Towards Dynamic Web Services

Luciano Baresi
Dipartimento di Elettronica e Informazione
Politecnico di Milano – Milano (Italy)
baresi|guinea@elet.polimi.it

International Conference on Software Engineering
Shanghai (China) May 20, 2006

Instructor

- Luciano Baresi
  - Associate professor, DEI – Politecnico di Milano (Italy)
  - Software Engineering group
  - Research interests
    - Web services, dynamic software architectures, software specification

2

Overview

- Introduction
- Main research problems
- Common vocabulary
- Publication
- Composition
- Monitoring
- Reaction strategies
- Conclusions
- Open discussion

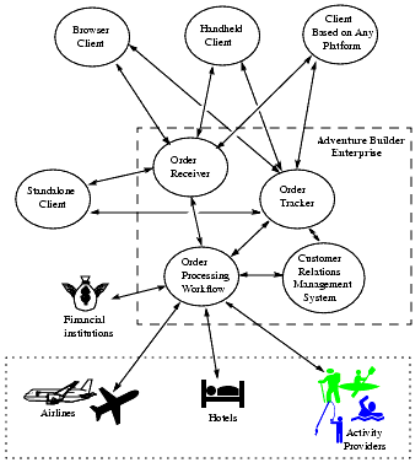
3

Early days vs. nowadays

- Early days
  - Monolithic organizations
  - The world was closed, fixed, static, and centralized
  - No attention to evolution
  - Changes should be avoided
    - they disrupt "normal" flow causing schedule and cost problems
- Nowadays
  - Changes originate in the business world
    - Agile networked organizations
    - Fast organizational responses to rapidly changing requirements
    - Intra and extra organizational changes require continuous changes in the information system
  - Modernization of legacy systems

4

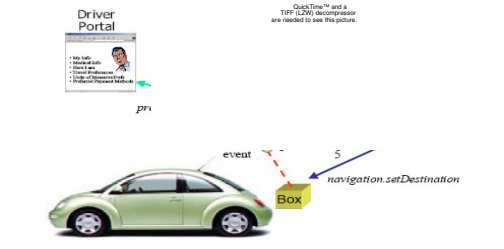
Integration of heterogeneous systems



5

Autonomic systems

- The system needs to "change" according to the context
  - Some parts can "disappear"
  - New functionality can be discovered
  - The system must r
    organize itself a
    time



6

Web services

Service-oriented architectures support dynamic, goal-oriented, opportunistic federations of organizations

"Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. … Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service."

IBM web service tutorial

7

More on services

- Components encapsulating business functions of possible value for others
  - Different level granularity – coarse grained business services vs. fine grained objects
- Services must support explicit contracts to allow independent party access
  - Allow for SLAs that deal not just with functionality
- Services can be the basis for service compositions
  - New value is created through integration and composition
  - New components are recursively created

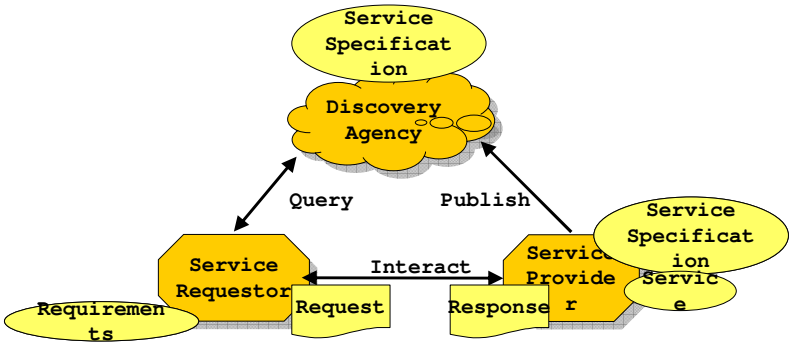
8

Services vs. components

- Both are developed by others
- Components are run in the application's domain
- Services are run in other domains
- Services imply less control and require more trust
- Components chosen and bound together at design/construction time
- Services may be chosen and bound at run-time

9

Main players



10

Service composition

- Service composition is the development task in SoAs
  - Applications are created by combining the basic building blocks provided by other services
  - Service compositions may themselves become services, following a model of recursive service composition
- Composition
  - Requires given functional requirements
  - Is often based on QoS parameters
  - Uses a P2P conversational interaction
  - Implies multi-party interactions
- Many composition models are possible/available

11

Composition and dynamism

- Composition can be defined at
  - Design time (static composition)
    - Services are identified and selected while conceiving the composition
  - Deployment time
    - Services are identified and selected while installing the application
    - Different installations can select different services
  - Run-time (dynamic composition)
    - Services are selected while executing the composition
    - Designers only define abstract processes

12

Possible problems

- Services
  - Do not answer
    - At least, they do not react within given time frames
  - Propagate faulty conditions
    - They send error messages to notify anomalous conditions
  - Violate established contracts
    - Both functional and QoS requirements
      - New versions of supplied services
      - Services cheat on their clients
- New services become available

13

Main research problems

- Publication
  - UDDI is not enough. What about push, pull mechanisms? What about different architectures? What about selective publication (only some persons can see the service)
- Dynamic composition (and dynamic binding)
  - Get a service or a set of services capable of providing a desired functionality with a certain QoS
- Run-time monitoring
  - How can we define what cannot go wrong? How long are we willing to wait to find out if something has gone wrong at run-time?
- Recovery strategies
  - When something does go wrong, what can we do to keep things rolling?

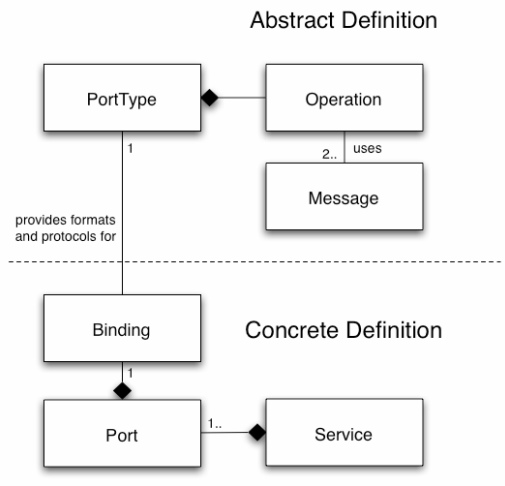
14

Background

WSDL

WSDL

(A purely syntactic interface)



16

Common vocabulary

Semantic Web

Ontolgies

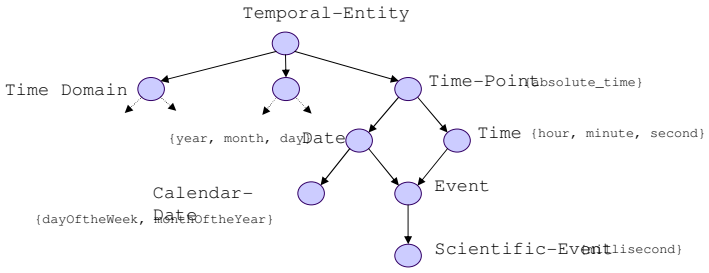
WordNet

Need for a common vocabulary

- Messages
  - Definition of data in input and output messages of web services
  - Suitable annotations (ontologies)
- Functional Semantics
  - Representation of the capabilities of web services
  - Pre and post conditions
- Behavior
  - Representation of the execution flow of operations (in a service)
  - State Machines, Petri nets, activity diagrams etc.
- QoS
  - Description of the operational metrics of web services (SLA)
  - QoS models and QoS ontology for web services

18

Semantic Web Services



- When Web services are semantically described, we may them Semantic Web Services
- An ontology includes a vocabulary of terms, and some specification of their meaning
- The goal is to create an agreed-upon vocabulary and a semantic structure for exchanging information about a domain

19

WordNet
(a lexical database for the English language)

- English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept
- Different relations link the synonym sets
- Create a lexical thesaurus (not a dictionary) which models the lexical organization used by humans
- Approximately 95,000 different word forms

http://wordnet.princeton.edu/

20

Publication

UDDI

Meteor-S

WSMO

UDDI

(Universal Description, Discovery & Integration)

- Access service descriptions, service typologies, and service providers using a well structured data structure
- Abstract from the used technology
- Search for a service using different search keys (taxonomies)
- Search can be done manually or by a program
- Globally available repository

22

UDDI is useful for

Broader B2B 

A mid-sized manufacturer needs to create 400 online relationships with customers, each with their own set of standard and protocols

Smarter Search 

A flower shop in Australia wants to be "plugged in" to every marketplace in the world, but doesn't know how

Easier Aggregation 

A B2B marketplace cannot get catalog data for relevant suppliers in its industry, along with connections to shippers, insurers, etc.

Describe Services

Discover Services

Integrate Them Together

www.uddi.org

23

Registry data

White Pages

- Business Name
- Text Description
  - list of multi-language text strings
- Contact info
  - names, phone numbers, fax numbers, web sites…
- Known Identifiers
  - list of identifiers that a business may be known by - DUNS, Thomas, other

Yellow Pages

- Business categories
  - For example …
    - Industry: NAICS (Industry codes - US Govt.)
    - Product/Services: UN/SPSC (ECMA)
    - Location: Geographical taxonomy
  - Implemented as name-value pairs to allow any valid taxonomy identifier to be attached to the business white page

24

Registry data

Green Pages

- New set of information businesses use to describe how to "do e-commerce" with them
  - Nested model
    - Business processes
    - Service descriptions
    - Binding information
  - Programming/platform/implementation agnostic
  - Services can also be categorized

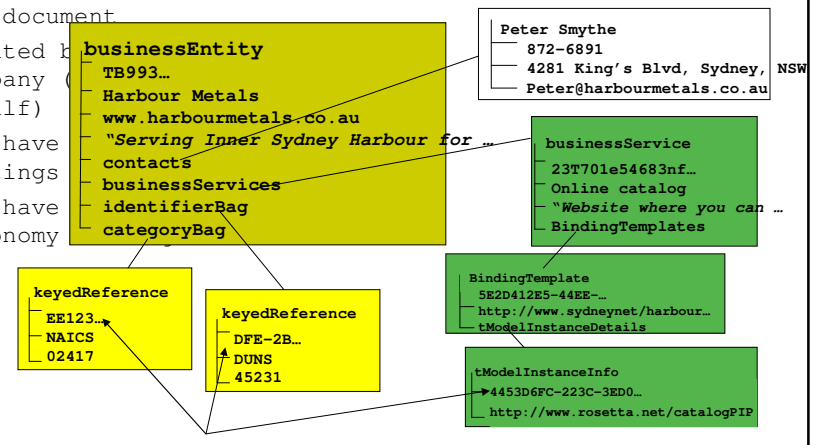
Service Type Registrations

- Pointer to the namespace where service type is described
  - What programmers read to understand how to use the service
- Identifier for who published the service
- Identifier for the service type registration
  - called a tModelKey
  - Used as a signature by web sites that implement those services

25

Business Registration

- XML document
- Created b… company (… behalf)
- Can have … listings
- Can have … taxonomy



26

UDDI v3

- Subscription
  - Synchronous
  - Asynchronous
- Digital signature support for authenticating provenance
- Custody transfer
- Explicit node replication API
- Migration of data between registries
  - UBR as registry of key generators
- UDDI Policy modeling
- UDDI Extensibility
- Additional query modifiers, category groups, internationalization, etc.

27

Next steps
(Conclusions)

- Compatibility changes for
  - SAML, WS-A, WS-I, WS-Policy, BPEL, etc.
- Better external taxonomy support (OWL-S) etc.
- Better searching (e.g. range searching, semantic searching)
- More granular access control (by role, entity, action)
- Life of data (stale data)
- Trustworthiness (integration of trust and identity services)
- Federation (representing registries within registries)
- Different comparisons for category groups

28

METEOR-S

- Interesting example of “semantic” approach

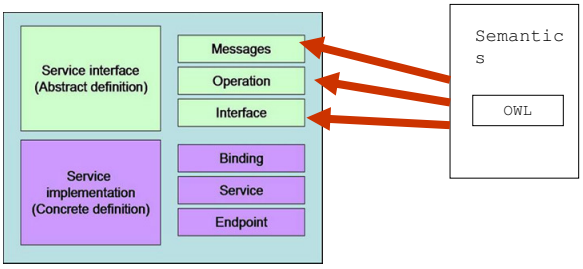
- MWSCF: Semantic Web Process Composition Framework
- MWSDI: Scalable Infrastructure of Registries for Semantic publication and discovery of Web Services
- MWSAF: Semantic Annotation of WSDL (WSDL-S)

lsdis.cs.uga.edu/Projects/METEOR-S/

29

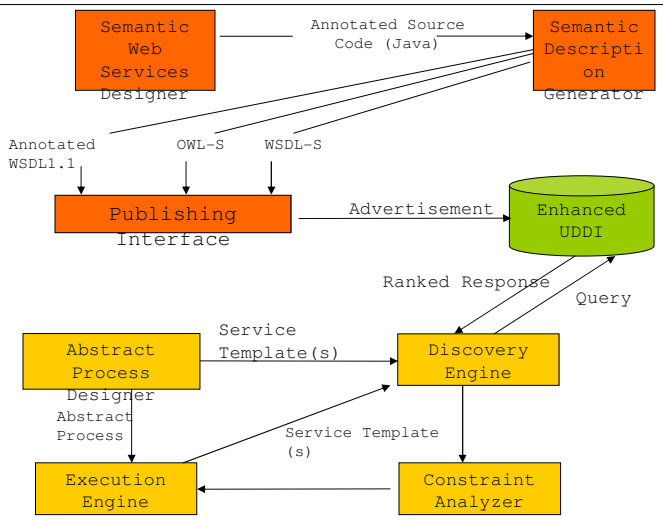
WSDL-S

- Adding semantics inline to WSDL abstract definition
  - Inputs and output messages
    - Annotated with domain concepts
  - Operations
    - Annotated with preconditions and effects
  - Service
    - Interface annotated with category information



30

Architecture



31

WSMO
(Web Service Modeling Ontology)

- Ontologies
  - Provide the formally specified terminology of the information used by all other components
- Goals
  - Objectives that a client may have when consulting a Web Service
- Web services
  - Semantic description of Web Services
    - Capability (functional)
    - Interfaces (usage)
- Connectors
  - Connectors between components with mediation facilities for handling heterogeneities

32

Composition

BPEL
WSCDL
WS-Notification
OWL-S
Research prototypes

BPEL

- Business Process Execution Language for Web Services
- Objective:
  - Behavior model specification of web services throughout a business process
- Syntax based on XML
  - Describes how the process works
  - An orchestration engine guides the process centrally

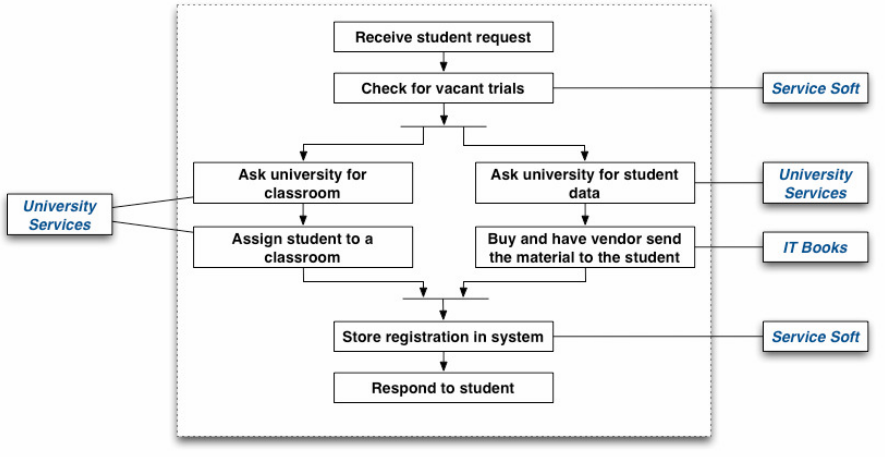
34

BPEL Activities

- Base Activities
  - Receive, Reply, Invoke, Assign
- Workflow activities
  - Sequence, Switch, While, Flow, Pick
- Scopes define a behavior context for nested BPEL activities
- WSDL is used heavily
  - BPEL processes are services themselves and so they provide their own WSDL interface
  - WSDL message types are used within the BPEL process to define the internal variables
  - WSDL message types define the structures of the messages exchanged with the outside world

35

Example



36

BPEL's Limits

- Closed solution
  - In treating abstract BPEL processes only dynamic binding is supported
    - dynamic modification of partnerLinks
- The execution is a possible bottleneck
  - Many research projects tend towards distributed processes
- Limited expressiveness
  - Pushes engine producers to introduce non-standard activities to invoke components that are not web services (for example, using WSIF)

37

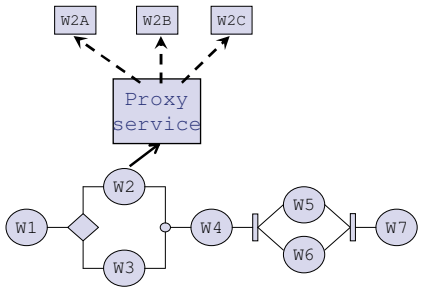
Annotated BPEL

- Nothing but an attempt to improve current technologies
- Abstract BPEL processes are usually annotated with
  - Pre and post conditions on required operations
    - Other semantic information for service discovery
  - Characteristics of foreseen bindings (e.g., duration)
  - QoS parameters to negotiate with service providers
  - Monitoring rules
  - Test data

38

Abstract services in BPEL

- If we redeploy the BPEL process every time
  - No possibility of rebinding
- Replace invocations to concrete services with invocations to proxy services



39

WSCDL

- From a common viewpoint we describe (in XML) the observable and complementary behavior of web services – through message exchanges
- We give a global definition of ordering conditions and constraints under which messages are exchanged:
  - Control flow dependencies
  - Data flow dependencies
  - Message correlations
  - Time constraints
  - Transactional dependencies
- Each participant uses the definition to produce compliant services
  - A Choreography Language is not an "executable business process description language" but should lead to code skeletons and/or abstract processes

40

Dynamic Systems
(Channels)

- A channel is a point of collaboration between parties
- The interesting part is that channels can be passed among parties
  - This provides for static and dynamic message destinations
- Channels can be prepared by a broker service and then one can be chosen at run-time and sent to the choreography under request

41

Dynamic systems
(Guard, Repetition, and Block conditions)

- Guard expression – a boolean XPATH expression that must evaluate to true before the actual work is done (pre-condition candidate)
- Repeat expression – a boolean XPATH expression that is evaluated after correct completion of the work activity and that can cause the work unit to be repeated
  - The guard expression is re-evaluated (post-condition & recovery strategy candidate)
- Block value – true or false. It indicates whether we must wait for certain variables to become available before evaluating the guard expression

42

Conclusions on WSCDL

- No clear distinction between model and exchange syntax
- No mapping to a precise formalism
- It is more about design than implementation
- We have shown some of the good ideas that have arisen in the WS-CDL camp (guards, repetitions, channels, etc.)
- They are insufficient in terms of:
  - expressiveness of the language chosen for defining the guard and repetition expressions (XPATH)
  - degree of clear separation between the collaboration logic and other aspects such as monitoring and recovery

43

Orchestration vs choreography

- Choreography takes global point of view
- Orchestration takes point of view of one participant
- Difficult to map WSCDL to BPEL abstract processes
  - Control flow is orthogonal – should use same control flow activities
- We would like to map design notation (WSCDL) to executable notation (BPEL) for model driven service oriented development environments

44

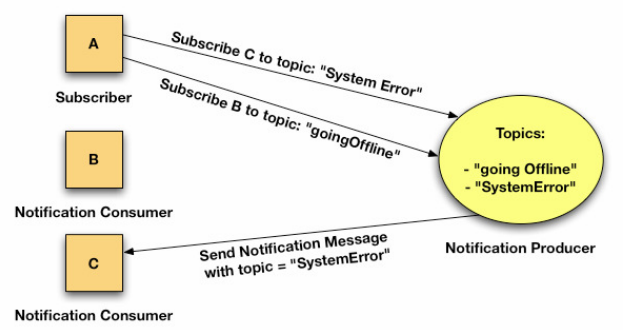
WS-Notification

- A standard for publish-subscribe systems
- Its goal is to standardize the terminology, concepts, and message exchanges for the notification pattern
- It also provides a common language for describing topics

- **Notifications** (can be sent in two ways)
  - Raw notification message to the consumer
    - The producer keeps trace of each message-type, one for each consumer.
  - Notification message data using a Notify message
    - Contains WS-Notification info such as topic + application data (real message)
    - Consumer can accept many Notify messages with one operation in the WSDL – cannot be done with raw notification
    - Producer does not need to make a specific message for each consumer

45

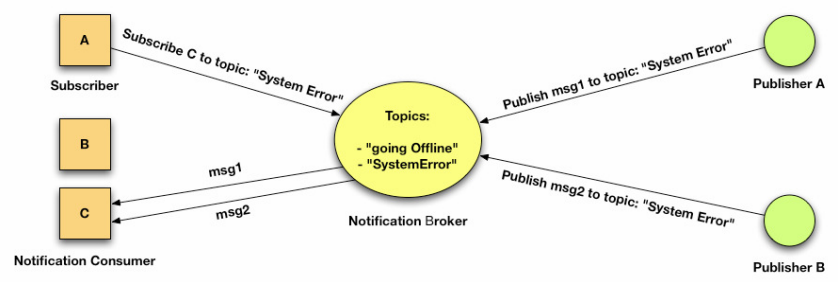
Notification roles



- Defines normative WS interfaces for the following key roles (among others):
  - Notification producer
  - Notification consumer

46

Broker-based solution



- Real publish-subscribe advantages with the use of broker
  - WS-Notification defines the interface for Notification Brokers

47

Conclusions on WS-Notification

- Advantages and disadvantages of publish-subscribe architecture
  - Can be easily used for discovery purposes
  - Does not have the strengths of real publish-subscribe middleware
- No middleware implementations are available at the moment
  - Incomplete implementation under the Apache Web Service (Pubscribe)
- Does not provide any specific help for monitoring

48

OWL-S – the semantic web approach

- OWL-S sets out to enable the following tasks:
  - Automatic Web Service Discovery
  - Automatic Web Service Invocation
  - Automatic Web Service Composition and interoperation
  - Automatic Web Service Monitoring (work hasn't started on this part)
- In order to achieve its goals OWL-S introduces one upper-ontology and several sub-ontologies
- Semantic description of all aspects of web services

49

Self-Serv

- Adopts a model-driven approach and specifies composite services through UML statecharts
- Self-Serv distinguishes among three types of services
  - elementary services (i.e. simple services)
  - composite services
  - service communities
    - containers of substitutable services that provide the description of the desired service without referring to any actual provider
- Web Services are specified by an identifier (SOAP access point), a set of operations (WSDL interface) and a set of attributes
  - Attributes can be advertised, provider-supplied, and community-supplied
- At run-time, communities are responsible for selecting the correct service implementation to best fit a particular user profile in a specific situation

50

Composition as planning
(Traverso et al.)

- Available services are described in OWL-S
  - Rendered as non-deterministic transition systems (OWL-S process model)
- Clients specify main executions to follow
  - Along with side paths that are typically used to resolve exceptional circumstances
  - These goals are expressed using the EAGLE language
    - It permits the definition of conditions on the whole behavior of the composed service and preferences among different sub-goals
- The composition is seen as an (extended) planning problem
  - The plan generation phase exploits the MBP (Model Based Planner)
  - It generates plans that are automata and that can be easily translated into BPEL executable processes

51

analysis

- Hull et al.
  - The description of available services (peers) is given in terms of Mealy machines that exchange messages through predefined communication topologies (channels), and that store messages in bounded queues
  - At each step of a conversation, a peer can send, receive, or consume a message, or perform an empty move to change state
  - The requirements are a desired global behavior in LTL formula
    - The result is a Mealy machine for peers such that their conversations comply with the LTL specification
- Berardi et al.
  - The composition requires
    - Finite state transition systems of the available services
    - A Finite state transition system of target service
  - The composition automatically produces a composed service that realizes the client request
    - Each action of the target service is delegated to one of the available services, according to its behavior

52

AOP-BPEL
(Finkelstein et al.)

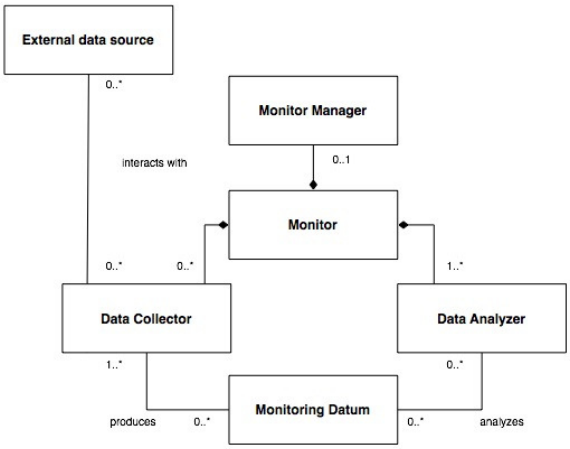
- Based on AOP and the visitor design pattern
- More flexible BPEL engine: an open, extensible, and configurable BPEL interpreter
- Aspects help
  - Easily extend or modify its behavior
  - Select or replace Web Services after deployment time (i.e., dynamic discovery and binding)
  - Plug or unplug features (aspects) in/from the engine on demand
  - Hot fix the workflow (e.g., compose new Web services on demand)
- AOP should also be applied on BPEL processes
  - This is to tackle the problems of hot deployment and bug fixing of Web services

53

Monitoring

Service monitoring
System monitoring
System probing

Conceptual model



55

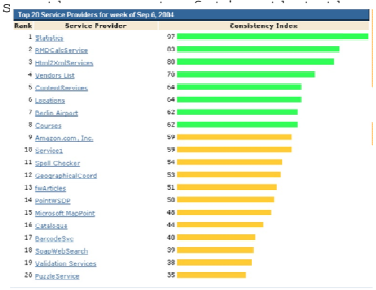
Monitoring levels

- Service execution and monitoring proceed in parallel: monitoring "oversees" the execution of the selected process, and can be seen as
  - **Service monitoring** is the lowest level probing activity and works at the level of the messages that services send and receive
  - **System monitoring** applies to composed services and studies the events generated during the execution of composed services
  - **System probing** inserts special-purpose probes in the process under analysis

56

Service Monitoring
(Message-based probing)

- They work at the level of SOAP messages and compute metrics like
  - The **Consistency Index** expresses the regularity of the service response time
  - The **Availability Index** express
    service
    is available
  - The **Absolute Performance Index** expresses the average response time for services in milliseconds



Web Services Performance Index (Computer Associates)

57

System monitoring
(Spanoudakis et al.)

- Types of deviation beyond classical inconsistencies
  - Inconsistencies w.r.t observed behavior
  - Inconsistencies w.r.t expected behavior
  - Unjustified behavior
- Identification of primitive monitorable events & extraction of behavioral properties from the specification of the composition process of a service-based system
- Specification of additional monitorable requirements in terms of the identified primitive monitorable events (bottom-up approach)
- Deployment of event calculus as the specification framework of the monitorable properties (via an XML-based language defined for this purpose)

58

Example

- Assumptions
  - A car booking service should not report a car as available if it is not:

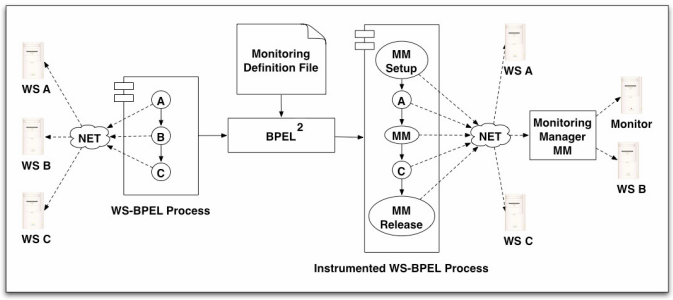
Happens(ir:FindAvail(l,veh),t1,ℜ(t1,t1)) ∧ **Happens**(as:A1(veh),t1, ℜ(t1,t1)) ∧ ¬**HoldsAt**(Available(v),t1−t_u) ⇒ ¬**Initiates**(as:A1(veh), equalTo(veh,v),t1)

  - Between the release and the return of a car key, a car should not be available:

Happens(rc:RelKey(v,c,l),t1,ℜ(t1,t1)) ∧
Happens(rc:RetKey(v,l),t2,ℜ(t2,t2))
∧ (t1 ≤ t3) ∧ (t3 ≤ t2) ⇒ ¬**HoldsAt**(Available(v),t3)

59

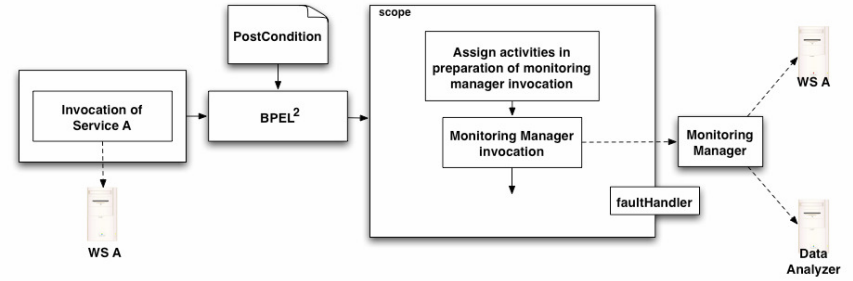
Service probing
(Monitoring Contracts)



- Separation of concerns between business and monitoring logic
- Monitoring defines pre- and post-conditions on invocations (and some other activities)
  - Written in WSCoL (Web Service Constraint Language)
- Static weaving at deployment time
- Analysis is done by external monitors (that also act as proxies)

60

Service probing
(Monitoring Contracts – BPEL^2)



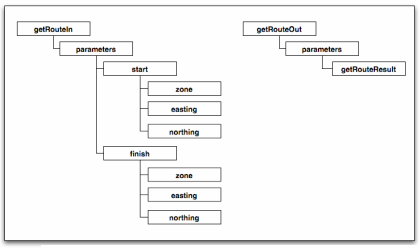
- Assign activities prepare the data to be sent to the monitor
- The Monitoring Manager acts both as a proxy and as a gateway to the data analyzers
- If everything is OK the process proceeds
- If everything is not OK
  - An exception is thrown to the process
  - The exception is caught by an appropriate faultHandler that starts a reaction

61

Assertion examples

Internal variables:

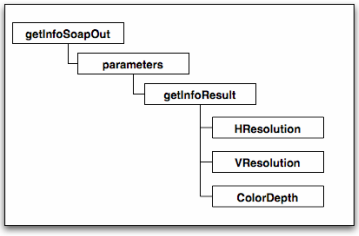
```
($getRouteIn/parameters/start/easting).len
   gth() == 7
($getRouteIn/parameters/start/easting).end
   swith('E')
```



External variables:

```
($returnInt('WSDL', 'getInfo',
   ($getRouteOut/parameters/getRouteResult
   ),

   '//parameters/getInfoResult/Hresolution
   ') <= 80
```



62

Adding dynamicity to the approach

- The actual monitoring activities performed are not set in stone!
- Monitoring parameters act as a knob that, when turned, can dynamically switch on and off certain monitoring activities
- Parameters that can be associated to WSCoL properties:
  - Priority
  - Validity
  - Certified providers

63

SLA-based monitoring

- Done by:
  - IBM WSLA framework
  - WS-Agreement (Cremona)
  - Fabio Casati at HP Labs (does not use WSLA)
- Data is typically collected server-side (not enough? What is true server-side could not be true client-side due to nature of internet)
- Some component is responsible for getting the data which is checked against conditions by a checker

64

Recovery

BPEL (compensation, fault, and event handlers)

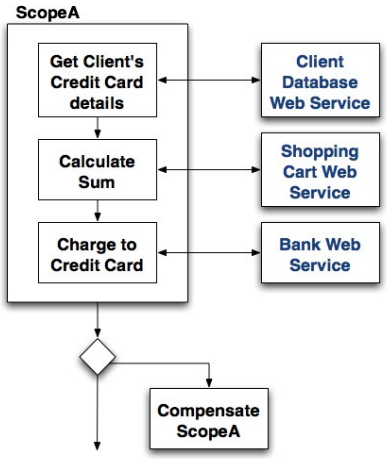
Reaction strategies

Integer programming and genetic algorithms

Rebinding

Compensation handlers

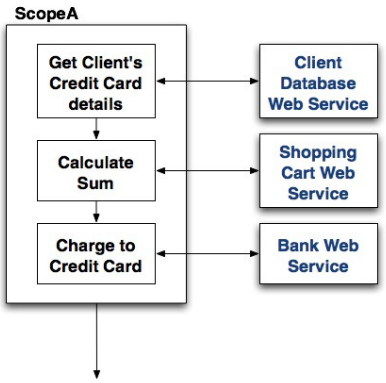
- A compensation handler is attached to a scope
  - It can only start after the scope has been completed correctly
  - It sees a snapshot of all the variables of the scope
  - It cannot update live data
  - It can be used from within fault-handlers or compensation handlers
- default compensation handlers call all compensation handlers for the immediately enclosed scopes in reverse order



66

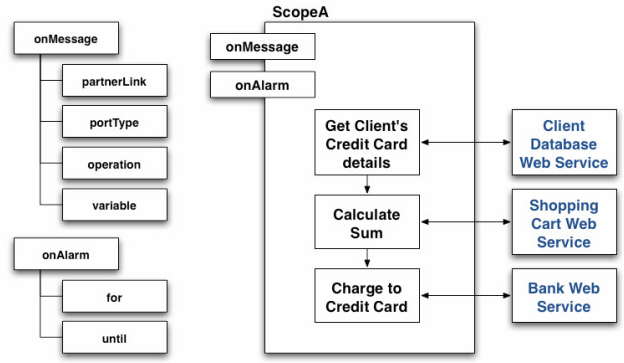
Fault handlers

- It is considered "reverse work" to undo the partial and unsuccessful work of a scope
- If a fault handler is invoked a compensation handler can never be invoked for that same scope
  - the first thing it does is to terminate all the activities contained within the scope



67

Event handlers



- Events are invoked concurrently and can be:
  - Incoming messages (onMessage)
  - Alarms (onAlarm)

68

Recovery actions might …

- Recall the same service
  - If the system (designer) decides it is only a transient failure
- Select a new service with the same characteristics
  - New discovery and negotiation are implied
- Enable new monitors to probe the execution of some services
  - Monitors must be switched on and off at runtime
- Renegotiate some quality parameters
  - If already negotiated contracts do not apply anymore
- Activate a process-specific handler
- Reorganize the (sub)process
  - A web process might be created on-the-fly to provide the same functionality (with similar QoS) as the faulty service

69

Some examples

- MAIS allows the process executor to
  - Select new services according to a given ontology
    - MAIS supplies an extended version of UDDI repository
  - Bindings can expire
- Self-Serv uses linear integer programming to select the best set of services that meet a given requirement
  - Runtime reorganizations are possible if the delta becomes too high
- Canfora et al. (Uni. Sannio) propose the use of genetic algorithms to tackle the QoS-aware composition of services
  - They do not propose ad-hoc monitors, but need monitors to be able to apply the approach dynamically
  - They propose an algorithm to select the sub(process) that must be reorganized

70

Comparison between techniques
(Integer programming and genetic algorithms)

- IP is a widely adopted technique to solve this kind of problem
- GAs better scale up when the number of available concrete services for an abstract service is very high
  - Possible scenarios: widely used services (e.g. hotel booking, search URL, etc.)
- GAs do not pose any limitation on the linearity of the fitness function and of the QoS aggregation formulae
  - Linearization necessary for IP → not to consider for user-defined, domain specific QoS attributes
    - e.g. temperature service may have QoS attributes such as refresh rate, accuracy, etc.
  - Alternative: nonlinear IP → serious scalability problems

71

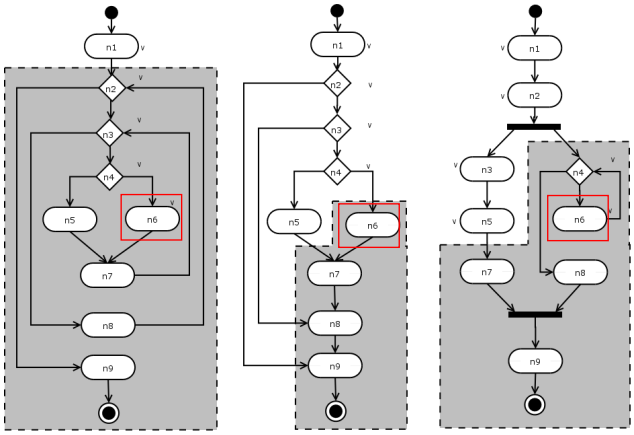
Example reaction strategy
(Rebinding)

- A service may not be available
- or, better services can be available
- QoS values deviate from the estimate
- Unlikely paths are followed
  - Branches unlikely to be executed
  - # of iterations largely different from the estimated value
- This may lead to:
  - Impossibility to continue the execution
  - Constraint violation
  - Poor optimization of the objective function
- Always consider re-binding overhead!

72

Determining the rebinding slice

(Canfora et al.)



Loop Node	Estimated Iterations	Actual Iterations	Current Iterations
n3	k3	k'3	j3
n2	k2	k'2	j2

Remaining n3 iterations: k'3-j3

Remaining n2 iterations: k'2-j2

d)

73

Rebinding may fail…

- No service available for replacement
- No way to recover from constraint violation
  - e.g., timing constraints already violated
- No way to optimize the objective function
- What to do
  - Suspend the execution
    - replace the unavailable service
  - Terminate the execution
    - Nothing can be done
  - Continue anyway
    - Constraints not so hard
    - Try to limit the violation

74

How can we restructure the process?

- Validity of changes
  - Single iteration
  - Single instance
  - Whole process definition
- Problems
  - Data consistency
  - Too heavy solutions (# of services)
- Example
  - Planning techniques (Traverso et al.)
  - Graph transformation systems

75

Our approach

… just to wrap up

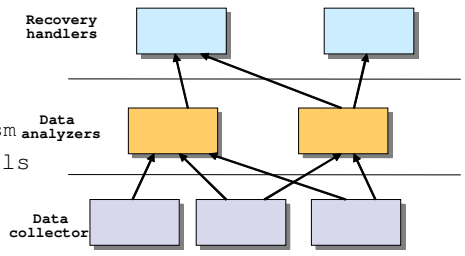
Compositions we can trust

- We need to provide tools and methodologies that can assure high levels of robustness and client perceived trustworthiness in service compositions. We need compositions we can trust
- Design-time testing and validation are not enough
  - Services chosen at design-time can still change during execution!
  - We might decide to choose the services at deployment-time or at run-time…

77

Hierarchy of elements

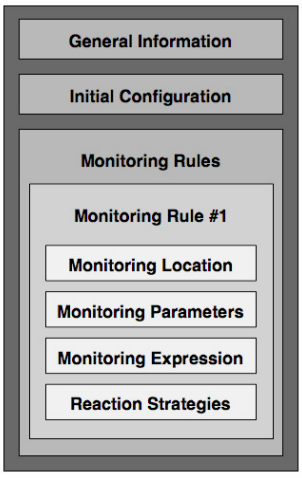
- Two main hypothesis
  - Standard technology
  - Separation of concerns
- Clever annotations
  - WSCoL
  - Flexibility and dynamism
- Two main conceptual tools
  - Proxy-based solution
  - Aspect orientation
- Annotated BPEL



78

Monitoring rules

- Monitoring Location
- Monitoring Expression (WSCoL)
- Monitoring Parameters
  - Priority
  - Validity
  - Certified Providers
- Reaction strategies



79

This means that

- Dynamic monitoring is of key importance
  - The trade-off between performance and timeliness in discovering erroneous situations cannot be fixed, but must depend on when, where and who is running the process
  - Even though our weaving is done at deployment-time, the amount of monitoring is still modifiable at run-time
- Our approach keeps the business logic and the monitoring logic separate
  - This is what permits such an “easy management of the monitoring activities”

80

Recovery actions

- Integration with the other elements/aspects of the framework
  - Retry
  - Rebind
  - Reorganize
  - Change monitoring rule
  - Change monitoring parameters
  - Renegotiate
  - Call handlers provided by services
  - Warn and stop

81

References

Books and Web pages

- Plenty of papers and articles
  - ICSOC, ICWS, ICSE, VLDB, ESEC, ASE, …
- Books
  - … (> 5000 references on Amazon)
  - Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, Donald F. Ferguson . "Web Services Platform Architecture : SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More", Prentice Hall (2005)
- Web pages
  - http://www.w3.org/2002/ws/
  - http://www.oasis-open.org/home/index.php
  - http://www.uddi.org/
  - http://www-130.ibm.com/developerworks/webservices
  - http://java.sun.com/webservices/index.jsp
  - http://msdn.microsoft.com/webservices/
  - http://www.webservices.org/
  - http://ws-i.org/
  - http://www.daml.org/services/

83

Conclusions

Open problems range from business strategies to software processes and service technology

First comments

- We are moving from experiments to real applications
  - Available technology has proven its capabilities on example applications
  - We still need to better assess how it behaves with real systems
- Web services are a good solution in many cases:
  - They provide functional richness and interoperability
  - They require a sacrifice in terms of complexity and performance
- Too many standards or standardization efforts
  - In many cases they are driven by competitive industrial coalitions
  - Many of them are nothing but preliminary ideas
  - Just a few good supporting tools
- Emphasis on more dynamic/decentralized compositions
  - Event-based systems

85

Questions?

Thank you !!!

"Things should be made as simple as possible, but no simpler"

Albert Einstein