Semantic SOA - Realization of the Adaptive Services Grid

results of the final year bachelor project

Bastian Steinert,
ASG bachelor project team
Outline

• review of midterm results
• engineering methodology
• service development
• build-up of ASG software stack
• positioning in overall ASG project
migration of Services Grid Infrastructure (C5) from JBoss AS / Postgres environment to an IBM environment (WebSphere AS / DB2)

specification and implementation of a Mobile Service Provider scenario
work@NIWA web solutions

development of a dynamic supply chain scenario with industrial background
Outline

• review of midterm results
• engineering methodology
• service development
• build-up of ASG software stack
• positioning in overall ASG project
Define your Business ...

- identify business requirements
- collect and acquire structured domain knowledge
- detect pre-existing internal and external functionality
Service Landscaping - Registration

Diagram showing the relationship between DomainName, DirectCheck Domain, DomainState, VerisignCheck Domain, DomainState, DomainName, contact, adminContact, nameServers, and DirectiRegister Domain.
Service Landscaping - Payment

- **saferpay** Credit Card Authorization
  - AmountOfMoney
  - CreditCardInfo for MC, VISA, AMEX, Diners or JCB

- **saferpay** Payment
  - AmountOfMoney
  - TransactionHandle
  - TransactionData

- **paypal** Instant Credit Card Payment
  - AmountOfMoney
  - CreditCardInfo for MC, VISA, Amex, Discover
  - person
  - TransactionData
Service Landscaping

- find appropriate granularities; reuse vs. coarse-grained design (loose coupling / strong cohesion)
- find common concepts and data types
- define service capabilities; input and output
- discover dependencies; preconditions and effects
- derive ASG required artifacts
  - common domain ontology
  - semantic service specifications
Semantic SOA - Realization of the Adaptive Services Grid

Application Engineering

Composition

Negotiation

Enactment

Invocation

Semantic Specification

Ontology

uses concepts defined in

grounding

tool support needed in

uses data types defined in

WSDL

XML Schema

1:1 mapping necessary
Service Engineering

Atomic Service Layer

AS
- XML-RPC Client
  - PaypalPayment

AS
- SMTP Client
  - DenicRegisterDomain

AS
- Web Service Client
  - DirectiCheckDomain

AS
- XML-RPC Client
  - PleskWebhosting
  - ...
Essential Artifacts in ASG Methodology

Service Landscaping

Atomic Service Development

Ontology
- DomainName
- DomainState
- CreditCard
  - number
  - expDate
  - CVV

Semantic Service Specification
- preconditions
- postconditions
- service grounding

XML-Schema
```
<xs:schema
targetNamespace...
xmins:tns...
xmins:xsd...
elementFormDefault...
...
</xs:schema>
```

WSDL
```
<portType>
<binding name="...
<soap:binding... 
<operation name... 
</operation>
```

methodology explained in-depth: “Adaptive Solutions for Internet Services’ Supply Chains”
Outline

• review of midterm results
• engineering methodology
• service development
• build-up of ASG software stack
• positioning in overall ASG project
Selected Services

- Payment Services
  - PayPal Payment
  - Saferpay Authorization
  - Saferpay Payment

- Domain Services
  - Directi Check Domain (.com, .net, .org, .biz, .name, .info, ...)
  - Directi Register Domain
  - Verisign Check Domain (.com, .net)
  - Denic Check Domain (.de)
  - Update Local Nameserver
MexMan4AS

Bastian Steinert, ASG bachelor project team

Semantic SOA - Realization of the Adaptive Services Grid
Atomic Service Development

- external service providers offer client libraries (e.g. WS-Clients, XML-RPC clients, SMTP-Clients)

- gain detail knowledge about provided service’s concepts, data-types, and behavior

- transformation of ASG concepts, names and data-types according the particular service

Interacting with real world services is non-trivial and time-consuming
PayPal API
Services and Security

- interaction with external services, especially payment, must be secure
- typically realized with standard authentication and encryption mechanisms
  - usage of Java security functionality
  - authentication is based upon certificates
Services and Security (2)

• **2 major problems**

  • client use Sun JDK specific encryption algorithm but: WebSphere AS runs on top of it’s own JDK

  • client APIs enforces an input parameter specifying the path to the certificate/key file
    • this means: the client code and thus the EJB does I/O operations
    • but a standard compliant EJB 2.1 must not use I/O
Services and Security (3)
J2EE Connector Architecture (JCA)

Web Service Client

Resource Adapter

Atomic Service EJB
Atomic Service EJB
Atomic Service EJB
Atomic Service EJB

Connection Manager
Security Manager
Transaction Manager

J2EE Application Server

External Application
Outline

- review of midterm results
- engineering methodology
- service development
- build-up of ASG software stack
- positioning in overall ASG project
The overall ASG stacks

Composition
- plan
  - discover service specification
  - discover service grounding

Negotiation
- negotiate
  - renegotiate
  - replan

Enactment
- enact
  - instantiate & negotiate
  - invoke

Instantiation & Invocation

Discovery
- Ontology
- Semantic Service Specifications
- Service Groundings

Facade
Discovery

- execute queries on semantic service specifications
  - requests from composition and negotiation
- encapsulates a reasoner (Flora) that can evaluate requests in logic expression (F-Logic)
Discovery - Improvements

- replaced mocks with fully functional discovery component
  - run as a singleton/synchronize requests
  - semantic service specifications and underlying ontology are loaded at startup
  - extraction of suitable conditions for a given state
  - extraction of service groundings for negotiation based on preconditions and effects

- using RMI Server for Flora Reasoner
  - in-process usage of reasoner is unstable and leads to faulty behavior
  - reasoner installations show different behavior on different OS
Composition

- compose a process to fulfill the user request containing initial state, goal state and parameter values
- select conditions from semantic service specifications that lead to a semantic process execution plan
Composition - Improvements

• strong interaction with Harald Meyer; we provided:
  • service specifications and ontology for supply chain scenario
  • sample composer outputs (composed services) as needed by negotiation
  • test cases for composer

• separation of concerns between Composition and Negotiation was mandatory
  • selection of condition sets instead of semantic service specifications necessary for negotiation

• main problem: unstable Flora reasoner
  • development without tool support
Negotiation

- negotiate with service agents QoS parameters
- select concrete services for semantic service invocations based on the negotiation process
- create BPEL compliant execution plan
Negotiation - Improvements

- input: execution sequence with semantic invocations, variables, and assignments

- only conditions (preconditions & effects) are specified instead of concrete service specification
  - fault-tolerance implemented

- tasks:
  - find service groundings for each condition set
  - create service instances and negotiate with them
  - select service fulfilling the requirements (availability)
  - replace semantics with executable invocations and essential variable assignments
the WSDL issue

- negotiation requires interface descriptions (WSDL)
- are now retrieved during instantiation and attached to the composed service, should be used by enactment
- negotiate method
Enactment

- show flexibility of ASG reference architecture

- evaluation of several open-source workflow engines according the following criterias
  - usability
  - standard compliance (BPEL)
  - legal issues, e.g. licenses
  - integration efforts

- engine of choice: Fivesight PXE

- explained in detail: “Open-source Workflow Engine Integration into ASG platform”
Enactment - Improvements

Integration of PXE

- challenge: adaption to ASG specific needs
  - usage of WS-Addressing for invocation of atomic services (to realize the stateful behavior)
  - automatic generation of PXE deployment descriptor

- transform enactment input (composed service) into a deployable bundle
  - BPEL compliant execution plan
  - interface data (in WSDL) of all services to invoke
  - PXE deployment descriptor
Service Invocation

Facade

Composition

Plan

Negotiation

Discover service specification

Discovered service specification

Enact

Composition

Discover service grounding

Discovered service grounding

Replan

Instantiation & Invocation

Invoke

Ontology

Semantic Service Specifications

Service Groundings

Service Implementation
Outline

• review of midterm results
• engineering methodology
• service development
• build-up of ASG software stack
• positioning in overall ASG project
What has been done?

• proof of general ASG concepts
• extended implementation of main ASG components
• detection and partial solution of structural pitfalls
  • namespace awareness
  • web service styles - document/literal (wrapped), ...
  • mapping between service groundings and service specification / condition sets
  • mapping between ontology concepts and XML types/elements
ASG Key Features

- **Seamless integration of heterogeneous external services**
  - Semantic-annotation of services basing on domain ontology
  - Methodology and tools
  - Proof of architecture’s flexibility

- **On-demand creation of composed services**
  - Semantic-enabled composition of supply chain processes

- **Reliable service provision with assured end-to-end quality of service**
  - Enactment of composed services
  - Simple negotiation
Restrictions / open issues

• fault handling, re-planning, re-negotiation
  • retrieve semantic information for XML process data

• intermediate result handling in case of faults
  • PXE monitors on process and activity level
    (current state, assignments of variables)

• compensation activities
  • credit card refund
  • nameserver entries
  • ...

Demonstration
Demonstration: Infrastructure

ASG Stack
- GUI Servant
- Composer
- Negotiation
- Discovery
- Enactment

SOAP
- JMX
- JBoss AS with Web Container
- JBoss AS: Atomic Services
- Websphere AS: Atomic Services

tb3-1

tb3-2
- C5 Container
- IBM DB2: C5 Database
Demonstration: Exemplary Compositions

[Diagram showing various service interactions and flow processes related to payment and domain management]
Thanks for your attention.

Any questions?