Semantic SOA - Realization of the ***Adaptive Services Grid***

results of the final year bachelor project

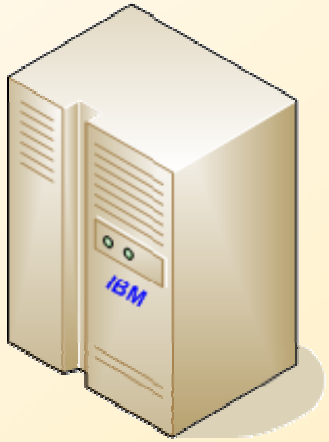
Bastian Steinert,
ASG bachelor project team

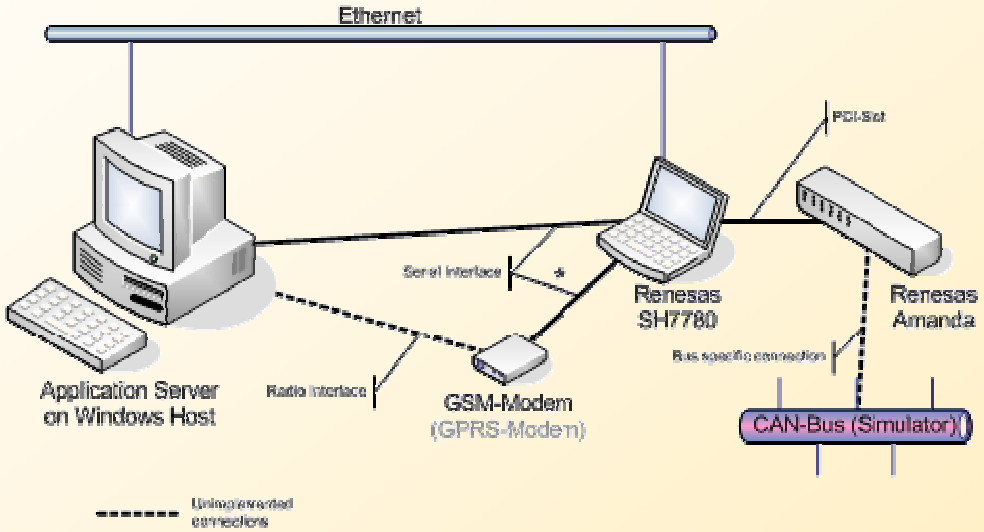
Outline

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

work@DaimlerChrysler Research

migration of Services Grid Infrastructure (C5)
from JBoss AS / Postgres environment
to an IBM environment (WebSphere AS / DB2)

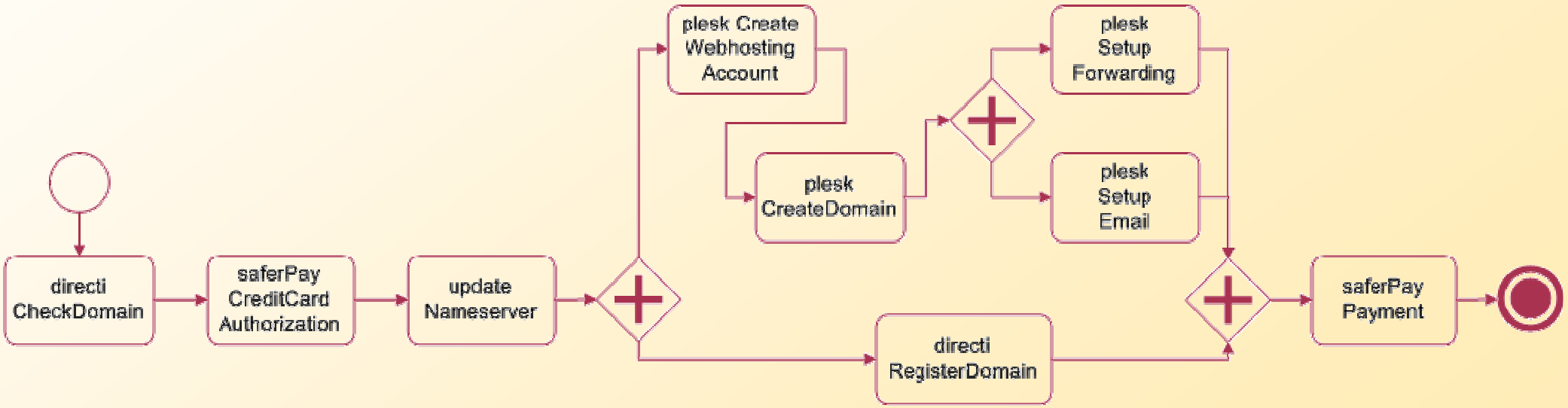




specification and implementation of a
Mobile Service Provider scenario

work@NIWA web solutions

development of a
dynamic supply chain scenario with
industrial background

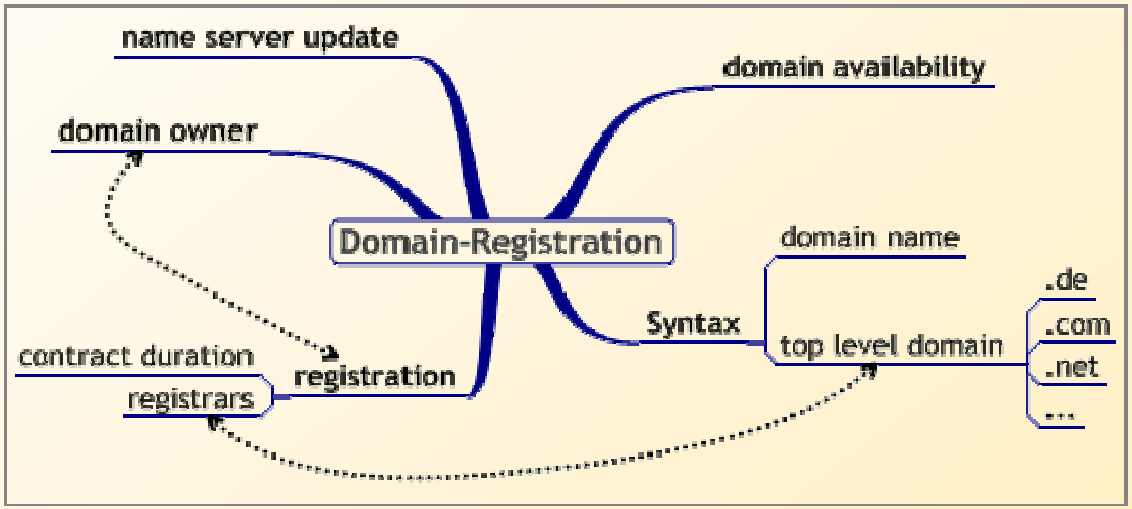


Outline

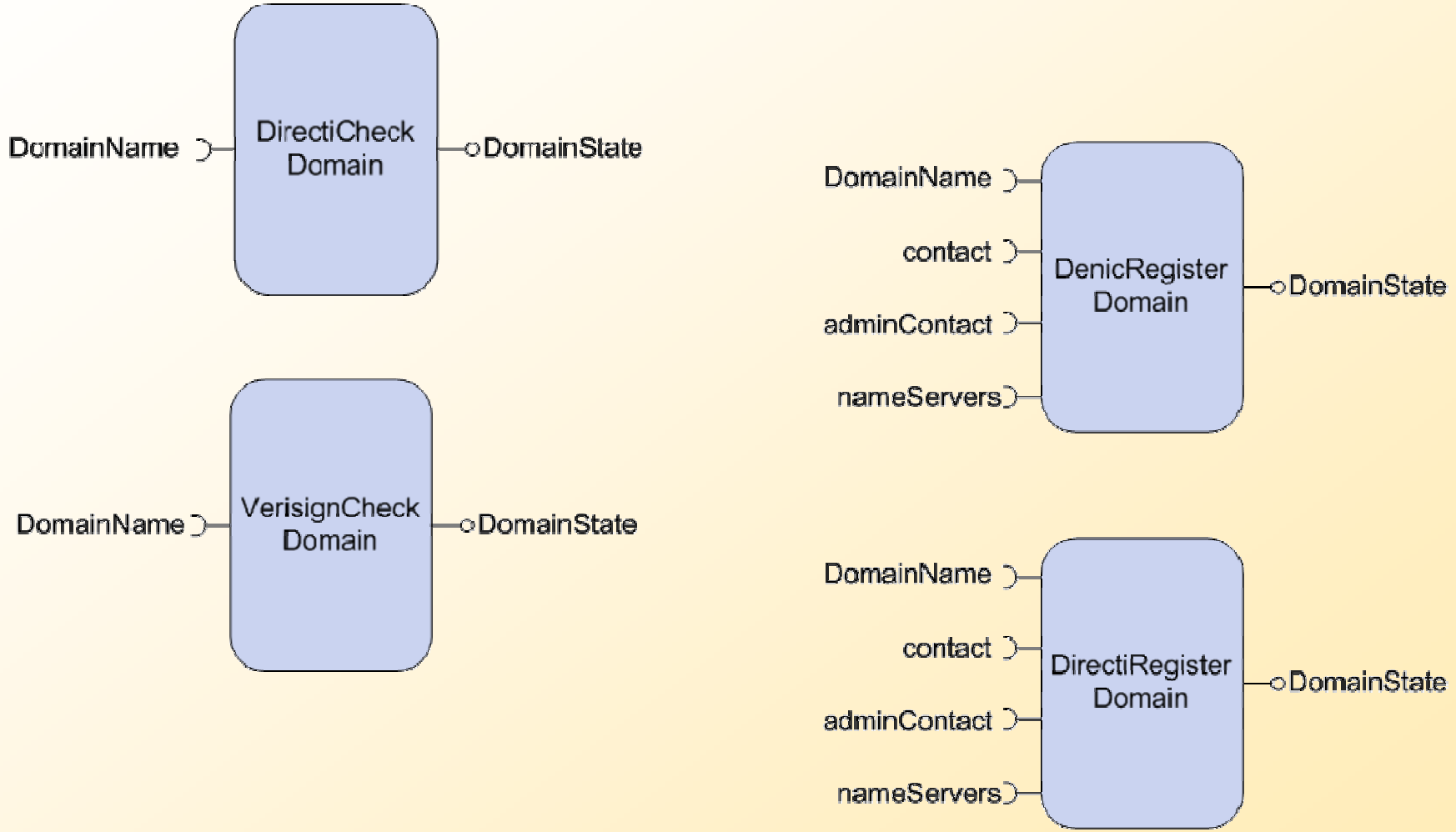
- review of midterm results
- engineering methodology
- service development
- build-up of ASG software stack
- positioning in overall ASG project

Define your Business ...

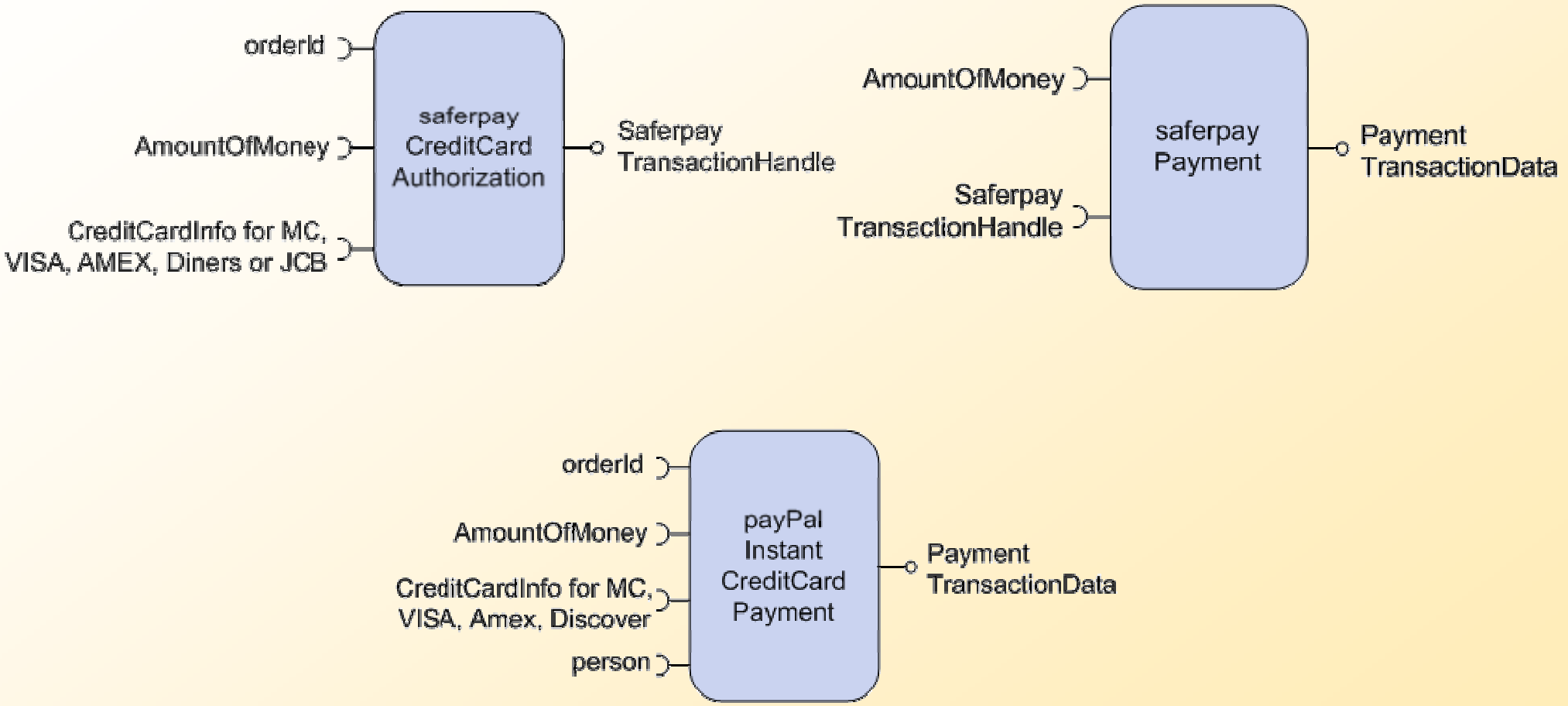
- identify business requirements
- collect and acquire structured domain knowledge
- detect pre-existing internal and external functionality



Service Landscaping - Registration



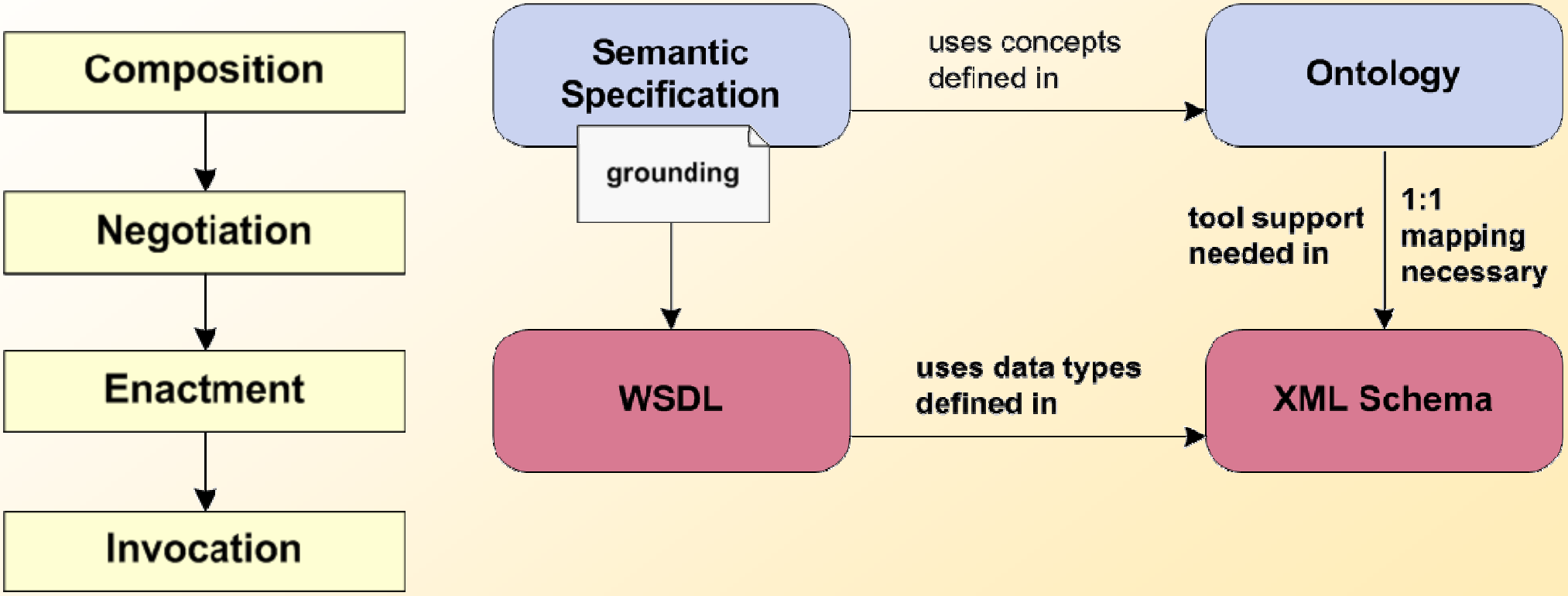
Service Landscaping - Payment



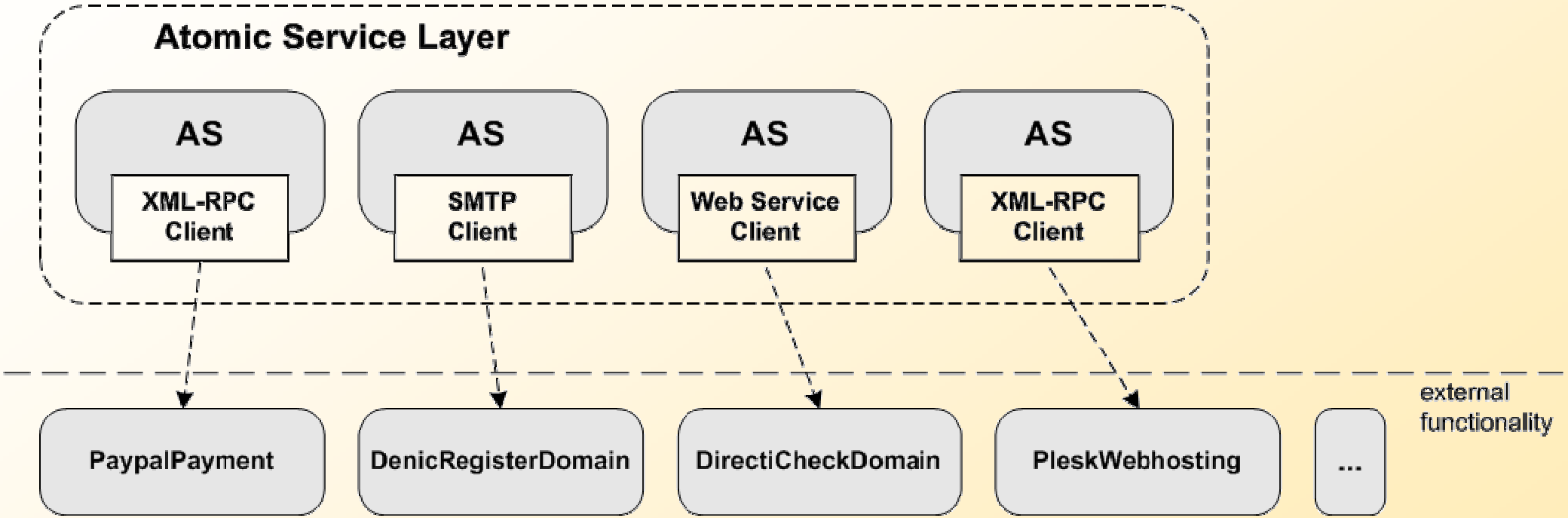
Service Landscaping

- 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

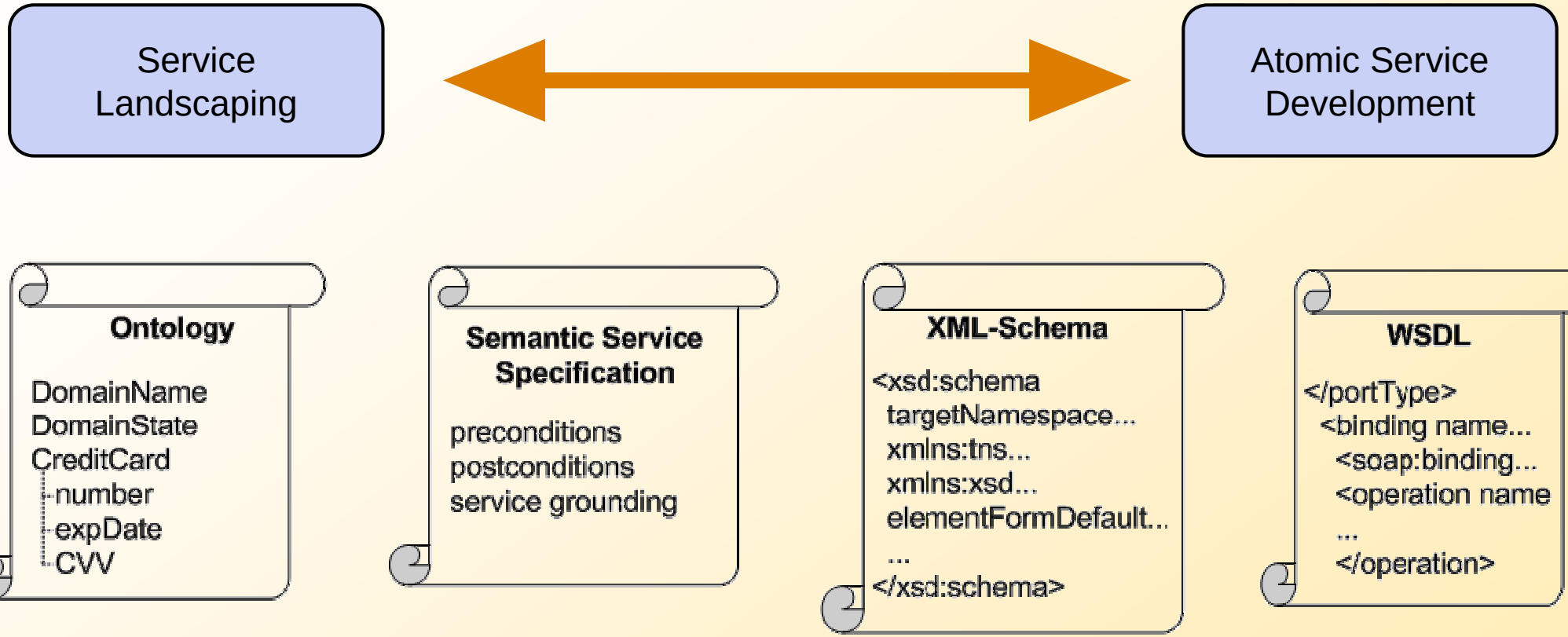
Application Engineering



Service Engineering



Essential Artifacts in ASG Methodology



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

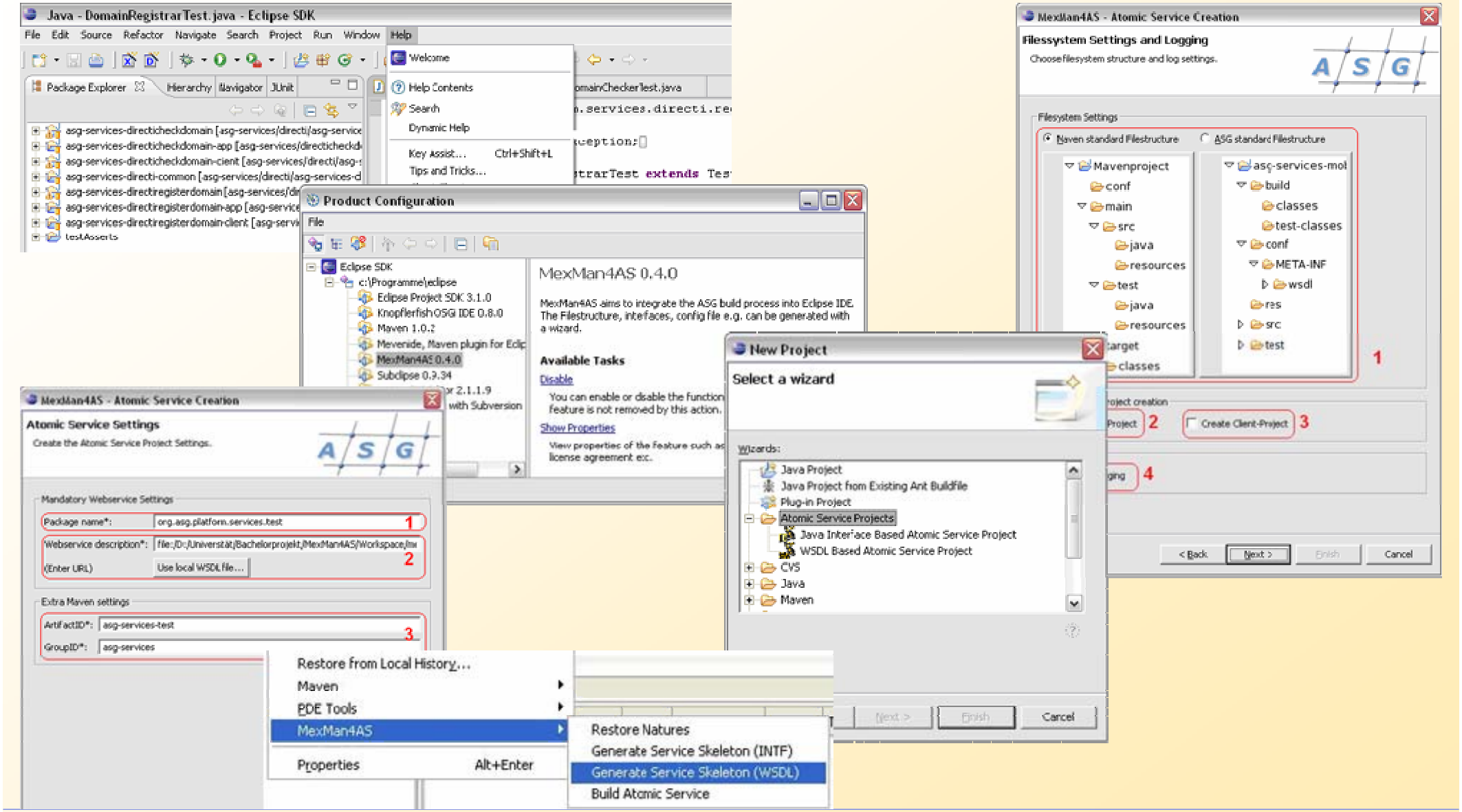
Outline

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

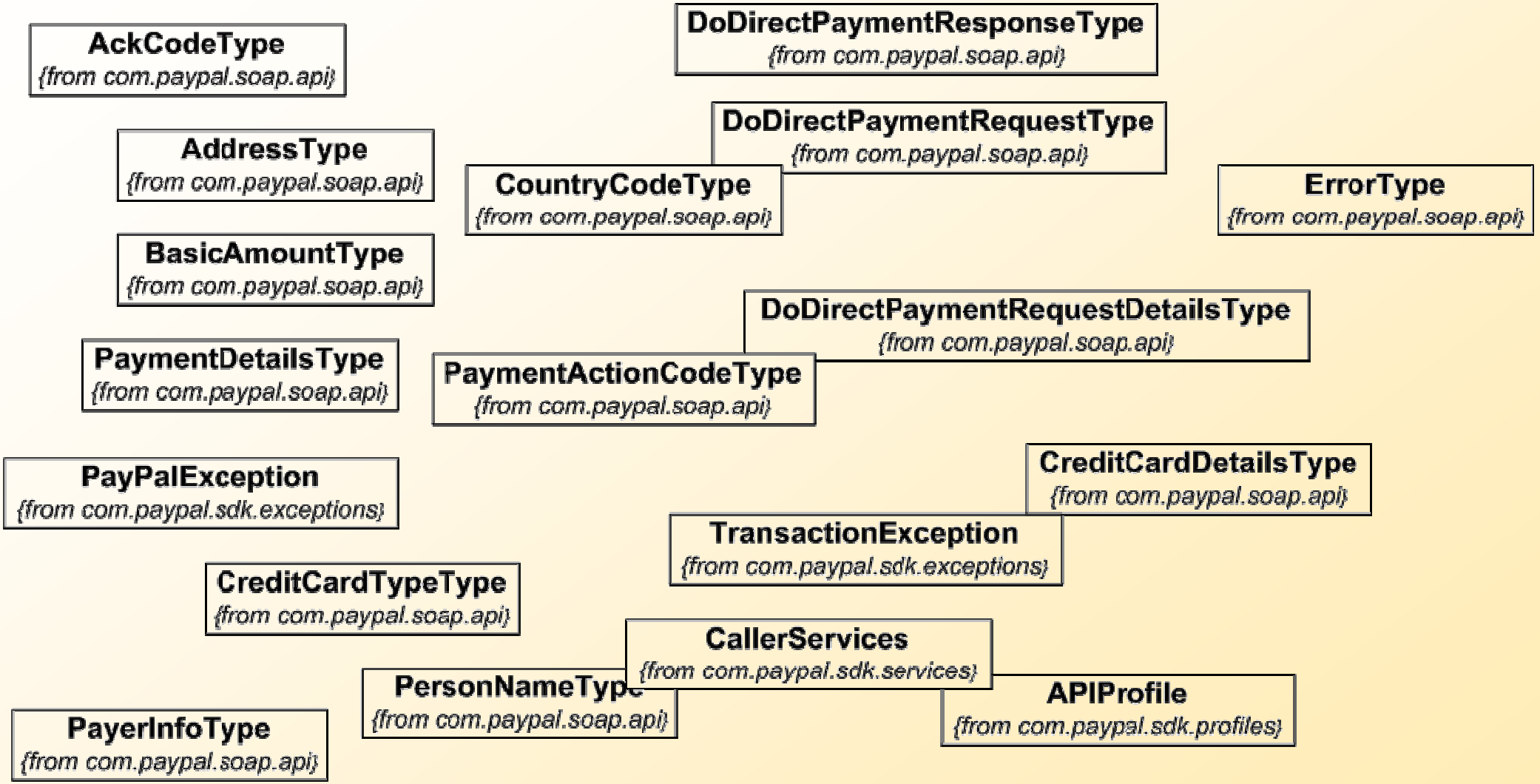
Selected Services

- 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


Java - DomainRegistrarTest.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome
Help Contents
Search
Dynamic Help
Key Assist... Ctrl+Shift+L
Tips and Tricks...
Package Explorer
Hierarchy
Navigator
JUnit
Product Configuration
File
Eclipse SDK
c:\Programme\eclipse
Eclipse Project SDK 3.1.0
KnopflerfishOSGi IDE 0.8.0
Maven 1.0.2
Mevenide, Maven plugin for Eclip
MexMan4AS 0.4.0
Subclipse 0.9.34
MexMan4AS 0.4.0
MexMan4AS aims to integrate the ASG build process into Eclipse IDE. The Filestructure, interfaces, config file e.g. can be generated with a wizard.
Available Tasks
Disable
You can enable or disable the function feature is not removed by this action.
Show Properties
View properties of the feature such as license agreement etc.
MexMan4AS - Atomic Service Creation
Filesystem Settings and Logging
Choose filesystem structure and log settings.
Filesystem Settings
Maven standard Filestructure
ASG standard Filestructure
Mavenproject
conf
main
src
java
resources
test
java
resources
target
classes
asg-services-mol
build
classes
test-classes
conf
META-INF
wsdl
res
src
test
1
Create Client-Project
3
2
4
MexMan4AS - Atomic Service Creation
Atomic Service Settings
Create the Atomic Service Project Settings.
Mandatory Webservice Settings
Package name*:
org.asg.platform.services.test
1
Webservice description*:
file:/D:/Universität/Bachelorprojekt/MexMan4AS/Workspace/tw
2
(Enter URL)
Use local WSDL file...
Extra Maven settings
ArtifactID*:
asg-services-test
3
GroupID*:
asg-services
New Project
Select a wizard
Wizards:
Java Project
Java Project from Existing Ant Buildfile
Plug-in Project
Atomic Service Projects
Java Interface Based Atomic Service Project
WSDL Based Atomic Service Project
CVS
Java
Maven
< Back
Next >
Finish
Cancel
Restore from Local History...
Maven
PDE Tools
MexMan4AS
Properties
Alt+Enter
Restore Natures
Generate Service Skeleton (INTF)
Generate Service Skeleton (WSDL)
Build Atomic Service

PayPal API

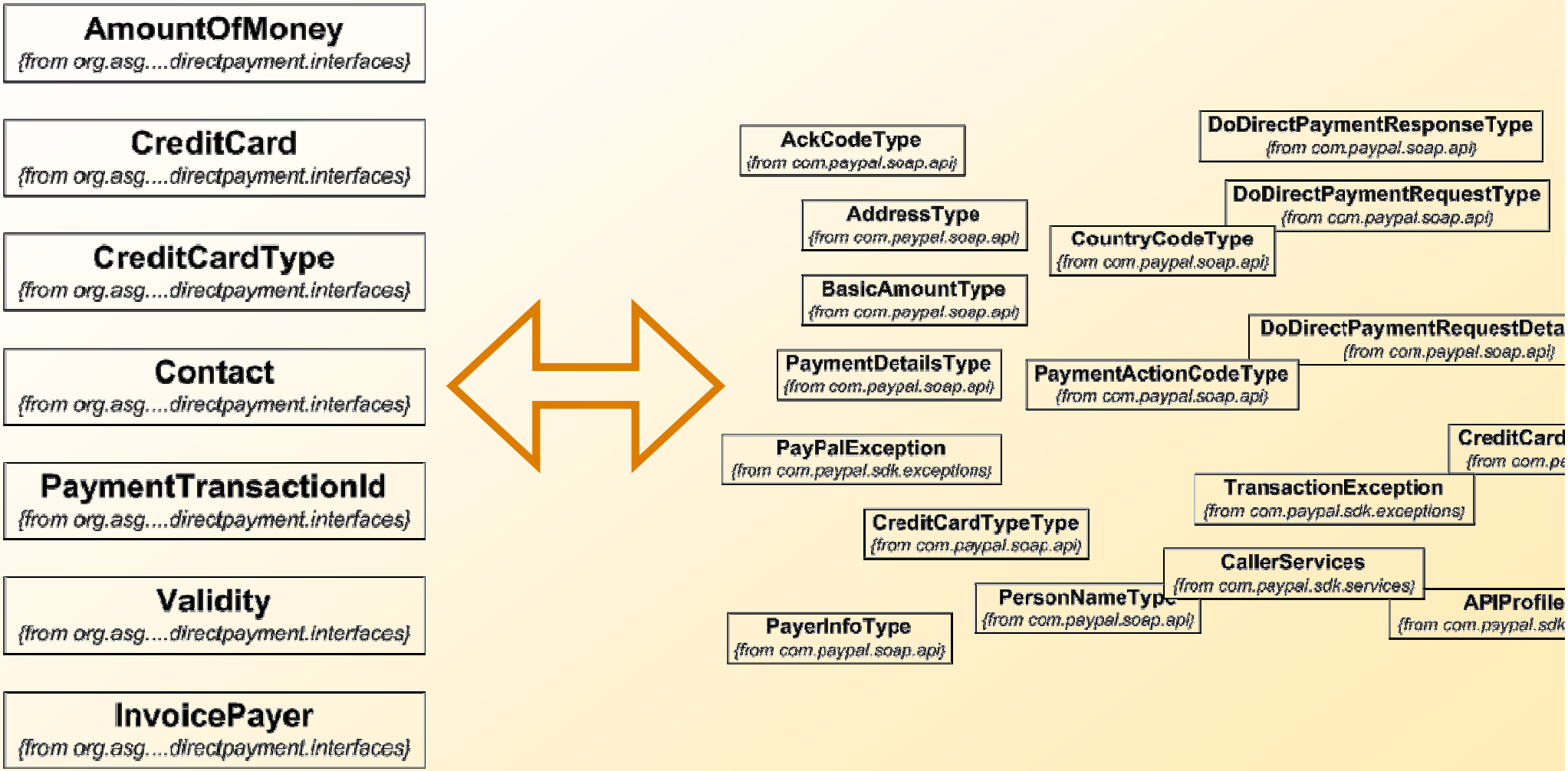

AckCodeType
{from com.paypal.soap.api}
DoDirectPaymentResponseType
{from com.paypal.soap.api}
DoDirectPaymentRequestType
{from com.paypal.soap.api}
AddressType
{from com.paypal.soap.api}
CountryCodeType
{from com.paypal.soap.api}
ErrorType
{from com.paypal.soap.api}
BasicAmountType
{from com.paypal.soap.api}
DoDirectPaymentRequestDetailsType
{from com.paypal.soap.api}
PaymentDetailsType
{from com.paypal.soap.api}
PaymentActionCodeType
{from com.paypal.soap.api}
PayPalException
{from com.paypal.sdk.exceptions}
CreditCardDetailsType
{from com.paypal.soap.api}
TransactionException
{from com.paypal.sdk.exceptions}
CreditCardTypeType
{from com.paypal.soap.api}
CallerServices
{from com.paypal.sdk.services}
PersonNameType
{from com.paypal.soap.api}
APIProfile
{from com.paypal.sdk.profiles}
PayerInfoType
{from com.paypal.soap.api}

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


AmountOfMoney
{from org.asg....directpayment.interfaces}
CreditCard
{from org.asg....directpayment.interfaces}
CreditCardType
{from org.asg....directpayment.interfaces}
Contact
{from org.asg....directpayment.interfaces}
PaymentTransactionId
{from org.asg....directpayment.interfaces}
Validity
{from org.asg....directpayment.interfaces}
InvoicePayer
{from org.asg....directpayment.interfaces}
AckCodeType
{from com.paypal.soap.api}
AddressType
{from com.paypal.soap.api}
BasicAmountType
{from com.paypal.soap.api}
PaymentDetailsType
{from com.paypal.soap.api}
PayPalException
{from com.paypal.sdk.exceptions}
CreditCardTypeType
{from com.paypal.soap.api}
PayerInfoType
{from com.paypal.soap.api}
DoDirectPaymentResponseType
{from com.paypal.soap.api}
DoDirectPaymentRequestType
{from com.paypal.soap.api}
CountryCodeType
{from com.paypal.soap.api}
DoDirectPaymentRequestDeta
{from com.paypal.soap.api}
PaymentActionCodeType
{from com.paypal.soap.api}
CreditCard
{from com.pe
TransactionException
{from com.paypal.sdk.exceptions}
CallerServices
{from com.paypal.sdk.services}
PersonNameType
{from com.paypal.soap.api}
APIProfile
{from com.paypal.sdk

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

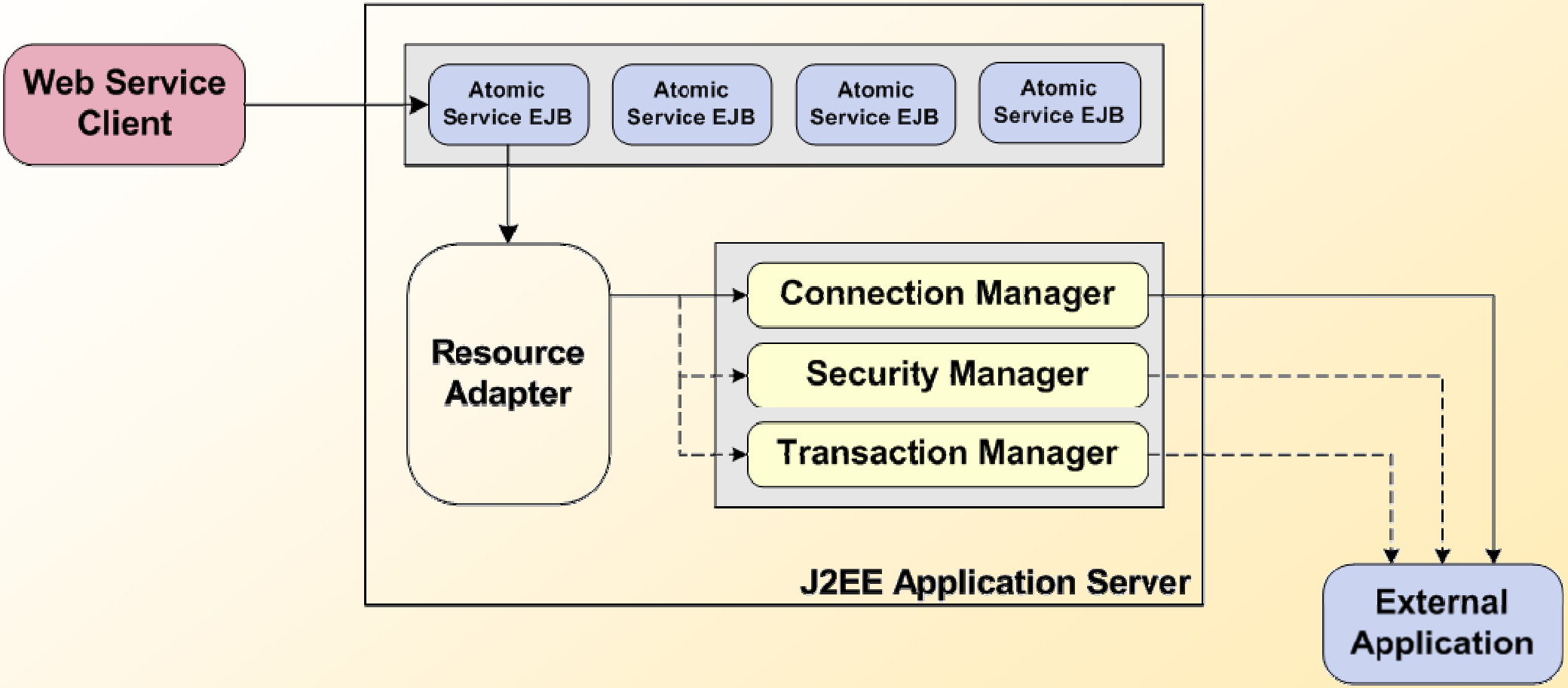
Services and Security (2)

- ***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

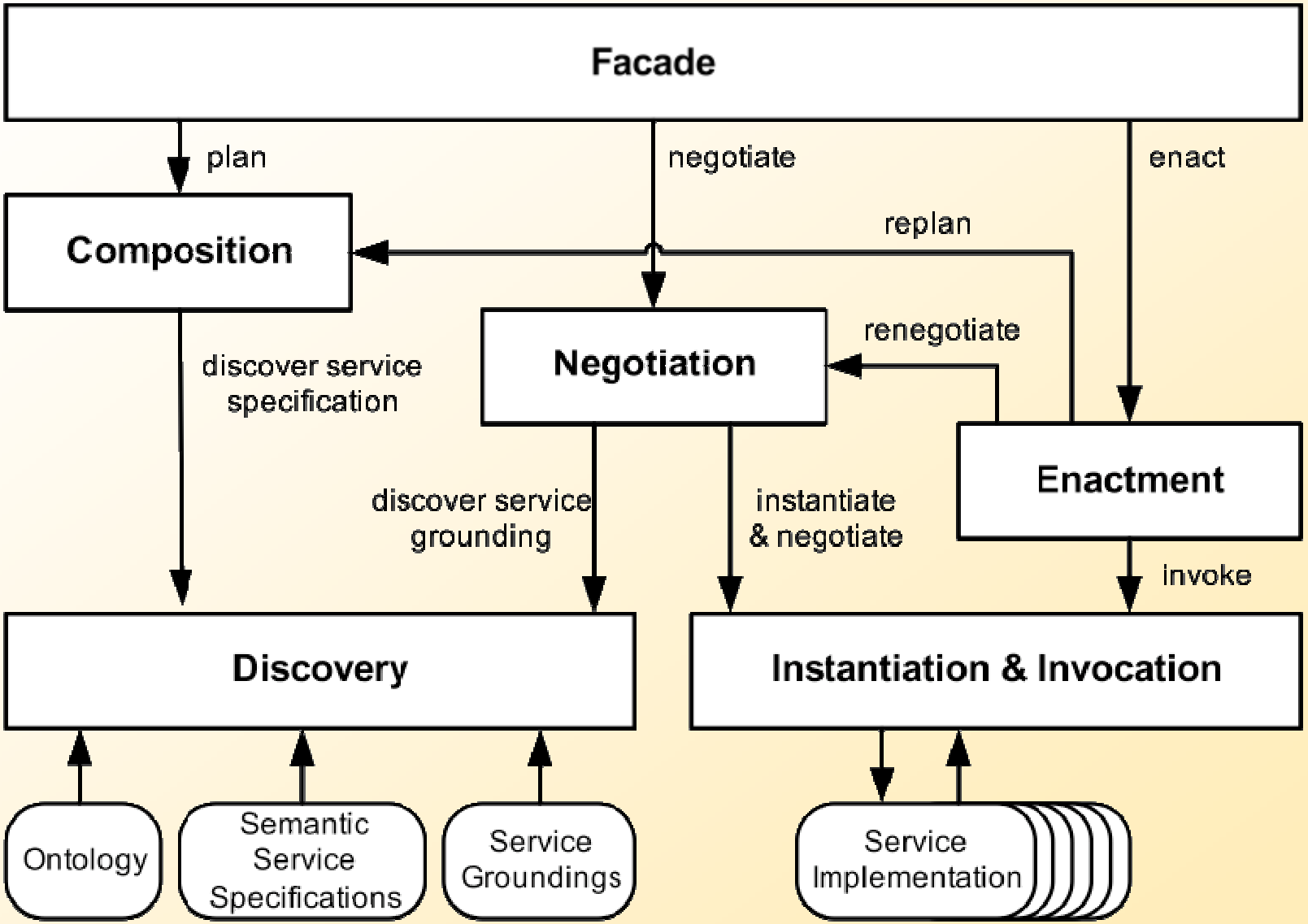
Services and Security (3)
J2EE Connector Architecture (JCA)



Outline

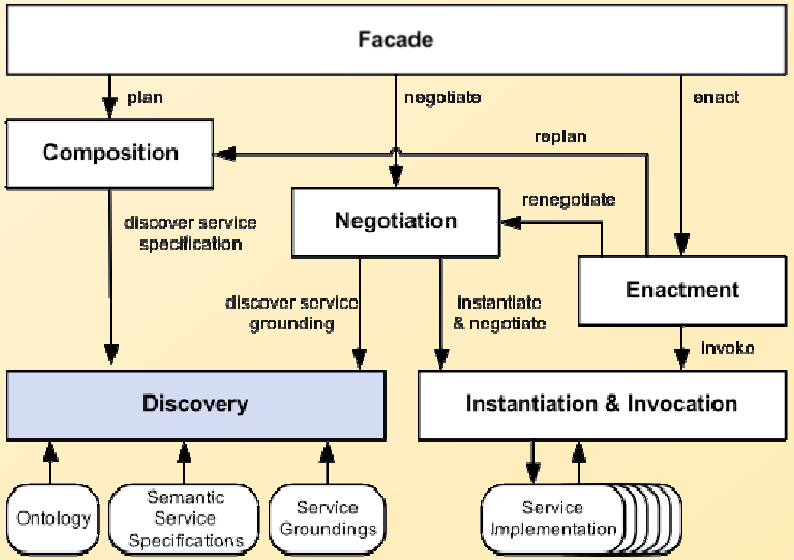
- review of midterm results
- engineering methodology
- service development
- build-up of ASG software stack
- positioning in overall ASG project

The overall ASG stacks



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)

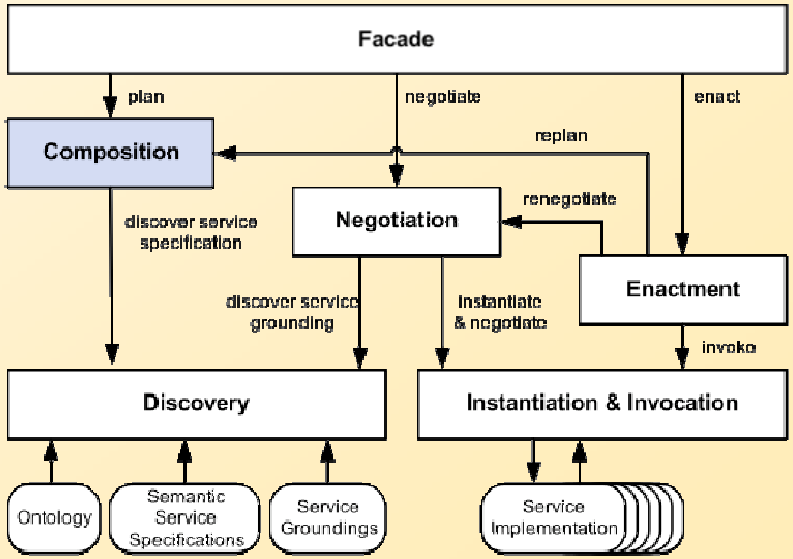


Discovery - Improvements

- 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

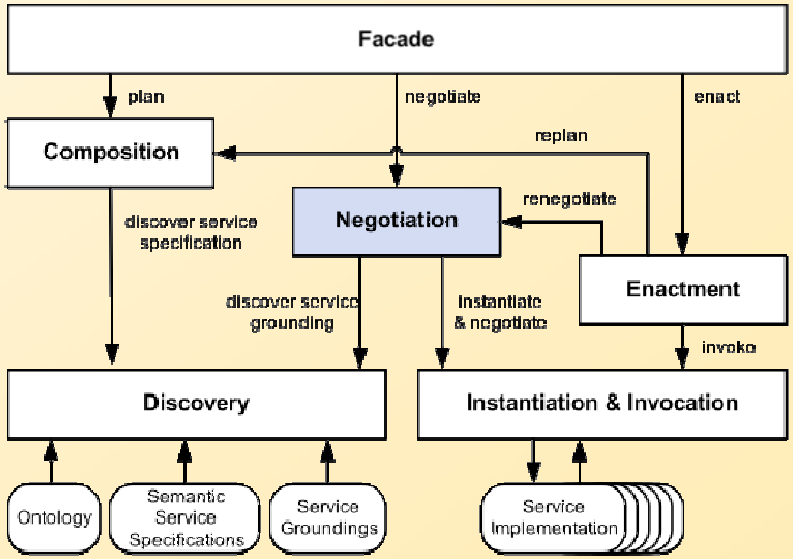


Composition - Improvements

- 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

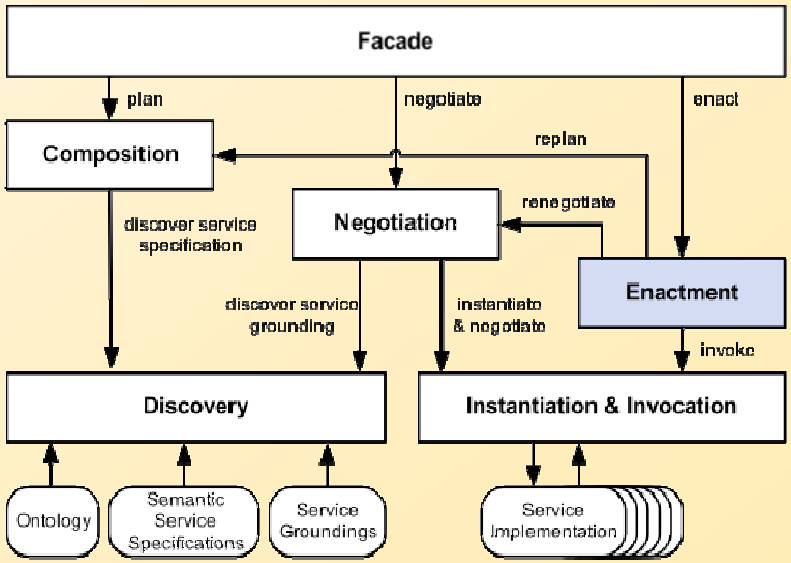
- input: execution sequence with semantic invocations, variables, and assignments
- only conditions (preconditions & effects) are specified instead of concrete service specification
  - fault-tolerance implemented
- tasks:
  - find service groundings for each condition set
  - create service instances 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

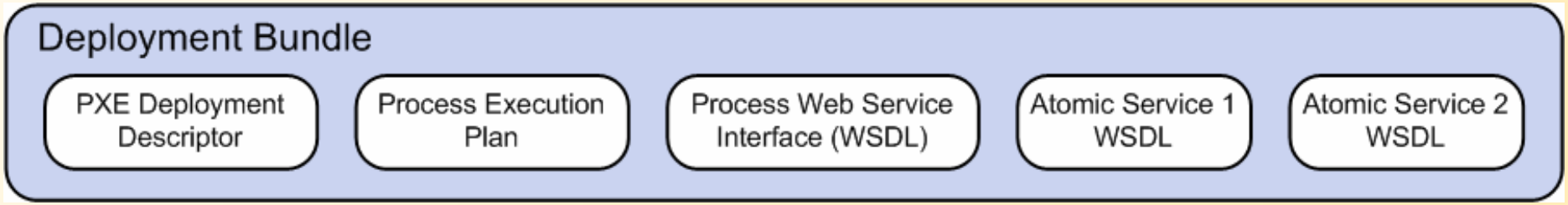
- show flexibility of ASG reference architecture
- evaluation of several open-source workflow engines according the following criterias
  - usability
  - standard compliance (BPEL)
  - legal issues, e.g. licenses
  - integration efforts
- engine of choice: Fivesight PXE
- explained in detail:
  “Open-source Workflow Engine Integration into ASG platform”



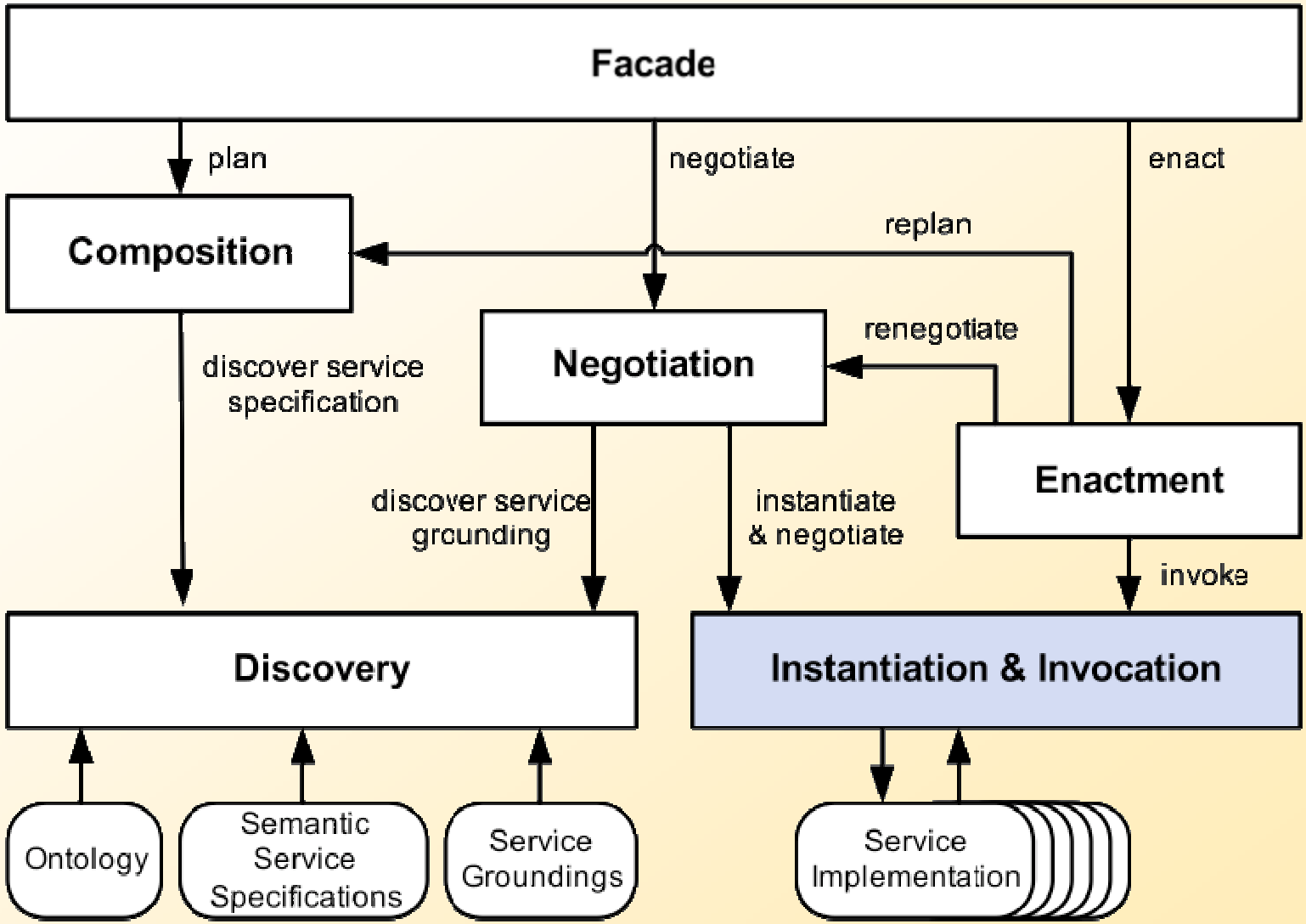
Enactment - Improvements

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



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

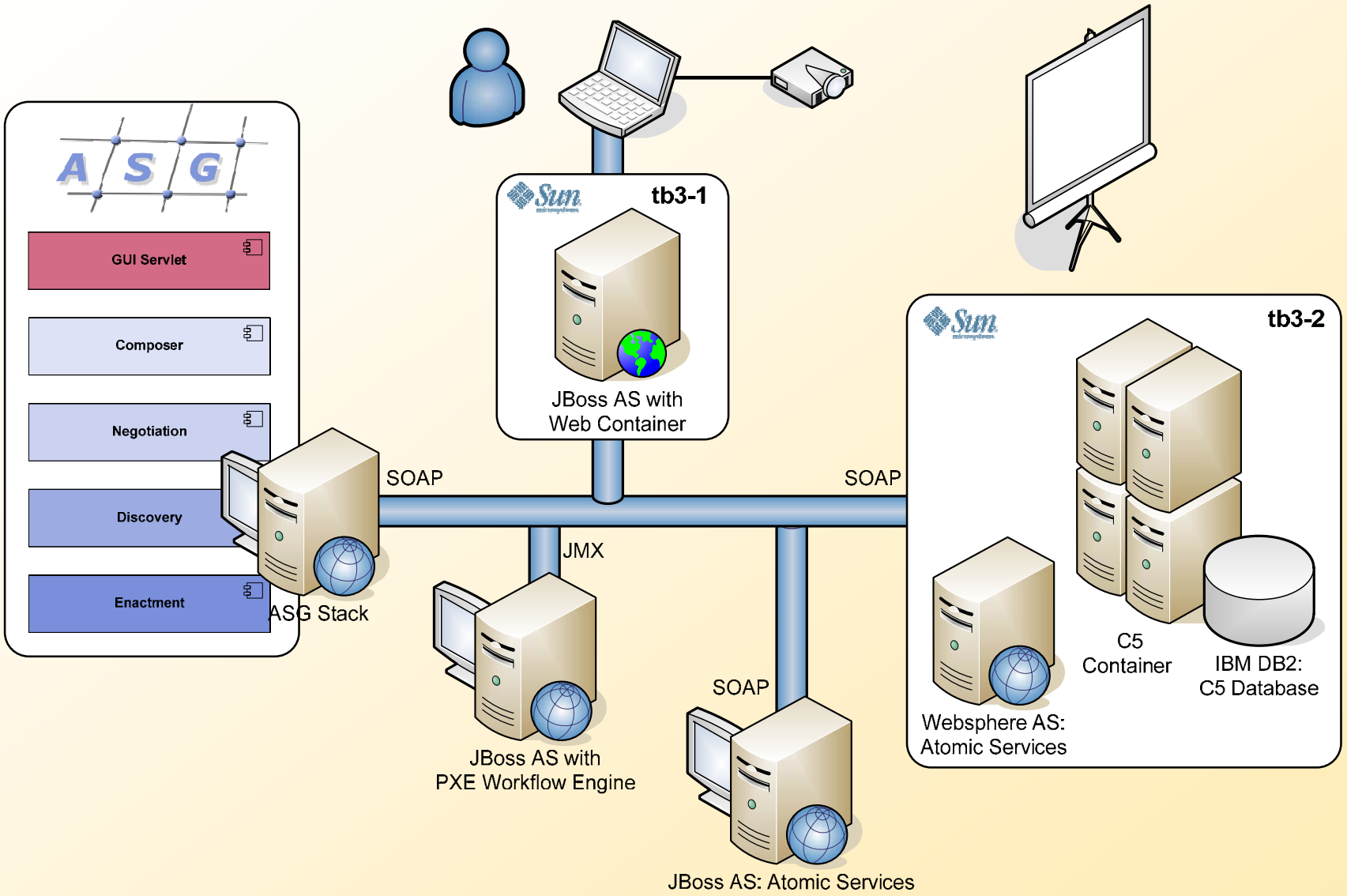
- *Seamless integration of heterogeneous external services*
  - Semantic-annotation of services basing on domain ontology
  - Methodology and tools
  - proof of architecture's flexibility
- *On-demand creation of composed services*
  - Semantic-enabled composition of supply chain processes
- *Reliable service provision with assured end-to-end quality of service*
  - Enactment of composed services
  - Simple negotiation

Restrictions / open issues

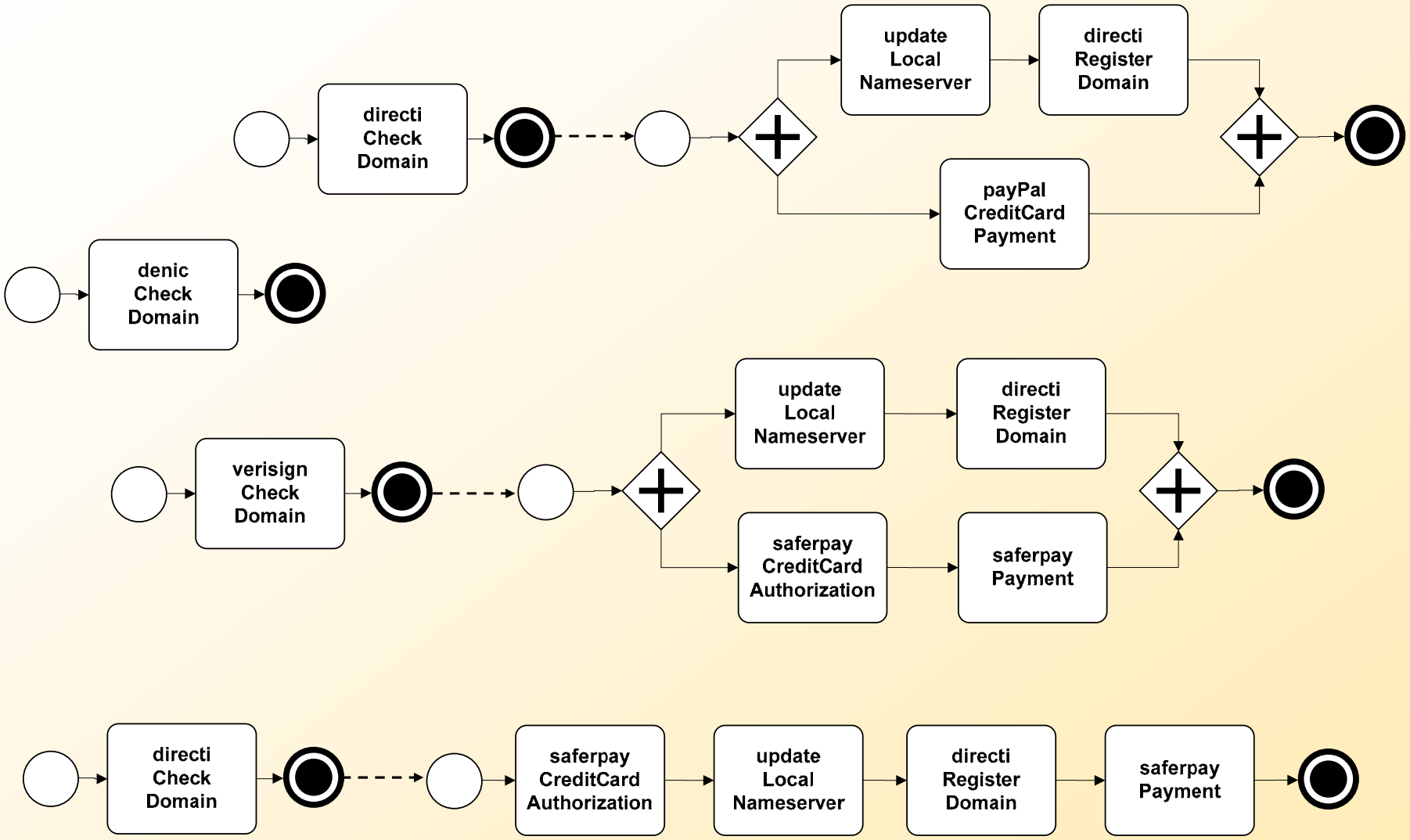
- fault handling, re-planning, re-negotiation
  - retrieve semantic information for XML process data
- intermediate result handling in case of faults
  - PXE monitors on process and activity level (current state, assignments of variables)
- compensation activities
  - credit card refund
  - nameserver entries
  - ...

Demonstration

Demonstration: Infrastructure



Demonstration: Exemplary Compositions



Thanks for your attention.

Any questions?