SASSY: A Framework For Self-Architecting Service-Oriented Systems

Presenter
Dat Tien Le
dattl@ifi.uio.no

INF5360: Seminar on Dependable and Adaptive Distributed Systems

Department of Informatics
University of Oslo

February 11, 2015
Papers to present

- **SASSY: A Framework for Self-Architecting Service-Oriented Systems**
  - By Daniel A. Menascé, Hassan Gomaa, Sam Malek, and João P. Sousa
SASSY: A Framework for Self-Architecting Service-Oriented Systems

- Solve the challenge of self-architecting systems to **deliver optimal QoS**.

- Topics covered:
  - Self-adaptive software
  - Self-managing
  - Self-optimizing
  - Software engineering
Challenges in engineering QoS SOA systems

- Lack of separation of concerns between **functional** and **non-functional** requirements.
- A large number of various QoS patterns should be considered by humans.
- QoS requirement may change after design phase while the design is static.
A show case

(a)

(b)

Fault tolerant

Occupancy awareness 1

Occupancy awareness 2

Approximate awareness

Smoke detector

911 dispatcher coordinator

Building category finder

Building locator

Load balancer

Building locator 3

Building locator 2

Building locator 1

Component

Connector

Interface
Self-architecting approach

Develop and register services

Service directory

Service discovery

Develop QoS architectural patterns

QoS-pattern library

Specify SASs and system service architectures (SSAs)

Generating base SSA architecture

Self-architecting

Service binding and deploying coordination logic

Develop software adaptation patterns

Adaptation-pattern library

Software engineers

Domain experts

Service activity schema (SAS) + SSAs

Base architecture

Near-optimal architecture

Running system

Model read/write

Model read

Human-computer interaction

Communication
Self-architecting approach

Developed by third-party
Self-architecting approach

Dynamically adapt system in run-time

Develop and register services

Develop QoS architectural patterns

Develop software adaptation patterns

Service directory

QoS-pattern library

Adaptation-pattern library

Service discovery

Developed by third-party

Specify SASs and system service architectures (SSAs)

Generating base SSA architecture

Self-architecting

Service binding and deploying coordination logic

Service activity schema (SAS) + SSAs

Base architecture

Near-optimal architecture

Running system

Model read/write

Human-computer interaction

Model read

Communication
Self-architecting approach

Dynamically adapt system in run-time

Develop and register services
Develop QoS architectural patterns
Develop software adaptation patterns

Software engineers

Specify SASs and system service architectures (SSAs)

Domain experts

Generating base SSA architecture

Self-architecting

Service binding and deploying coordination logic

Service discovery

Developed by third-party

Service directory
QoS-pattern library
Adaptation-pattern library

Service activity schema (SAS) + SSAs
Base architecture
Near-optimal architecture

Model read/write
Model read
Human-computer interaction
Communication

Dat Tien Le INF5360, ifi/UiO  SASSY: A Framework For Self-Architecting Service-Oriented Systems- Feb. 11, 2015
Self-architecting approach

- Dynamically adapt system in run-time
- Developed by third-party

1. Develop and register services
2. Develop QoS architectural patterns
3. Develop software adaptation patterns

- Service directory
- QoS-pattern library
- Adaptation-pattern library

- Service discovery

- Service coordination + QoS goals

- Specifying SASs and system service architectures (SSAs)

- Generating base SSA architecture

- Self-architecting

- Service binding and deploying coordination logic

- Service activity schema (SAS) + SSAs

- Base architecture

- Near-optimal architecture

- Running system

- Model read/write
- Human-computer interaction
- Communication
Service activity schema (SAS)

- Is used to
  - Specifying local activities and service usages
  - Describe the sequence of interactions among service usages and activities
  - Support gateways:
    - Inclusive
    - Exclusive
    - Parallel

- QoS requirements is described in Service sequence scenarios (SSSs) graph
- Has the semantics for generating executable architectural models
- Support formal verification
Service activity schema (SAS)
Service sequence scenarios

![Diagram of service sequence scenarios](image-url)
A base system service architecture is generated from SAS.

Consists of structural and behavior models:
- Structural models are based on xADL language.
- Behavioral models are based on finite state process language.
Self-architecting

- Determining an derived architecture that maximizes a utility function.
- Is a NP-hard problems: given p architectural patterns, s service providers and n service instances, (ps)^n solution exists
Self-architecting

- The problem is solved by hill-climbing approach to find sub-optimal solution.

The problem with hill climbing is that it gets stuck on "local-maxima"
Self-architecting

- The problem is solved by hill-climbing approach to find sub-optimal solution.

The problem with hill climbing is that it gets stuck on "local-maxima".
Self-adaption in runtime

- Following the MAPE-K (monitor, analyze, plan, execute, knowledge) approach.
Summary

● Paper main goal
  ● Introducing a new approach in designing and engineering service-oriented system satisfying QoS requirement.

● To achieve this goal
  ● Describe the design, development and deployment process of the system
  ● Explain how the self-architecting and self-adapting of the system work
Interesting points

- Key issues for traditional software design to satisfy QoS requirement
- An inventive approach with self-architecting
- Separation of concerns between functional and non-functional requirement
- Reasoning as to design’s choices in languages and models.
Questions and discussion

- The authors claim that they have developed an framework for self-architecting system
  - The infrastructure is described at an abstract and concept level
  - No details of how an implementation of the infrastructure works
  - How are architectural constraints expressed and checked in detail?
  - No reasoning why hill-climbing is chosen?
The END