Towards A Knowledge-Based Approach to Architectural Adaptation Management

John Georgas, Richard Taylor
Institute for Software Research
University of California, Irvine

October 31, 2004
Objectives and Introduction

- **Challenge:** Self-adaptive behavior is often *embedded* into implementation code and expressed as *static, preplanned* responses.

- **Goal:** *Decouple* adaptive behavior from system implementations, and allow for the *dynamic* runtime addition and modification of adaptive behavior.

- **Approach:** Develop architecture-centric self-adaptive software using knowledge-based reasoning techniques.
  - Model adaptation policies as *explicit* architectural constructs.
  - *Deploy* policies alongside runtime system and allow for their *independent* and *dynamic* evolution.
  - Base decision-making on *knowledge-based expert system* reasoning.
An Overview

- Constraint Resolver: Maintain architectural constraints and invariants.
- Architectural Configuration Manager: Maintains architectural configuration version information and implements recovery operations.
- Architectural Adaptation Manager: Based on the knowledge-based system’s directives, enacts architectural adaptations, and dynamically modifies adaptation policies.
- Architectural Model: A model of the system’s components, connectors, and links.
- Architectural Evolution Manager: Maintains runtime consistency between architectural model and implementation.
- Implementation Runtime: Deployed component-based software system which may be instrumented for monitoring.
- Knowledge-Based Expert System: Perform knowledge-based reasoning based on observations and dynamic adaptation policies.
- Adaptation Policies
- Observations
- External Events: Events which may add knowledge or influence adaptations.

Legend:
- Software component
- Information flow
Elements of Decision-Making

- Architectural Observations
  - Known information about a system.
  - Structural Observations: System-independent information relating only to architectural structure.
  - Behavioral Observations: System-dependent knowledge based on specific system goals.

- Adaptation Responses
  - Modifications to a system’s architectural model.

- Adaptation Policies
  - A policy is a mapping from observations to responses that determines the specifics and timing of adaptations.
  - Rule-based specification following an IF-THEN structure.
Benefits

- Better *decoupling* between systems and the policies governing self-adaptive behavior.
  - Independent and parallel development of policies and system implementations.

- *Finer-grained* manipulation of adaptive behavior.
  - Modifications can be made at the level of individual policies.

- *Dynamic*, runtime modification of adaptive policy.
  - Post-deployment modification of adaptation policy during system runtime.
  - Less foresight needed at design-time.
Present and Future

- Prototype system developed using:
  - Schema extensions to xADL 2.0 for policy definition.
  - *Java Expert System Shell (JESS)* to support expert system-based reasoning.
  - Modifications of existing *Critic* components for constraint management.

- Future plans:
  - Policy representation refinements and tool-support.
  - Expand general feature set (especially constraint management).
  - Additional experimentation and validation (particularly attention to computational overhead).
  - Transitioning techniques to *distributed* settings.
    - Integration of both local and remote knowledge.
    - Driving global adaptive behavior through local decisions.
Summary

- Architecture-based adaptation policy modeling, representation, and enactment.
  - Adaptation policies are modeled at the architectural level.
  - Application of knowledge-based methods/tools for policy enactment.
  - Dynamic and independent runtime modification of adaptation policy.

- Targeting systems with dynamic adaptation needs.
  - Allowing for post-deployment inclusion/modification of adaptation policies.
  - Less foresight needed at design-time.