Models and Technologies for Service Composition (issues and solutions)

L. Baresi, M. Matera, and P. Plebani
Politecnico di Milano
Dipartimento di Elettronica e Informazione

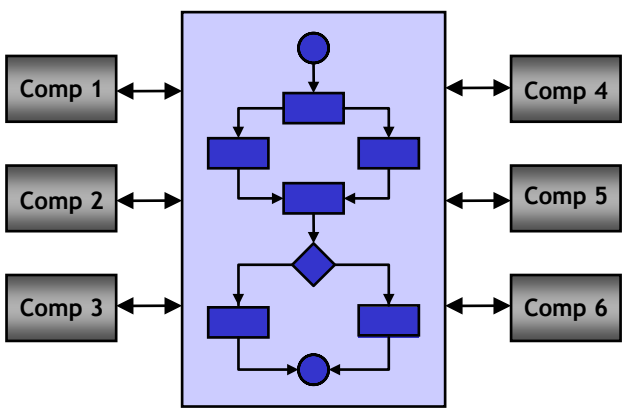
Instructors

- Luciano Baresi
  - Associate professor
  - Software engineering
- Maristella Matera
  - Assistant professor
  - Database and Web technology
- Pierluigi Plebani
  - Ph.D. student
  - Information Systems

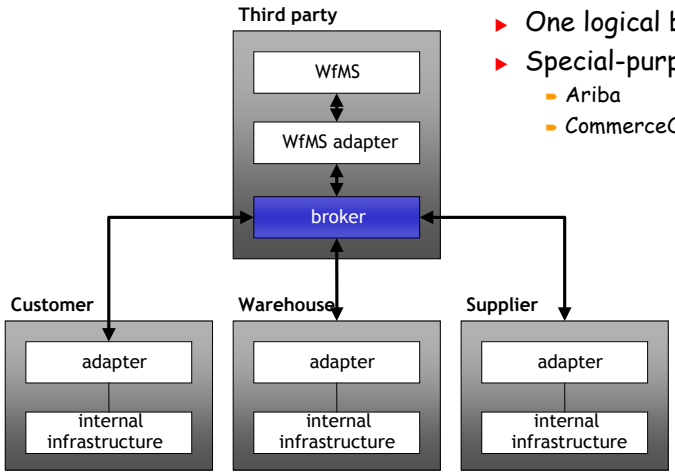
Outline

- Introduction to composition and services
- Background concepts for provision and invocation
- Basic approaches to composition
- Main issues about composition
- Advanced technologies for composition
- Analysis
- Conclusions

The problem



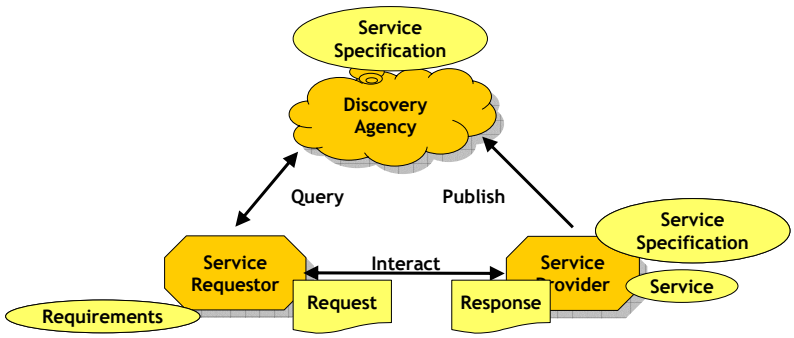
Integration through "standard" middleware technology



- One logical broker
- Special-purpose companies
  - Ariba
  - CommerceOne

Alonso, Casati, Kuno, and Machiraju
Web Services (Concepts, Architectures and Applications)
Springer, 2003

Service-oriented systems

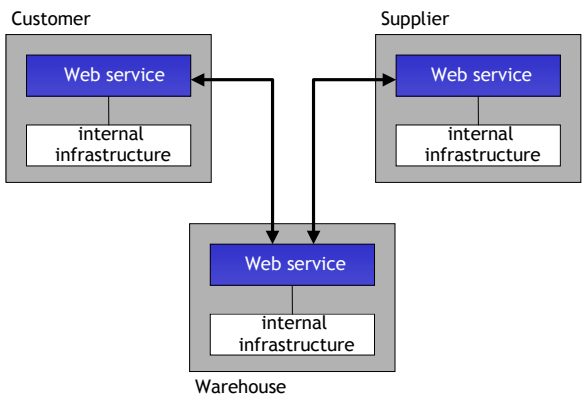


Web services

- A Web service is a platform and implementation independent software component that can be:
  - Described using a service description language
  - Published to a registry of services
  - Discovered through a standard mechanism
  - Invoked through a declared API
  - Composed with other services

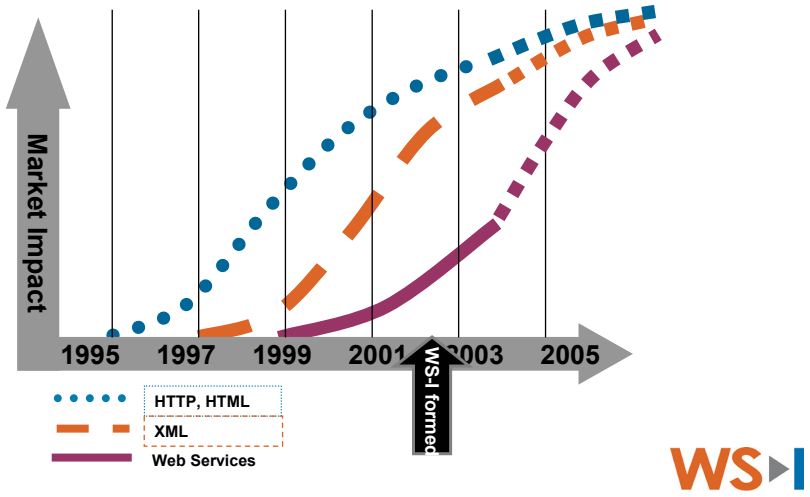
Building Web Services with Java

Integration through Web Services



Alonso, Casati, Kuno, and Machiraju
Web Services (Concepts, Architectures and Applications)
Springer, 2003

The opportunity



Advantages

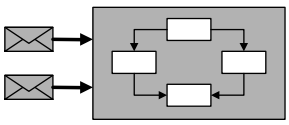
- Supply an interoperable solution to application integration
  - Act as "lingua franca"
  - Separate collaboration logic from application logic
- Replace and encapsulate proprietary solutions
- Facilitate the integration of applications between different organizations
  - Integrate Web services within the enterprise and across enterprise boundaries
- Save massive amounts of money through integration of business processes
- Supply flexibility and speed in B2B applications
- Enable long running business processes with the power of Web services

Challenges

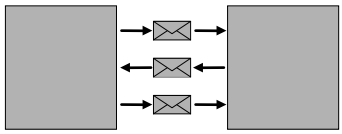
- Support the interactions among parties
  - Coordinate asynchronous communication between services
  - Correlate the exchange of messages among parties
- Implement parallel processing activities
  - Synchronize independent entities
- Transform data among interactions
  - Manipulate proprietary formats
  - Support transformations between formats
- Support for long running business transactions

Composition models

- Orchestration
  - Intra-process
  - Process controlled by one party



- Choreography
  - Inter-processes
  - Sequence of observable messages
  - Conversation among equals



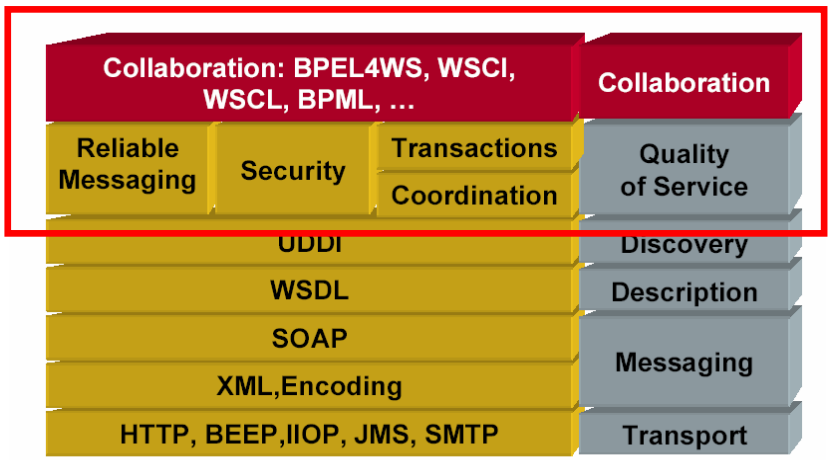
Further dimensions

- When
  - Static composition happens at design/compile time
  - Dynamic composition happens while executing the process
- How
  - Non functional aspects like
    - QoS
    - Security
- Transactional behavior
  - Need for long-lived atomic interactions

How many standards do we need?

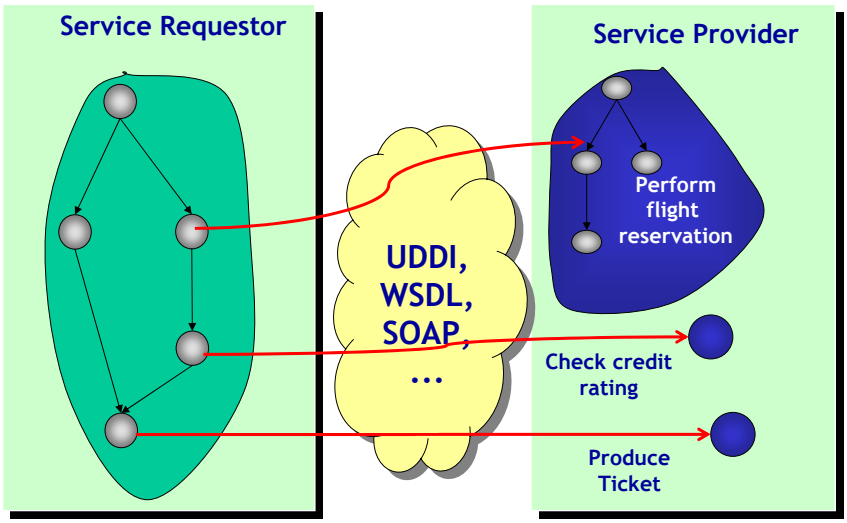
WSLA
WSCI
BPEL4WS
WSFL
WS-Coordination
WSXL
WS-I
WS-Transaction
XLANG
BPML
DAML-S
WS-Security
WSRP
WS-Policy
WSCL

Technology stack



Background concepts for provision and invocation

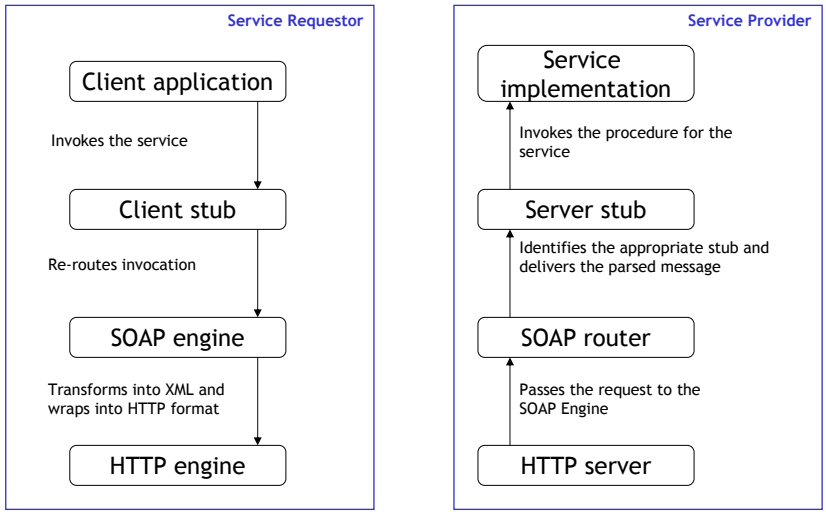
SOA in Action



SOAP: Simple Object Access Protocol

- Defines the strucure of messages to be exchanged
  - Interoperability amongst a wide range of programs, on a wide range of platforms
- Use of existing technologies wherever possible
  - HTTP and SMTP as transport protocols
  - XML as interaction (RPC and messaging) encoding scheme

SOAP implementation



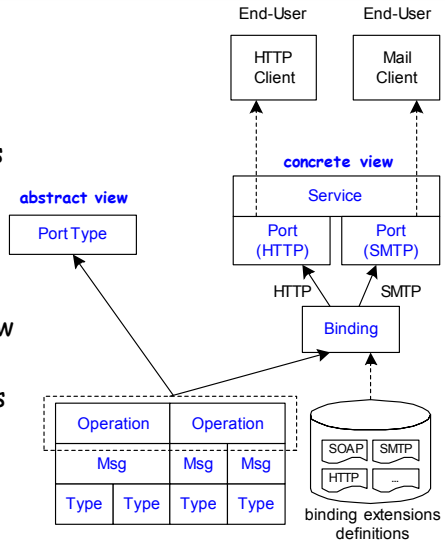
Alonso, Casati, Kuno, and Machiraju
Web Services (Concepts, Architectures and Applications)
Springer, 2003

How to define and publish services

- What is needed for publishing our own service (both a simple function or complete business process)?
- WSDL (Web Services Description Language): a language to describe:
  - How others can invoke our service
  - What to expect when it responds
- Abstract view: service description through the exposed operations
  - Port types as collections of related operations exchanging messages
- Concrete view: protocol binding
  - Bindings, ports, services

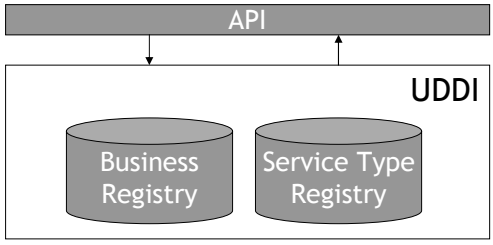
Making services available

- Definition of data types used in the document
- Definition of messages exchanged by operations
- Port types support operations
- Given a port type, the binding defines how to encode messages and how to invoke operations
- Ports implement bindings
- Services are groups of ports

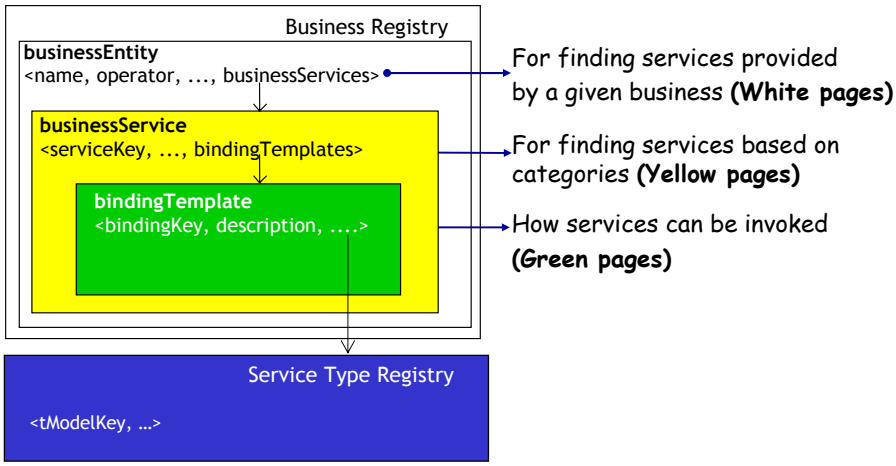


How to find published services?

- UDDI (Universal Description, Discovery and Integration)
  - A globally available directory, for system description and discovery



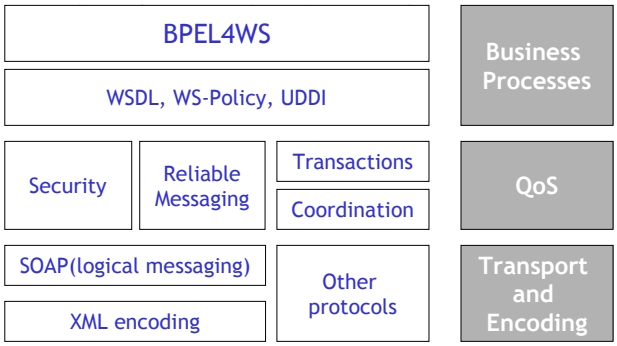
Registry data



Basic approaches to composition

Beyond the basic framework

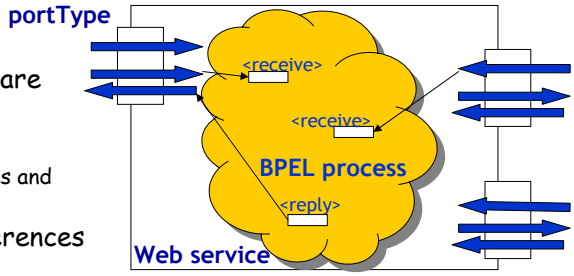
- New requirements for supporting robust business interactions
  - Service composition for business processes



(Curbera et al. – CACM,October 2003)

BPEL4WS
Business Process Execution Language for WS

- Definition of a new Web service as composition (process) of a set of existing services
- The process and the interacting partners are described as WSDL abstract services
  - collection of portTypes and operations
- It assumes that references to specific ports are specified at deployment- or run-time



BPEL4WS – Interacting partners

- BPEL4WS processes involve:
  - **Invoking** other services, through `<invoke>` activities
  - **Receiving invocations** from clients, through `<receive>` **and** `<reply>` activities
- **Partners**: services interacting with a process
  - **Invoked partners** vs **Client partners**
  - Some partners can be **both** client and invoked: e.g., in asynchronous interaction
  - Modeling client partners allows specifying that certain operations may be invoked by certain clients only

BPEL4WS – Partner definition

- Service link type definition:
  - A pair of roles that exchange messages and
  - WSDL port types that the services playing the role must implement
- Partner definition:
  - A name and a service link type
  - The role of the process
  - The role of the partner

```
<partners>
  <partner name="customerP"
   serviceLinkType="orderLT"
    myRole="supplier"
    partnerRole="customer"/>
</partners>
```

- Association of a partner to a specific port
  - Not part of BPEL specification – it is performed at deployment time
  - Also discovered and assigned dinamically at run-time

BPEL4WS – Variables definition

- Variables are used for mantaining the state of the process and for manipulating control data
- **Containers** store data of messages exchangend among partners
  - Input and output parameters in operation invocations, with reference to WSDL message types

```
<containers>
  <container name="order" messageType="OrderRequest"/>
</containers>
```


BPEL4WS - Activities

- Each step in a process is an **activity**
- **Primitive activities:**
  - Invokation of WSDL operations performed by services playing some roles over other services playing some other roles
  - **<invoke>, <receive>, <reply>, <wait>, <assign>, <throw>, <terminate>, <empty>**
- **Structured activities:**
  - Perform orchestration, by defining ordering constraints among (primitive or structured) activities
  - **<sequence>, <switch>, <pick>, <while>, <flow>**

BPEL4WS – Process instance

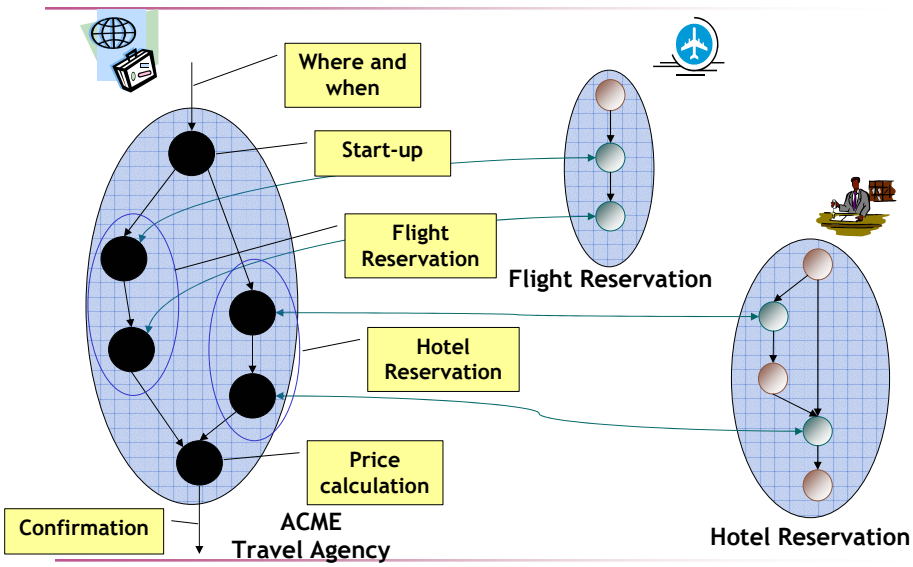
- Process instances are created when a new message arrives for the process
- **key fields** within exchanged messages are used to correlate messages received from a partner to a specific conversation
  - e.g., invoice number in an order fulfillment service
- **Message correlation** allows processes to participate in stateful conversations
  - Messages arriving at a process starting point create a new instance if a matching instance is not available
  - Messages arriving at a non-starting point are accepted only if a matching instance is identified

BPEL4WS - Exceptions and transactions

- Each BPEL activity defines a scope, for which it is possible to specify
  - Fault handlers
    - <catch> element, defining the fault to be managed
    - When the fault occurs, the BPEL engine terminates all the activities in the scope, and executes the activity in the fault handler
  - Event handlers
    - Monitoring an event and executing some actions when the event occurs
    - Always active when its scope is active
  - Compensation handlers
    - Implementing compensating actions for irreversible actions
    - Invoked by fault handlers

Main issues about composition

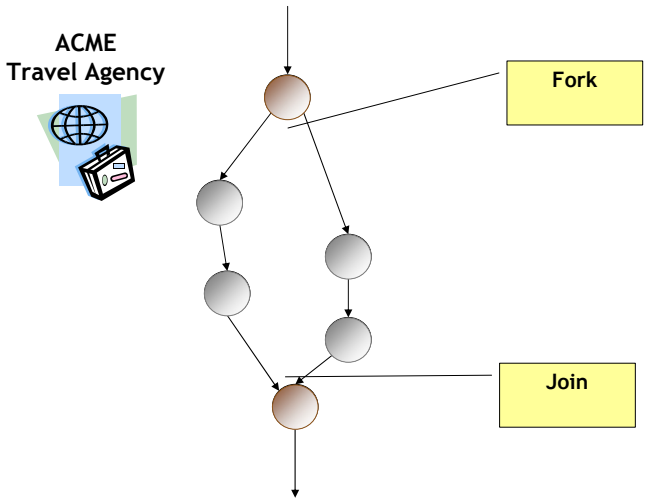
Composition scenario



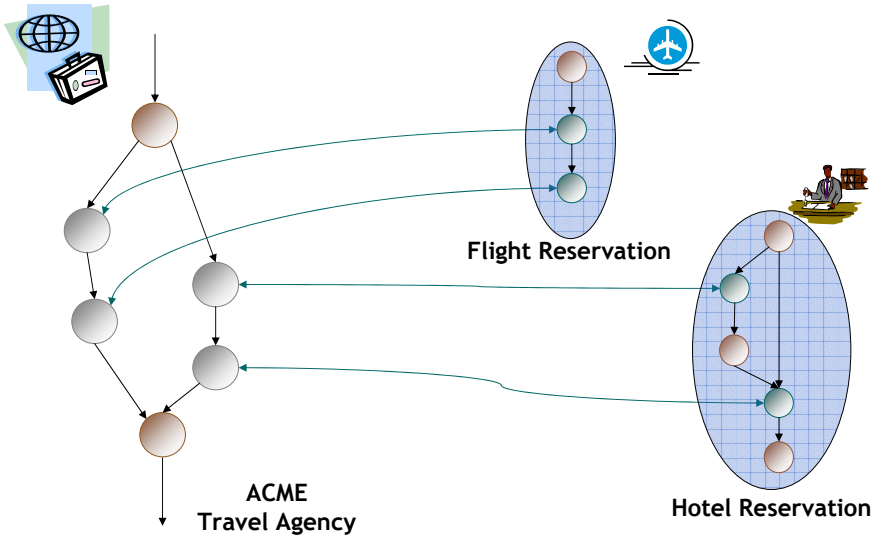
Issues about composition

- How can we define the service-based process?
- How can we associate process activities with available services?
- When do we have to operate such an association?
- How can selected services work correctly?
- How can we ensure correct execution of the process?
- Can we see a service composition as a single service?

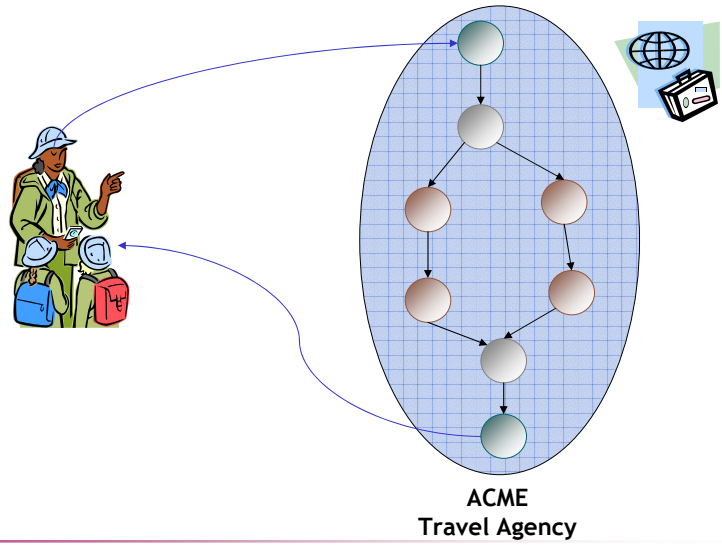
Defining service-based processes



Associating services



Composing service scenarios



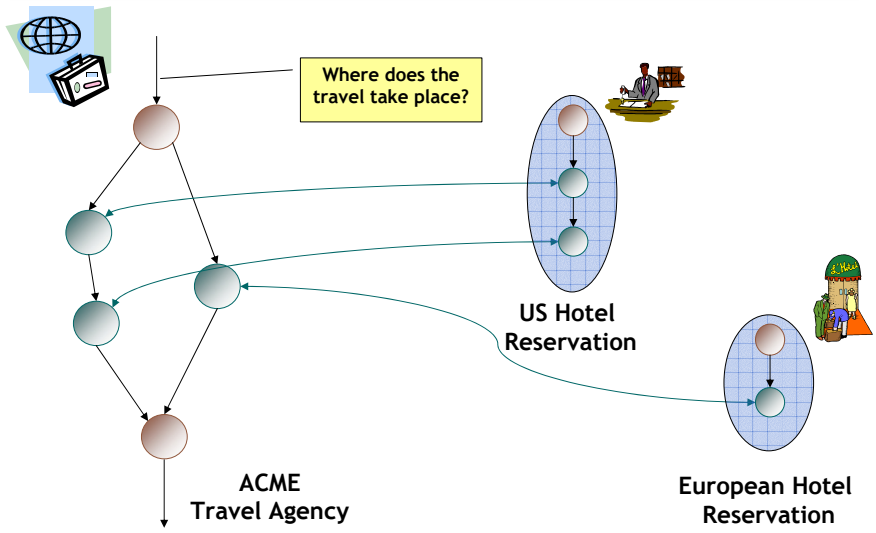
Service-based process definition

- Includes the definition of the application logic of the process
- Some activities could be performed by external applications available as services
  - Interested activities have to define how the service could be invoked and used
- BPEL, WSCI, and … provide models and languages to describe service-based processes

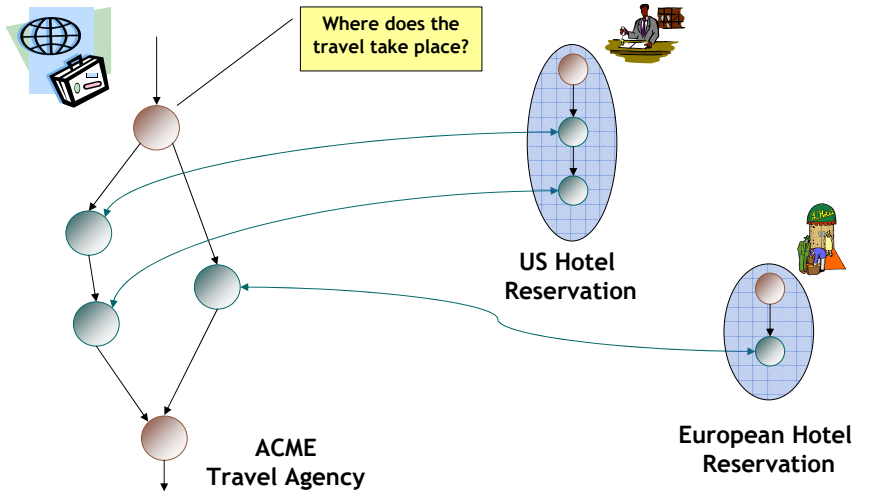
Activities and services

- Two approaches
  - Bottom-up
    - We try to create the requested process out of a set of available services
  - Top-down
    - We start from a well-defined process and try to associate each activity with a suitable service
- Two moments
  - During the process phase (static)
  - At run-time (dynamic)

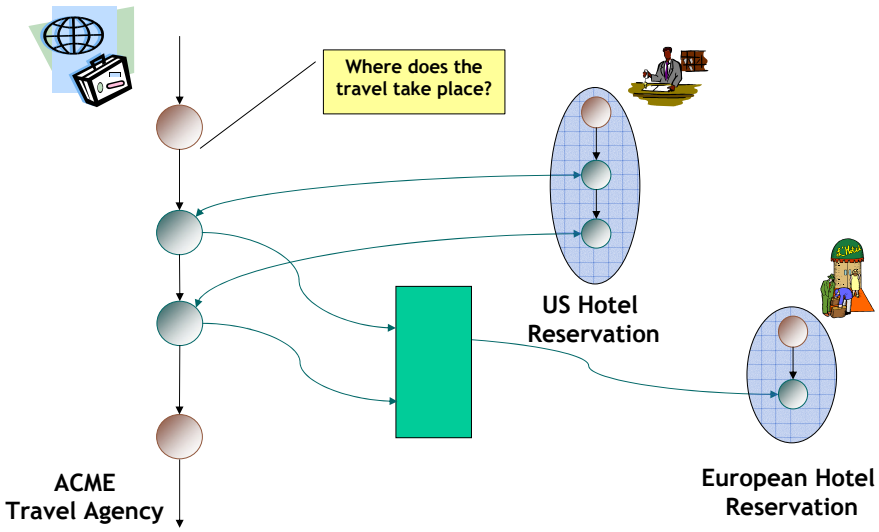
Bottom-up – design time



Bottom-up – run time



Top down

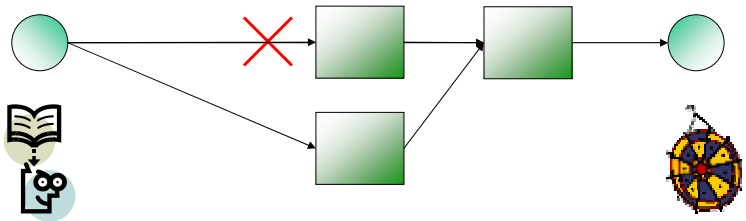


Associations depend on

- Semantic aspects
  - Context in which the service operates
  - Goals of the service
- Syntactic aspects
  - Name of the operations provided
  - Name and order to the requested parameters
  - Order in which the operations have to be invoked

Back-tracking planning

- Is based on approaches used in the agent community
- Algorithm
  - We start looking for the service that matches my goal
  - The input of the selected service is the new goal
  - The composition terminates when the goal and available inputs match



Service directory

- Service composition depends on the discovery phase
- Can be done both at design time and at run-time
  - Given an activity we can try to find the "most suitable" service
  - But, what is the meaning of "most suitable"?
- UDDI provides
  - A way to search for services
  - A business oriented classification
  - A set of standard taxonomies
  - Yellow, white, green pages
- UDDI does not provide
  - Information about quality
  - Content-based discovery

Service models

- Matchmaking mechanism necessarily relies on a specific service models
- A classical model involves
  - Interface
  - Behavior
  - Quality

Interface

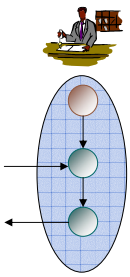
- Defines
  - What the service requires
  - What the service provides
- From a syntactic perspective defines
  - Exchanged data types
  - Order of parameters in the operations
  - Supported protocol
- WSDL specifications match the majority of these requirements

Behavior

- A service has two kinds of behaviors
  - Internal or non-observable behavior
  - External or observable behavior
- During the composition we have only to consider the external behavior and satisfy related constraints
- The service could be composed of a simple request/response invocation or could require a more complex interaction
- If the interface includes different operations the behavior could define the order in which such operations have to be invoked

Behavior

- The behavior is usually modeled as a state machine
- WSCL provide a language to describe such a model



Hotel Reservation

```
<Conversation name="HotelReservationConv"
        InitialInteraction="Start" finalInteraction="End">
  <ConversationInteractions>
    <Interaction InteractionType="Empty" id="Start"/>
    <Interaction InteractionType="Empty" id="End"/>
    <Interaction InteractionType="Receive" id="InfoReq">
       <InBoundXMLDocument id="…" hrefSchema="…">
    </Interaction>
    <Interaction InteractionType="Send" id="HotelRes">
       <OutBoundXMLDocument id="…" hrefSchema="…">
    </Interaction>
  </ConversationInteractions>
  <ConversationTransitions>
    <Transition>
      <SourceInteraction href="Start">
      <DestinationInteraction href="Start">
      …
  </ConversationTransitions>
</Conversation>
```

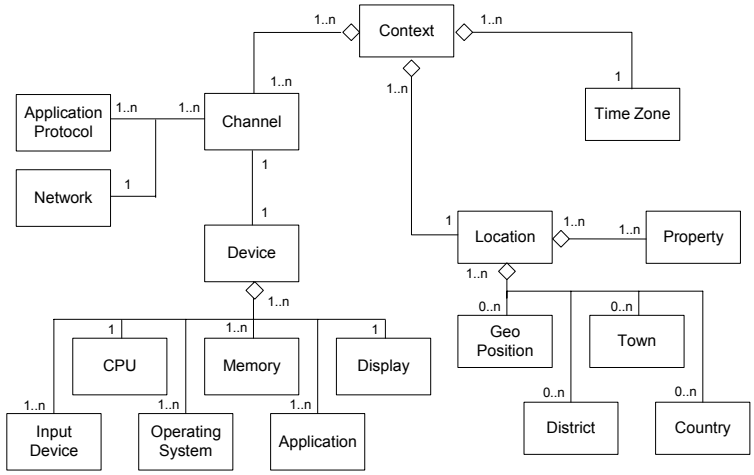
Quality of service

- Each service provides a particular quality
- Even a service-based process must provide a quality
- What is quality for services?
  - Performance issues (what it offers)
  - Economical issues (how much it costs)
  - Resource consuming issues (what it requires)
- Quality parameters are specific to different domains

Enhanced service model

- Besides the classical service model we could consider the context in which the service operates
- The service context could be defined by
  - The channels
  - The time-zone
  - The location

Context



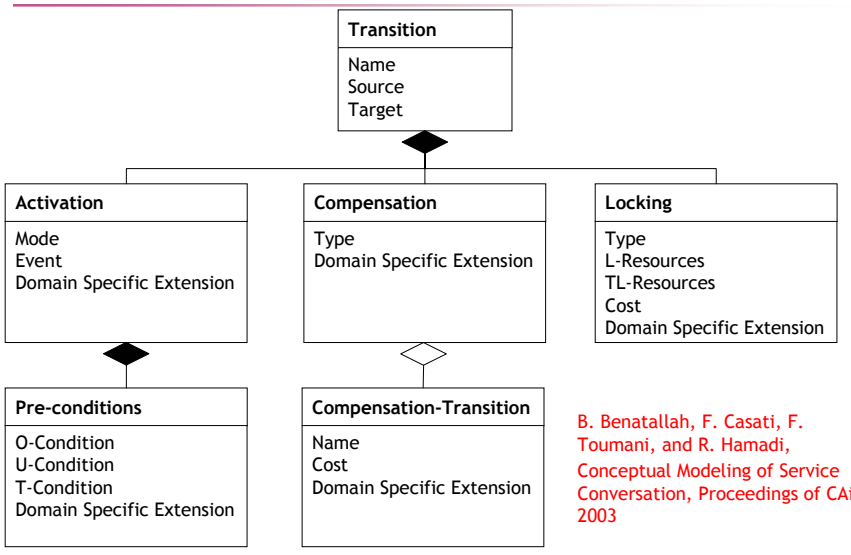
Conversation

- A service execution requires a correct exchange of messages
- The process has to satisfy the constraints on the external behavior
- The state machine model used to describe the service behavior could be also used to describe the process conversation

Conversation aspects

- Transition:
  - Explicit: if it refers to explicit operation invocations
  - Implicit: if it depends on the application logic
  - Timed: if it occurs automatically
- Compensation: similar to roll-back but from a user perspective and not from a database perspective
- Resource Locking: e.g. the flight seat
- Conditions and instance-specific properties: transition may require that certain conditions be verified in order to be enabled

Conversation meta-model



B. Benatallah, F. Casati, F. Toumani, and R. Hamadi, Conceptual Modeling of Service Conversation, Proceedings of CAiSE 2003

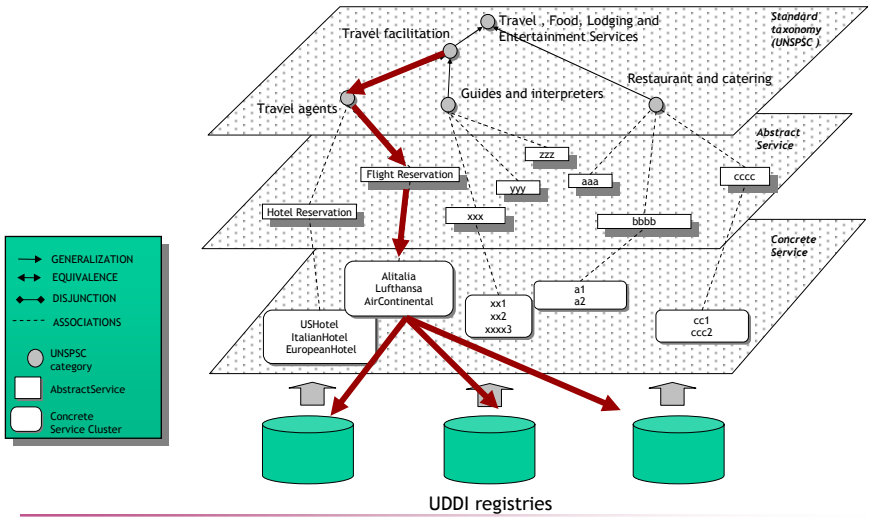
Compatibility classes

- To support run-time selection we need **compatibility classes**
- A compatibility class is a set of services that can be mutually substituted
- A compatibility class is represented by an abstraction of the services that compose the class called abstract service
- The members of the class are called concrete services
- VISPO relies on a set of ontologies, two of them are:
  - Service ontology on which the services are organized in order to create the compatibility class
  - Domain specific ontology which organizes the more relevant concepts in the specific domain

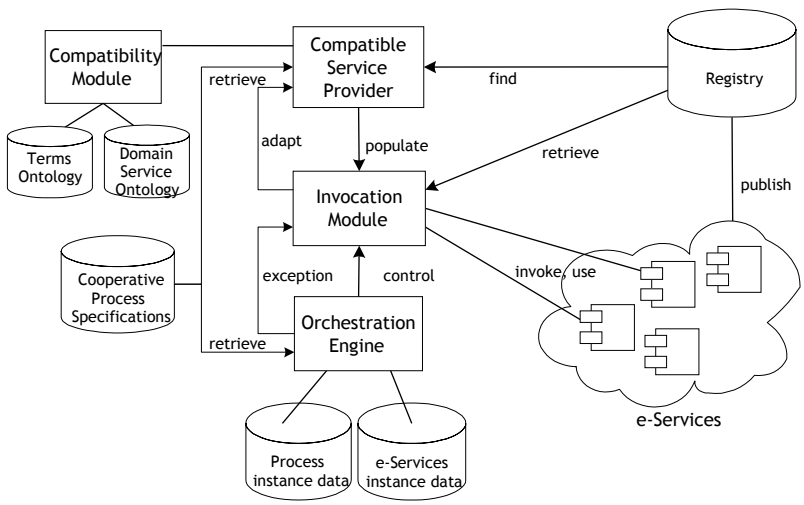
VISPO (Virtual district Internet-based Service PlatfOrm)

- Is an Italian project that aims at proposing an architecture supporting the dynamic execution of cooperative processes
- Cooperative processes are composed of several services that interact to reach a common goal
- VISPO proposes an architecture that enables a run-time selection of the services

Service Ontology



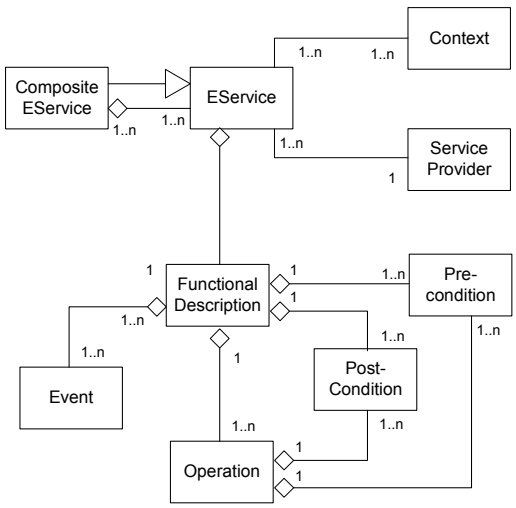
VISPO Architecture



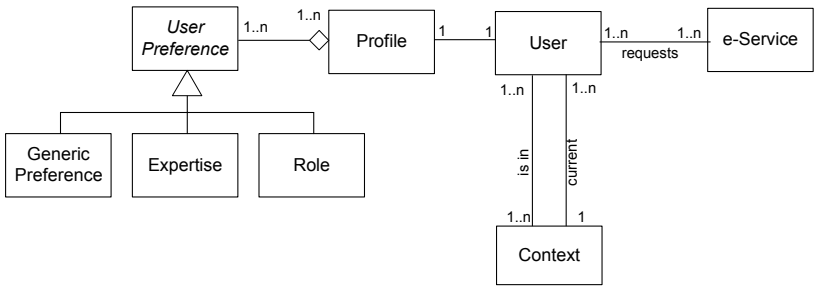
Context-aware composition

- Includes issues about channels and quality of service
- When we consider services we separate:
  - Application logic (the real service)
  - Presentation logic that depends on the used channel
- Two different modeling perspectives:
  - Request perspective (user)
  - Provisioning perspective (provider)
- MAIS (Multi-channel Adaptive Information Systems) Project

MAIS – Service provisioning

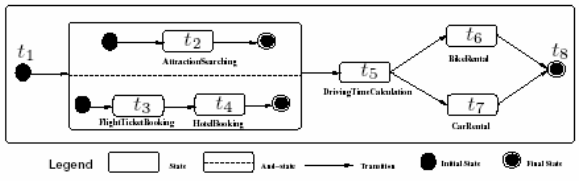


MAIS – Service request



Quality Driven composition

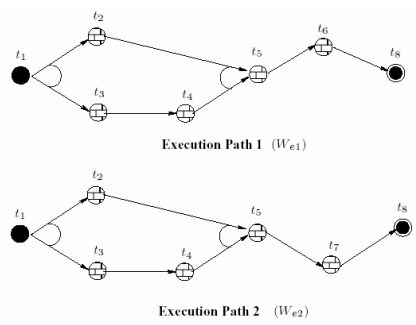
- Proposes a global approach to planning to optimally select component services during the execution of a composite service
- Defines a quality model and a quality-driven service selection



L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, Q.Z. Sheng, Quality Driven Web Service Composition WWW 2003, Budapest

Quality Driven composition

- Given a composite service the execution path and the execution plan are defined
  - Execution paths are sequences of states
  - Execution plans are execution paths in which each activity is performed by a service
- The problem is to select for each activity the service that not only satisfies local requirements but also the global ones



Advanced technologies for composition

Transactional behavior

- Web services
  - Require the same coordination behavior provided by a traditional transaction mechanism to control the operations of an application
  - Also require more flexible transaction models
- non-ACID transactions
  - Collaborations, workflow, etc.
  - Grouping of Web services into applications that
    - need some form of correlation
    - but do not necessarily require transactional behavior

Supporting reliable transactions

- WS-Coordination and WS-Transaction support reliable, transactional coordination of Web Services
- Can be used to extend the BPEL composition model with distributed coordination capability
  - BPEL offers constructs for composing existing Web services
  - WS-Coordination implements coordination types by offering a shared context
  - WS-Transaction defines two coordination types for short- and long-running transactions

WS-Coordination

- A meta-specification, that governs concrete forms of coordination, for example transactions
  - Coordination context: the shared context that is propagated between distributed interacting services (to be included in a SOAP header)
  - Activation service: the service used by clients to create a coordination context
  - Registration service: the service used by participants to register resources (ports) to take part in a coordination protocol

Services

- Activation Service
  - Begins a new activity
  - Specifies the coordination protocols available to the activity
  - Allows the user to specify a relationship between a newly created activity and an existing activity (that is, to establish a subordinate or nested relationship between the activities)
- Registration Service
  - Allows a Web service to register and to select a protocol for the activity:
    - Enrollment and selection allow the Web services involved in the activity to establish the traditional roles of coordinator and participant
    - The registration process identifies the specific protocol used for activity coordination

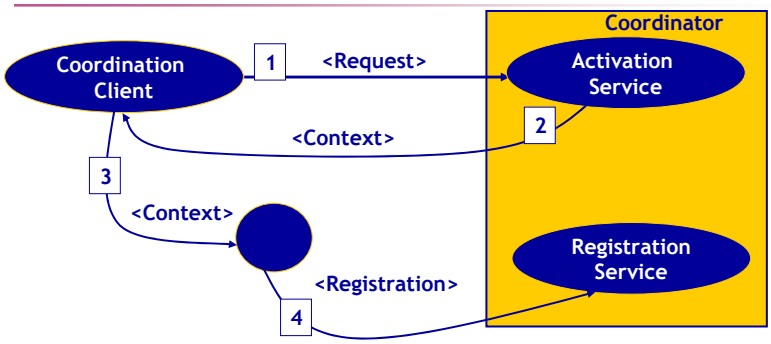
Services

- Coordination Service
  - Controls the activity completion for the registered Web services using the selected coordination protocol (defined in WS-Transaction)
- Operations are identified by role
  - For example, for atomic transactions
    - The coordinator provides an interface to the application to direct completion (that is, commit and rollback)
    - The participant provides an interface to the coordinator to direct agreement (that is, prepare, commit, rollback)

Coordination context

- For each newly created activity, the activation service returns a **CoordinationContext** that contains the following fields:
  - Identifier: a unique name to identify the **CoordinationContext**
  - Expires: an activity timeout value
  - CoordinationType: a set of coordination protocols that describe the supported completed processing behaviors
  - Registration Service: address of the registration service; the service is used to register interest and participation in a coordination protocol for determining the outcome of the activity
  - Extensibility element: provides for optional implementation-specific extensions

Coordinating services



1. **Activating a coordination**
2. **Creating a Coordination Context**
   `<GlobalID, ExpirationDate, RegistrationService Port,... >`
3. **Propagating context** when a service is invoked
4. **Registering an invoked service** to the coordination protocol
   - by means of the Registration Service located at the port specified within the Coordination Context

WS-Transaction

- A standard protocol for long-running transactions (business activities)
- It also provides a set of specifications for short transactions (atomic transactions)
- Builds upon WS-Coordination
  - Assumes the existence of coordinators coordinating transactions

Protocols for atomic transactions

- Handle activities that are short-lived
- Atomic transactions are often referred to as providing a two-phase commitment protocol
  - (atomicity) The transaction scope states that all work must be completed in its entirety
  - (isolation) The results of the activity are available to other users only upon successful completion

Protocols for business transactions

- Handle long-lived activities
- Activities can take much longer to complete
  - (compensation) Mechanisms for fault and compensation handling are introduced to reverse the affects of previously completed business activities
  - (non-atomicity) The results of interim operations are released before the overall activity has completed
  - Minimizes latency of access by other potential users of the resources used by the activity

Other known proposals for service coordination

- … Web Services Acronym Hell (WSAH) …
- WSFL (graph-structured) and Xlang (block-structured): the ancestors of BPEL4WS
- BPML and its "sibling" WCSI
- ebXML and BPSS
- … …
- van der Aalst, Dumas and ter Hofstede provides an analysis and a comparison of these different approaches

WS Choreography Interface (WSCI)

- A language for defining coordination protocols
  - Focuses only on coordination, without introducing a specific protocol
  - Built on top of WSDL
- Extends WSDL interfaces with constraints on the order in which WSDL operations can be executed
  - Several WSCI interfaces associated with the same WSDL interface

WSCI: main extensions

- **Exception Handling**
  - e.g., parts of the protocol are executed in case of a fault message as response to an operation invocation
- **Transactions**
  - Specifies atomicity or compensation for parts of the protocol
  - Does not define how the two properties are implemented by the service (as WS-Transaction does)
- **Correlators**
  - Denote that certain data items of exchanged messages are unique identifiers of the conversation
  - Conversation controllers can associate messages to conversations
- **Time Constraints**
  - Specify that a time interval should elapse between the invocation of two operations

Other horizontal protocols

- SOAP is not intended for guaranteeing QoS (security, reliability, preferences and capability, …)
- Some other protocols have been proposed for facing such requirements
  - WS-Security
  - WS-Policy
  - … …

WS-Security

- A SOAP extension, addressing the need for end-to-end application-level security
- Enables the encryption of portions of SOAP messages
- It defines:
  - A SOAP header block (Security) that carries a signature
  - Which elements are included within the block (e.g.: UsernameToken)
  - How such elements must be processed by the receiving node

WS-Policy

- A flexible and extensible grammar for expressing service requirements, preferences, and capabilities as policy
- Policy: A collection of one or more policy assertions
- Provides a representation of service requirements shared by clients and services in the matchmaking process
  - Does not provide a negotiation solution for Web services!

WS-Policy…

- Does not define domain specific policies – other standards do that
  - For example: it does not provide a language for specifying authentication policy, rather a language for grouping them
- Does not define how policies are attached to services
  - WS-PolicyAttachment describes how to attach policies to WSDL ports, WSDL type definitions and UDDI entities

Policy assertions

- Traditional requirements and capabilities that have externally visible manifestation
  - e.g., authentication scheme, transport protocol selection
- Requirements and capabilities without visible manifestation - yet critical to proper service selection and usage
  - e.g., privacy policy, QoS characteristics
- Five usage qualifiers: **Required**, **Optional**, **Rejected**, **Observed** and **Ignored**

WS-Policy operators

- For specifying complex assertions, made of several alternative acceptable assertions:
  - `<All>`: all of its child elements be satisfied
  - `<ExactlyOne>`: exactly one of its child elements be satisfied
  - `<OneOrMore>`: at least one of its child elements be satisfied
  - `<Policy>`: the same semantics as `<wsp:All>`

WS-Policy: example

```
<wsp:Policy xmlns:wsse="..." xmlns:wsp="...">

   <wsp:ExactlyOne>

        <wsse:SecurityToken wsp:Usage="wsp:Required"
                            wsp:Preference="100">
        <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
        </wsse:SecurityToken>
        <wsse:SecurityToken wsp:Usage="wsp:Required"
                              wsp:Preference="1">
        <wsse:TokenType>wsse:X509v3</wsse:TokenType>
       </wsse:SecurityToken>

   </wsp:ExactlyOne>

</wsp:Policy>
```

Analysis

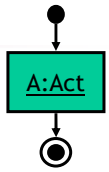
What can we do?

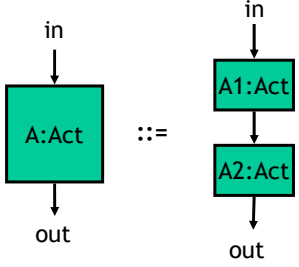
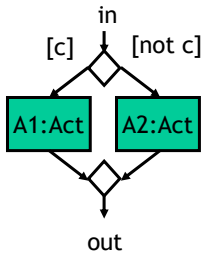
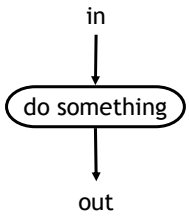
- Static analysis
  - Behavior analysis
    - Activity and actions (WS communication)
    - Consistency (results of communication)
    - Service usage (expected vs. actual)
  - Verification
    - Behavior compatibility (design vs. imp)
    - Composite service traces (poss. paths)
    - Equivalence (processes and actions)
  - Model-checking, Petri nets, automata
- Run-time monitoring
  - Adequacy checks
  - Goal satisfaction
  - Assertions, goal-based approaches

BPEL and CSP

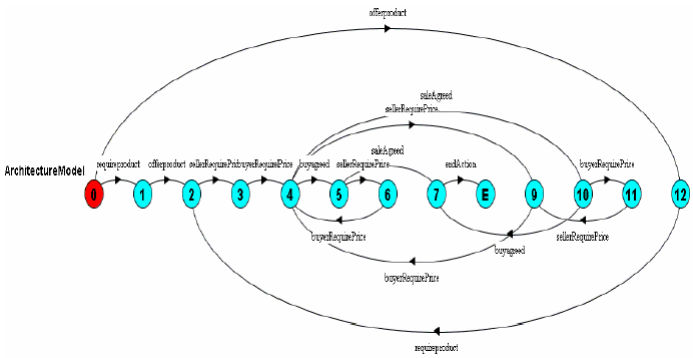
R. Heckel, J. Küster, S. Thöne, H. Voigt
Towards Consistency of Web Service Architectures
Proc. of the 7th World Multiconference on
Systemics, Cybernetics, and Informatics (SCI 2003)
Orlando, Florida USA, July 2003

Source: well-formed activity diagrams



	Target: CSP	*Proc(A)*
		
Proc(A) ::= *Proc(A1)* → *Proc(A2)*	**if** [c] **then** *Proc(A1)* **else** *Proc(A2)*	do something

BPEL and LTS



Model-based Verification of Web Service Compositions
Howard Foster, Sebastian Uchitel, Jeff Magee, Jeff Kramer
ASE 2003

Conclusions

Some Early Adopters

- Amazon, Google (publicly available APIs)
- According to Forrester, 52% of US companies > $1B have rolled out or are rolling out a Web services project
- Yankee Group says that only 12% of all businesses are currently doing Web services

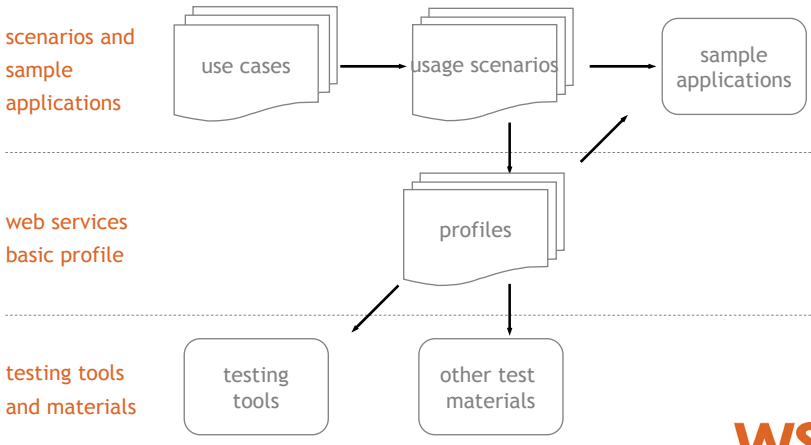
WS-I

- Promote a common, clear definition for Web services
- Integrate specifications from various standards bodies
- Build industry consensus to reduce early adopter risks
- Provide a forum for end users to communicate requirements
- Act as a customer advocate to raise awareness of business requirements
- Offer implementation guidance and best practices
- Deliver tools and sample applications
- Provide a forum for Web services developers to collaborate and share expertise

WS-I: objectives

- Profiles
  - Named groups of specifications at given version levels with conventions about how they work together
- Use cases and usage scenarios
  - Solution scenarios based on customer requirements
- Sample code and applications
- Test suites and supporting materials
  - Conformance testing tools
  - Supporting documentation and white papers

WS-I: deliverables



WS-I basic profile 1.0

- What is a profile?
  - A set of specifications at specific version levels
  - Guidelines and conventions for using the specifications together
- What is the Basic Profile 1.0?
  - SOAP 1.1, WSDL 1.1, UDDI 2.0, XML 1.0, XML Schema and HTTP 1.1
  - More than 200 interoperability issues resolved
  - Conventions around messaging, description, discovery such as:
    - Deprecation of RPC-encoded (use schema as the interoperable type system)
    - Support and guidelines for RPC/lit
    - Unique signatures for input messages
    - Fault and error handling clarifications

Old wine in new bottles?

- … An interesting paper by van der Aalst, Dumas and ter Hofstede
- Very often, the proposed languages recall concepts of WfMSs
- In some cases, composition languages are more expressive than traditional WfMSs
  - Explicit support for basic communication patterns
- Little efforts have been however devoted to evaluating their real advantages and limitations

What to investigate for

Some aspects need to be systematically investigated:

- Expressiveness
- Adequacy
- Orthogonality
- Formal characterization (reachability)

References – part 1

- G. Alonso, F. Casati, H. Kuno, V. Machiraju. Web Services. Concepts, Architectures and Applications. Springer Verlag, 2003.
- WS-Policy. www-106.ibm.com/developerworks/webservices/library/ws-polfram/
- BPEL4WS. www-106.ibm.com/developerworks/webservices/library/ws-bpel/
- W.M.P. van der Aalst, M. Dumas and A.H.M. ter Hofstede. Web service composition languages: Old Wine in New Bottles? Proc, of EuroMicro'03 Conference
- R. Khalaf, F. Leymann, On Web Services Aggregation, Proceedings of VLDB-TES 2003 Workshop.
- B. Benatallah, F. Casati, F. Toumani, and R. Hamadi, Conceptual Modeling of Service Conversation, Proceedings of CAiSE 2003, LNCS 2681, pp-449-467

References – part 2

- F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana, The Next Step in Web Service, Communications of the ACM, October 2003, vol. 46, no. 10
- L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, Q.Z. Sheng, Quality Driven QWeb Service Composition, Proceedings of WWW 2003.
- T. Freund and T. Storey, Transactions in the world of Web services, Part 1: An overview of WS-Transaction WS-Coordination. www-106.ibm.com/developerworks/library/ws-wstx1/
- R. Heckel, J. Küster, S. Thöne, H. Voigt, Towards Consistency of Web Service Architectures
  Proc. of the 7th World Multiconference on Systemics, Cybernetics, and Informatics (SCI 2003). Orlando, Florida USA, July 2003
- H. Foster, S. Uchitel, J. Magee, J. Kramer, Model-based Verification of Web Service Compositions. ASE 2003

Q & A

Instructors

- Luciano Baresi
- Maristella Matera
- Pierluigi Plebani

- Politecnico di Milano
  Dipartimento di Elettronica e Informazione
  piazza L. da Vinci, 32 – I20133 Milano (Italy)
  baresi|matera|plebani@elet.polimi.it

Thank you!