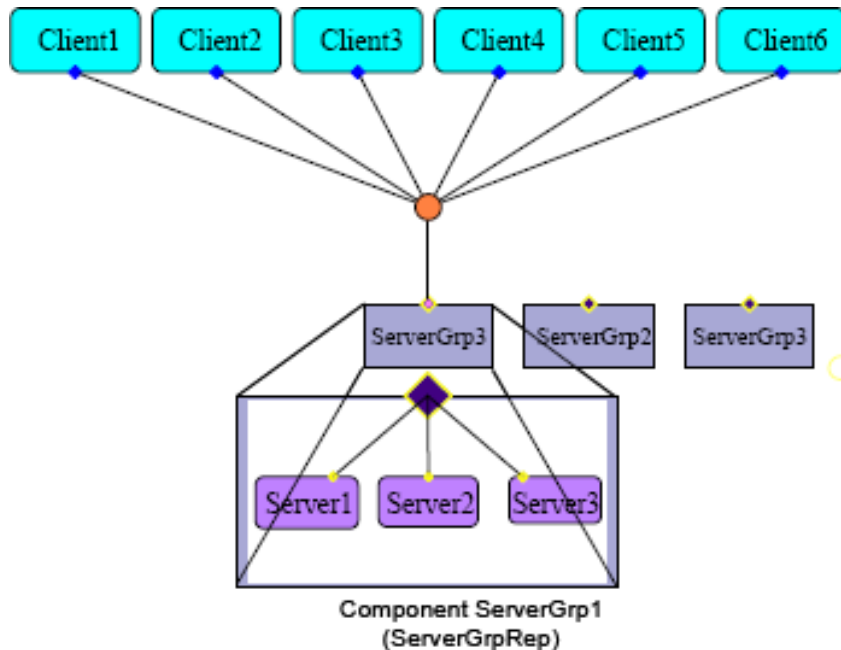
The architectural style

- The architectural style is used to determine:
 - what properties should be monitored
 - what constraints need to be evaluated
 - what to do with violated constraints
 - how to carry out repairs in terms of high-level architectural operators

The architectural style (cont.)

- The architectural style specifies:
 - a set of specific architectural operators
 - a collection of repair strategies associated with style's constraints
- Data collection:
 - the low-level monitored informations are processed by gauges that relate them to the architectural model

Example



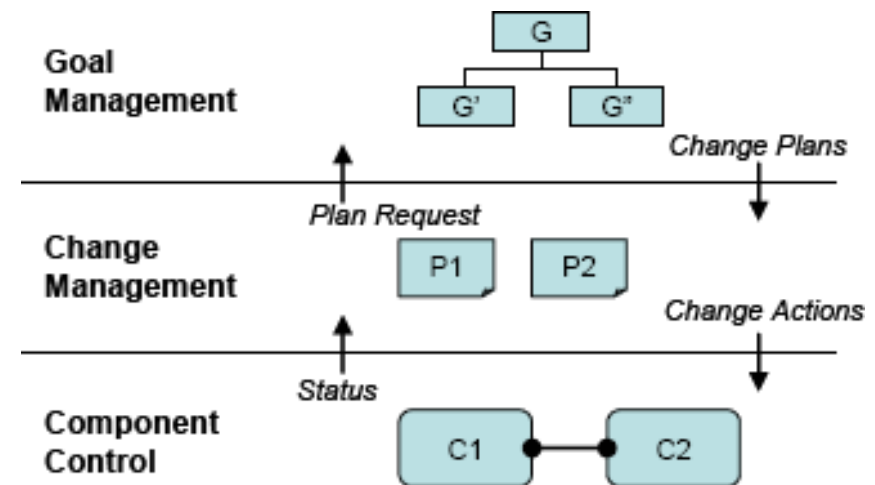
- Goal: performance adaptation
- Monitoring
 - gauge: average latency for each client
 - probes: SNMP Sensors
- Adaptation operators:
 - addServer
 - move(SGroup)
 - remove

The ROBOT Analogy

- The goal of a robot is to achieve goals without human interventions.
- SPA architectures:
 - **Sense**: sense the environment
 - > *collect data through monitors and gauges*
 - **Plan**: take a decision
 - > *consider the different plans*
 - **Act**: modify the environment
 - > *modify the architecture of the system*

Modern ROBOT architectures

- Gat in '97 describes a layered architecture:
 - **Control:**
 - reactive feedback control
 - **Sequencing:**
 - reactive plan execution
 - **Deliberation:**
 - planning



Component control

- In robotics:
 - this layer consists of sensors, actuators and control loops.
- In self-managed systems:
 - consists in the set of interconnected components
 - facilities to report the current status
 - support component creation, deletion and interconnections

CC ~ Research issues

- The component has to expose part of its internal state for management purpose
 - Available technologies:
 - SNMP
 - JMX
 - DTrace
- Find scalable algorithms that minimize the effects of a change on a system, ensuring that safety properties are not violated.

Change Management

- Sequencing layer:
 - react to the changes in the environment
 - execute plans that select new control behaviour
- Speed:
 - consists of a set of pre-specified plans
 - can respond quickly
 - if the situation is “new” , is required a planning (higher level)

CM ~ Research Issues

- Deal with distribution and decentralization
 - need of some level of local autonomy
 - preserving the global consistency
 - obtain a consistent view of the system state
 - design algorithms that:
 - can tolerate inconsistencies
 - terminate in a state that satisfies constraints
 - do not violated safety constraints while converging to a stable state

Goal Management

- Planning:
 - takes the current states and the goal
 - produce a plan to achieve that goal

GM ~ Research Issues

- Goal specification:
 - comprehensible for human
 - machine-readable
- The planning is an expensive task
 - are we able to respond in time?
 - (part of) the planning can be done off-line
 - the planning problem is reduced to a verification one

Thank you
for listening

