Feature Based Design of Web Service Transaction Compensations

Peter Dolog with Michael Schäfer
dolog@cs.aau.dk
CS Department
Intelligent Web Information Systems
http://www.cs.aau.dk, http://iwis.cs.aau.dk

SOAPL 2008 @ SPLC 2008, September 2008, Limerick, Ireland

Outline

IWIS group and background
General problem
Business transactions
Middleware for advanced compensations
Service provider and client feature modelling
Matchmaking and restriction model
Further Challenges

Outline

IWIS group and background

General problem

Business transactions

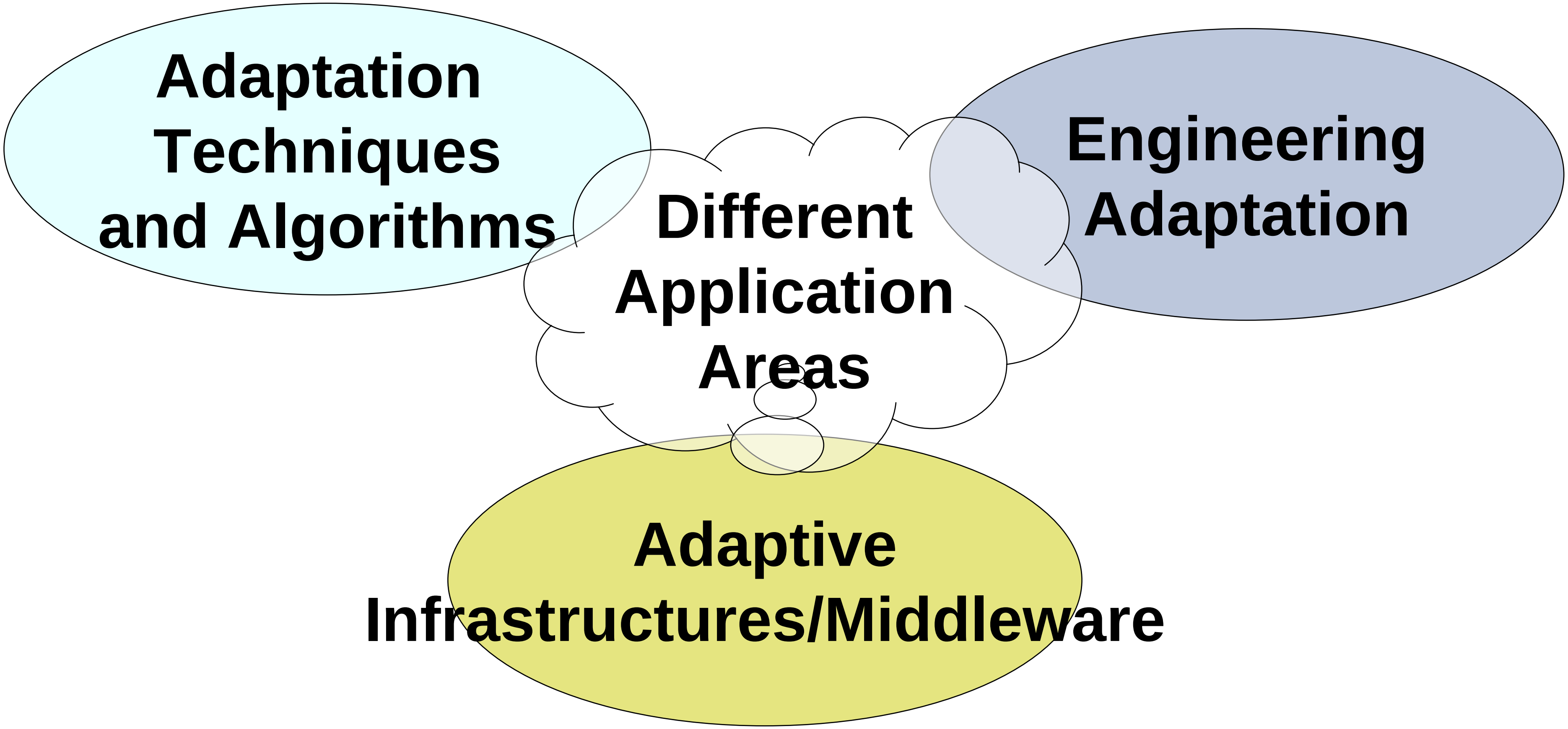
Middleware for advanced compensations

Service provider and client feature modelling

Matchmaking and restriction model

Further Challenges

Intelligent Web and Information Systems
http://iwis.cs.aau.dk



Adaptation Techniques and Algorithms

Different Application Areas

Engineering Adaptation

Adaptive Infrastructures/Middleware

Adaptation/Customization

- Customization by humans (designers)
- Dynamic adaptation by a system itself
- Adaptation is about decision on which information resource or function variant to provide or recommend access to,
- We need a knowledge to decide about appropriate information or service configuration in a certain processing step (user or other):
    - Resource and information access environment
    - Application domain
    - User/Context
    - And their configuration – variants and their meaningful combinations for certain purposes

Outline

- IWIS group and background
- General problem
- Business transactions
- Middleware for advanced compensations
- Service provider and client feature modelling
- Matchmaking and restriction model
- Further Challenges

Open Web Service Environment

Service Providers

- A number of autonomous service providers exist
- They can provide similar functionality
- They can dis-/appear any time
- Each wants to maximize its profit for executing provided services by external consumers

Service Consumers

- Number of consumers with similar requirements exist
- They want to achieve high value for their expense
- To maximize their service
- By composing matched available services from different providers

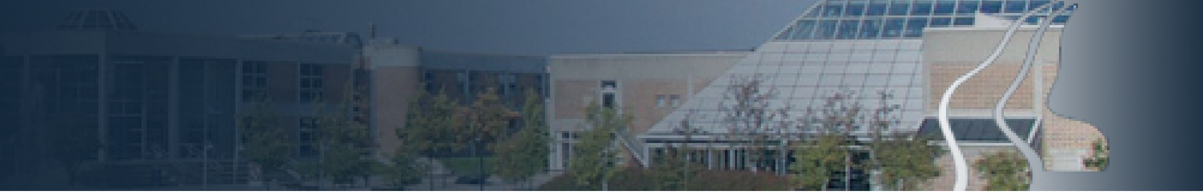
Software Product Lines

Software Providers

- Number of reusable software assets exist
- They may vary in its functionality
- They want to maximize its profit by providing the assets in an application in a family mostly from one company

Software Consumers

- Number of consumers with similar requirements
- They want to achieve high value for their expense
- To maximize their service
- By composing a final application from the reusable assets



Difference

Client is composing in web service world

Client is composing from different providers in web service world

Services used in the composition may be exchanged

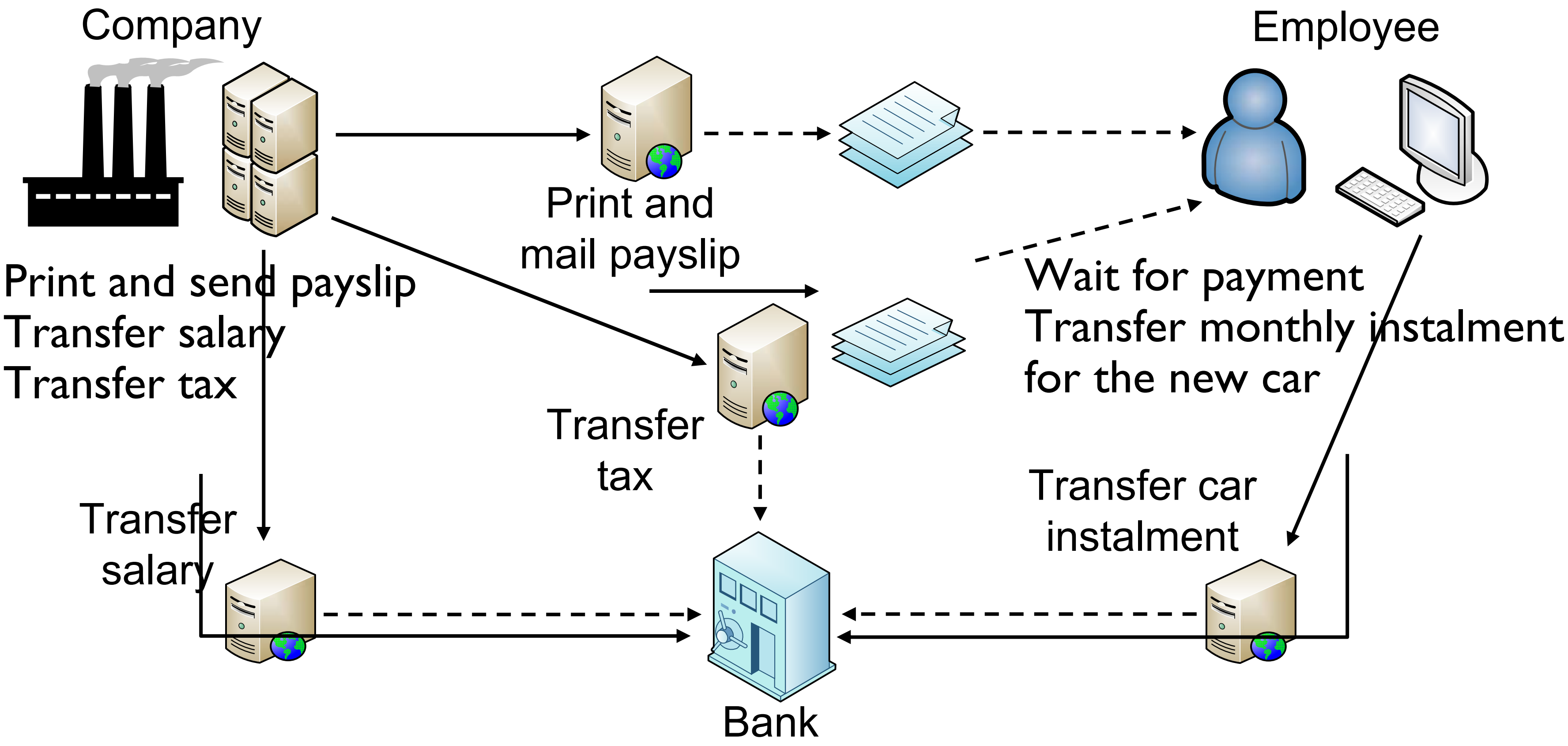
Question:

- What can be achieved by current state of the art software product lines techniques?

Outline

- IWIS group and background
- General problem
- Business transactions
- Middleware for advanced compensations
- Service provider and client feature modelling
- Matchmaking and restriction model
- Further Challenges

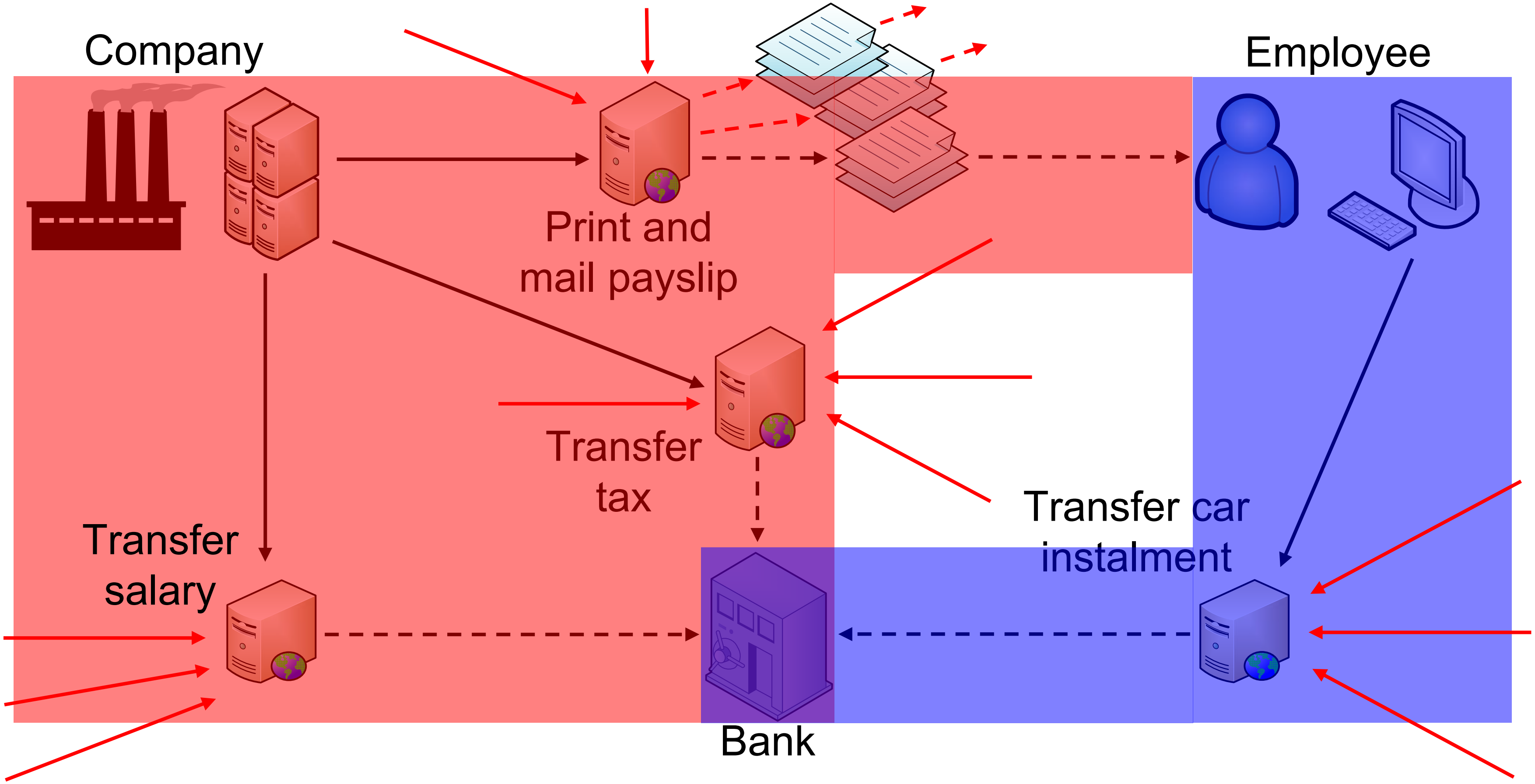
Payroll Scenario



Print and send payslip
Transfer salary
Transfer tax

Wait for payment
Transfer monthly instalment
for the new car

Service Oriented Payroll Scenario



To reach mutually-agreed outcome (commit/cancel)
In environment with concurrent access

Transactions

Control the execution of the required operations on the external services.

Consist of a set of operations (e.g. database operations) that are performed by multiple participants.

Control the collective outcome of the operations.

Distributed transactions control the execution of operations on multiple providers.

- Participant
- Coordinator

Error Compensation

Different transaction specifications exist for different purposes

Backward recovery

Normally, predefined *rollback operations* are executed in order to restore the state before the transaction.

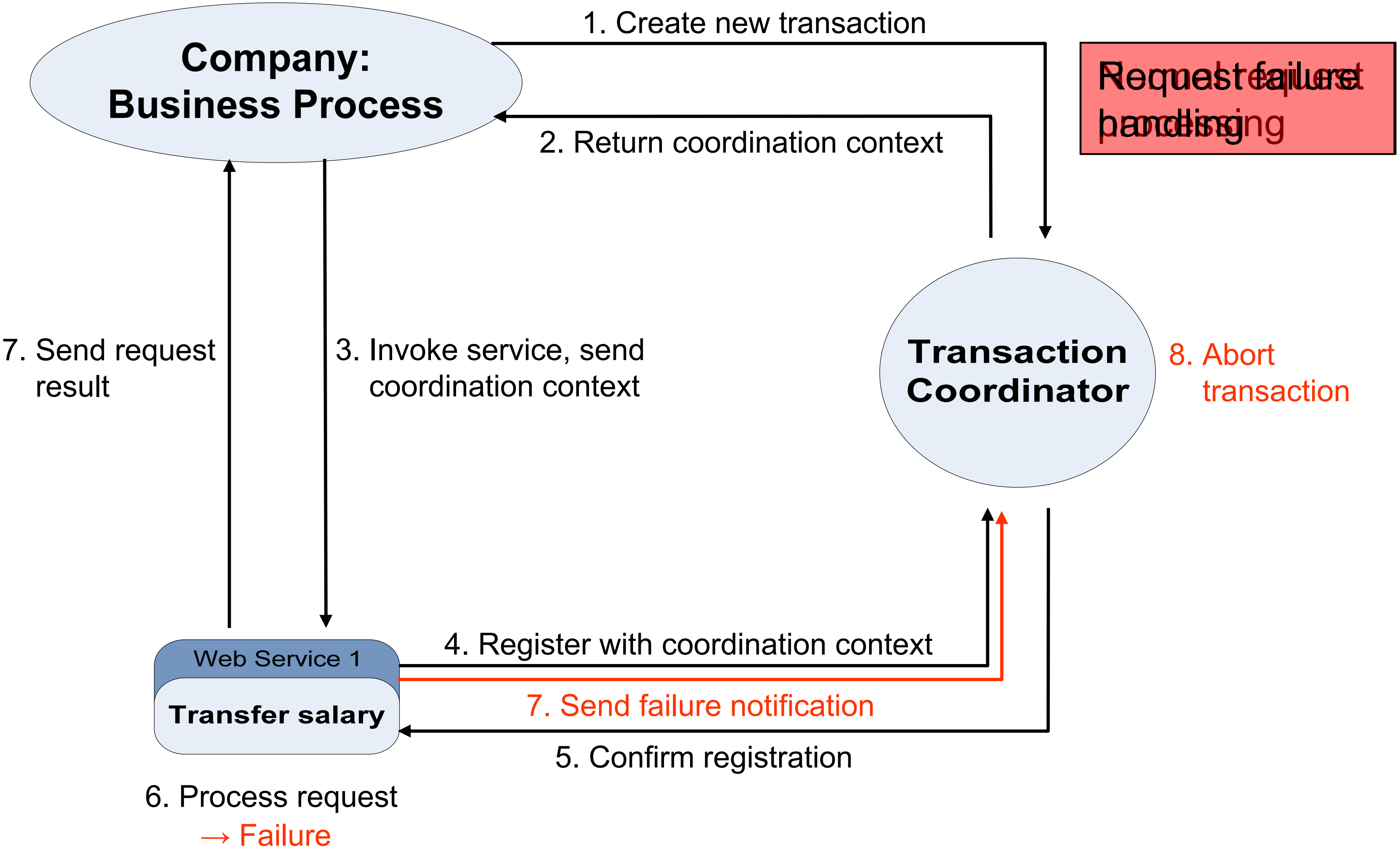
- Time and money is lost
- *Dependent transactions* also have to roll back (*domino effect*)

Forward recovery

Aims at changing pro-actively the state of the participant or transaction to enable a successful execution after a failure.

- Complex
- Can normally only be performed semi-automatically

Traditional WS-Transaction Coordin. Structure



Company:
Business Process

1. Create new transaction

2. Return coordination context

Request failure handling

Normal request processing

Transaction Coordinator

8. Abort transaction

7. Send request result

3. Invoke service, send coordination context

Web Service 1

Transfer salary

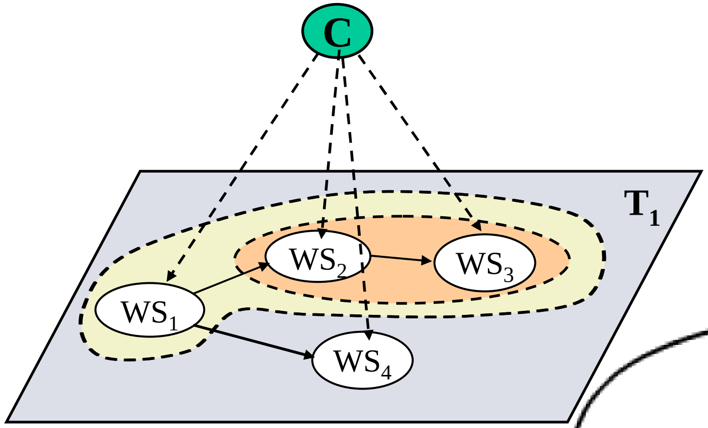
4. Register with coordination context

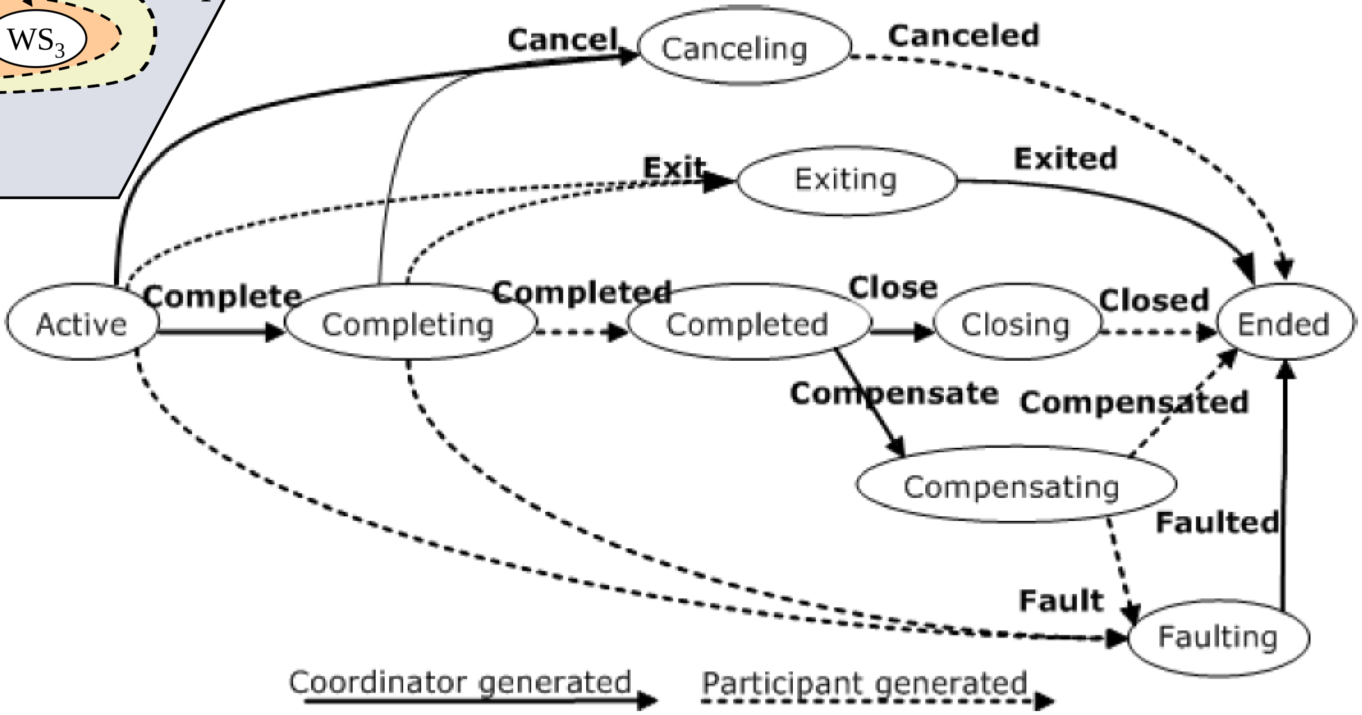
7. Send failure notification

5. Confirm registration

6. Process request
→ Failure

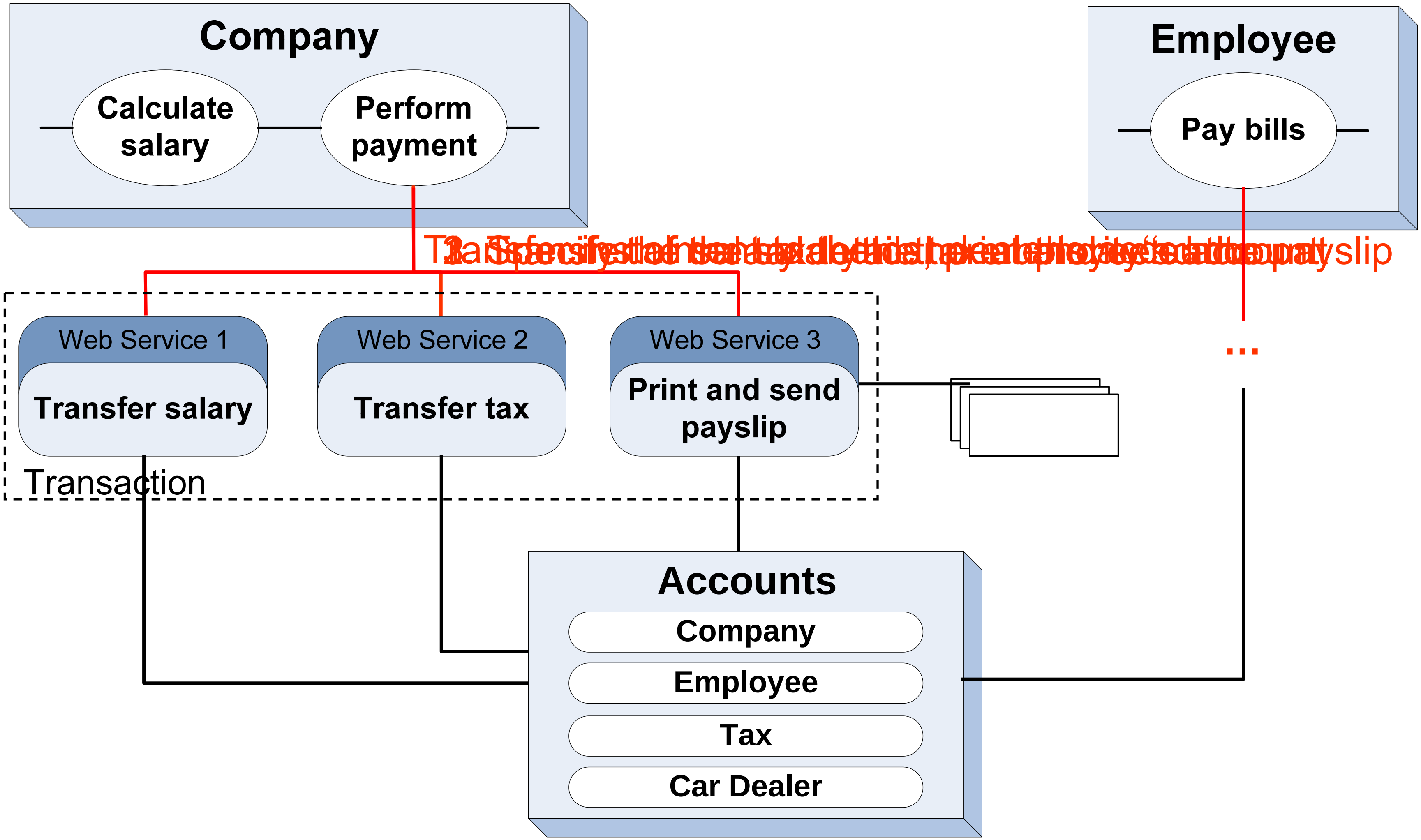
WS – Tx / Business Activity Coordination Type



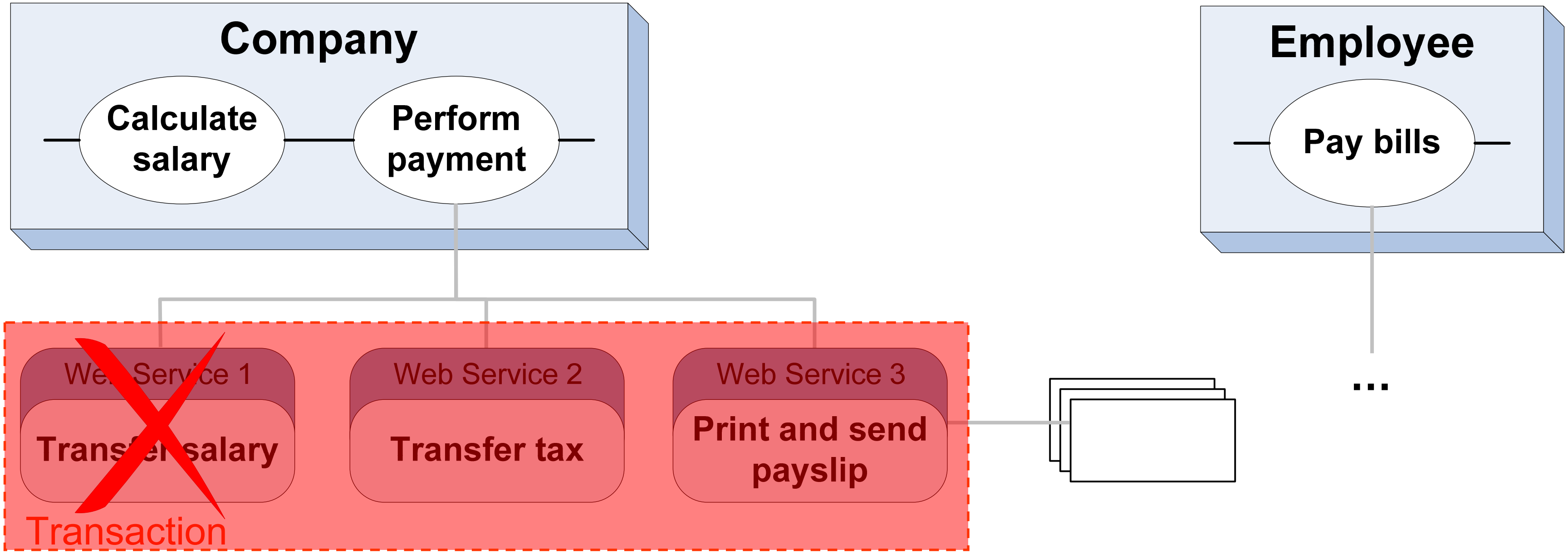


abstract state diagram

Payroll Processing



Motivating Scenario – Problem



A service fails due to an internal error.
The error can only be compensated by aborting the complete transaction.
Why should the transaction be aborted, if a different service exists that can perform the same operations?

Outline

IWIS group and background

General problem

Business transactions

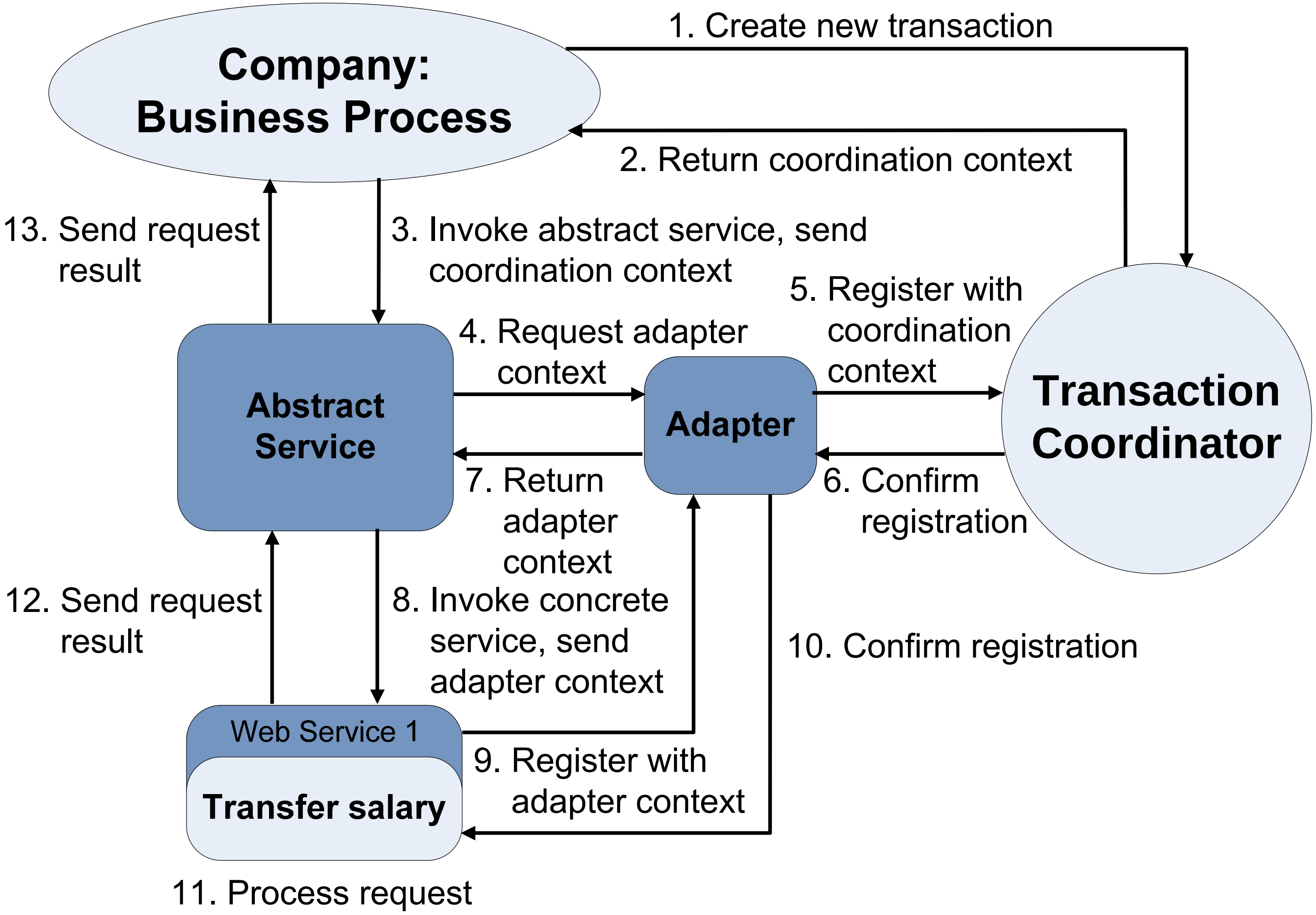
Middleware for advanced compensations

Service provider and client feature modelling

Matchmaking and restriction model

Further Challenges

Extended Transaction Coordination Structure



Company: Business Process

1. Create new transaction

2. Return coordination context

3. Invoke abstract service, send coordination context

4. Request adapter context

5. Register with coordination context

6. Confirm registration

7. Return adapter context

8. Invoke concrete service, send adapter context

9. Register with adapter context

10. Confirm registration

11. Process request

12. Send request result

13. Send request result

Abstract Service

Adapter

Transaction Coordinator

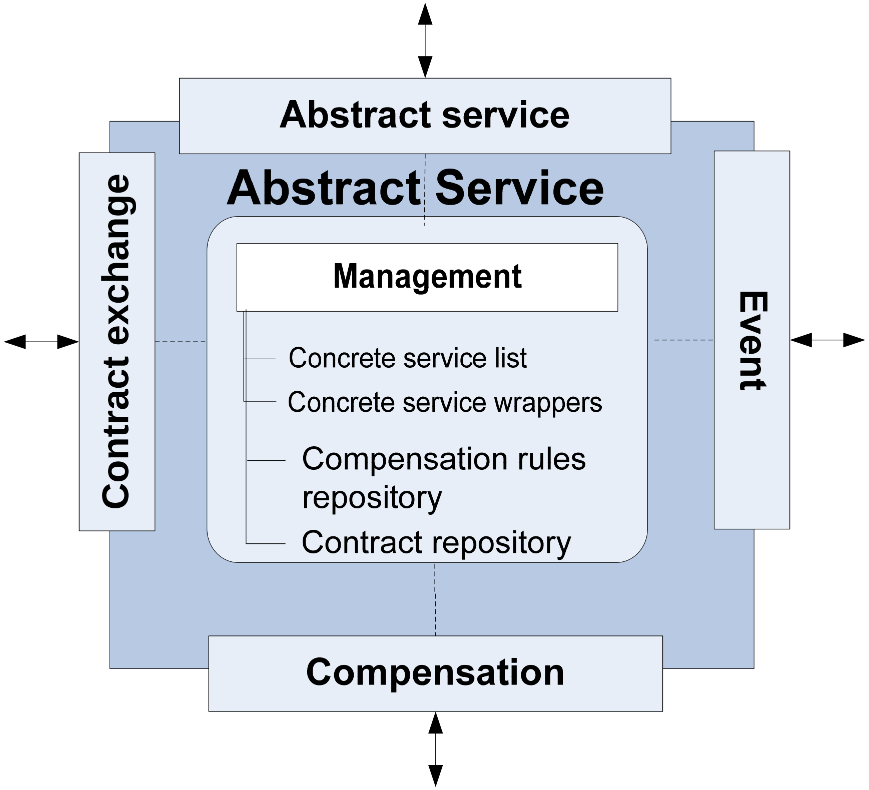
Web Service 1

Transfer salary

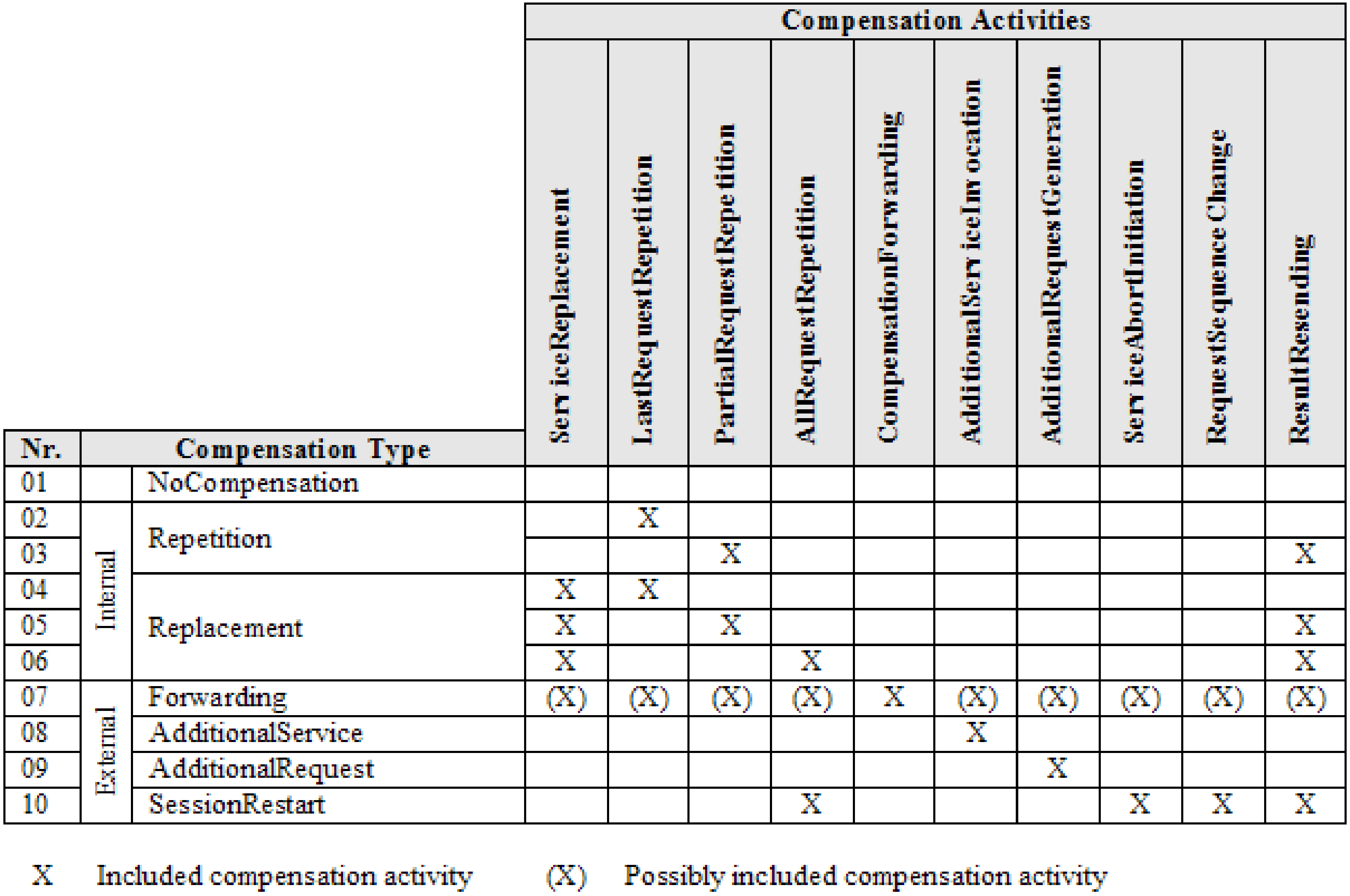
New Components - Abstract Service

Does not directly implement functionalities.
Manages a list of *concrete services*.
Is a mediator between the client and the concrete service.
Manages and performs compensation actions.
Interfaces:

- Service
- Event (internal compensation handling)
- Compensation (external compensation handling)
- Contract exchange



Compensation Activities and Types



			Compensation Activities									
Nr.	Compensation Type		ServiceReplacement	LastRequestRepetition	PartialRequestRepetition	AllRequestRepetition	CompensationForwarding	AdditionalServiceInvocation	AdditionalRequestGeneration	ServiceAbortInitiation	RequestSequenceChange	ResultResending
01		NoCompensation										
02	Internal	Repetition		X								
03					X							X
04		Replacement	X	X								
05			X		X							X
06			X			X						X
07	External	Forwarding	(X)	(X)	(X)	(X)	X	(X)	(X)	(X)	(X)	(X)
08		AdditionalService						X				
09		AdditionalRequest							X			
10		SessionRestart				X				X	X	X

X Included compensation activity (X) Possibly included compensation activity

Example: Internal Compensation Rule

```
<cmp:InternalCompensationRule identifier="internalFailureLastRequestResending">
  <cmp:CompensationCondition>
    <cmp:ParticipantEvent eventCode=
   "http://sourceforge.net/projects/frogs/AdapterInteraction/ParticipantFault"/>
    <cmp:ParticipantState
      stateType='http://schemas.xmlsoap.org/ws/2004/10/wsba/Faulting' />
    <cmp:ReplacementService exists="true" isDirectReplacement="true" />
    <cmp:RequestSequence>
      <cmp:Request identifier="transferSalaryMethod" />
    </cmp:RequestSequence>
  </cmp:CompensationCondition>
  <cmp:CompensationPlan>
    <cmp:Compensation>
      <cmp:ServiceReplacement/>
    </cmp:Compensation>
    <cmp:Compensation>
      <cmp:RequestResending lastN="1" />
    </cmp:Compensation>
  </cmp:CompensationPlan>
</cmp:InternalCompensationRule>
```

[illegible]

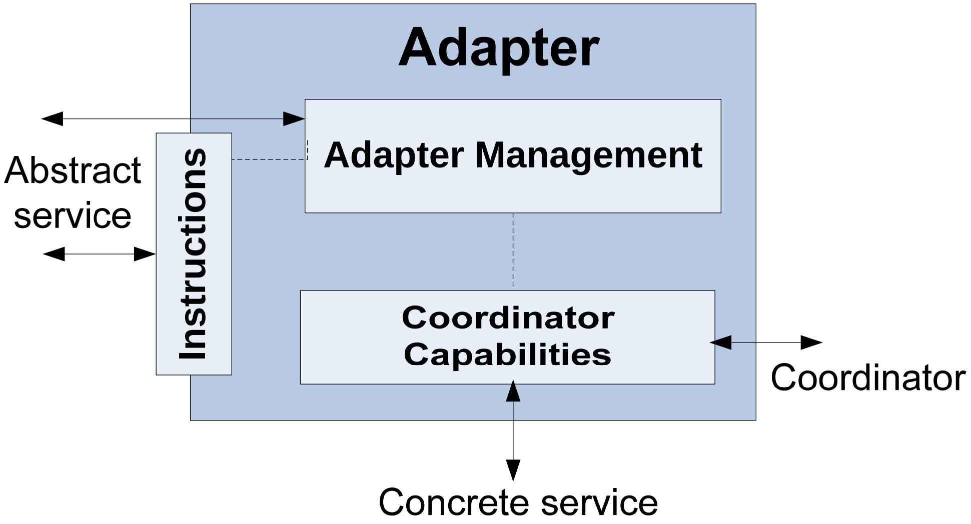
New Components - Adapter

Encapsulates coordinator-specific functionality.

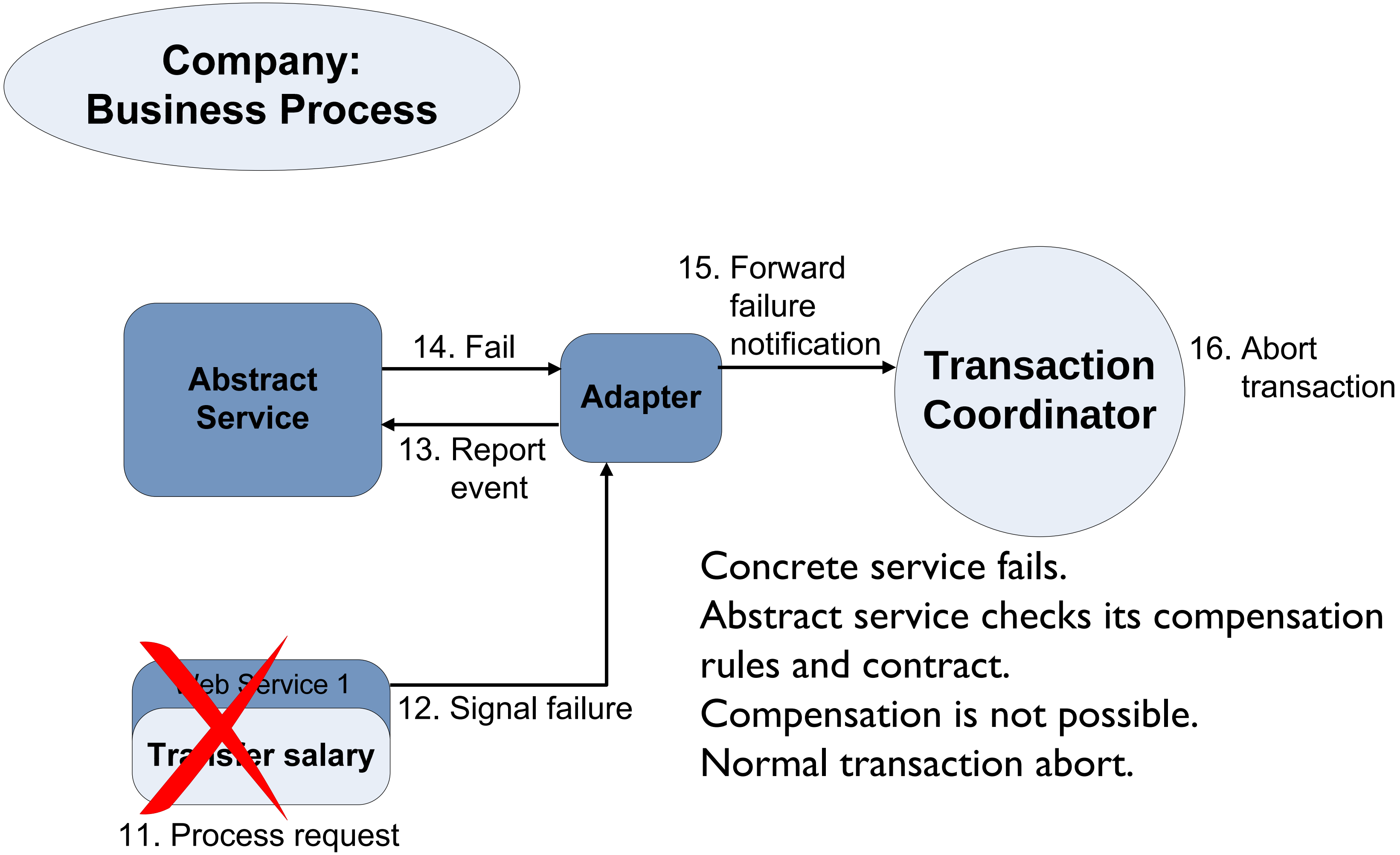
Functions as a coordinator for the concrete service.

Manages messaging:

- Forwards normal messages between the real coordinator and the concrete service.
- Intercepts failure messages and informs the abstract service.
- Creates additional notifications as part of a compensation process.

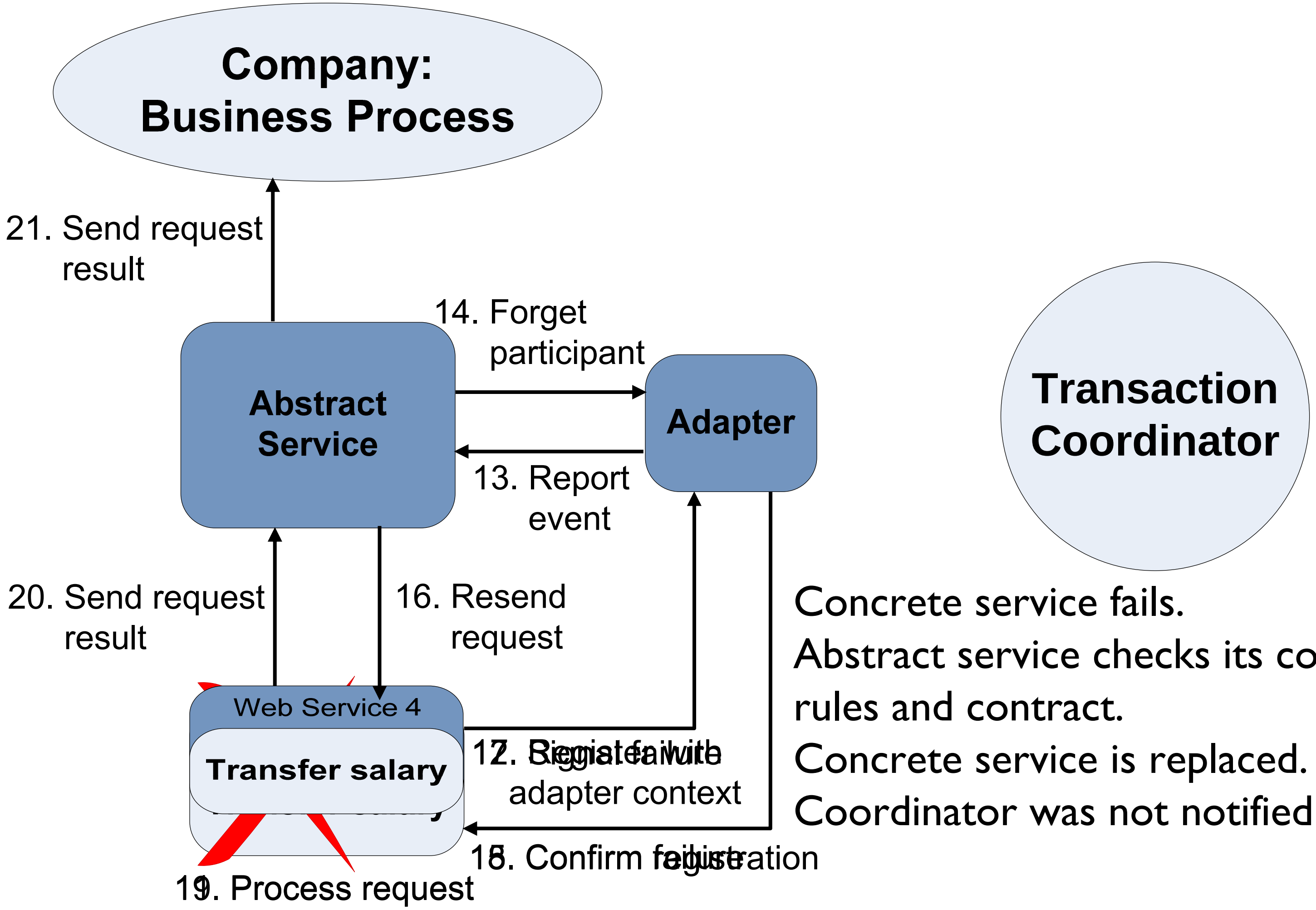


Internal Compensation Handling – No Action



Concrete service fails.
Abstract service checks its compensation rules and contract.
Compensation is not possible.
Normal transaction abort.

Internal Compensation Handling – Replacement



Concrete service fails.

Abstract service checks its compensation rules and contract.

Concrete service is replaced.

Coordinator was not notified!

Evaluation

Multiple scenarios for internal and external compensation handling have been implemented and tested.

An evaluation model has been created, which calculates *net values* for the standard environment and the abstract service environment.

- ➢ Allows an assessment whether the utilization of the new design is economical and beneficial.

Experiment performed on a simalated environment

More in ACM TWEB paper

Outline

IWIS group and background

General problem

Business transactions

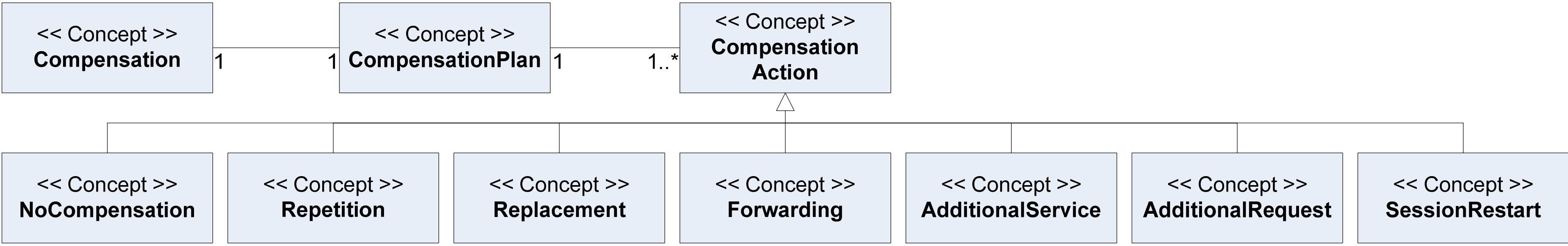
Middleware for advanced compensations

Service provider and client feature modelling

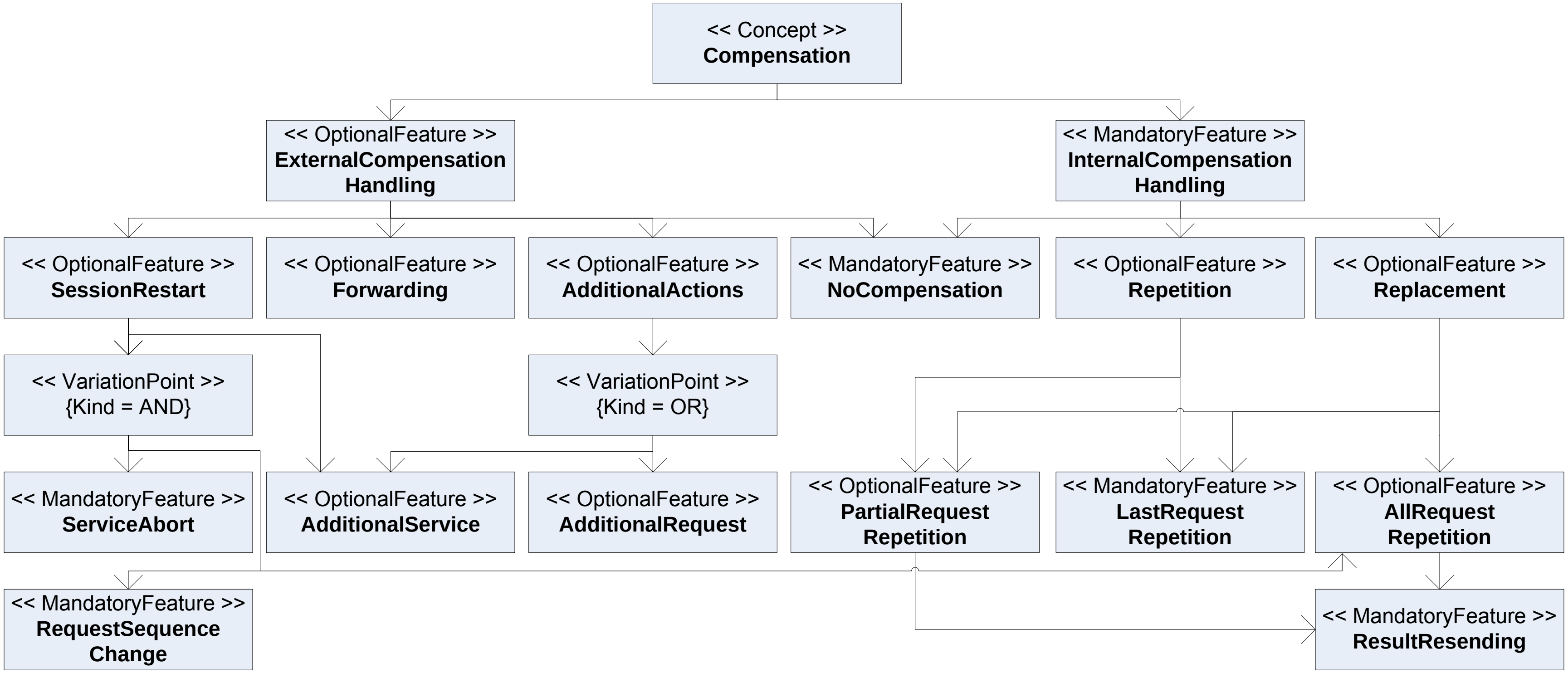
Matchmaking and restriction model

Further Challenges

Compensation Types

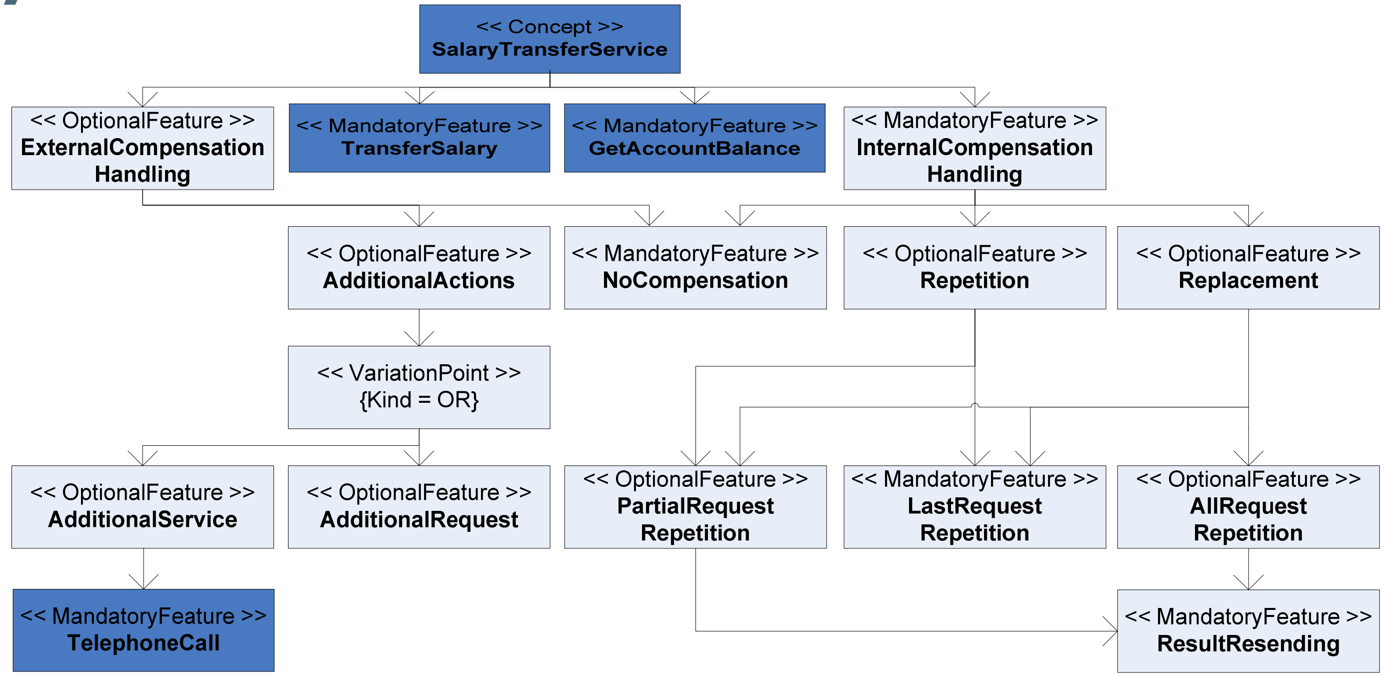


Compensation Features



<< Concept >> Compensation
<< OptionalFeature >> ExternalCompensation Handling
<< MandatoryFeature >> InternalCompensation Handling
<< OptionalFeature >> SessionRestart
<< OptionalFeature >> Forwarding
<< OptionalFeature >> AdditionalActions
<< MandatoryFeature >> NoCompensation
<< OptionalFeature >> Repetition
<< OptionalFeature >> Replacement
<< VariationPoint >> {Kind = AND}
<< VariationPoint >> {Kind = OR}
<< MandatoryFeature >> ServiceAbort
<< OptionalFeature >> AdditionalService
<< OptionalFeature >> AdditionalRequest
<< OptionalFeature >> PartialRequest Repetition
<< MandatoryFeature >> LastRequest Repetition
<< OptionalFeature >> AllRequest Repetition
<< MandatoryFeature >> RequestSequence Change
<< MandatoryFeature >> ResultResending

Capability Feature Model

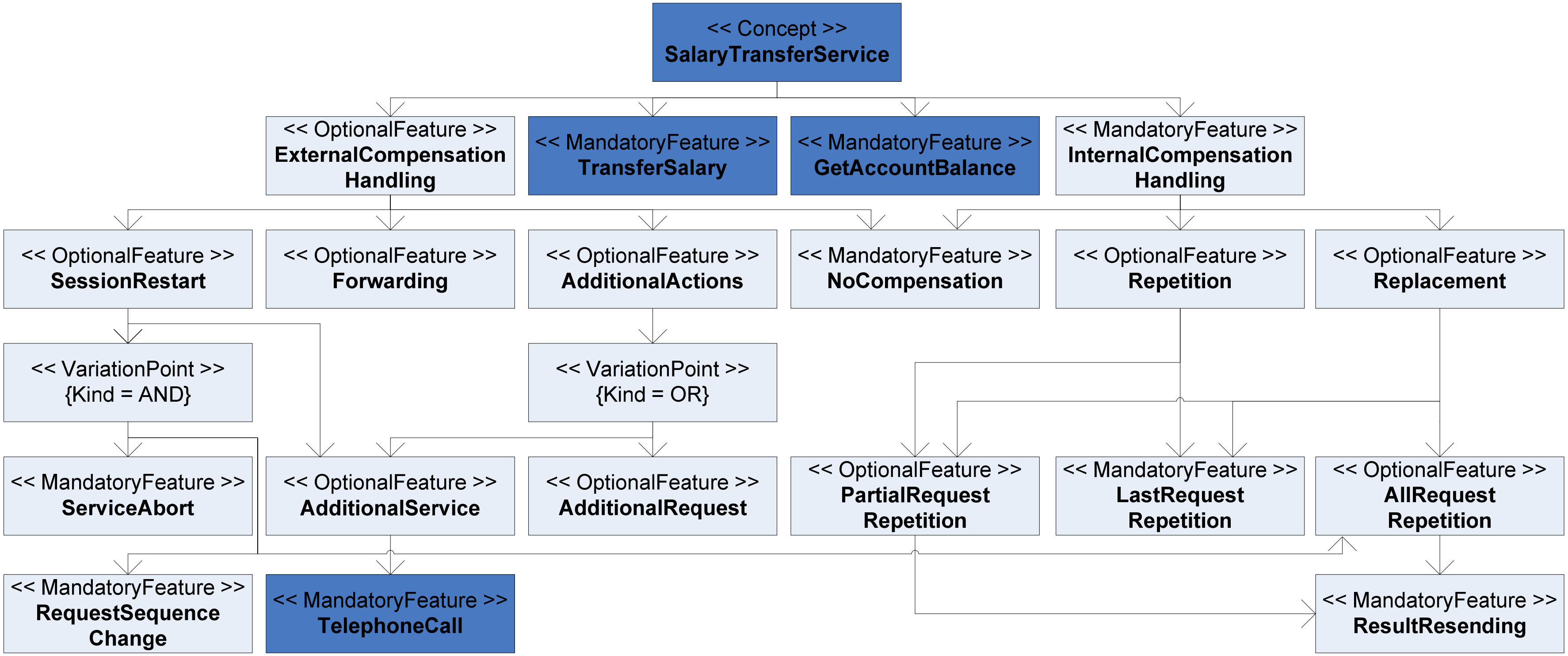


Consists of:

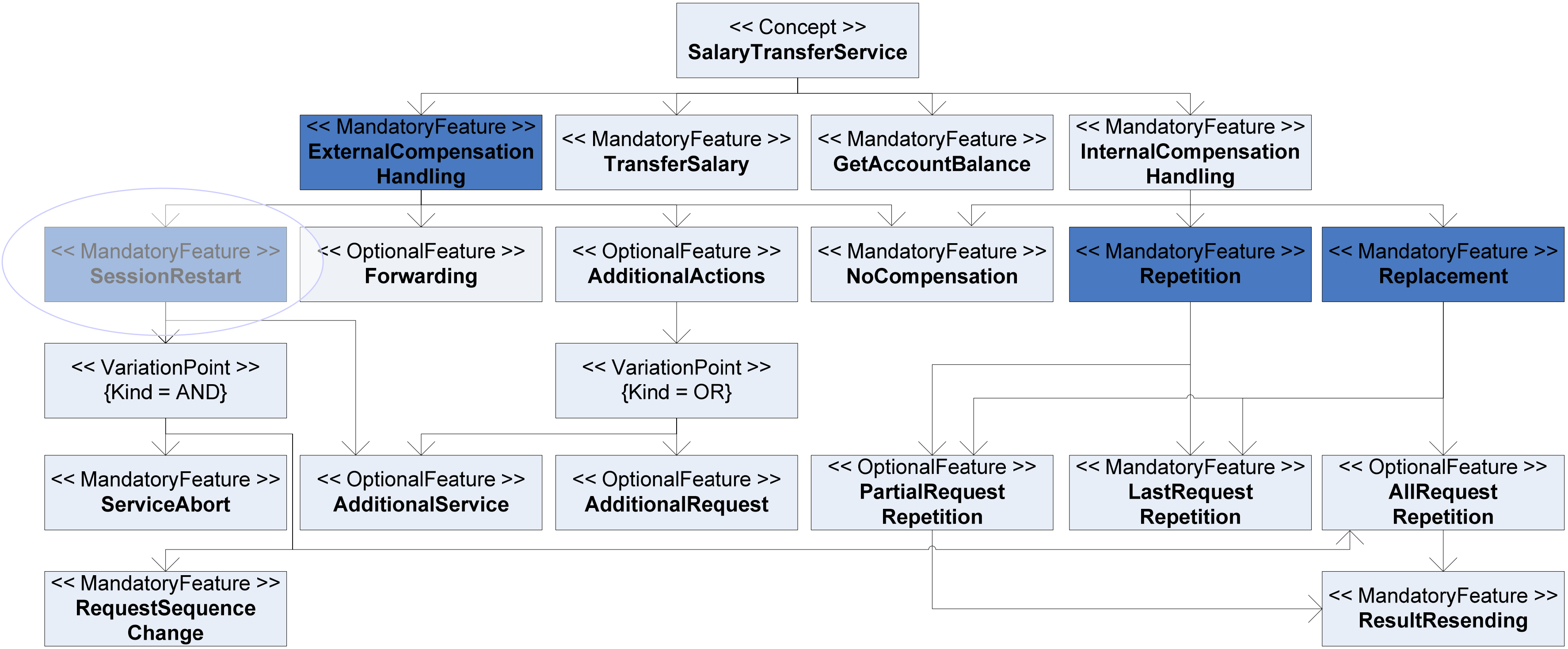
- *functionality feature model*
- *compensation feature model*

The compensation feature model can contain custom features.

Service Capabilities



Consumer Requirements



<< Concept >> SalaryTransferService
<< MandatoryFeature >> ExternalCompensation Handling
<< MandatoryFeature >> TransferSalary
<< MandatoryFeature >> GetAccountBalance
<< MandatoryFeature >> InternalCompensation Handling
<< MandatoryFeature >> SessionRestart
<< OptionalFeature >> Forwarding
<< OptionalFeature >> AdditionalActions
<< MandatoryFeature >> NoCompensation
<< MandatoryFeature >> Repetition
<< MandatoryFeature >> Replacement
<< VariationPoint >> {Kind = AND}
<< VariationPoint >> {Kind = OR}
<< MandatoryFeature >> ServiceAbort
<< OptionalFeature >> AdditionalService
<< OptionalFeature >> AdditionalRequest
<< OptionalFeature >> PartialRequest Repetition
<< MandatoryFeature >> LastRequest Repetition
<< OptionalFeature >> AllRequest Repetition
<< MandatoryFeature >> RequestSequence Change
<< MandatoryFeature >> ResultResending

Outline

IWIS group and background

General problem

Business transactions

Middleware for advanced compensations

Service provider and client feature modelling

Matchmaking and restriction model

Further Challenges

Matchmaking between service and consumer feature models

Compatibility score calculation

Iteratively compares feature models

Features must appear at the same place in the graph

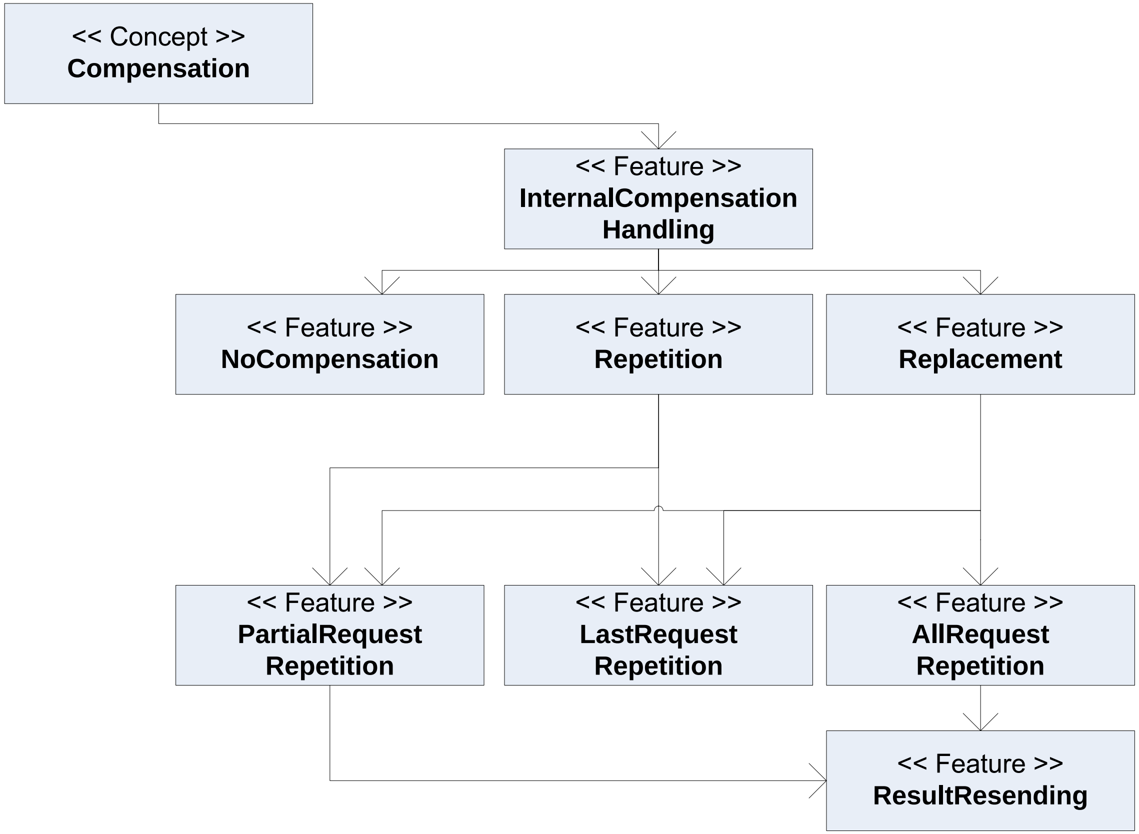
Mandatory features must all match but do not contribute to the compatibility score

If a mismatch is found in a mandatory feature, algorithm stops and a negative score is returned

Optional features add to the compatibility score when a match is found (in our case +1)

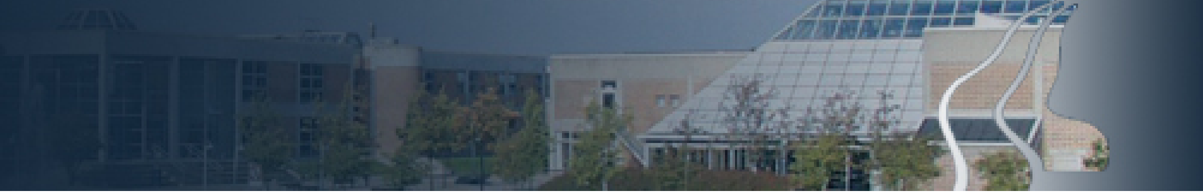
Additional features may contribute with different scores

Restriction Feature Model



Example: Internal Compensation Rule

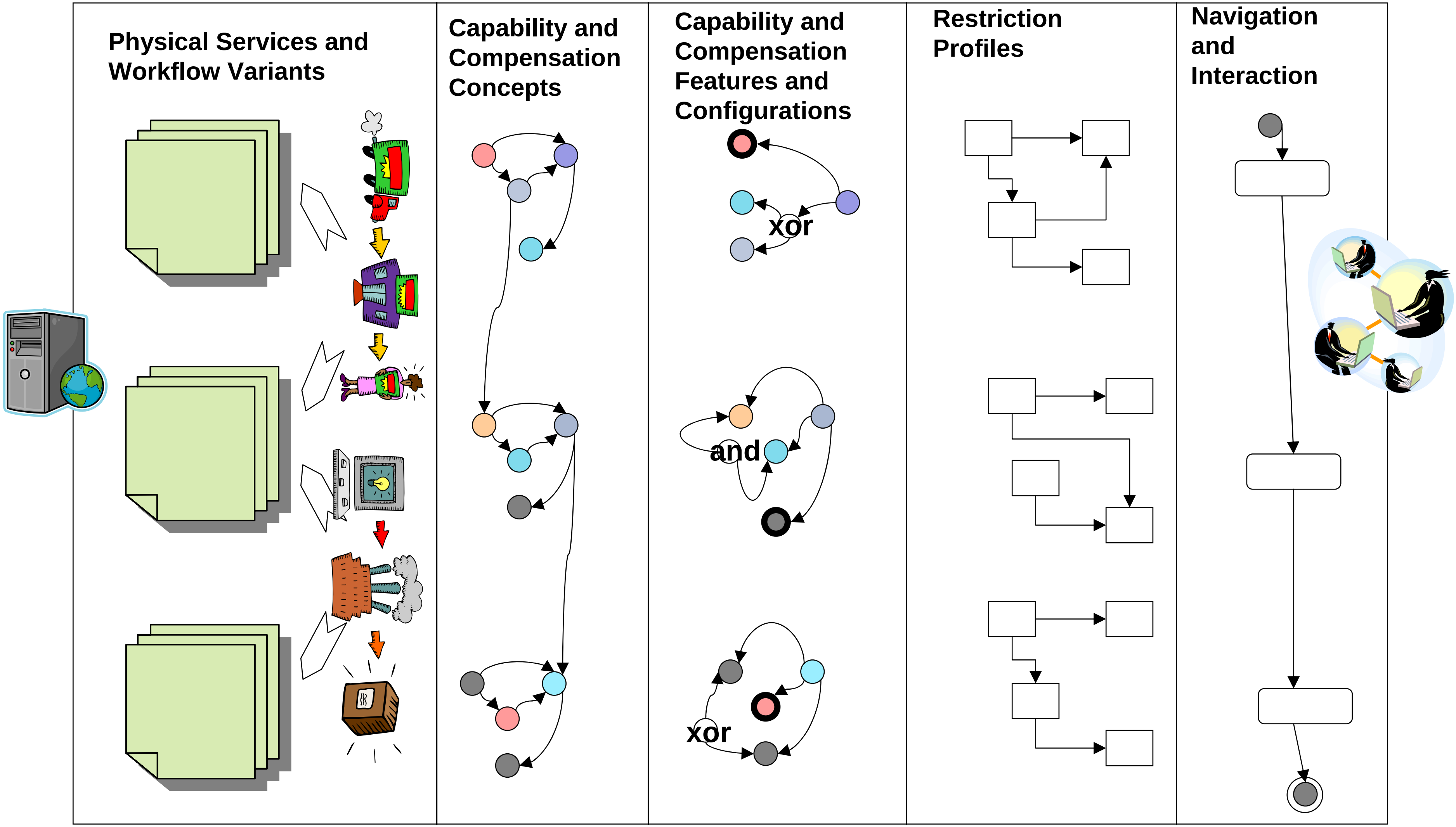
```
<cmp:InternalCompensationRule identifier="internalFailureLastRequestResending">
  <cmp:CompensationCondition>
    <cmp:ParticipantEvent eventCode=
   "http://sourceforge.net/projects/frogs/AdapterInteraction/ParticipantFault"/>
    <cmp:ParticipantState
      stateType='http://schemas.xmlsoap.org/ws/2004/10/wsba/Faulting' />
    <cmp:ReplacementService exists="true" isDirectReplacement="true" />
    <cmp:RequestSequence>
      <cmp:Request identifier="transferSalaryMethod" />
    </cmp:RequestSequence>
  </cmp:CompensationCondition>
  <cmp:CompensationPlan>
    <cmp:Compensation>
      <cmp:ServiceReplacement/>
    </cmp:Compensation>
    <cmp:Compensation>
      <cmp:RequestResending lastN="1" />
    </cmp:Compensation>
  </cmp:CompensationPlan>
</cmp:InternalCompensationRule>
```



Feature Model

```
<feature  name="Compensation" type="NONE" id="compensation">
  <feature  name="InternalCompensationHandling" type="NONE”
    id="internalCompensationHandling">
    …
   <feature  name="PartialRequestRepetition" type="NONE"
    id="reference3IXIpartialRequestRepetition">
        <feature  name="ResultResending" type="NONE"
    id="reference3IXIreferenceIXIresultResending">
        </feature>
      </feature>
    </feature>
    <feature  name="Replacement" type="NONE" id="replacement">
      <feature  name="LastRequestRepetition" type="NONE"
    id="reference4IXIlastRequestRepetition">
      </feature>
      <feature  name="PartialRequestRepetition" type="NONE"
    id="reference5IXIpartialRequestRepetition">
        <feature  name="ResultResending" type="NONE"
    id="reference5IXIreferenceIXIresultResending">
        </feature>
      </feature>
      <feature  name="AllRequestRepetition" type="NONE"
    id="reference6IXIallRequestRepetition">
        <feature  name="ResultResending" type="NONE"
    id="reference6IXIreferenceIXIresultResending">
        </feature>
      </feature>
    </feature>
  </feature>
    …
</feature>
```

Layers of Abstraction



Physical Services and Workflow Variants

Capability and Compensation Concepts

Capability and Compensation Features and Configurations

Restriction Profiles

Navigation and Interaction

Outline

IWIS group and background

General problem

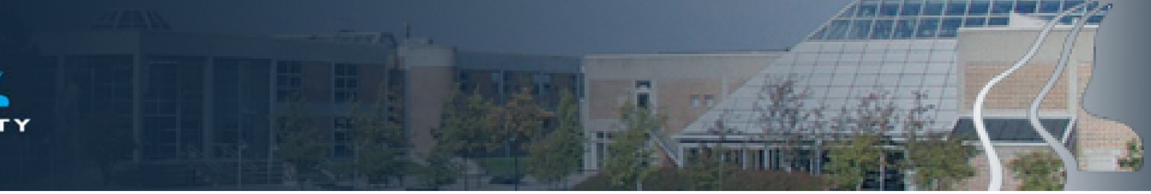
Business transactions

Middleware for advanced compensations

Service provider and client feature modelling

Matchmaking and restriction model

Further Challenges



Workflows vs. Middleware

Compensations and adaptations can be specified at the design level in workflows

Copensations and adaptations can be encoded in an intelligent middleware

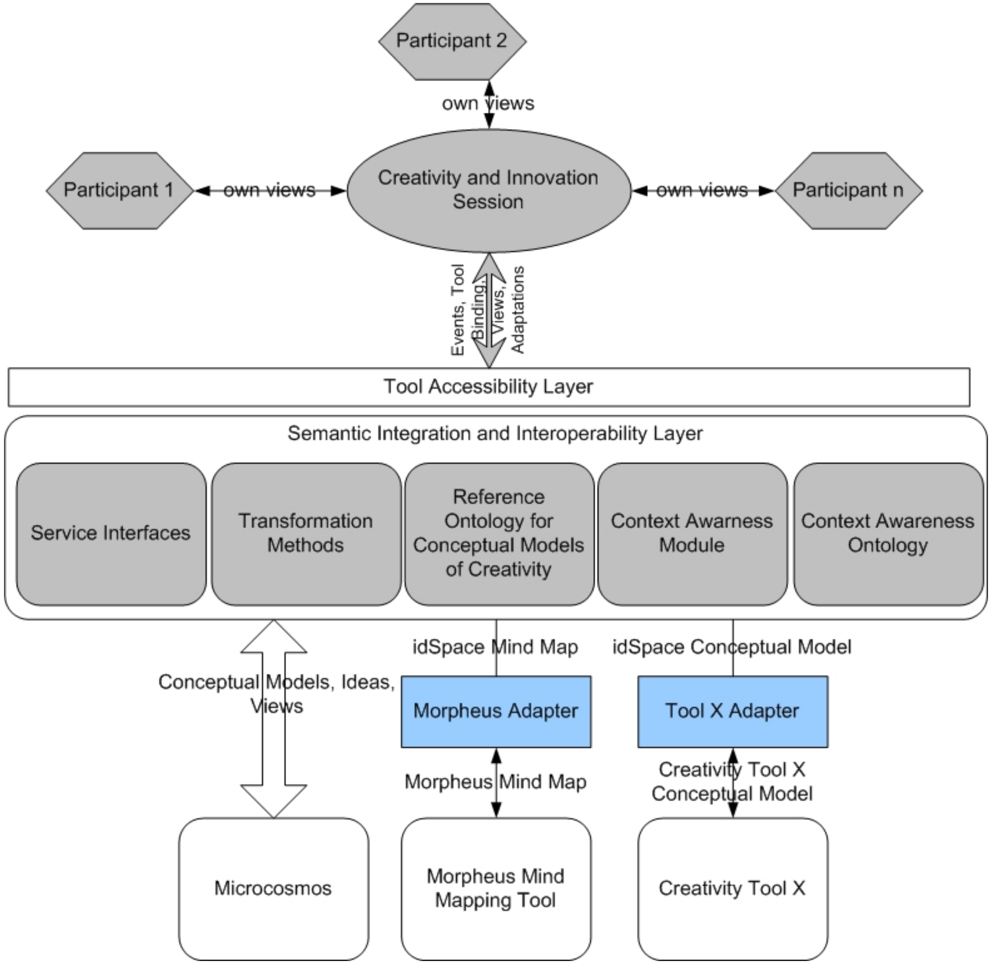
How to combine them

How to compose them

How to ensure consistency

…

FP7 ICT EU idSpace: Tooling of and training for collaborative, distributed product



References

- M. Schäfer, P. Dolog, W. Nejdl: An Environment for Flexible Advanced Compensations of Web Service Transactions. ACM TWEB, 2(2), 2008
- P. Dolog, W. Nejdl: Using UML-based feature models and UML collaboration diagrams to information modelling for web-based applications. UML 2004.

Thanks!!! Questions?

dolog@cs.aau.dk
http://www.cs.aau.dk/~dolog
http//iwis.cs.aau.dk