Towards Dynamic Web Services

Luciano Baresi

Dipartimento di Elettronica e Informazione

Politecnico di Milano - Milano (Italy)

baresi@elet.polimi.it

ICSE 2006 Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)

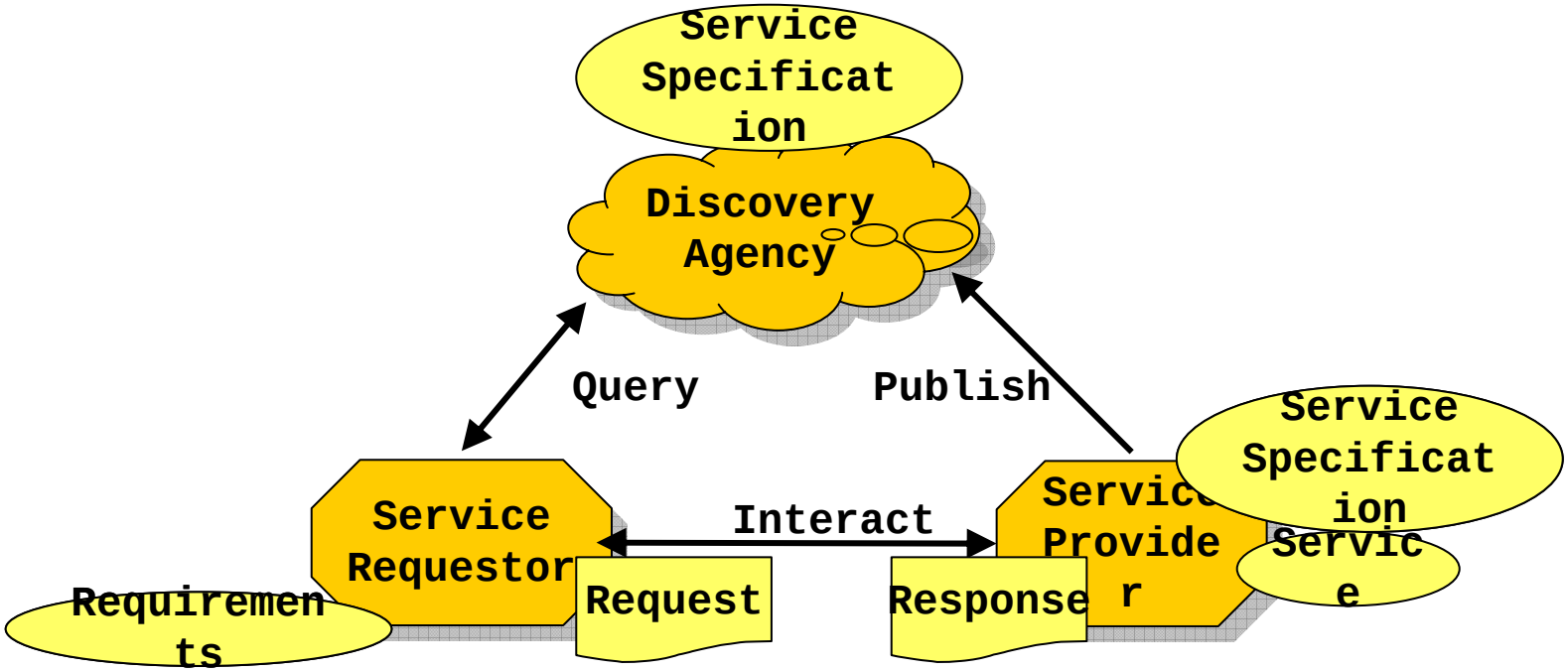
21-22 May 2006, Shanghai, China

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

Main players



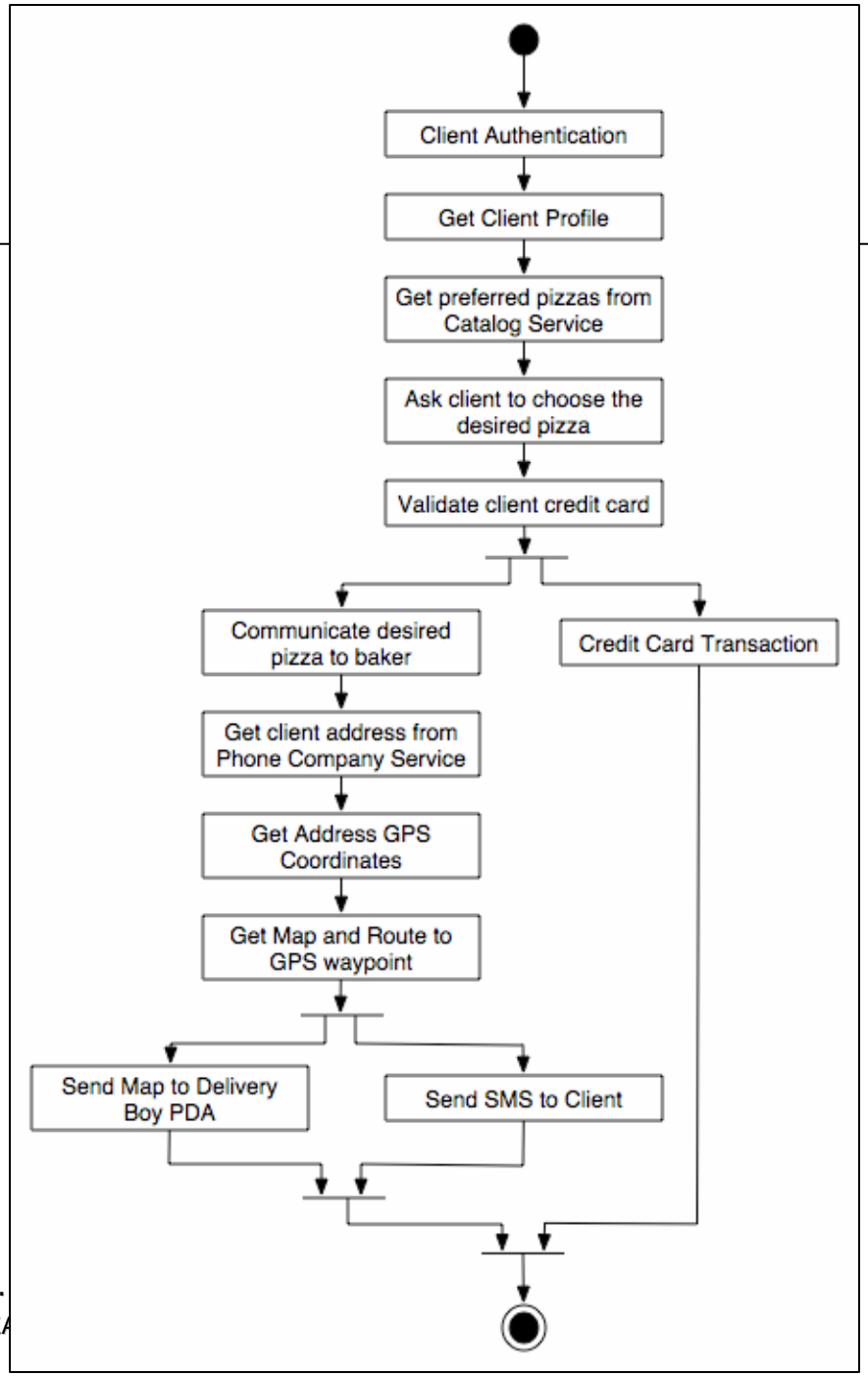
Service composition

- 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

Compositions and dynamism

- Compositions 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

Example


Client Authentication
Get Client Profile
Get preferred pizzas from Catalog Service
Ask client to choose the desired pizza
Validate client credit card
Communicate desired pizza to baker
Credit Card Transaction
Get client address from Phone Company Service
Get Address GPS Coordinates
Get Map and Route to GPS waypoint
Send Map to Delivery Boy PDA
Send SMS to Client

Map service

- The system needs to “change” according to the context
  - Some parts can “disappear”
  - New functionality can be discovered
  - The system must re-organize itself at run-time
  - Autonomic behavior
    - MAPE approach

QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

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

Compositions we can trust

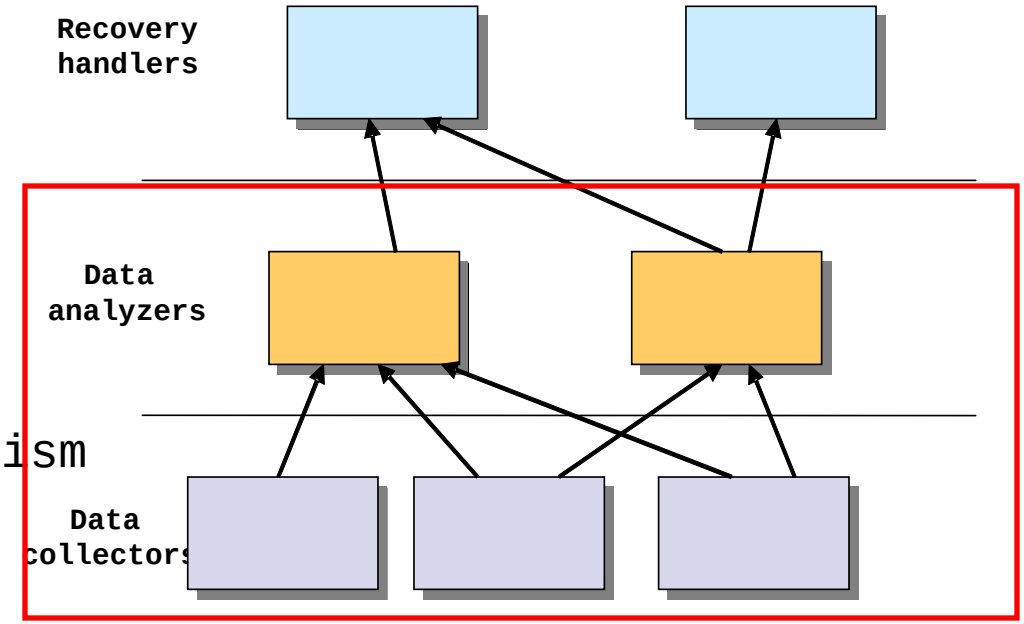
- We need to provide tools and methodologies that can assure high levels of robustness and client perceived trustworthiness in service compositions. We need compositions we can trust
- Design-time testing and validation are not enough
  - Services chosen at design-time can still change during execution!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

Main hypotheses

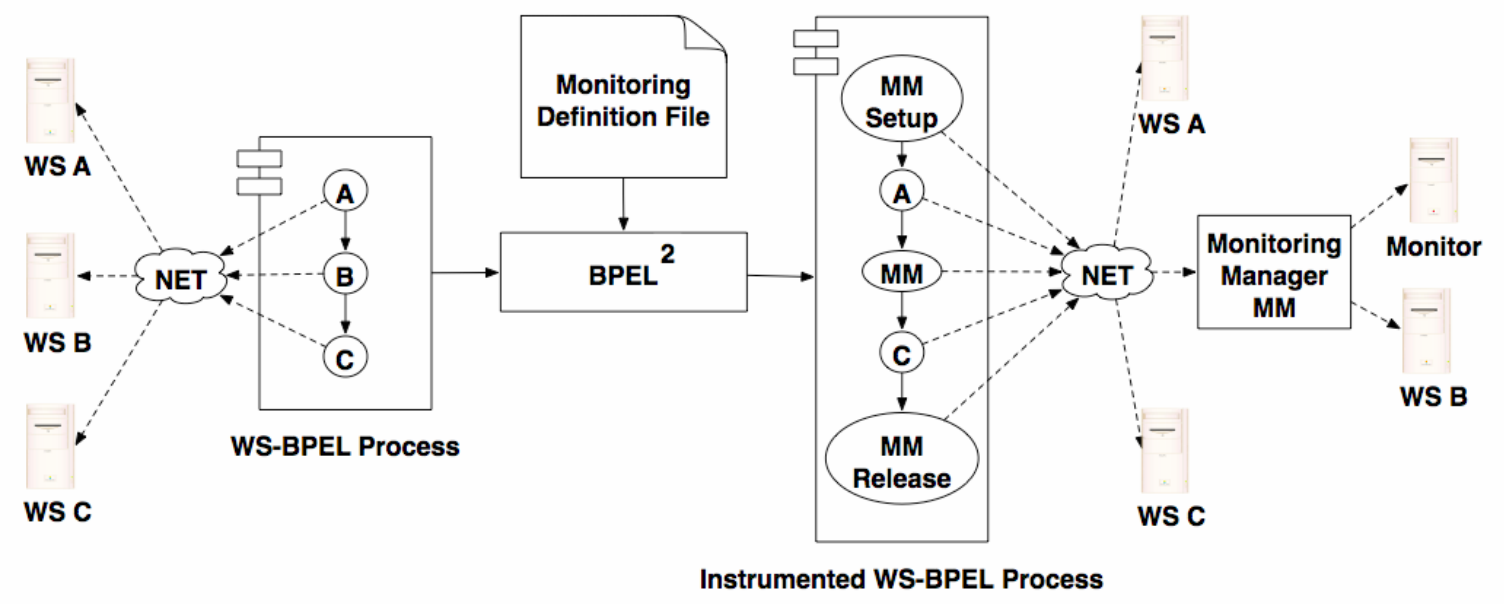
- Standard technology
  - BPEL is the de-facto standard for Web services compositions
  - Many interesting engines are available (for free)
  - Services can be described in many different ways
    - Web Service Description Language
- Separation of concerns
  - No defensive programming
  - No intertwining of business and supervision logics
  - Many possible supervision policies for the same business process

Our solution

- Two main conceptual tools
  - Proxies
  - Aspects
- Clever annotations
  - WSCoL
  - Reaction strategies
- Flexibility and dynamism
- Annotated BPEL



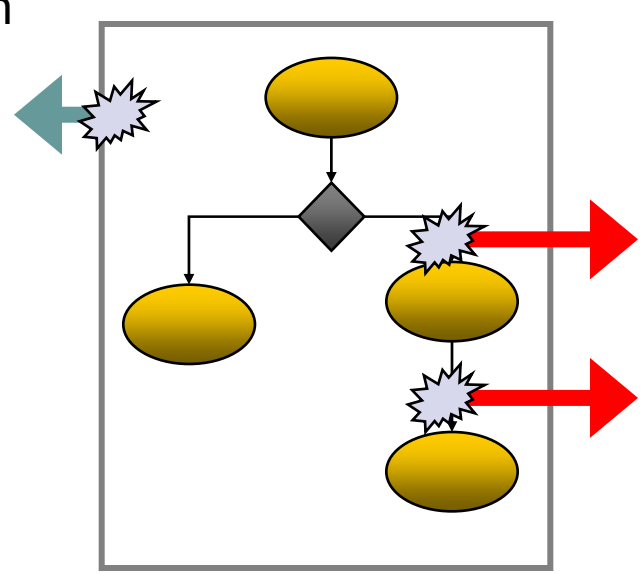
Main steps



- Design the process with any visual tool
- Import the process into the **Visual Annotation Tool** and create the monitoring rules
- Create the instrumented version of the process by using **BPEL^2**
  - We substitute each service invocation for which we want to monitor an assertion with a call to the Monitoring Manager
- Deploy and run it

Loose and strict monitoring

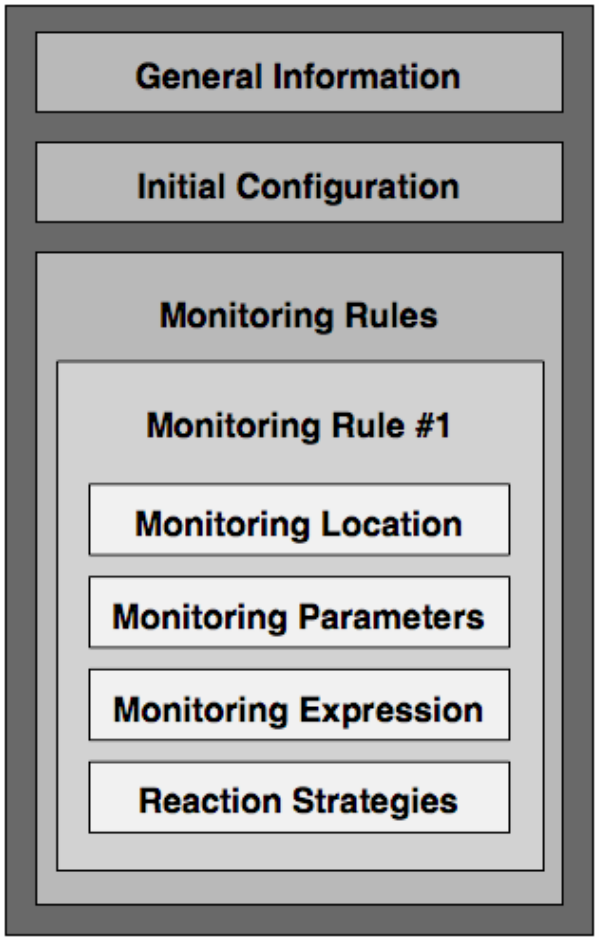
- Loose monitoring
  - Runs in parallel with main execution
- Strict monitoring
  - Intertwined with main execution



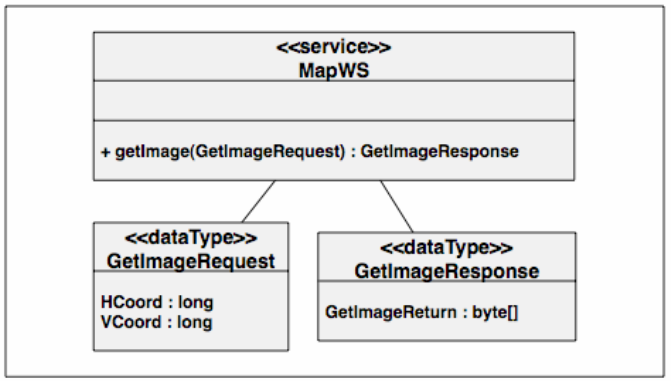
Monitoring rules

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

- Internal variables



Example monitoring rule



- Location
  - type = "post-condition"
  - path = "//:invoke[@partnerLink="lns:MapServicePartnerLink" and @operation="getMap"]"
- Parameters
  - priority = 3
  - validity = always
  - certifiedProviders = <empty>
- Expression
  - \returnInt(wsdlLoc, getResolution, 'image', GetImageResponse.GetImageReturn, HResolution) <= 80 && \returnInt(wsdlLoc, getResolution, 'image', GetImageResponse.GetImageReturn, VResolution) <= 60;
- Reaction strategy

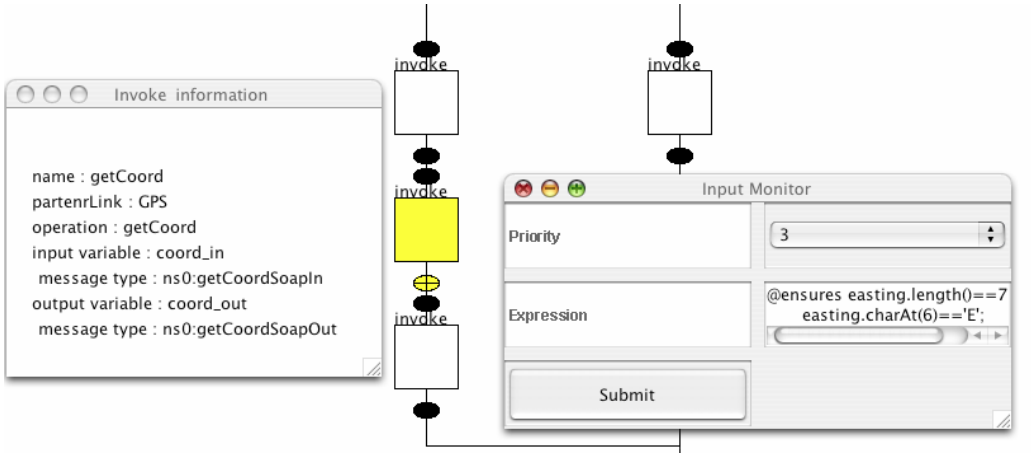
  If (!retry()) then

  If (!rebind()) then reorganize()

Reaction strategies

- Built-in solutions
  - Retry
  - Change monitoring rule
  - Change monitoring parameters
  - Call handlers provided by services
  - Warn and stop
- Third-party solutions
  - Rebind
  - Reorganize
  - Renegotiate
  - …
- Reaction strategies as on-the-fly BPEL processes

Visual annotation tool



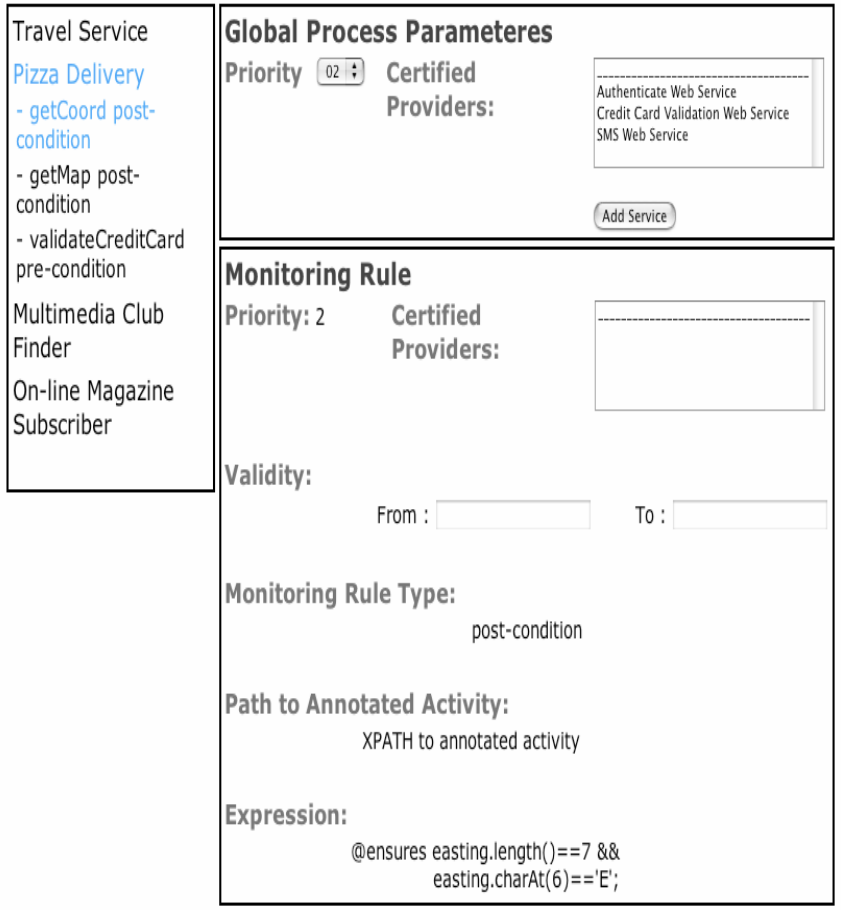
- It shows a stripped down version of the process
  - It only concentrates the structure of the process and its *invoke* activities
- Multiple windows help us gain information about the process

BPEL^2

- BPEL^2 substitutes each service invocation for which we want to monitor an assertion with a call to the Monitoring Manager
- It also adds:
  - An initial call to the monitoring manager for setup of the manager itself
    - Useful for sending less info later on
    - A heavy initial step is preferable to slowing down the execution
    - A ending call to the monitoring manager
    - Any configuration space on the monitoring manager is released

Dynamic monitoring

- At run-time we can access the monitoring manager and modify the monitoring parameters that are associated to the process in execution
- This changes the level of monitoring that is performed!



Conclusions

- Dynamic Web services are 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”
- Reaction strategies seem to be promising
  - We need further experiments

Future directions

- Second release of the set of tools
- More emphasis on reaction strategies
  - A language to compose the atoms
  - A means to analyze possible alternatives and choose among them
- Further experiments
- Further ideas
  - BPMN instead of BPEL
  - BPEL^2 as dynamic weaver (dynamic AOP techniques)
  - Integration of the approach into a BPEL execution engine
  - Further data collectors and analyzers
    - Different languages for describing different kinds of properties to be monitored

Credits

- People
  - Carlo Ghezzi
  - Sam Guinea
  - Marco Plebani
  - Master students at Politecnico

- Projects
  - SeCSE (EC project)
  - Cascadas (EC project)
  - ArtDecò (national project)
  - Discorso (national project)

Questions?

Thank you !!!

“Things should be made as simple as possible, but no simpler”

Albert Einstein