

# Contract-first Java Web Service development

*Experiences in Designing,  
Creating and Testing Java Web  
Services for use with BPEL*

## Titel



*Sjoerd Michels werkt als Informatie Architect voor de Oracle Solutions unit van Inter Access.*

*Hij heeft meer dan 12 jaren ervaring met het bedenken, ontwerpen en realiseren van maatwerk informatiesystemen.*

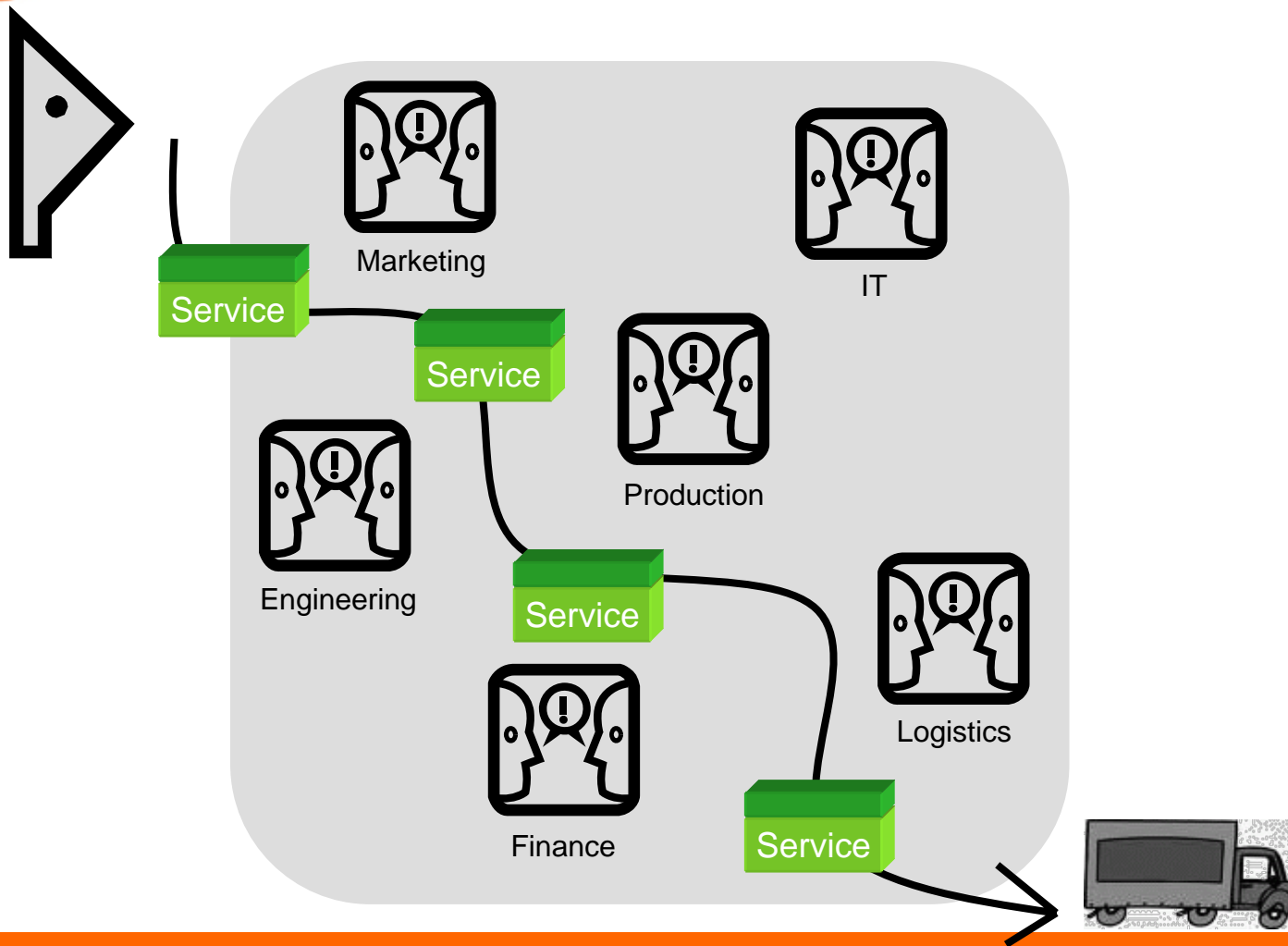
*Sjoerd programmeert in Java sinds 1997.*



Contract-first Java Web Service  
development

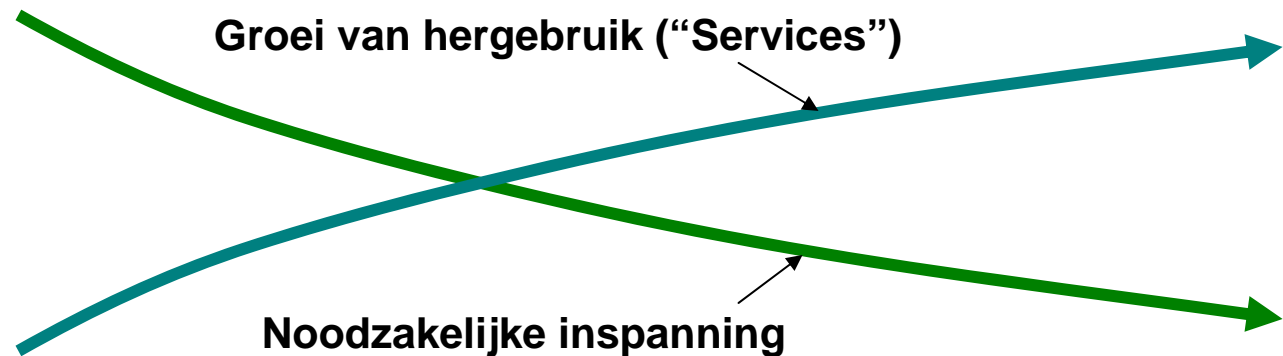
*Experiences in Designing, Creating and  
Testing Java Web Services for use with  
BPEL*

# Collaboration



## SOA: build robust, flexible systems fast

- Robust
  - Architecture defines standard interfaces
  - Service re-use
- Flexible
  - Multi-layered architecture
  - Loosely coupled services hide implementation details
- Fast
  - Specialization: make instead of buy
  - Parallel development
  - Assemble composite services



## Beware: SOA is not just ICT

Bottom-up, ICT-centric approach is tempting

- Back-office systems' complexity creeps into processes
- Tools let you quickly generate Web Services

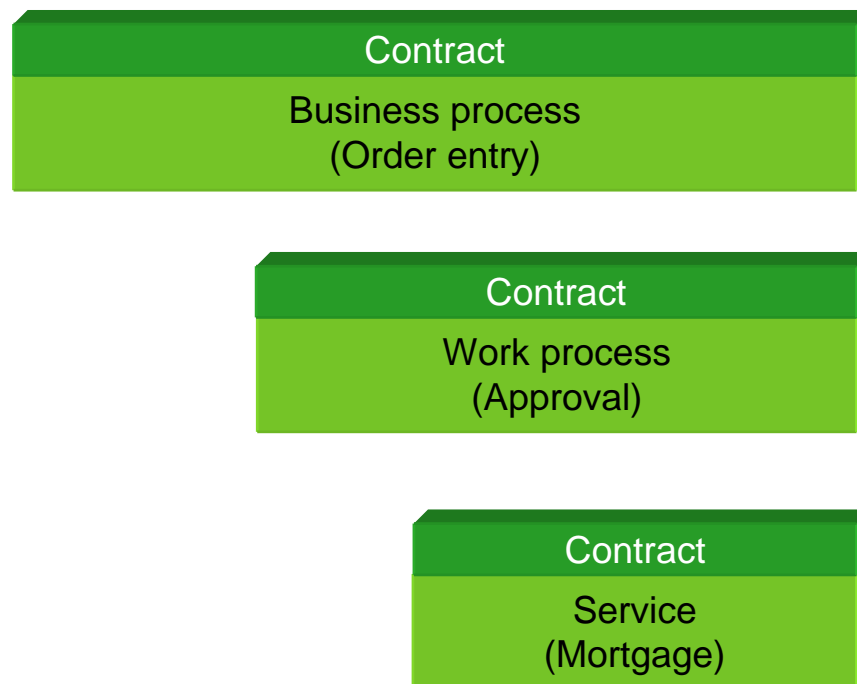
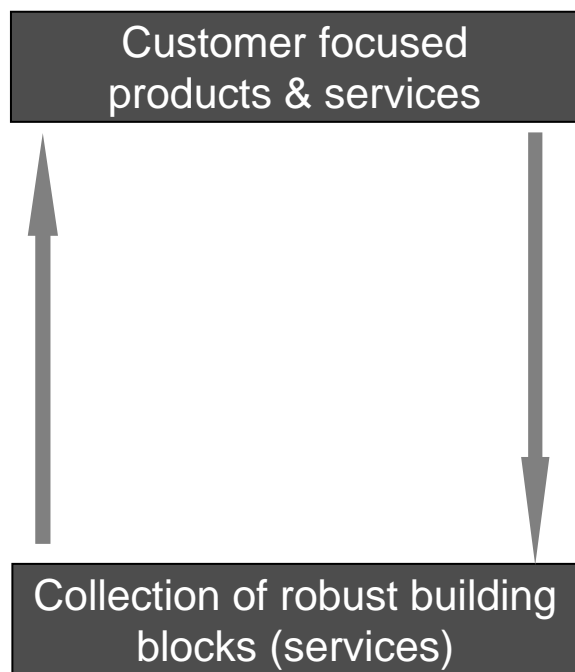


## SOA: business & ICT collaboration

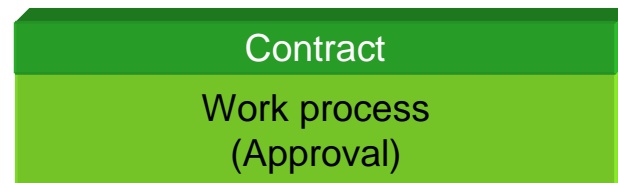
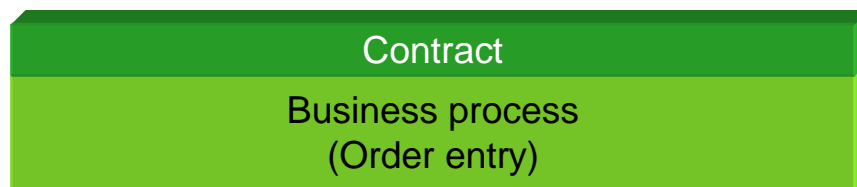
- Customer focus
- Process-centric
- Specialization
- Standards
- Loosely coupled
- Evolution



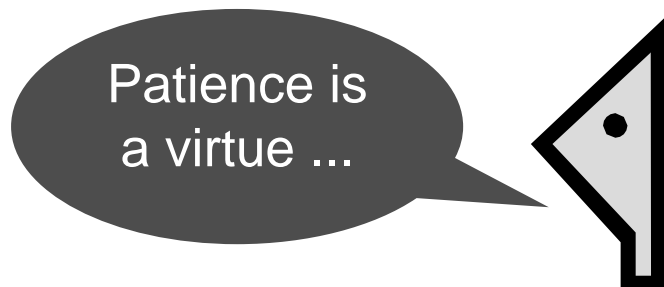
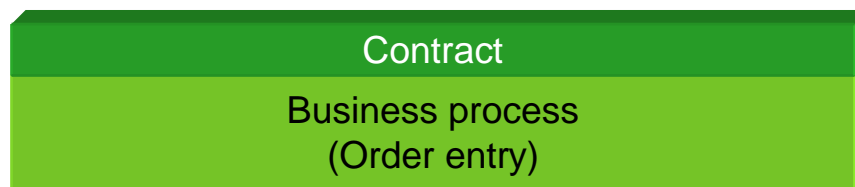
# Embrace 'contract-first' design



# Problems with 'Code-first' development



# Problems with 'Code-first' development

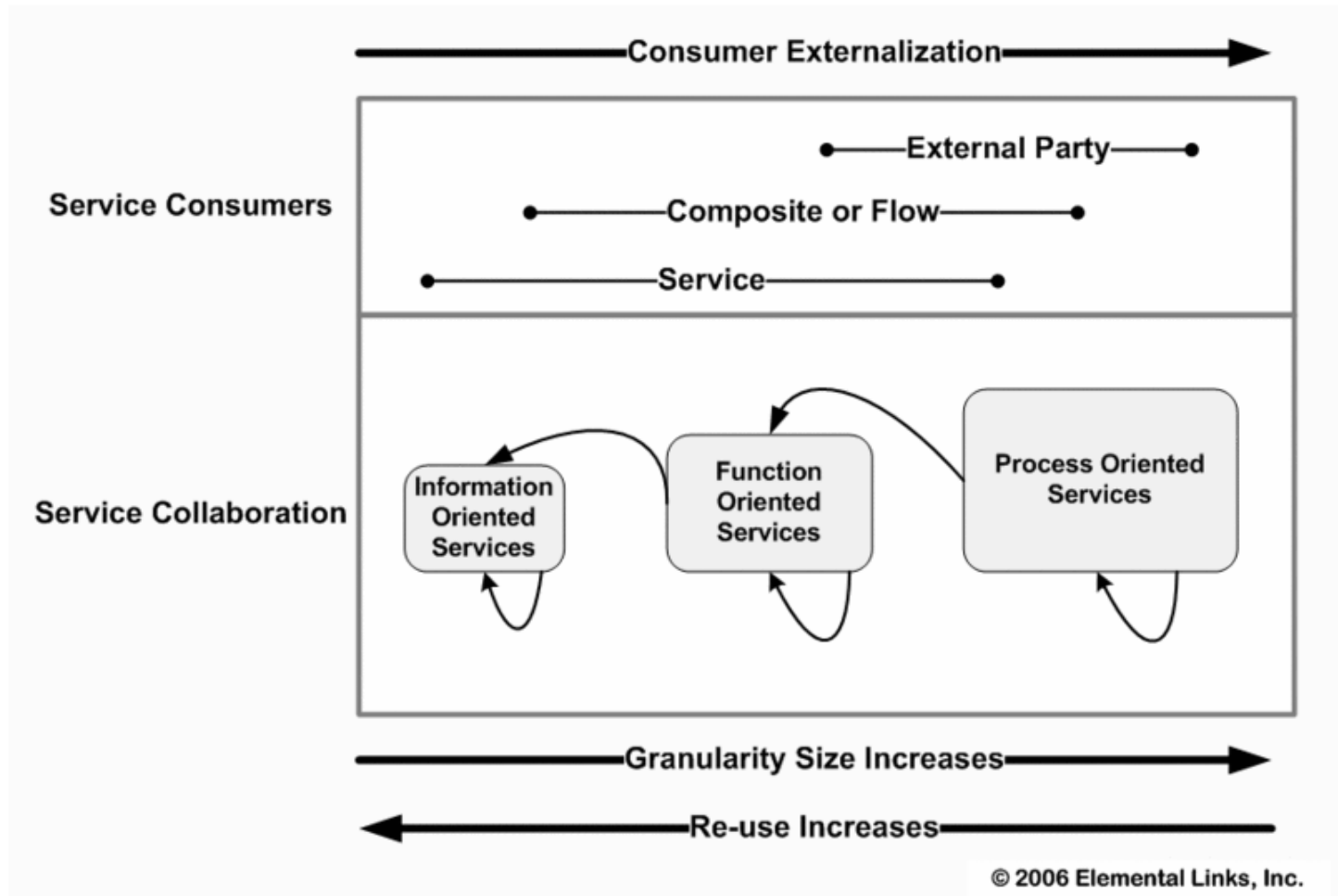


## Code-first brings complexity into the processes



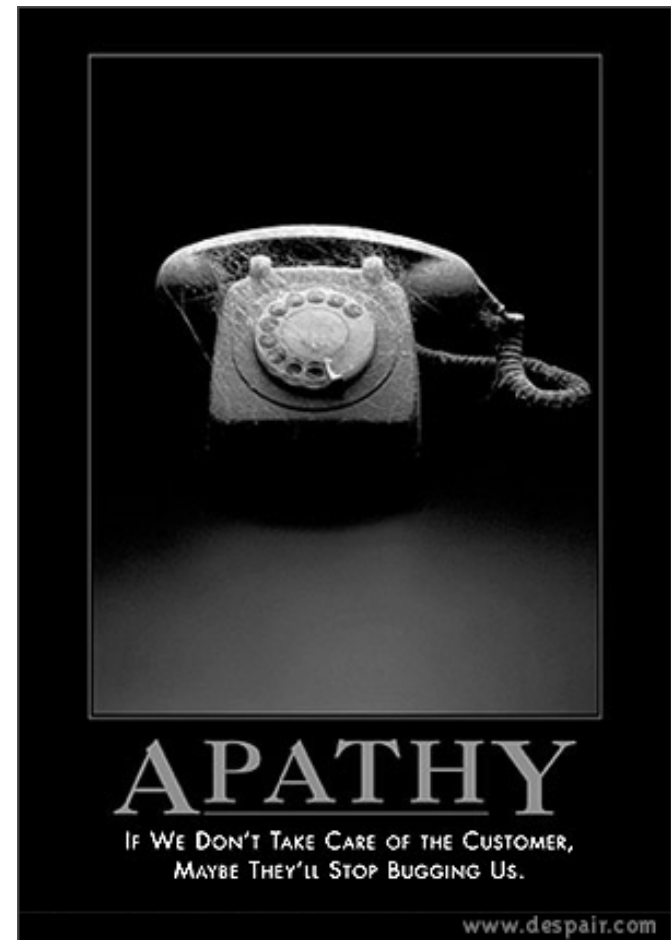
# BPEL processes?

# A design approach for SOA



## A design approach for SOA

1. Customer-focus
2. Customer-focus
3. Customer-focus
4. Products & services
5. Design processes
6. Define domain model
7. Service definition based on contracts
8. Develop systems



## Domain model: common ground semantics

```
<xsd:complexType name="pipe">  
  <xsd:sequence>  
    <xsd:element ...  
  </xsd:sequence>  
</xsd:complexType>
```



## 'Contract-first' design

Contract

<XML>

Operation X

- input
- output

Service implementation

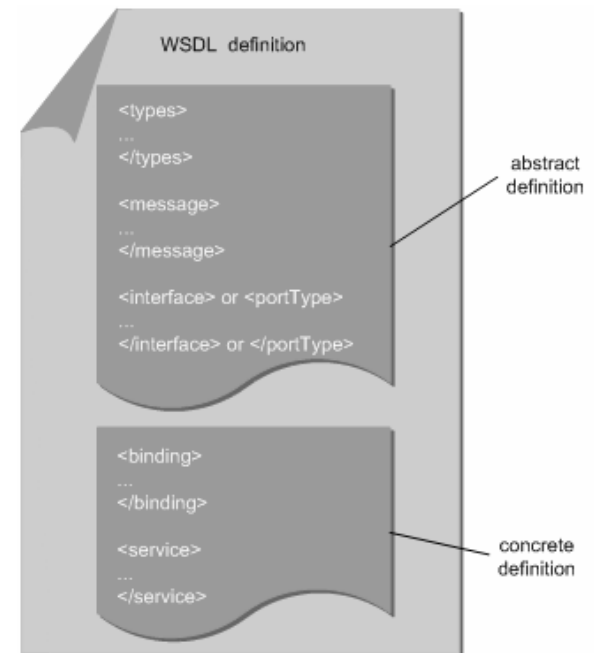
- Implementation details hidden
- May change over time

## XML: The lingua franca in SOA

- XML is the 'datamodel' in SOA
- XML messages between services
- Contract is formally specified in XML: Web Service Definition Language
- Best practice: document/literal style (defined by W3C XML Schema)

## WSDL: the ideal contract?

- Web Service Definition Language
- WSDL formally describes a contract
- Specification in XML



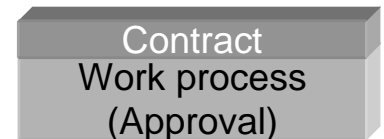
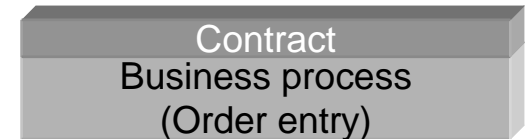
## Advantages of WSDL

- Abstract definition understandable for everyone
- Implementation independent, decouples technologies
- Division of work:
  - OurWSDL > Starting point for BPEL modeling
  - OurWSDL > Web Service implementation
- Generate Web Services infrastructure from WSDL?

## Creating Java Web Services from WSDL

### Requirements:

- WSDL is starting point
- Should create infrastructure from WSDL (boring bits and pieces)
- Productivity: allow focus on functionality, avoid SOAP extensions and WS\* stuff
- Integrate well with development process (develop, test, deploy)
- Java



## Frameworks

*axis*

~~X~~FIRE

The safe choice: Apache Axis

- Most widely used platform for Java SOAP Web Services
- Well-documented, loads of samples
- Not perfect, but solid roadmap

# AXIS?

Apache eXtensible  
Interaction System

## Apache Axis: usage recipe

- Axis: SOAP Server and Client
- Design XML schema and WSDL
- Wsd2Java → Java classes
- Start adding functionality!
- Test
- Deploy

Generate *infrastructure!*

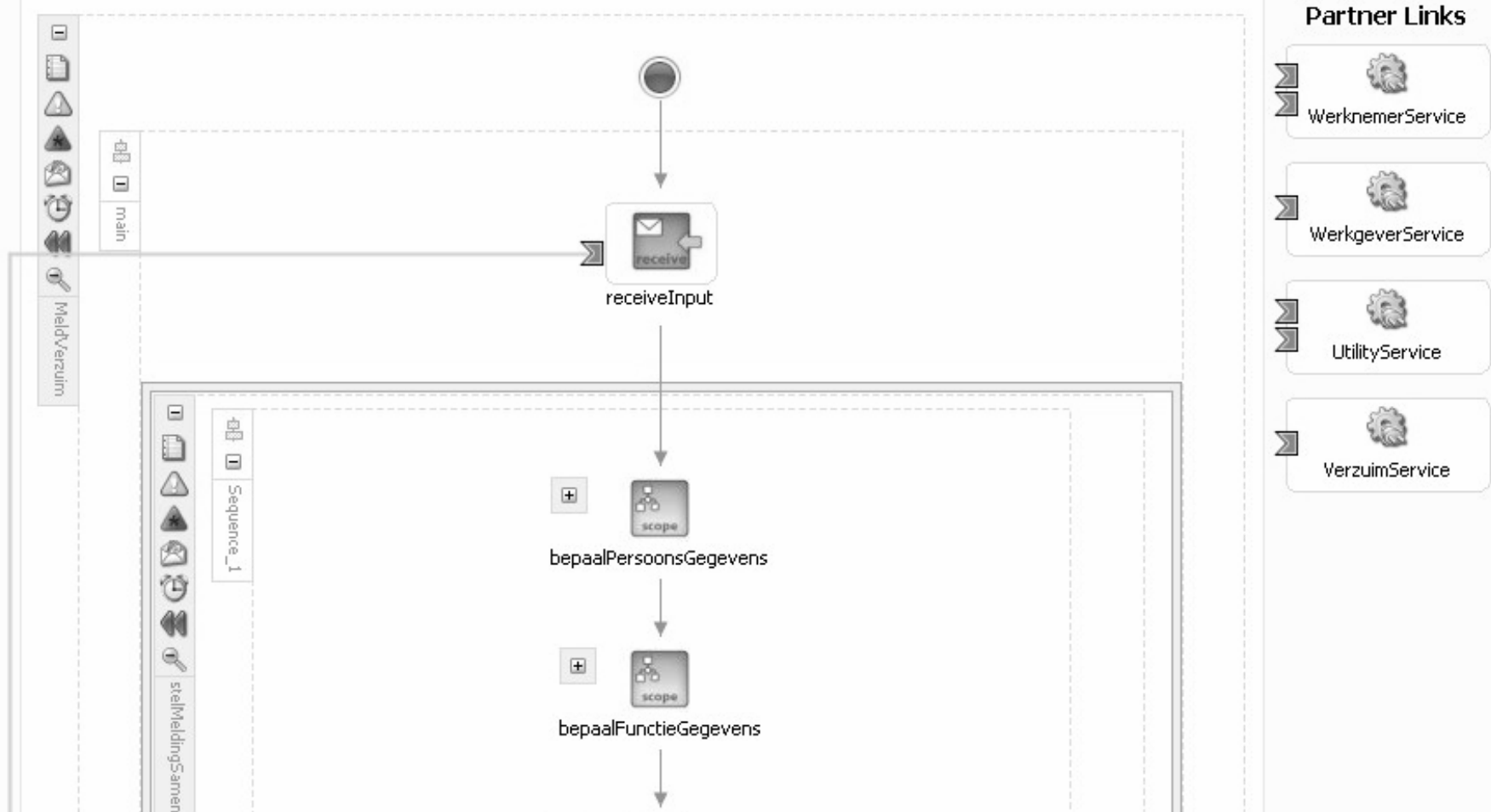
## Axis 1.4: no free lunch!

- Axis requires concrete WSDL definition
- Warning: Axis breaks the contract!
- How well does Axis support XML?
  - Designed around JAX-RPC 1.0
  - Limited binding capabilities
- Bugs
- Deployment

## Axis and Ant development automation

- Automate the development process
  - Set axis.classpath
  - axis-ant.jar
  - `<taskdef resource="axis-tasks.properties" classpathref="axis.classpath" />`
- In a 'contract-first' setting:
  - Wsdl2Java Axis Ant task

# BPEL: service orchestration



## Contract-first approach streamlines BPEL development

- Meaning is well-defined
- Services share the same semantics  
→ hardly any transformations
- Simple assign statements →  
copy complex types with a single  
assign statement

## Axis2: fulfilling the promise?

- Improved performance  
SAX → AXIOM (AXIs Object Model)
- Pluggable multiple data-binding frameworks (AxisDB, XMLBeans, ...)
- RPC-style → Message-style interactions  
Object-exchange → Message-exchange
- Improved support for WS\* standards
- Hot deployment of Web Service Archives

## Axis2: fulfilling the promise?

- Improved performance  
SAX → AXIOM (AXIs Object Model)
- Pluggable multiple data-binding frameworks (AxisDB, XMLBeans, ...)
- RPC-style → Message-style interactions
- Improved support for WSDL

**Not ready for production**

## Lessons learned

### Summary

- Meet in the middle: where business meets ICT
- SOA is all about carefully architecting and designing: let the architecture work for you!
- Contract-first design provides the common ground
- Axis supports contract-first development
- Automate the development process

# Questions?



[sjoerd.michels@interaccess.nl](mailto:sjoerd.michels@interaccess.nl)