Using Dynamic Workflows for Coordinating Self-adaptation of Software Systems

Carlos Eduardo da Silva
ces26@kent.ac.uk
Computing Laboratory
University of Kent, UK

Rogério de Lemos
rdelemos@dei.uc.pt
Dept. of Informatics Engineering
University of Coimbra, Portugal
Agenda

- Introduction
- Background
- Workflow generation
  - Applied to architectural reconfiguration
- Case study
- Conclusions
Motivation

- Self-adaptive systems should be able to define adaptation plans at run-time
  - Deal with variability
  - E.g., Web-based systems

Our approach

- Dynamic workflows for managing self-adaptation
  - Workflows for representing adaptation plans
  - Dynamic workflow generation
Background

Workflow management technology

- Coordination of the flow of work and information of business processes
- Workflows
- Significant maturity level
  - Heterogeneous environments/technologies, modelling languages, etc ...
  - Applied in several domains
    - Including self-adaptive systems [Valetto] [Shrivastava]
Dynamic Workflow

- Modification of workflows
  - Restructuring of workflows
  - Flexible composition
  - Flexible deployment

Workflow generation

- Composition process
  - Tasks
  - Interconnection of tasks
  - For achieving an objective
    - From a particular state
Workflow generation model

Basis for workflow generation model

- 3 layers reference model for self-managed systems
  - Feedback between the layers
- Workflow techniques from different domains
  - Grid computing/Web service composition/Pervasive systems
  - Abstract/concrete workflows
Workflow generation model

- Generate abstract workflows
  - Pre/post-conditions of task templates/workflow

- Generate concrete workflows
  - Populate abstract workflows with actual resources
Architectural reconfiguration based on dynamic workflows

- **Strategy**
  - Abstract configuration

- **Tactics**
  - Concrete configuration
Architectural reconfiguration based on dynamic workflows

- Detailed workflow for reconfiguration
- Control generation process

Diagram:

- Task library
- Workflow generator
- Abstract workflow
- Generate abstract workflow
- Concrete workflow
- Generate concrete workflow
- Selected abstract configuration
- Select configuration
- Identify abstract configurations
- Selected concrete configuration
- Connect configuration
- Identify concrete configurations
- Select concrete configuration
- Connect concrete configuration
- Reconfiguration Commands
- Registry
- Monitoring resources
- Available resources
- Target system

Key:
- Normal flow
- Exceptional flow
- Component
- Activity
- Generation activities
- Strategy activities
- Tactic activities
Example of reconfiguration workflow

◆ **Tactics level** *(Connect configuration)*

◆ **Concrete configuration**
Example of reconfiguration workflow

- Connect configuration
Example of reconfiguration workflow

- **Connect concrete configuration (stateless)**
Task template

- Instantiation of a structure based on atomic actions
  - For dealing with faults
- Applied to workflows that are generated
- Sub-workflow
  - E.g., Connect(Component1, Component2)
Case study

Web-based stock quotes application

- Requirements: dependable stock quotes
- Fault-tolerance techniques: dual and triple module redundancy
- Architectural reconfiguration in the presence of faults
Case study

Example of generated workflow (Connect instances)

- Configuration #1
Example of generated workflow (Connect instances)

- **Configuration #1**
Case study

Example of generated workflow (*Connect instances*)

- **Configuration #2**

![Workflow Diagram]

Key:
- Start
- End
- XOR-split task
- XOR-join task

Carlos Eduardo da Silva

ICSE 2009 SEAMS – May 2009 – 17
Case study

Prototype

- Web services, xADL 2.0, Yawl workflow management system

Some experiments

- Fault injection on components
  - During application execution
  - During reconfiguration
- Generation of concrete workflows
  - Pre-defined abstract workflows
- Generation of abstract workflows
Conclusion

We are using workflows for representing adaptation plans

- Dynamic workflow generation (e.g., depending on the resources availability)
  - Based on the pre- and post-conditions of the plan and a library of tasks

Future work

- Refine the generation model/implementation
  - Workflow meta-models
  - Selection of workflows
  - Exception handling and exceptional behaviour of task templates
  - Pre/post-condition representation
- Experiments with more complex scenarios
- Component-based integration testing