Moving Model-based Analysis at run-time

Antinisca Di Marco,
adimarco@di.univaq.it
Outline

» Motivation

» Constraints on Models at run time

» Our contribution
  – Model-based dynamic reconfiguration for performance management
  – QoS- and context-aware software services
Motivation

» Self-* systems
» Autonomic systems/services
» QoS negotiation
» Guarantee of SLA satisfaction
New Software Engineering Process

$P_S = \text{Standard Engineering Process}$

$P_F = \text{Future Engineering Process}$

**PROCESS**

- Model1
- Model2
- ... ModelN

**Frozen System**

**SYSTEM**

- $P_S$
- $P_F$

- Core Code
- Self-Evolving/Adaptive Code

-compiling process

Frozen → Self-evolving/adaptive
New Constraints on the Analysis Models

» Analysis models have to be modified and evaluated online:
   > models must be as flexible as possible to be automatically changed according to the reconfiguration\adaptation policy;
   > models should allow their analysis on-the-fly.

» The previous characteristics might be incompatible.
   > The first requires detailed models which permit to apply changes (re-parameterization or structural/behavioral modification) in order to reflect the new system configuration.
   > The second requires analytic or numerical models which are as simple as possible (omitting useless details of system behavior).
Challenges

» To design (numerical/analytical) analysis models expressive enough to describe sensible different alternatives with respect to QoS behavior.

> QoS models at the software architecture level guarantees such expressiveness without introducing too much complexity.

» Composition capabilities of models and analysis become the key aspect of Model-based analysis at run time.
Contribution: Model-based Dynamic Reconfiguration for Performance Management
Dynamic Performance Management

» Managing performance at run time

» Dynamic Reconfiguration
  > The application runs during the reconfiguration

» Monitoring
  > To discover performance problem
  > To parameterize the models of alternative configurations devised from the reconfiguration policies

» On line, model-based decision process
  > Application modeled at the Software Architecture level of details
  > Lightweight
  > No simulation models
The framework

Software architecture

First performance model

Data

Monitor

Current application configuration

Decision Step

New application configuration

Performance model

Solver

Results

Other factors

PaCo – Kick-off Meeting
22-23 ottobre 2008
Issues to address

» What is the relevant data to collect? And how to use it?
  > Data collected is more fine-grained than the performance model parameters.

» When should we reconfigure the application? Which are the reconfiguration alternatives?
  > It depends on the application.
  > From Pre-defined (static alternatives) to completely dynamic defined.

» Models have to be modified and evaluated online (fast solution techniques).
  > Which performance model should we use?

» How do we take the decision on the next configuration?
  > Different aspects should be considered (security, resources availability,...)
The framework for Siena
Publish/Subscribe
Siena Publish/Subscribe

» Scalable Internet Event Notification Architecture.

> Events, Publish/Subscribe Middleware, Scalability/Expressiveness
Siena Publish/Subscribe

Simple rules in the SR interactions:
• Sub are always forwarded to the master by each SR
• Pub are forwarded to the master of the SR
• when a SR receives an event it notify it to its interested sibling
Implementation details: LIRA

- Component
  - Component Agent
  - MIB
- Host Agent
  - MIB
- Manager
  - MIB
Reconfiguration Process and LIRA

1: Event: Notify(Ucomp)
2: CollectData
3: storeCurrentConfig
4: reconfigure
5: readCurrentConfigurationModel
6: store newConfigurationModel
7: writeReconfig(chosenReconfig)
8: return

Performance Manager

Application Manager

Comp Agent

Siena Router

Models DB

Monitoring and performance evaluation local to the Siena routers

The Siena Router Agent triggers the reconfiguration

Centralized decision management and reconfiguration

The Performance Manager decides the next configuration and reconfigures the system
Implementation details
Lesson learned

» The process defined is general and can be reused in several contexts.

» The application of the process in real contexts is feasible (we were able to reconfigure Siena without service interruption and human intervention).

» However the implementation of the Performance Manager component is tightly coupled to the application.

» The approach does not deal with reconfiguration that can change the behavior of the application (e.g., substitution of sw components.)

» The decision step must be designed and implemented carefully.
Some Open Issues

» More sophisticated decision step where different aspects are considered (e.g., the overhead imposed by the suggested reconfiguration).

» Measurement of the overhead introduced by (LIRA) reconfiguration framework

» Planning of Experiments
Contribution: QoS- and context-aware software services
B3G Networks are characterized by mobile and resource-limited devices that communicate through different kind of network interfaces.

Software services deployed in such networks shall adapt themselves according to possible execution contexts and requirement changes.
The main target of PLASTIC project is an integrated model-based solution supporting both the development of services deployable over B3G networks and the definition of their related SLAs.
The PLASTIC vision

The key idea of PLASTIC is that the QoS characteristics of the network and of the devices should be visible at the level of the service.

Which level of QoS can be guaranteed at the application service layer?

- The QoS can be managed at the application service layer to adapt to and exploit the platform capabilities.

- Traditional approaches

- PLASTIC approach
The PLASTIC development process
Two Layers Approach

Service Layer

Service: S3, S4, S5, S6, S7, S8, S9, S10

Service Request

Service Consumer

Software Developer

Service Description

Service Composition
Two Layers Approach

Service Layer

Service request

Service Composition

Component Layer

Component Assembling

Service
Service Description

Component
Component Description

S10
S8
S5
S9
S7

S3
S4

C1
C2

C3
C4

C5
C6

C7
C8

C9

Service Consumer

Software Developer
Two Layers Approach

Service Layer

Service request

Component Layer

Service Composition

Component Assembling

Service Consumer

Software Developer

Service Description

Component Description
PLASTIC modeling

» A PLASTIC UML profile has been devised. It allows modeling functional and extra-functional (QoS) aspects of the service.

» A PLASTIC service model is composed in several views used to structure the UML design in packages:

– Requirement view
– Service view
– Component view
– Implementation view
– Deployment view

PLASTIC Modeling for Performance Assessment

» Requirement view
PLASTIC Modeling for Performance Assessment

» Service view

Service Description Diagram

Business Process Description Diagram
PLASTIC Modeling for Performance Assessment

Component view

Service Specification Diagram

Elementary Service Dynamics Diagram
PLASTIC Modeling for Performance Assessment

» Implementation view
PLASTIC Modeling for Performance Assessment

» Deployment view