CLAM: Managing Cross-layer Adaptation in Service-Based Systems

Asli Zengin, Annapaola Marconi, Luciano Baresi and Marco Pistore
zengin@fbk.eu
SOA Research Unit, FBK-Irst

The IEEE International Conference on Service Oriented Computing & Applications
Irvine, Dec 14, 2011
Outline

- **Motivation**
  - State-of-the-art adaptation & analysis approaches for SBAs
  - Our research context

- **Proposed solution: CLAM**
  - CLAM approach
  - Illustration: motivating application scenario

- **Current Status of Work**
  - CLAM v1.0 & work-in-progress for CLAM v2.0

- **Future Work**
  - Research plan for the rest of time
Service-Based Systems
State-of-the-art Adaptation and Analysis Approaches for Service-Based Systems
**Research Problem:**
- We want to see the impact of an adaptation on the whole SBS to avoid
  - incompatible or useless adaptations for the application.

**Challenge:**
- **R1.** identify the problems that might occur in different concerns of the SBS layers due to a local adaptation.
- **R2.** tackle these problems by proposing new adaptations in a consistency with the overall system.

**Proposed Solution:** Cross-layer Adaptation Manager (CLAM):

>>> A platform that integrates different analyzers & adaptation mechanisms for cross-layer analysis and extension of an adaptation.
CLAM Approach

1. Cross-layer System Modeling

High level dependency model of the system parts: Make “already existing implicit dependencies” explicit!
Cross-layer Rule Engine

Predefined CLAM Rules

- **Analyzer Rules** associate analyzers with system elements.
  - E.g., $KPI \rightarrow \text{cost analyzer, time analyzer}$

- **Solver Rules** associate a set of solvers with each adaptation need defined in the system.
  - E.g., $\text{reduce process time} \rightarrow \text{process optimizer, service quality renegotiator, service replacer}$

- **Enactor Rules** associate a set of enactors with each adaptation action included in the system.
  - E.g., $\text{add service} \rightarrow \text{dynamic service binder}$
Motivating Application Scenario

- Call & Pay Taxi Workflow (CPTS composite service)
  - Telecom Co provides Parlay X services:
    - SMS, location service (LS), payment service (PS)
  - Taxi companies provide real taxi service (TS)
Analysis and Adaptation Tools used in CLAM for the Application Scenario
CLAM Approach

2. Supporting Architecture

- Get adaptation need then produce report:
  1) OK; adaptation
  2) NOK; no adaptation

- Get adaptation then produce report:
  1) OK
  2) NOK; aNeed
  3) NOK; no aNeed

Diagram:
- CLAM
- Cross-layer Rule Engine
- Model Updater
- Cross-layer Tree Constructor
- Adaptation Paths
- Selected Adaptation Strategy
- Enactors

Flow:
- Initial Adaptation
- Check adaptation, adaptation need
- Cross-layer Rule Engine
- New tree node
- Adaptation paths
- Cross-layer Tree Constructor
- Cross-layer SBS Meta Model
- Triggered adaptation
- Initial Adaptation Need
- Data mismatch solver
- Process optimizer
- DataNet analyzer
- Time analyzer
- Cost analyzer
- Report back
- Is new adaptation harmless?
- Instant System Configuration
- Infrastructure
- Service
- Instant Process Models
Illustration of CLAM:
*Output of CLAM impact analysis*

new configuration of system when A1 is applied to M0

Adaptation A1

list of components to be invoked for A1
Application after CLAM Impact Analysis
Current Status of Work

- We have implemented CLAM v1.0 and evaluated it on the presented application scenario.

- Currently we are working on
  - formalization and implementation design of CLAM v2.0.

CLAM v2.0 will enable the customization of the overall analysis:

- CLAM configuration parameters:
  - Check a proposed adaptation again via the same analyzer (that triggered the problem): ON / OFF
  - Set how to invoke solvers for a given adaptation need: invoke all / invoke the highest priority / custom selection
  - Set how to invoke analyzers for a given adaptation: invoke all / invoke the highest priority / custom selection
  - Set how to construct the tree: breadth first / depth first / whole tree
  - Set how to prevent infinite trigger of adaptations: stop at the repetition of an adaptation need / stop at a threshold number of iterations
Research Plan for Future Work

- Formalization & Implementation of CLAM v2.0

- Evaluation of the implementation on various case studies:
  - show that CLAM works well:
    - Flexibility
      - Changing configuration parameters
    - Run-time applicability
      - Trade-off between time performance vs having larger set of alternative paths
    - Scalability
      - See the performance when we increase the number of tools in the analysis
  - show that CLAM costs little:
    - Evaluate the efforts required to integrate a new tool
    - Evaluate the design-time efforts needed to use CLAM

- Validation:
  - Formally show that the proposed solution addresses the cross-layer adaptation problem.
Conclusions

- **CLAM makes a comprehensive impact analysis of an adaptation:**
  
  Given (i) the local adaptation and analysis tools and (ii) the cross-layer system dependency model:
  
  - **CLAM platform** validates an initial adaptation trigger for the entire service based system.

- **CLAM is generic:**
  
  Abstraction of the service-based system into the high-level dependency model enables a generic solution:
  
  - CLAM can work with **different application domains having different layers and system elements.**

- **CLAM is extensible:**
  
  One can always **plug in new tools** to the integration platform
  - to enhance analysis
  - to switch to another application domain
THANK YOU!

Questions?

Comments?
Solution Overview
Algorithm of the Overall Impact Analysis

COMPONENTS DISCOVERY (through CLAM rules)

- upon receiving adaptation
  - update current system configuration
  - identify a list of analyzers to be invoked, put them in the priority queue.

- upon receiving adaptation need
  - identify a proper solver (adaptation proposer) to be invoked, put it in the priority queue.

TREE CONSTRUCTION

- invoke components continuously
  - Upon receiving analyzer report for an adaptation:
    - If OK >> contact next component in the queue
    - If NOK & no AdaptationNeed proposed >> stop analysis for that SBS configuration
    - If NOK & AdaptationNeed >> find the proper solver, add it in the queue, contact next component
  - Upon receiving solver report for an adaptation need:
    - If OK & Adaptation >> find the relevant analyzers, add them to queue, contact next component
    - If NOK & no Adaptation found >> stop analysis for that SBS configuration

- update cross-layer adaptation tree:
  - Tree branches: analyzer/solver reports
  - Tree nodes: current SBS configuration + current queue (list of analyzers, solvers to be invoked)
  - Update tree upon receiving a report: add branch and add node