Self-Architecting Software SYstems (SASSY) from QoS-Annotated Activity Models

Sam Malek
Naeem Esfahani
Daniel A. Menascé
João P. Sousa
Hassan Gomaa

May 19th 2009
ICSE Workshop on Principles of Engineering Service Oriented Software Systems

Presented by Tejas Gandre (G01034008)
Outline

➢ Motivation

• Research Objective
• SASSY Framework
• A Taste of SASSY
  – Activity-Oriented Language
  – Architecture Generation
  – Methodology
• Future Work
Emerging SOA-Enabled Pervasive Systems

• We are witnessing the emergence of a new breed of software systems

• Driven by the proliferation of smart spaces, portable devices, and advances in wireless network connectivity

• Enabled by service-oriented architectures
  – Autonomy: decoupling of service providers from consumers
  – Flexibility: dynamically discovering and binding to services
  – Standards: alleviating interoperability and integration challenges
Key Characteristics

• Smart emergency response system
  – Smart buildings with sensors
  – Smart fire stations that monitor the sensors and dispatch fire engines
  – Smart fire engines wirelessly connected to the rest of the system
  – Smart hospitals monitoring the situation and dispatch ambulances

• Innately dynamic and unpredictable

• Functional requirements are often not completely known at design-time, and may change at run-time
Challenges

• Software engineers need to rethink the way they compose such software systems
  – Existing techniques rely on human reasoning and manual intervention

• There is a pressing need for
  – Rapid composition of software systems at run-time based on the users’ changing needs
  – Autonomous adaptation of the software system at run-time to satisfy the system’s functional and QoS requirements
Outline

• Motivation

➢ Research Objective

• SASSY Framework

• A Taste of SASSY
  – Activity-Oriented Language
  – Architecture Generation
  – Methodology

• Future Work
State of the Art

• Self-Adaptive Software: Software that is designed to automatically change its functionality (capabilities) at runtime
  – 3-layer model from Imperial
  – ArchStudio from UCI
  – Rainbow from CMU

• What is known: in the absence of explicit architectural model, the engineer is left to reason about adaptation from memory or from source code
  – Not feasible!

• Architecture should serve as the primary focus of reasoning

• Underlying assumption: software requirements are known at design time

• Key question: how can we design self-adaptive software in pervasive settings, where the system’s requirements are not completely known at design-time?
SASSY: Self-Architecting Software SYstems

• Pushes the envelope
  – From self-adaptive software → self-architecting software

• Doman expert, instead of a software engineer, specifies the functional and QoS requirements in an activity-oriented modeling language

• SASSY automatically constructs an optimal architectural model and the corresponding software system that satisfies the user’s requirements
Somewhat Similar to Lego
Fiction or Reality: Key Assumptions

• Focus on service-oriented systems
• Reliance on a domain ontology
  – Provides precise definition of domain concepts, including activities and services
  – Terminology that is commonly understood by domain experts
• Informed users that are trained in the requirements modeling language
Outline

• Motivation
• Research Objective
  ➢ SASSY Framework
• A Taste of SASSY
  – Activity-Oriented Language
  – Architecture Generation
  – Methodology
• Future Work
Outline

• Motivation
• Research Objective
• SASSY Framework
  ➢ A Taste of SASSY
    ➢ Activity-Oriented Language
    ➢ Architecture Generation
    ➢ Methodology
• Future Work
Service Activity Schemas (SAS)

- A small emergency response system modeled in SAS
- This shows the high-level requirements for the system
- The notion of coordinator is implicit
Internal Details of 911 Dispatcher

Emergency Phone System

Building Locator

Occupancy Awareness

Building Category Finder

Availability

Security

startToOp call

getLoc

smokeDet

routeToOp call

status

status.response == true

estimate

estimated

ocEstimat.num > 0

category.type == Critical

investigate

reqHelp

reqMultFS

reqSingFS
Modeling the QoS Requirements
Self-Architect the Base Architecture

- Identify the service providers satisfying the functional requirements
- (Near-) Optimally select the best providers with respect to QoS
Modeling the QoS Requirements
If the QoS requirements cannot be satisfied using existing providers, apply QoS patterns

In this example, we apply the replication and mediator patterns
Self-Architect Coordinator’s Behavior
Architecture Generation using GReAT

SAS Meta-model

Transformation Modeling

Transformation Executions

SAS Model

Input

SAS to Architecture Model Transformer

GReAT Engine

Transforms

Architecture Model

Model Transformation Specification in GReAT

Refers to

SAS to Architecture Model Transformer

GReAT Engine

Architecture Meta-model

Refers to
Outline

• Motivation
• Research Objective
• SASSY Framework
• A Taste of SASSY

➢ Future Work
Ongoing Research & Future Work

- Developing a repository of QoS patterns
- Fully automating application of patterns
- Utility function elicitation
- Optimal architecture selection
- Adaptation and evolution infrastructure
Questions

Domain Ontology

Service Activity Schema
Base Software Architecture Generator
Base Software Architecture
Service Discovery
Optimal Service Selection

Utility functions $u_r(r)$

QoS met? No
Critical SSS Determination
Alternative Architecture Generator

Yes
Architectural Patterns
Executable Software System

SASSY Run-time Support System