

# **SERVICE ORIENTED COMPUTING :** **Service Modelling**

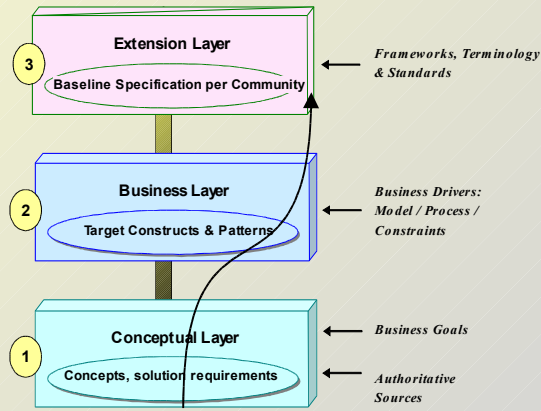


**Prof. dr. ir. Mike P. Papazoglou**  
**INFOLAB,**  
**Dept. of Information Systems & Mgt**  
**Tilburg University,**  
**The Netherlands**  
**email: [mikep@uvt.nl](mailto:mikep@uvt.nl)**  
**<http://infolab@uvt.nl/people/mikep>**

# Holistic Approach to Service-Oriented Computing

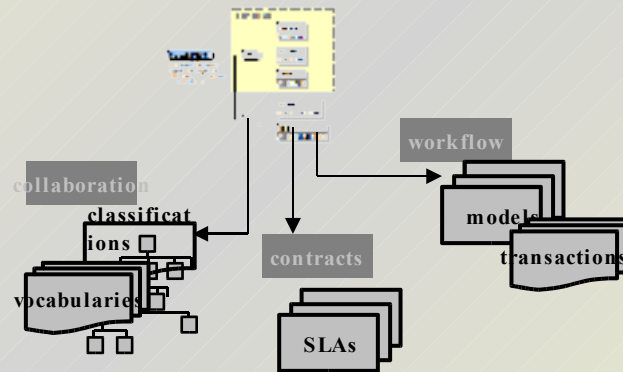
Step 1 :

Use Layers to analyse & define Business Requirements



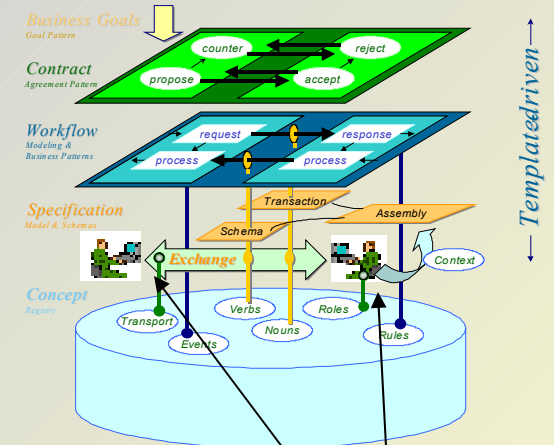
Step 2 :

Build service templates with Service Modelling Tools



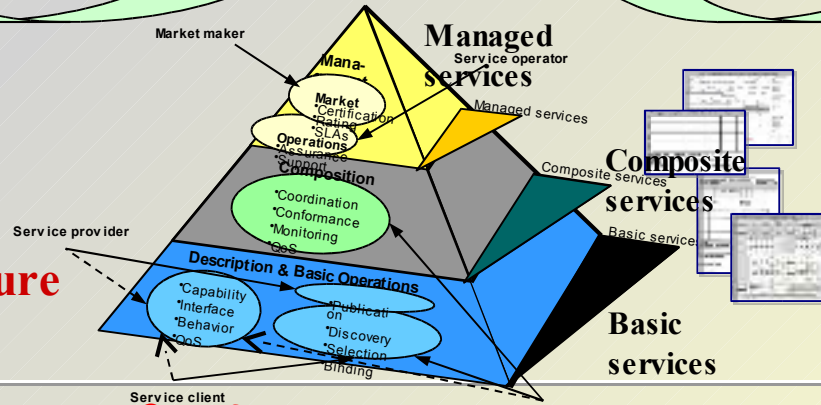
Step 3 :

Deploy with service interfaces & implementations

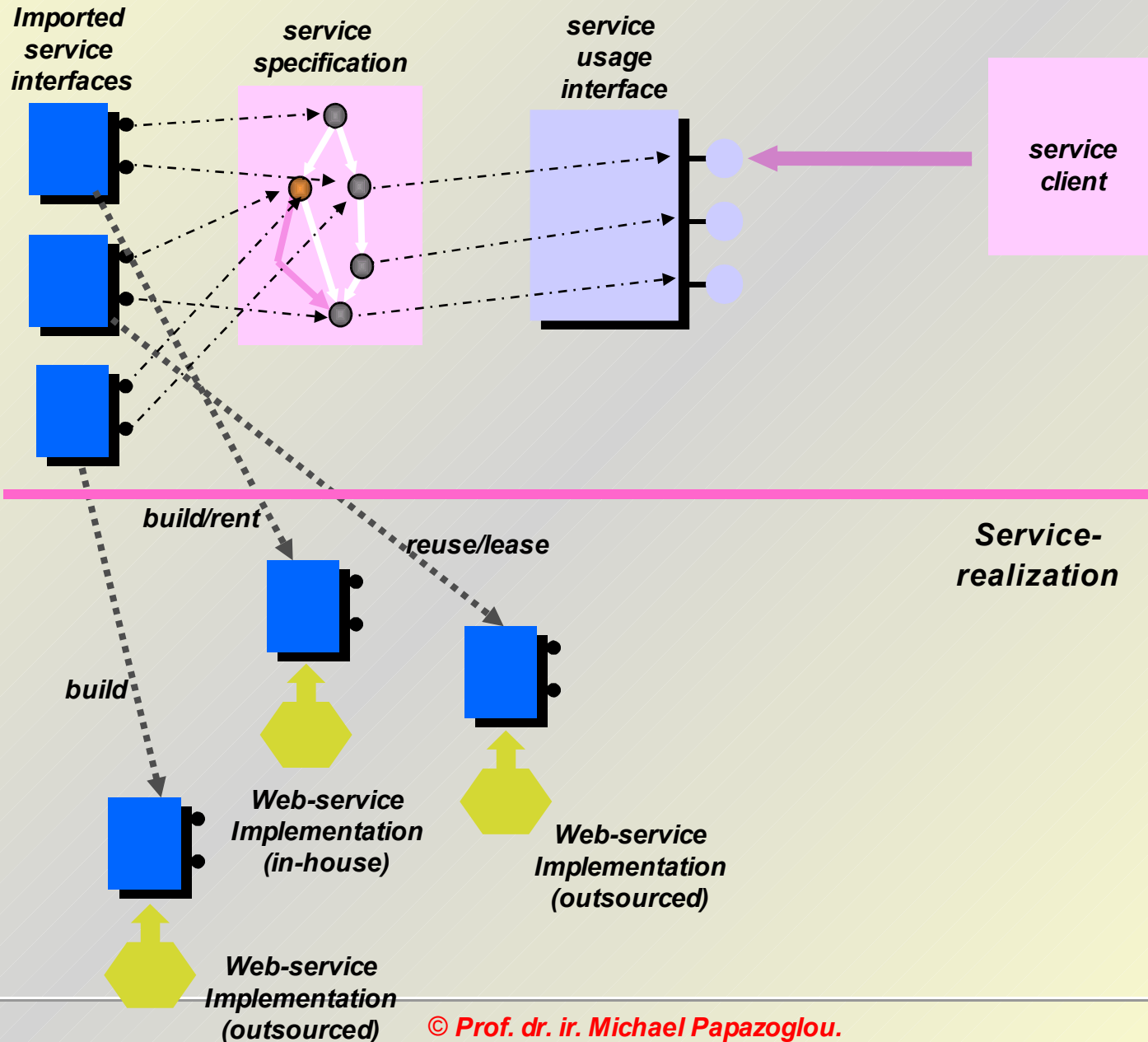


Cooperating organisations: "ocean" of services.

While referencing the Extended Service-Oriented Architecture



# Service Interfaces & Realization



# Service views in an enterprise



## Business Executive:

What interactions must we enable with customers and partners?

How do we secure those interactions?

What business processes need to be most adaptable?

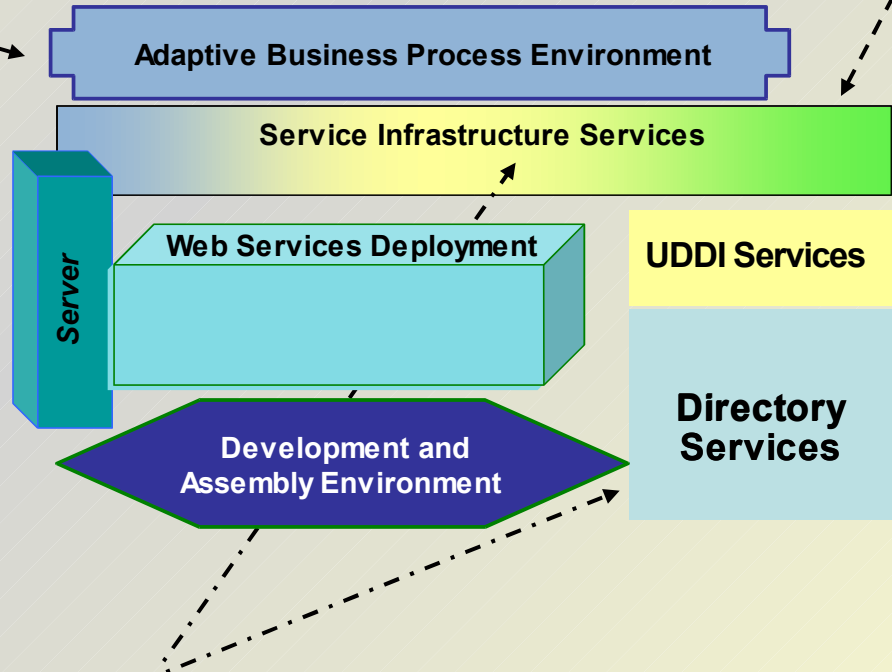
## IT Executive:

What services should be created and shared across the organization & between the organization & its customers?



## Consider:

- Service granularity &
- Service Connection to enterprise assets



## IT Architect:

What are the infrastructure functions that should be separated from application functions?

# *Describing business process features*

**Web-services characteristics impinge on the design characteristics intrinsic to business processes.**

A business process leverages the <port type> element in WSDL to define its basic process activity interface.

The fundamental value of the business process is that it provides the protocol that enables business activities (and hence web-service invocations) to have coherence with each other and achieve the desired result.

- a) **Determine objectives and describe the business process structure:**
2. **Identify, group and describe the activities that together implement a business process.** To assemble a higher-level service by combining other web services, the service designer needs to:
  - ✓ Select the services to compose by looking at how these services and their operations within a business process relate to one another.
  - ✓ Connect the usage interface of the business process to the interfaces of imported services and plug them together.
3. **Describe activity dependencies, conditions or synchronisation:** A process definition can organise activities into varying structures:
  - ***Hierarchical process activity definitions:*** activities have a hierarchical structure. e.g, the activity of sending an insurance policy for a travel plan can be divided into three sub-activities: compute the insurance premium, notify insurance, mail insurance premium to the customer.

- ***Conditional activity definitions:*** in process definitions that have a conditional activity structure activities are performed only if certain conditions are met, e.g, it may be company policy to send a second billing notice to a traveller when an invoice is more than two months overdue.
- ***Activity dependency definitions:***
  - ✓ In any process definition, sub-activities can execute only after their parent activity has commenced.
  - ✓ In other cases there might be an explicit dependency between activities: an activity may only be able to start when another specific activity has completed. e.g, an itinerary confirmation cannot be sent to a traveller unless all flights have been reserved by an airline.
- ***Describe the implementation of the business process:*** Write the application, e.g., provide a BPEL definition, that maps the operations interfaces of imported services to those of another in order to create the usage interface of the business process (higher- level web service).

***d) Describe business activity responsibilities (roles)***

# *Service Design Principles*

- Services should be self-contained, modular and support service composability, thus they should rely in:

## **2. Service coupling**

- The objective is to minimise coupling, i.e., make business processes as independent as possible by not having any knowledge of or relying on any other business processes.
- Low coupling between business processes indicates a well-partitioned system that avoids the problems of service redundancy and duplication.
- Coupling can be achieved by reducing the number of connections between services, & by reducing the number of necessary relationships, if possible.



- **Representational coupling:** The business processes should not depend on specific representational or computational details of one another. Think of abstract classes in o-o design. These concerns lead to the exploitation of interoperability and reusability for service design & leads to:
  - ✓ Interchangeable/replaceable services.
  - ✓ Multiple service versions.
- **Identity coupling:** Connection channels between services should be unaware of who is providing the service.
- **Communication protocol coupling:** A sender of a message should rely only on those effects necessary to achieve effective communication. e.g, one- way styles of operation where a service end point receives a message without having to send an acknowledgement places the lowest possible demands on the service performing the operation.

# 1. Service cohesion

Cohesion is the degree of the strength of functional relatedness of operations within a service. Guidelines to increase cohesion:

3. **Functional service cohesion:** A functionally cohesive business process contains services that all contribute to the execution of one and only one problem-related task. Operations in the services of the business process must also be highly related to one another.
4. **Communicational service cohesion:** A communicationally cohesive business process is one whose activities and services contribute to tasks that use the same input and output messages.
5. **Logical service cohesion:** A logically cohesive business process is one whose activities all contribute to tasks of the same general category. These tasks are selected and invoked outside the business process.

# Non-Functional Modelling Challenges

- Good understanding of QoS & composition of different QoS (e.g., overall cost, performance, reliability, availability, scalability, integrity)
- Service provisioning strategies
  - ✓ Service realization strategies (Gap analysis)
    2. Purchasing/leasing/paying per use for services.
    3. Outsourcing service design and implementation.
    4. Using wrappers and/or adapters.
- Service billing strategies
- Accounting/auditing services
- Service policy management models
- Security/authentication/authorisation service models
- Modelling transactional characteristics.

## **A Final Remark**

- SOC publications, e.g., service compositions, web service transactions, P2P services, etc .. can be found in:
  - ✓ <http://www.uvt.nl/infolab/pub/db/>