Towards Proactive Adaptation: A Journey along the S-Cube Service Life-Cycle

Andreas Metzger (University of Duisburg-Essen, Paluno), Eric Schmieders (University of Duisburg-Essen, Paluno), Cinzia Cappiello (Politecnico di Milano, DEI), Elisabetta Di Nitto (Politecnico die Milano, DEI), Raman Kazhamiakin (FBK-Irst), Barbara Pernici (Politecnico die Milano, DEI), Marco Pistore (FBK-Irst)

Funded by the European Community’s 7th Framework Programme FP7/2007-2013 / Objective 1.2

www.s-cube-network.eu
Agenda

- Motivation and Problem Statement
- Example Scenarios
- Solution
- Future Work
Motivation: Context S-Cube Service Life Cycle

- **Focus**
- **Identify Adaptation Need**
- **Operation & Management**
- **Requirements Engineering**
- **Evolution**
- **Design**
- **Realization**
- **Deployment & Provisioning**
- **Enact Adaptation**
- **Runtime**
- **Design Time**

Eric Schmieders – MESOA 2010, Timisoara, September 2010
Motivation: Problem Statement

- Problem in the context of Service Based Applications (SBA) is: it must be determined whether **SBAs might deviate from its requirements** during its future runtime

- SBAs shall **automatically and dynamically adapt** → Target is to keep large and/or massive distributed systems maintainable

- Major work on adaptation has been centered around **two reactive adaptation capabilities based on monitoring**. These approaches … monitor **individual services** → unclear whether the failure of a single service leads to a violation of the SBA’s requirements (**Service Monitoring**) and/or … are restricted to **monitoring of requirements** → Monitoring events might arrive so late that an adaptation of the SBA is not possible anymore (**Requirements Monitoring**)
Scenario 1: Service Monitoring

cumulative response time [ms]

end-to-end requirement

cumulative response time = allowed range according to SLA

failure = actual response time

S1 S2 S3 S4 S5

steps
Scenario 2: Requirements Monitoring

- **End-to-end requirement deviation**
- **End-to-end requirement**
- **Failure**

- **Steps:** S1, S2, S3, S4, S5
- **[ms]:**
  - 1300
  - 1200
  - 1100
  - 1000
  - 900
  - 800
  - 700
  - 600
  - 500
  - 400
  - 300
  - 200
  - 100

- **Symbols:**
  - ▲ = allowed range according to SLA
  - X = actual response time

Eric Schmieders – MESOA 2010, Timisoara, September 2010

© S-Cube - 6
Solution: Identify Adaptation Need

- **SBA elements** are related to **assumptions** (based on SLAs) about the SBAs context (the services to be executed)

- When a service invocation **deviates from its assumption** the following check is performed:
  - M is the monitored data
  - A’ are the assumptions about the services which are not invoked yet
  - S is the specification of the Workflow (e.g. in BIR)
  - R are the requirements (e.g. expressed in LTL)

- The past monitoring data (M) together with the assumptions about the not yet invoked services (A‘) and the specification (S) are checked against the requirements (R):
  - S, M, A' |= R

- If R is satisfied, then the workflow execution is **continued**

- If R is not satisfied, the SBA must be **adapted**
Solution: S-Cube Service Life Cycle Phases

S, A', M |= R

Identify Adaptation Need

Operation & Management

M |= A

Requirements Engineering

R := \{r_1, r_2 \ldots\}

S

Identify Adaptation Strategy

Adaptation

Design

Enact Adaptation

Deployment & Provisioning

Realization

A := \{a_1, a_2 \ldots\}

S, A |= R
Solution: Identify Adaptation Need

end-to-end requirement (R)

end-to-end requirement deviation

= Assumption A

= Assumption A’

= actual response time

= Identified adaptation need; based on S, A’, M |= R check

M ≠ A

X

S1 S2 S3 S4 S5

steps
Summary and Future Work

- Presented: **proactive adaptation technique for Service Based Applications**, with focus on identifying adaptation needs early in time
- Techniques have been introduced along the key phases of the **S-Cube service life-cycle**
- Validation: **Implementation of a prototype** (ongoing)
- Future Work:
  - Enrich current approach with **online testing** in order to predict the services quality ➔ **Critical Problems can be identified earlier**, even before a service is reacting e.g. too slow
  - Reduction of false-positives by analysing **historical data**