Design and runtime architectures to support autonomic management

Etienne GANDRILLE [1,2]
Catherine HAMON [1]
Philippe LALANDA [2]


MESOCA 2013
2013 IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems
Systems complexity increase

- Application size

v 7.2 : more than 8000 man x year... en 2002

3500 man x year

- Heterogeneous technologies
  - example: web interface built using more than 20 frameworks

- and also...
Dynamic composition (at runtime)

- Applications use heterogeneous resources
  - cloud services
  - devices: sensors, boxes, smartphones...

- Problem: no control over resources life-cycle
  - resources may be unknown before application starts
  - resources can appear and disappear
  → Composition must be dynamic
Adaptation and context

Resources availability is unpredictable... and it’s part of the “normal” life of the application.

Applications need to be adapted at runtime.

Context modification \rightarrow application adaptation \rightarrow behavior update

structure update

code hot swapping

Applications may expose **different architectures** during runtime to deliver the **same service**.
Administration

- Administration scope is getting much wider
  - It is the administrator duty to adapt the applications
    => shift from design towards runtime

- Infrastructures are getting bigger and more complex

- Administration tools didn’t evolve as quickly as administration scope
Autonomic computing [1]

- alleviate administrator from doing repetitive or overly complex tasks
- administrator focus on the *what*, not on the *how*

IBM approach: MAPE-K [2]

IBM approach: MAPE-K

**Benefits**

- clear separation of concerns
- adaptation logic is centralized, not melt with code
- evolution is easier
But: there is no magic!

Knowledge...

administrator

high level goals

low level directives

System
Problem: knowledge is often getting lost
Our approach

- Links traceability between architectures
- Maintaining links over components life cycle
A software architecture of a system is a set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.


- Architecture
  - elements (components)
  - relations
  - properties

- An architecture is a model
Maintaining links over components life cycle
From component to components

Component

Abstract component

Implemented component

Runtime component

properties definitions

[parameters definitions] parameters definitions parameters values

[state variables definitions] state variables definitions state variables values
Example: Filter Component

```
FilterComponent

ports
1 in-port : stdin
1 out-port : stdout

FilterSpec

FilterImpl

FilterImpl#124

property

isSecured

isSecured=false

filterValue

filterValue=[default value 1]

filterValue=10

nbProcessMsg

nbProcessMsg=252

nbProcessMsg=252

avgProcessDuration

avgProcessDuration=0.2ms

avgProcessDuration=0.2ms
```
Design architecture

- Conception (specification) knowledge

- Represents the valid set of architectures for the application

- Design architecture has
  - cardinalities
  - components specifications and implementations
  - constraints: for selecting implementation using properties (at runtime)
Design and runtime architecture

What is executed: execution knowledge

What is valid: conception knowledge
The autonomic manager responsibility

1. Administrator sets high level goals.
2. **Autonomic manager** compares and adjusts to conform with high level goals.
3. Design architecture.
4. Runtime architecture represented by system running on a platform.
Validation: Digital Home services

- A rich ecosystem
  - more and more sensors and actuators
  - connected to the cloud
  - dynamic: smartphones (camera, GPS, accelerometers...) enter and leave home

- Adaptation
  - mandatory, to update application as soon as resources availability change
  - no “expert” for administration tasks
Actimetrics use-case

- dynamic
- flexible
- heterogeneous
- distributed

implantation structure
Cilia framework

- Programming model and execution framework designed for mediation applications
- Component based
- Dynamic
Cilia concepts

Cilia execution framework

scheduler
processor
dispatcher
ports

a Cilia chain
mediator
binding
adaptateur
Cilia Workbench
Conclusion

- Architecture is a relevant abstraction for understanding and updating a system.

- Understanding and formalizing the relationships between components along the life-cycle is a key feature.

- Approach validated on the MEDICAL projet:
  - [www.medical.imag.fr](http://www.medical.imag.fr)

- Work in progress: advanced administration use-case involving code deployment.
Thank you!

Etienne Gandrille
etienne.gandrille@orange.com