Classification of SOA Contract Specification Languages

Joseph C. Okika, Anders P. Ravn
{ojc, apr}@cs.aau.dk

Department of Computer Science
Aalborg University, Denmark

FLACOS 2008 – Malta, November 27-28, 2008

AALBORG UNIVERSITET

Why SOA Contract?

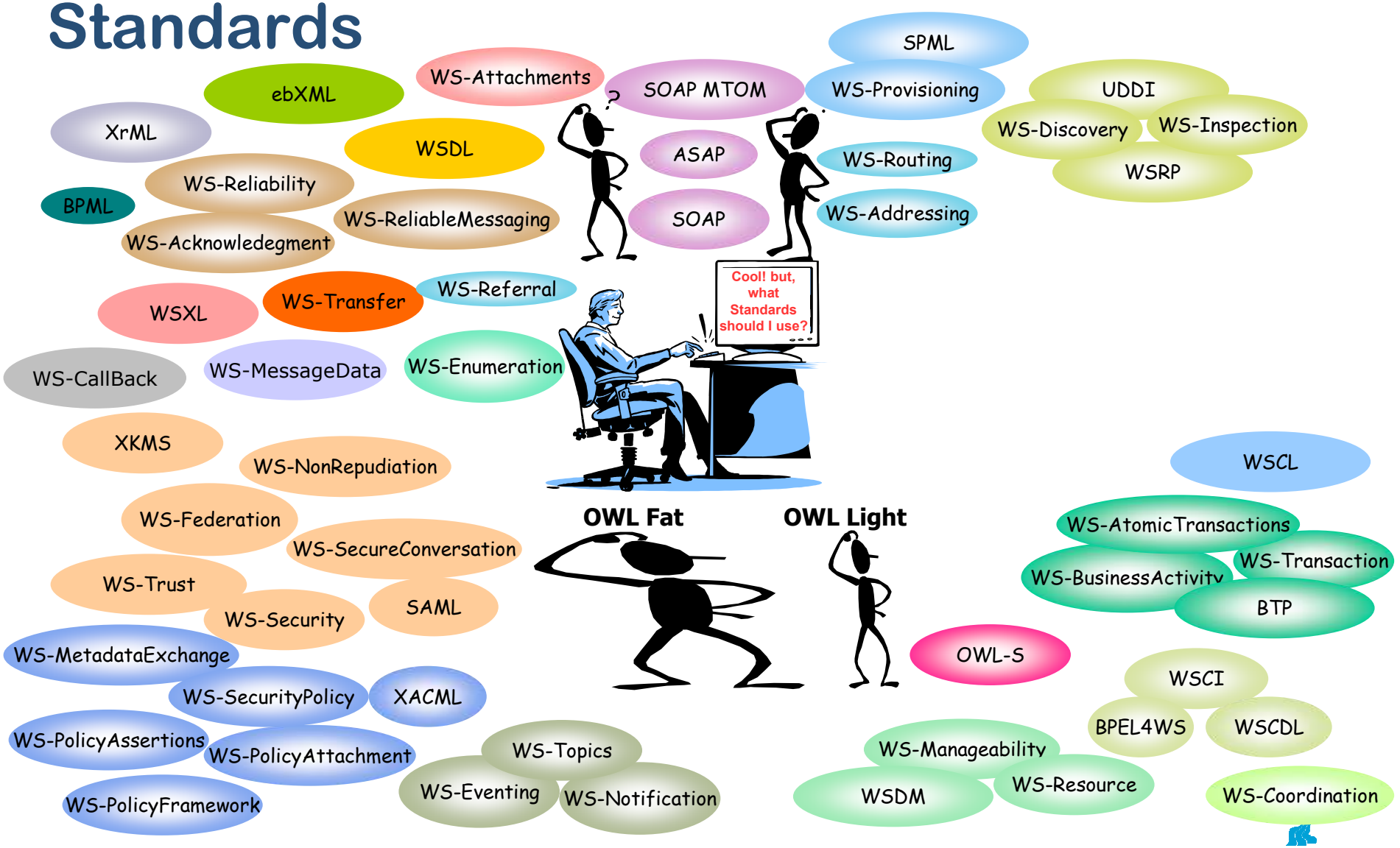
We should integrate our systems using **Web Service** Technologies …

Manager

services

AALBORG UNIVERSITET

Standards



Ref: *Benatallah* ``Interoperability Specifications'', *May 06, IEEE Computer, by. Motahari, Benatallah, Casati and Toumani*

Surveys

- [Lippe et al 2005]
  - Cross-Organisational Business processes
  - concepts
  - coverage of development process
- [Beek et al 2006]
  - semantics of languages
  - mappings between languages
  - applicability

Classification – Compatible notations

BPEL4WS

WSCDL

WS-Coordination

WS-SLA

WS-Addressing

WS-Referral

WS-Routing

WS-Trust

WS-Security

WS-PolicyFramework

WS-Reliability

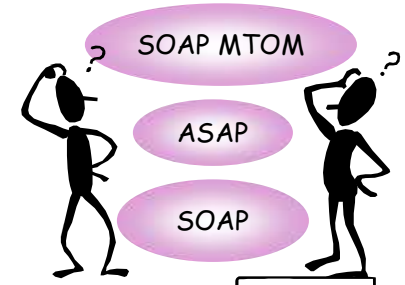
WS-Transaction

WS-Notification

WS-MessageData

WS-Eventing

WSDL



ebXML

ebCPA

ebCPP

ebBPSS

OWL Fat

OWL Light

OWL-S

WMSO

What are SOA Contracts?

Defining contract:

- A contract is about certain aspects of a service:
  - interfaces.
  - behaviour
  - functionality
  - quality
- A contract language expresses properties of some aspects

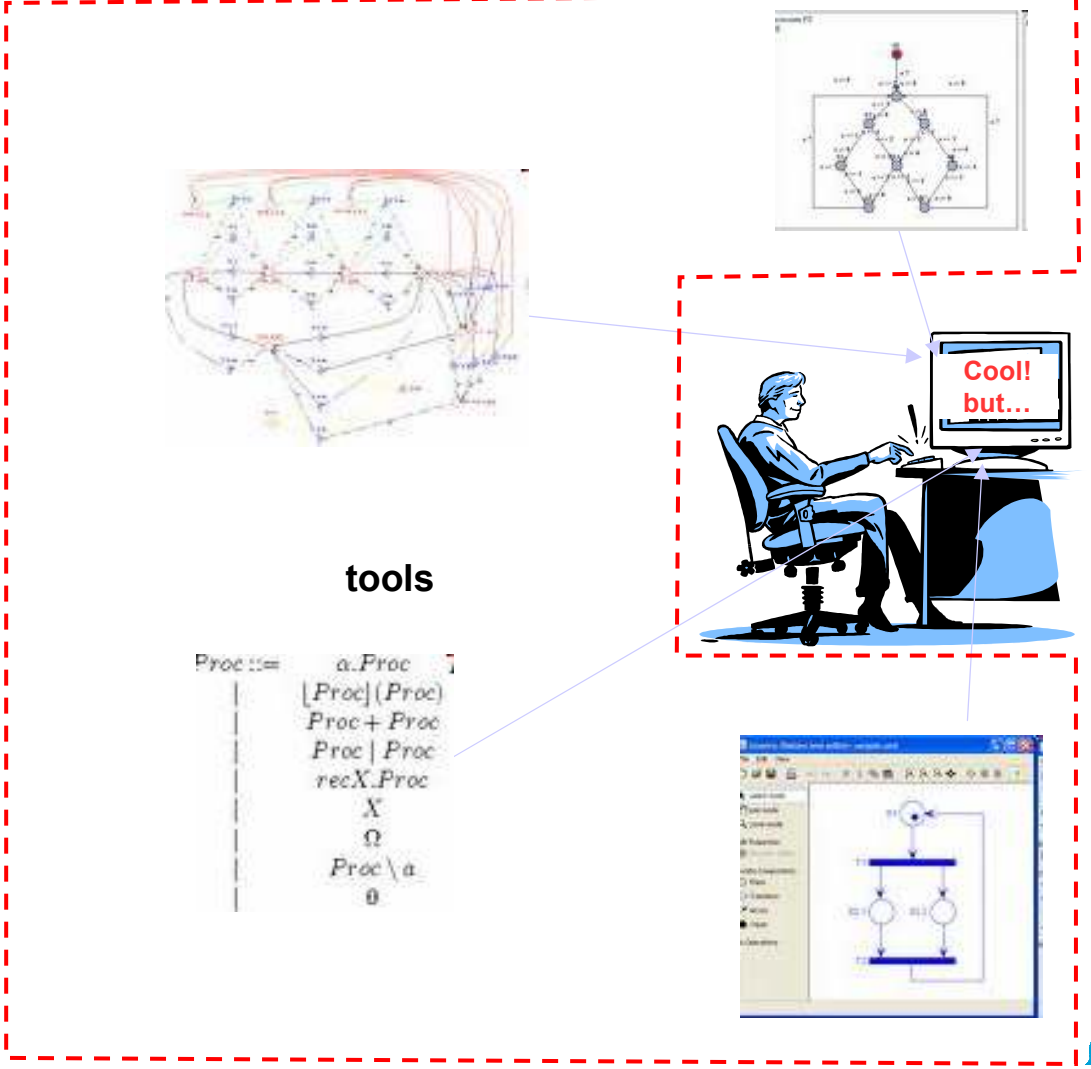
- Properties define proper service collaborations and /or interactions

Contract Aspects

- Interface:
  - defines syntax of communication
- Functionality:
  - operations and their properties
- Protocol:
  - behaviour: sequencing of events, signals, messages, etc.
- Security:
  - techniques and practices that ensure confidentiality
- Extra functional properties:
  - constraints on a concrete service property: performance, availability, reliability, accessibility, etc.

Tools for Analysis I

- based on:
  - Automata
  - Petri nets
  - Abstract State Machines
  - Process Algebra
  - etc.



Tools for Analysis II

- Interface:
  - compilers
- Functionality:
  - test tools
- Protocol:
  - monitoring
  - model checking
- Security:
  - monitoring, model checking
- Extra functional properties:
  - simulation, monitoring

	Semantic models		
Characteristics	Automata	Petri nets	Process algebras
Connectivity	[16, 17]	[20, 21, 37, 39, 40, 41]	[23, 44]
Exception handling/ Compensations	[18, 30]	[19, 21, 36, 41]	[23]
Correctness	[16, 17, 18, 30, 33]	[20, 21, 22, 37, 39]	[23, 44]
QoS	[16, 18, 33]	[19, 37, 42]	[23, 44]

- [Beek et al 2006]

Classification- Aspect and Family

Aspects		Contracts Languages/Approaches		
		Web Services (WS-*)	Semantic Web (*-S)	Electronic Business (eb-*)
Interface		WSDL	OWL-S	ebBSI
Functionality		WS-BPEL, WSOL	OWL-S (IOPE), WSMO	ebBPSS
Protocol		WS-BPEL, WS-CDL	WSMO	ebBPSS
Security		WS-Security		ebCPA(SecurityPolicy)
Quality	policy	WS-Policy		
	trust	WS-Trust		ebCPP(XMLDSIG)
	availability	WSOL		
	performance	WSLA, WSOL	WSMO/WSML	ebCPA

Using the Classification

Aspects	WS - *	Tool Options
Interface	WSDL	Compilers, XML-parsers
Functionality	WS-BPEL	Model checking ? Proof techniques (cf. JML)
Protocol	WS-BPEL, WS-CDL	Model checking
Security	WS-Security	Model checking
Quality	WSLA	Simulation

Example I: WSDL

- Describes the public interface to a particular web service
  - Data Types <types>
  - Messages <message>
  - Operations <portType>
  - Communication Protocols <binding>
- Serves as a (syntactic) contract between service providers and service consumers
- WSDL tells a client what functions are available

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions xmlns:http="http://schemas.xmlsoap.org
      /wsdl/http/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"  ...
 <portType name="EmployeeTravelStatusPT">
   <operation name="EmployeeTravelStatus">
    <input message=
      "tns:EmployeeTravelStatusRequestMessage"/>
    <output message=
     "tns:EmployeeTravelStatusResponseMessage" />
   </operation>
 </portType>  ...
 <message name=
 "EmployeeTravelStatusRequestMessage">
  <part name="employee" type="tns:EmployeeType"/>
    </message>
 <message name=
 "EmployeeTravelStatusResponseMessage">
 <part name="travelClass"
       type="tns:TravelClassType"/>
 </message> ...
```

Example II - BPEL

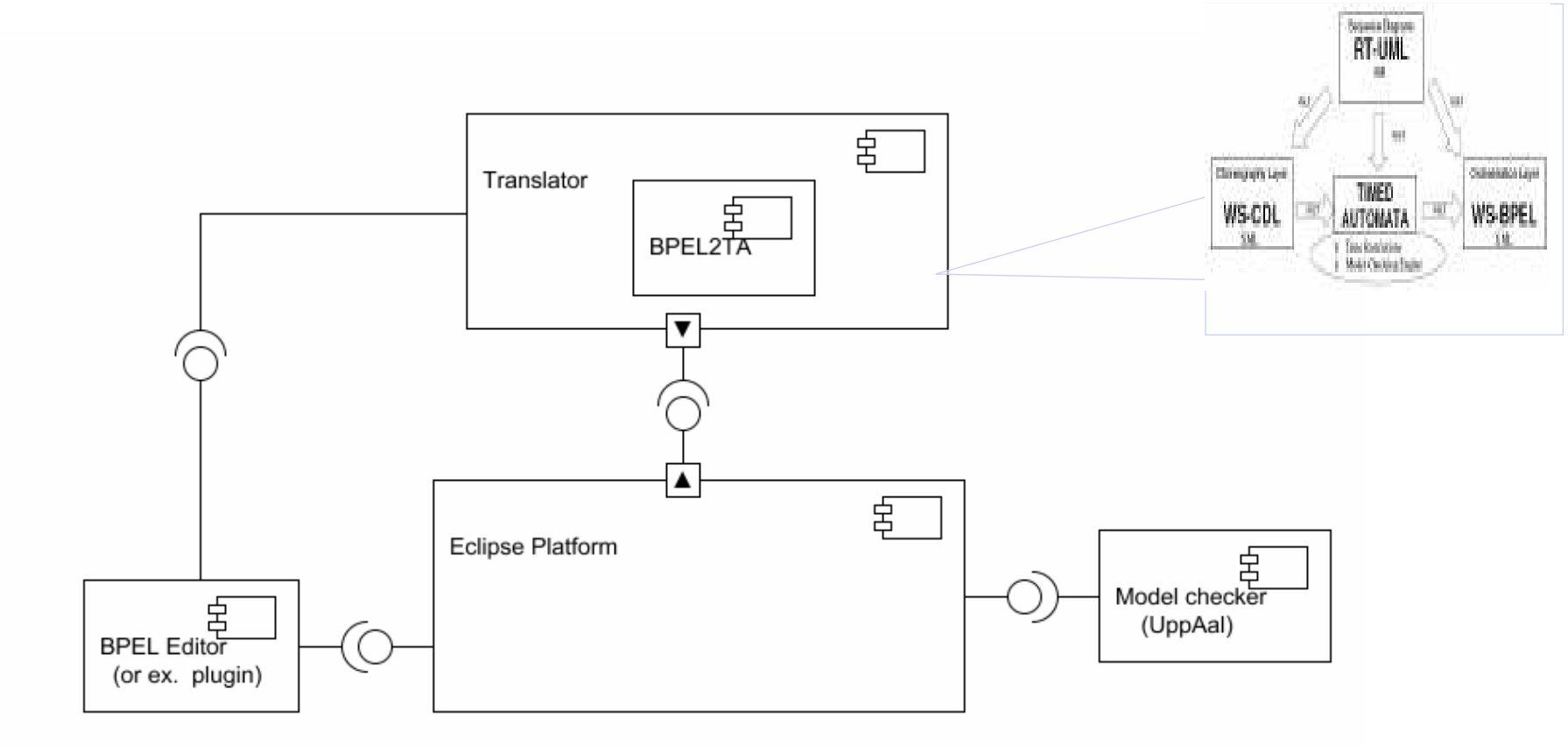
- Specifies behavioural aspects of service contract
- Uses partnerlinks and activities to model the service interaction

```
<?xml version="1.0" encoding="utf-8" ?>
<process name="Travel"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/
business-process/"
...
<partnerLinks>
    <partnerLink name="client"
        partnerLinkType="trv:travelLT"
        myRole="travelService"/> ...
    <partnerLink name="employeeTravelStatus"
        partnerLinkType="emp:employeeLT"
        partnerRole="employeeTravelStatusService"/>
...
```

```
...
 <partnerLink name="AmericanAirlines"
      partnerLinkType="aln:flightLT"
      myRole="airlineCustomer"
      partnerRole="airlineService"/>
  <partnerLink name="DeltaAirlines"
       partnerLinkType="aln:flightLT"
       myRole="airlineCustomer"
       partnerRole="airlineService"/>
</partnerLinks>
<!-- Variables are declared here-->
<sequence>
      <receive partnerLink="client"
          portType="trv:TravelApprovalPT"
          operation="TravelApproval"
          variable="TravelRequest"
          createInstance="yes" />
```

Why do we do this?

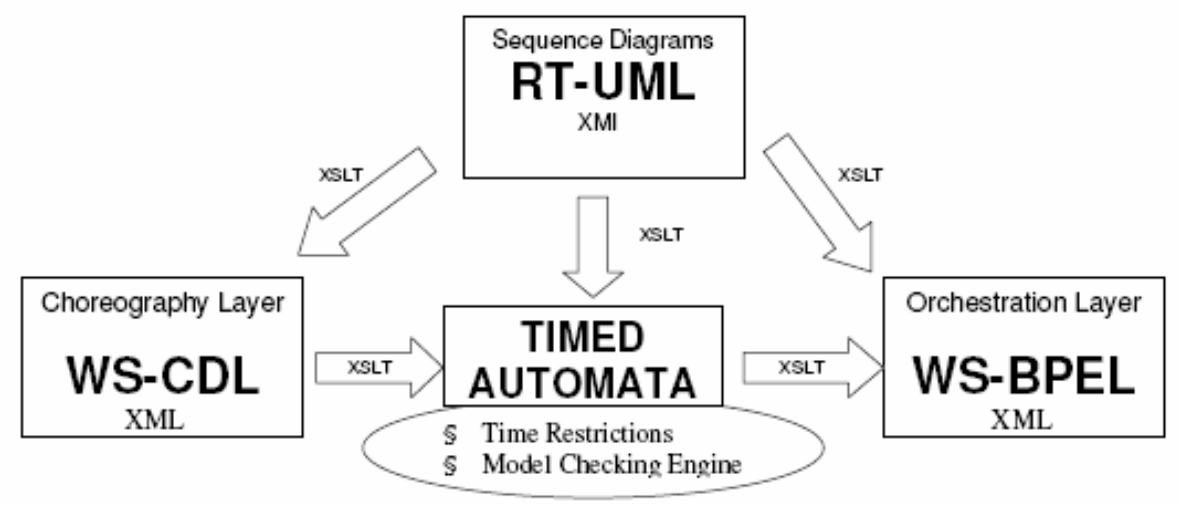
A Tool Framework for BPEL Analysis



[Ongoing work with Cambronero]

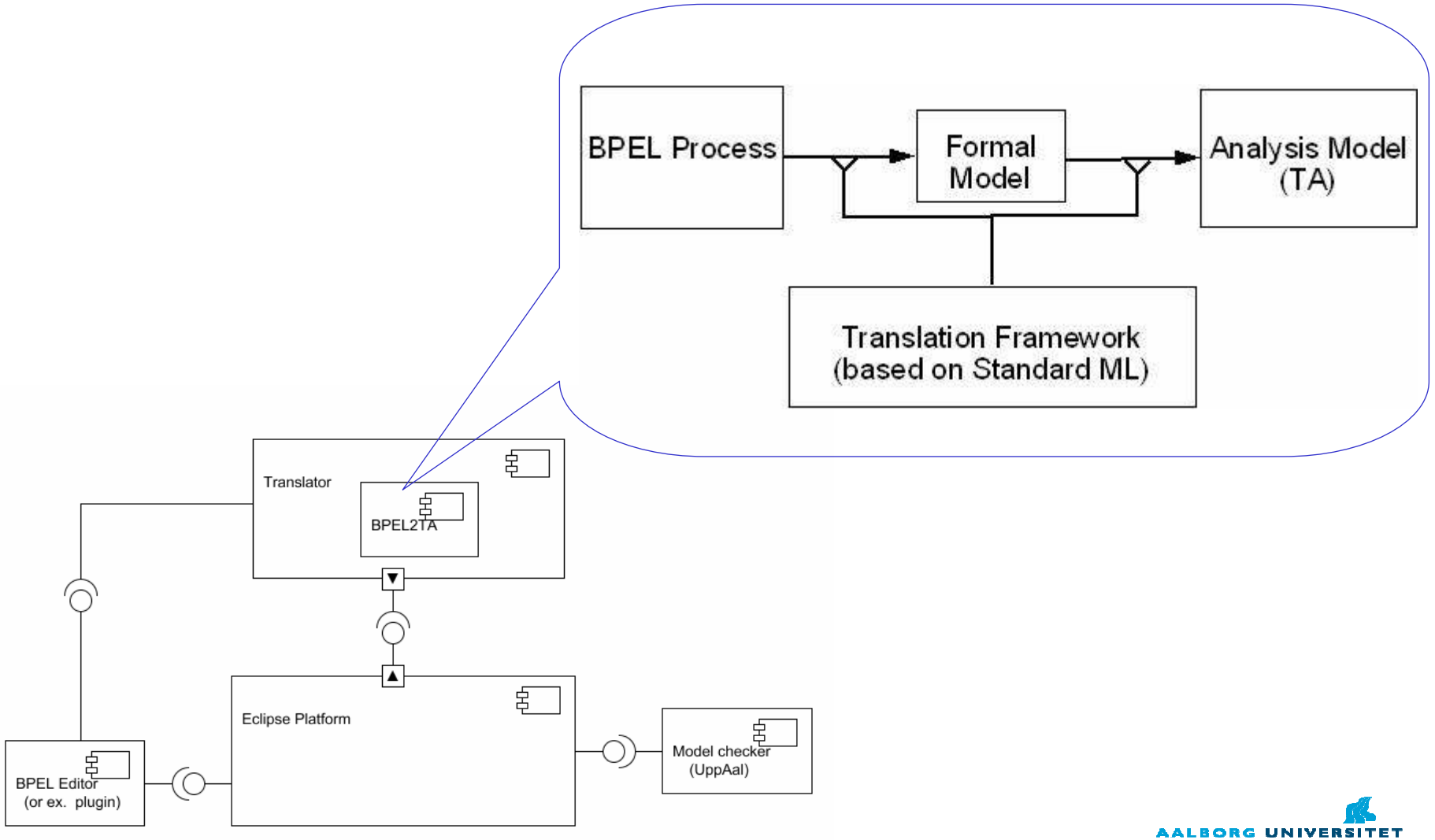
A Tool Framework for Behavioural Aspect

[Cambronero 2007]

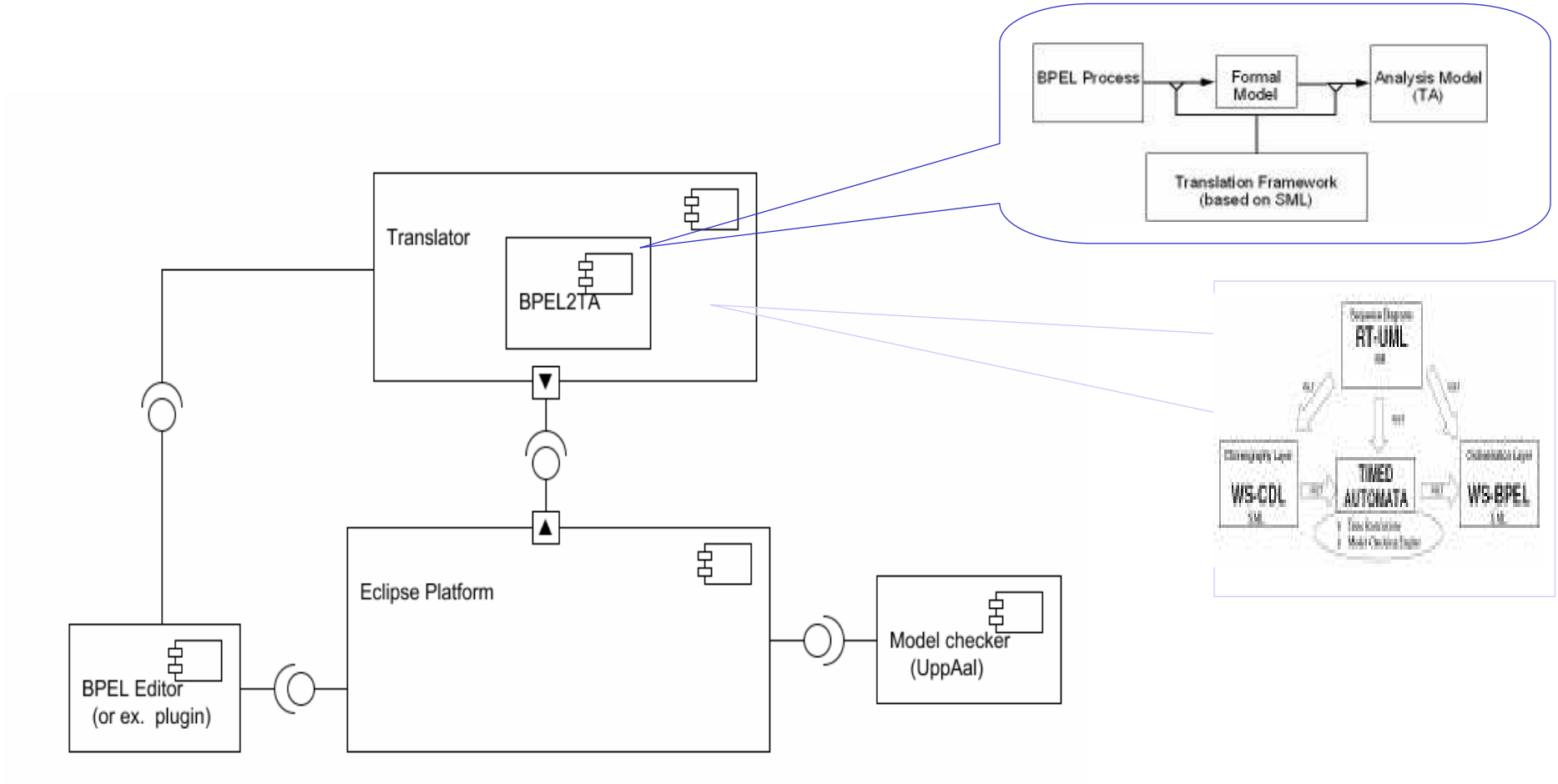


Are the translation function reasonable?
What about the semantics?

A Tool Framework for Behavioural Aspect



A Tool Framework for BPEL Analysis



The Difficult issues in Semantics

- BPEL compensation, faults and flow
- UppAal model of a compensation manager.
- Completion condition of concurrent activities
- Data dependent behaviour

What Properties should a service have?

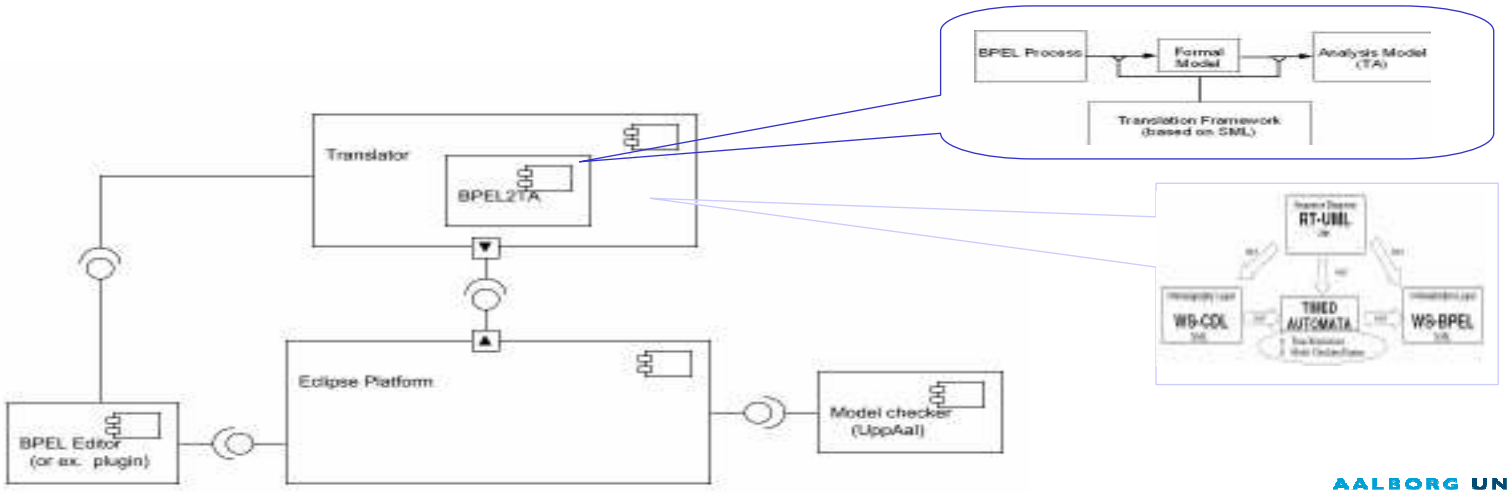
- No deadlock
- Completion - reaching an end point
- Inter-operability/Consistency with choreography
- ?

Conclusion

WS-* family covers the major aspects of SOA contract

The classification gives a blue print for:

- ❑Web Services specification language families
- ❑integrated analysis tools



```
            * Eventhandlers * Activityl
and      Onalarm = Onalarm of For_Or_Until * AScope

datatype Extension2 = Extension of Import * Partnerlinks * Messageexchanges * Variabl
            * Correlationsets * Faulthandlers * Eventhandlers * Activity
            * Extensionattr
type Extension = string

datatype BPEL_Process = bpelprocess of BPA * Extension * Import list * Partnerlinks
                    * Messageexchanges * Variables * Correlationsets
                    * Faulthandlers list * Eventhandlers list
                    * Activityl
```

```
con Actvityl = Actvityl : Onmessage
con Onalarm_P = fn :
(Expr_Attrs * Expr_Attrs * Expr_Attrs) * Activityl -> Onalarm_P
con Faulthandlers = fn : CatchAttrs * Catchall -> Faulthandlers
con Eventhandlers = fn : Onevent * Onalarm -> Eventhandlers
con AScope = fn :
Scope_Attrs * PLA list * Messageexchange_Attrs list * V_Attrs list *
CRS_Attrs list * Faulthandlers * Compensationhandler * Terminationhandler *
Eventhandlers * Activityl -> AScope
con Onalarm = fn : (Expr_Attrs * Expr_Attrs * Expr_Attrs) * AScope -> Onalar
con Extension = fn :
Import_Attrs * PLA list * Messageexchange_Attrs list * V_Attrs list *
CRS_Attrs list * Faulthandlers * Eventhandlers * Activity * Extensionattr ->
Extension2
con bpelprocess = fn :
{name : string, parallel : bool, supressJoinFailure : bool,
 targetNamespace : string, xmlns : string, xtns : string} * string *
Import_Attrs list * PLA list * Messageexchange_Attrs list * V_Attrs list *
CRS_Attrs list * Faulthandlers list * Eventhandlers list * Activityl ->
BPEL_Process
```