Université de Lausanne
Ecole des Hautes Etudes Commerciales (HEC)

beat michel
CONSEIL EN INFORMATIQUE

Workshop EMMSAD - CAISE 2006 - Luxembourg 5th, 6th june 2006

Modeling Services using Contracts
Identifying dependencies in Service–Oriented Architectures

Thibault Estier – Beat Michel – Oliver Reinhard

thibault.estier@unil.ch - michel@beatmichel.ch

Agenda

- Context of this work
- Motivation for service contracts and model
- Service model
- Towards a methodology for SOA migration

Context: migrating from ad hoc architecture to service architecture

- Inside the same IT platform
  - Accent on relationships between applications of the same organization/company rather than relationships with applications outside the organization .
- Migration rather than implementing from scratch
  - Replacing existing information exchange between legacy applications by a systematic service oriented approach.
- Not limited to web service technology
  - This is not a paper on pre- and post-conditions with WSDL.
- Granularity is that of applications
  - Not companies, nor EJBs

Motivation for service contracts and service model

- Separation of concerns
  - Establish clear responsibilities of each application
- Loose coupling
  - Avoid impacts between applications
- Reusability
  - Allow for reuse of functionality. The service provider makes no assumptions about a service consumer, others than those specified in the service contract.

Service contracts and service model

- Service contract
  - mutual agreement between a service provider and a service consumer, based on the description of what the provider offers,
  - the service offer is described by a *model*
- Service model

  formal description of several aspects of a given application service offer

  1. **Operations and Structures** – the service operations, their parameters and parameter types,
  2. **Semantics and Scope** – the behavior of the service operations, returned results and side effects; do they overlap the information scope of other services,
  3. **Failures** – behavior when the consumer-provider interaction will not succeed,
  4. **Quality** – availability, what non-functional requirements does the service comply to.

Aspect 1: Service Operations and Parameters – Syntax to invoke

- A service is a set of **closely related** operations.
  - Operations are closely related if they share the same parameter types and access or modify the same persistent information.
- Each operation has a **signature** (name, formal parameters, failure signals)
- Closure of types of all parameters data structure define a **Service Type Model**

Aspect 2: Semantics and Scope – About producing effects

- The Post-condition describes the result of a service call: information returned to service consumer or state transition of the service provider.
- The Pre-conditions states what must be respected by the service consumer. Design-By-Contract paradigm:

  preconditions $\wedge$ **operations** $\Rightarrow$ **post-conditions**

- The **Core Type Model** contains the Service Type Model (parameters types) **plus** all the necessary (persistent) types to express semantics of each operation.

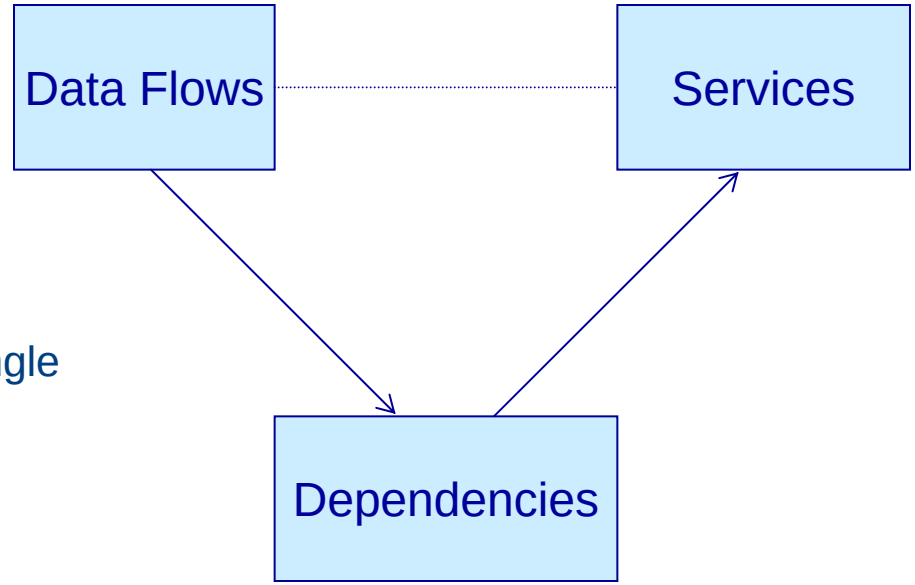
Aspect 3: Failures - About not producing effects

- The service model declares how **failures** will be handled.
- Two possible precondition failure handling:
  - **Checked** preconditions – The service consumer can safely rely on the provider checks.
  - **Unchecked** preconditions – The service consumer cannot make any assumption concerning the provider behavior in case of precondition violation.

SOA and existing application landscapes

- What you see:
  - **Data flows** between applications
- What it means:
  - **Dependencies** between applications
- What you aim to:
  - **Service contracts** between applications

Yet another SOA triangle



Steps of a Service Architecture Migration methodology

1. Identify the scope of migration effort and identify all data flows between applications in the scope.
2. **Determine, clarify and optimize dependencies between applications, starting from existing data flows.**
3. Formally specify the service contracts.
4. Chose the most appropriate technology for implementing each service model (from SOAP to file transfer).
5. Implement and deploy services.

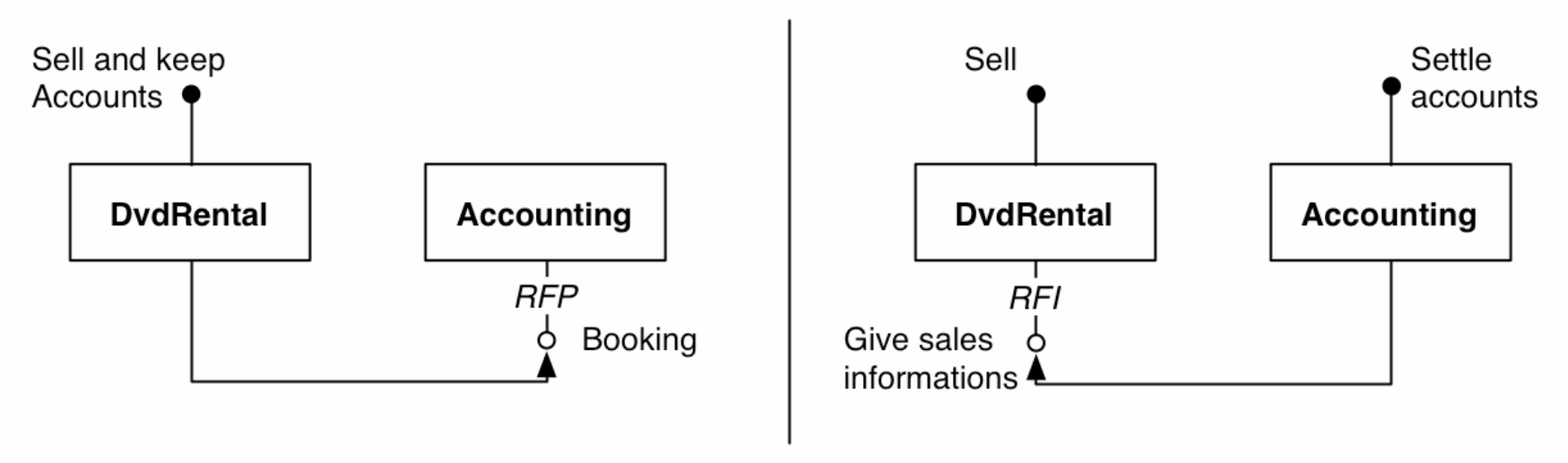
Business services vs application services

- A *business service* is provided to a business process, typically through a user interface.
- An *application service* is provided to an other application.
- An application uses services of other applications to provide its own (business or application) services.

Orientation – Establishing a service relationship

- Relationship between service provider and consumers is asymmetric
  - The consumers knows all about the providers service model.
  - The provider knows nothing about the consumer (except what is stated by the preconditions).
- Two types of dependency
  - **information dependency** – the consumer needs information supplied by the service provider, in which case he makes a request for information (**RFI**)
  - **delegation dependency** – the consumer delegates part of its processing to the service provider, in which case he makes a request for processing (**RFP**).
- In practice it may be difficult to decide **who** has to play the service **provider** role.

Information dependency vs delegation dependency



DvdRental delegates booking to the *Accounting* application (Requests For Processing : RFP)

Accounting needs sales information from *DvdRental* (Requests For Information : RFI)

In both cases data flow is from *DvdRental* to *Accounting*.

Blocking vs. Non-blocking

- Blocking
  - RPC-like information service: the service consumer asks for information to the service provider.
  - Delegation Service: the service consumer expects an information about successful processing (or failures)
- Non-blocking
  - Publish-subscribe pattern information service: the service provider publishes information, the service consumer subscribes to the information.
  - Delegation Service without return information: the service consumer submits its request for processing. The service provider alone is committed by contract to handle the request and all possible failure situations (as long as the preconditions are validated)

Making choices when modelling services

- Define the semantics of a data flow either as
  - information dependency, or as
  - delegation dependency.
- For delegation dependency choose between
  - return information on result or
  - no return information.
- For information dependency choose between
  - RPC-like or
  - publish/subscribe pattern.

A case for event based service architecture?

- Loose coupling is argument for a general tendency from RPC-type delegation services to publish-subscribe-type information services.
- This looks like a case for event propagation as a basic mechanisme of a service oriented architecture.

The third author … in migration.

Questions ?