

# Self-adaptive Service Compositions

**Luciano Baresi**

Politecnico di Milano

Dipartimento di Elettronica e Informazione

Millano (Italy)

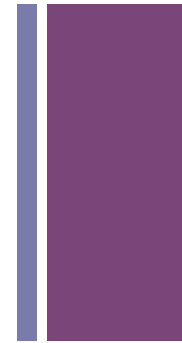
# + Software services

- Software components that can be used but are not owned
- Service-oriented applications constructed by composing and configuring software services
  - Often provided by “third parties”
  - Services are not under control

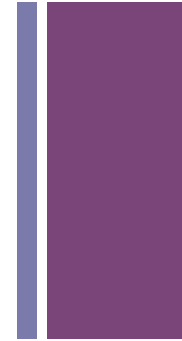
(Traverso 2006)

# + 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 functional requirements
  - Is often based on QoS parameters
  - Implies multi-party interactions
- Many composition models are possible/available

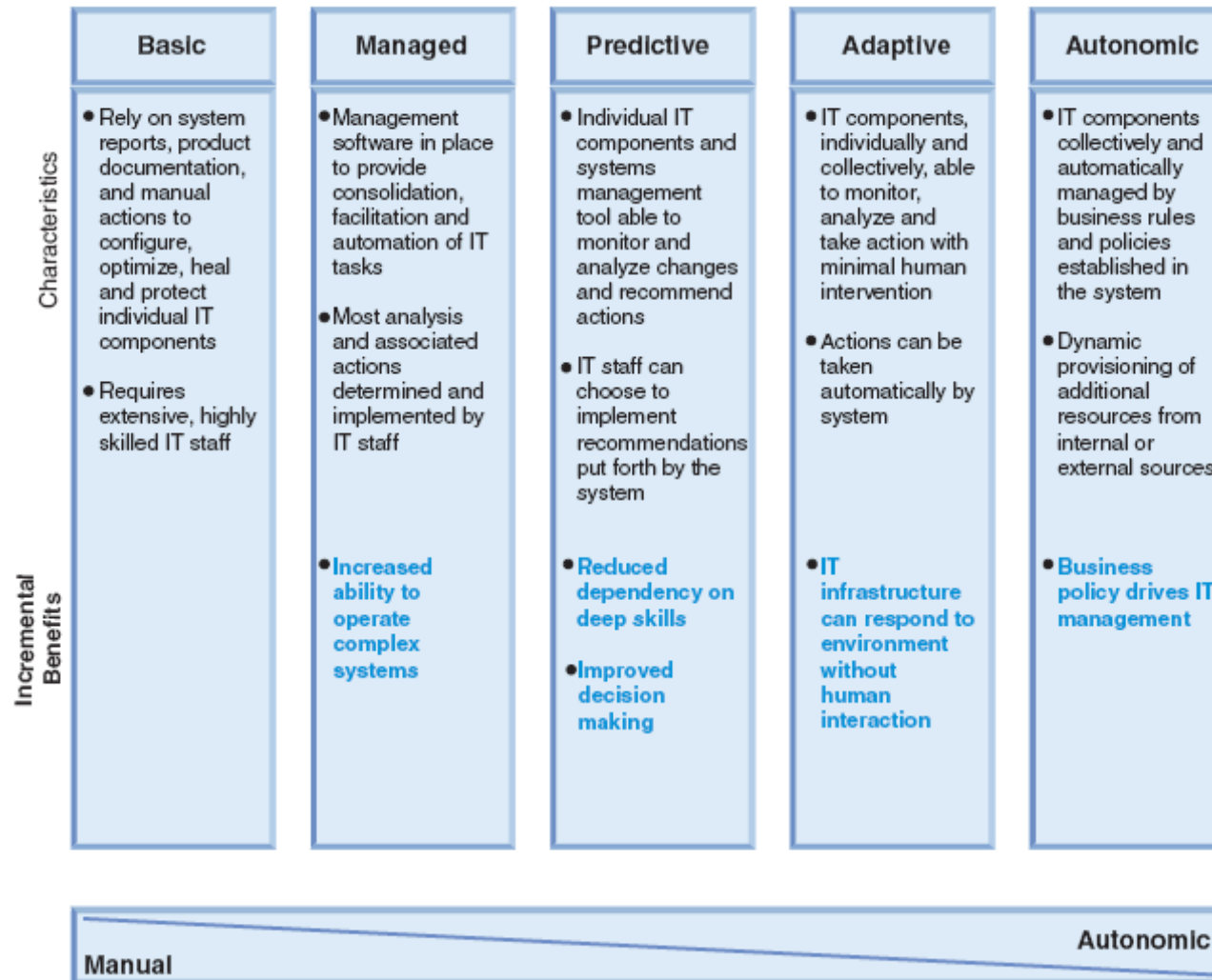


# + Possible problems



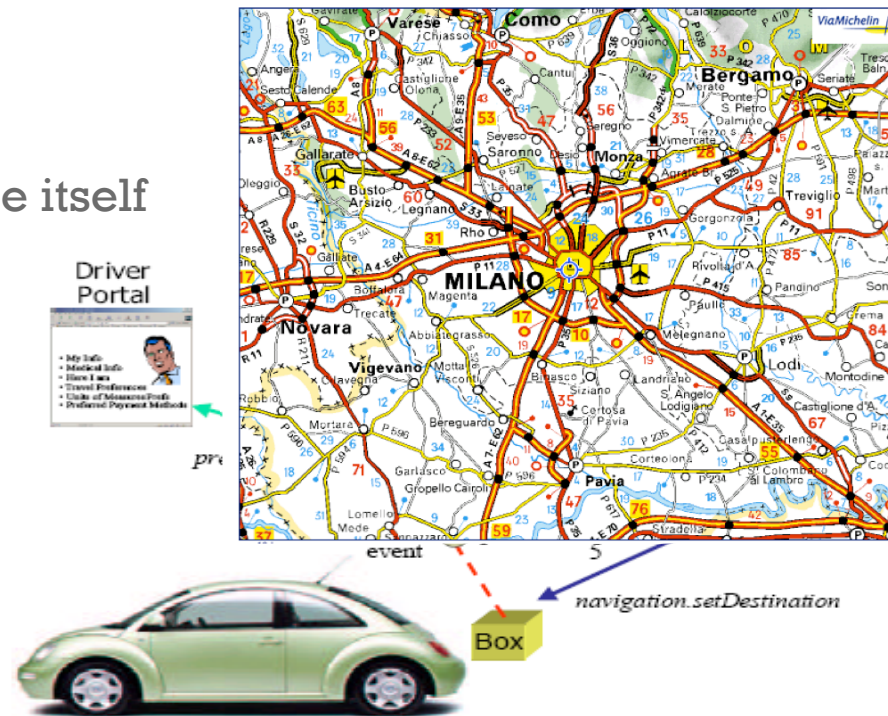
- Services
  - Do not answer
    - At least, they do not react within given time frames
  - Propagate faulty conditions
    - They send error messages to notify anomalous conditions
  - Violate established contracts
    - Both functional and QoS requirements
      - New versions of supplied services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

# + Maturity and sophistication

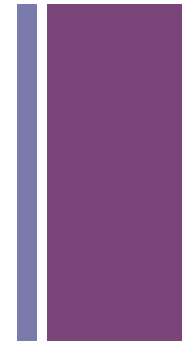
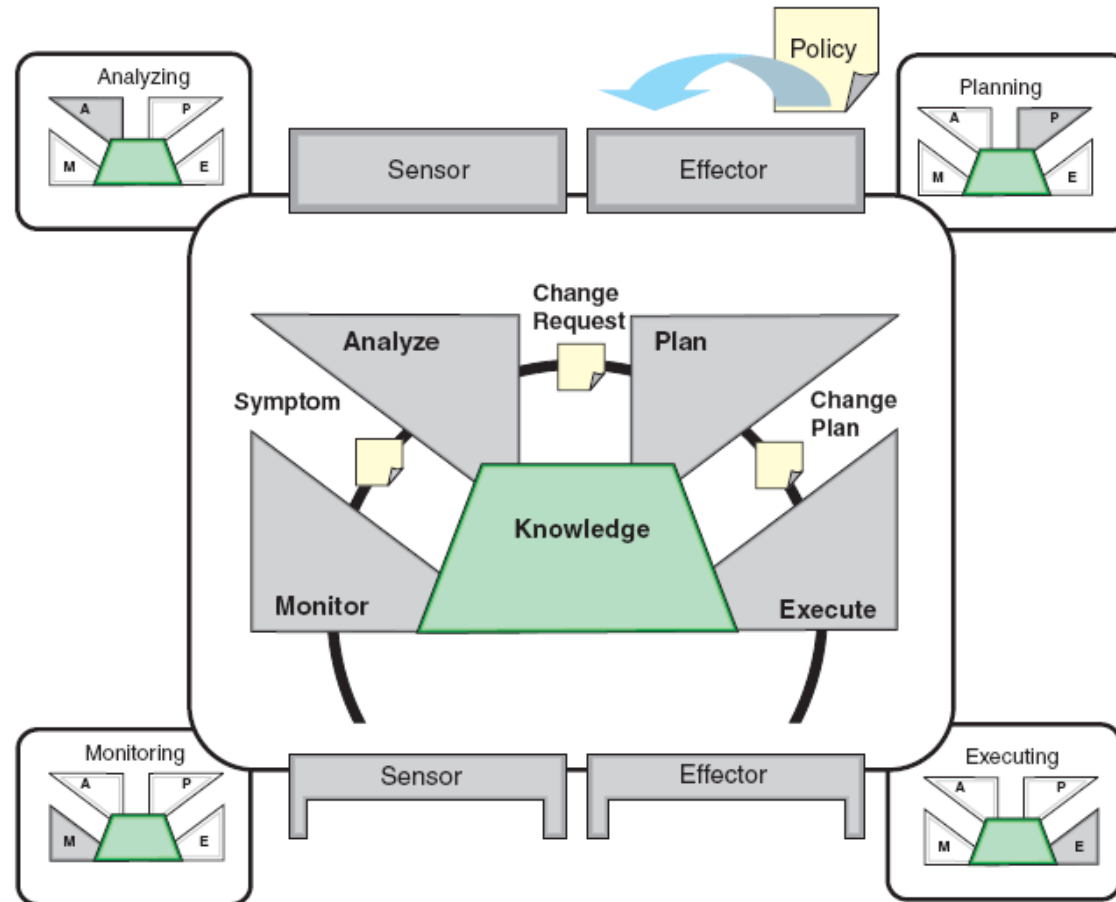


# + Self-adaptive systems

- 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

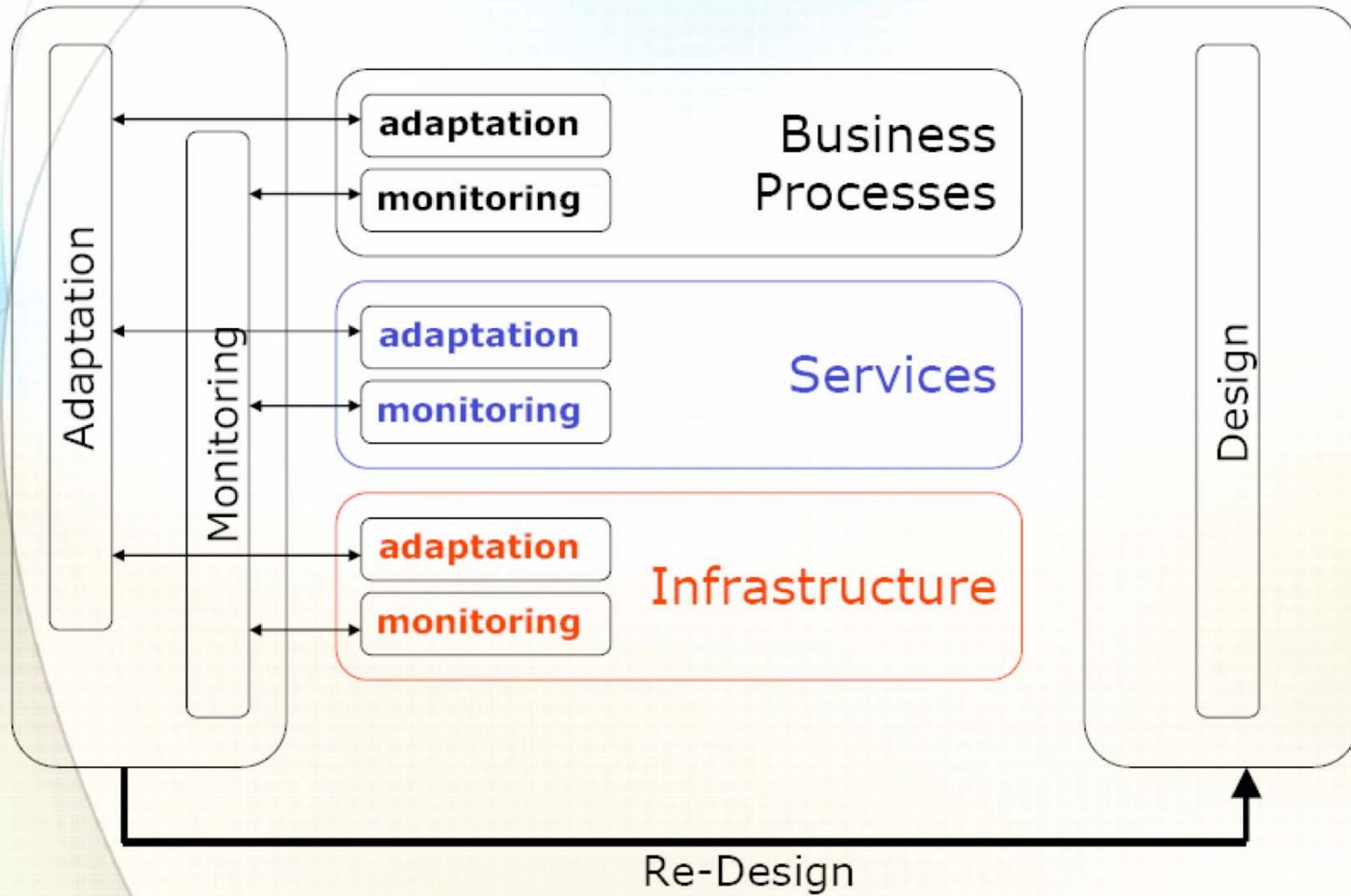


# + Autonomic systems

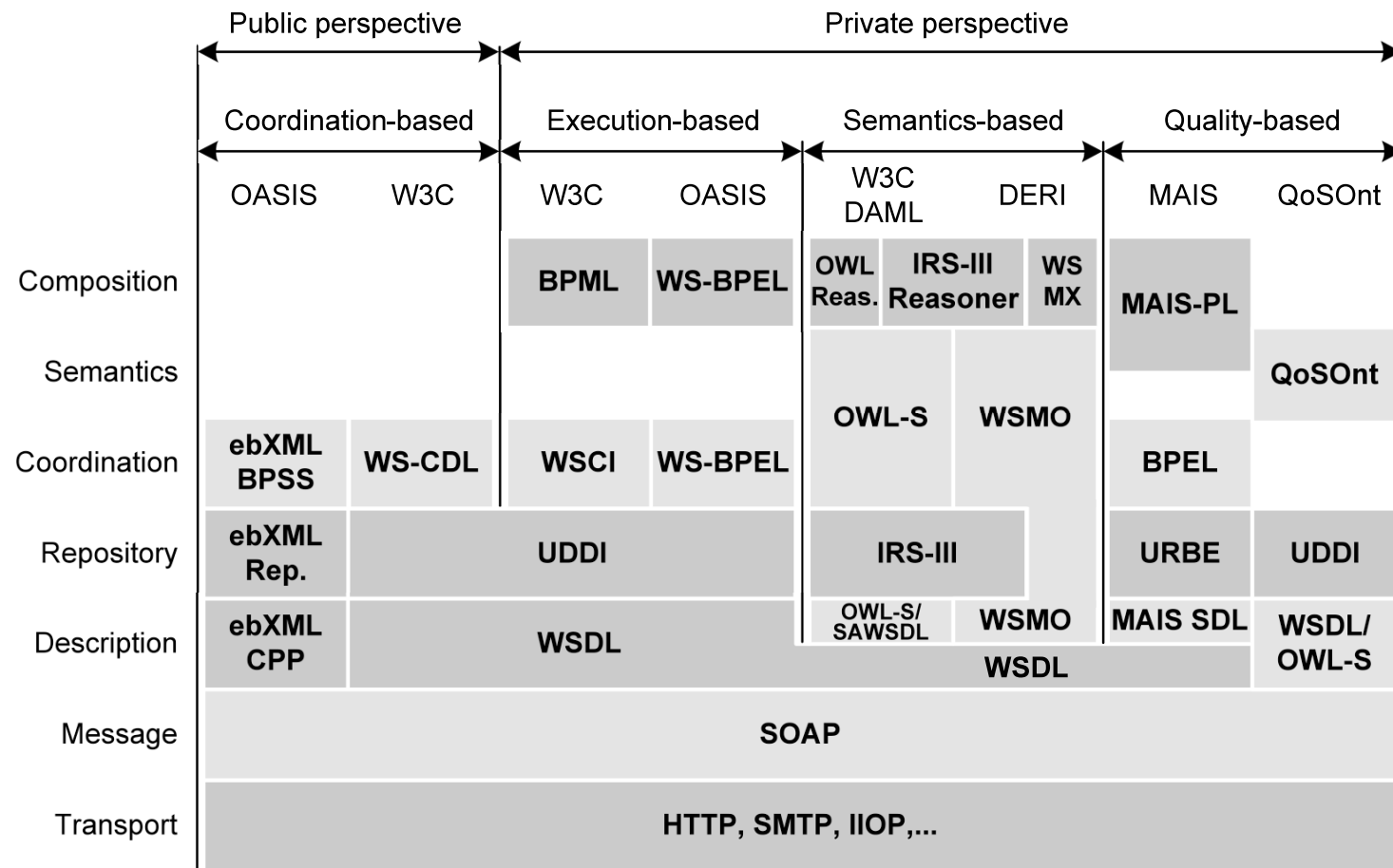




# A Framework for Monitoring & Adaptation



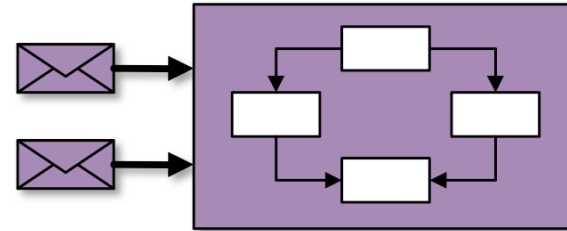
# + Some proposals



# + Composition models

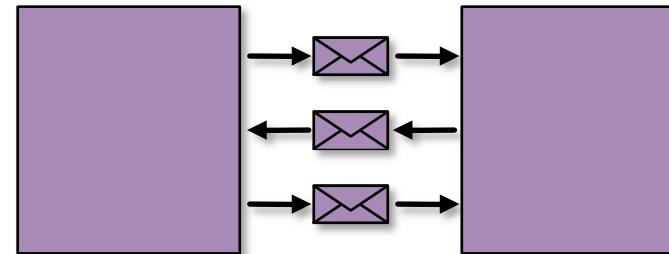
## ■ Orchestration

- Intra-process
- Process controlled by one party



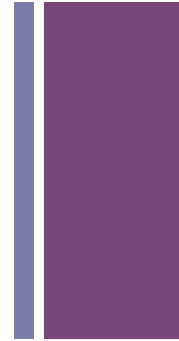
## ■ Choreography

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

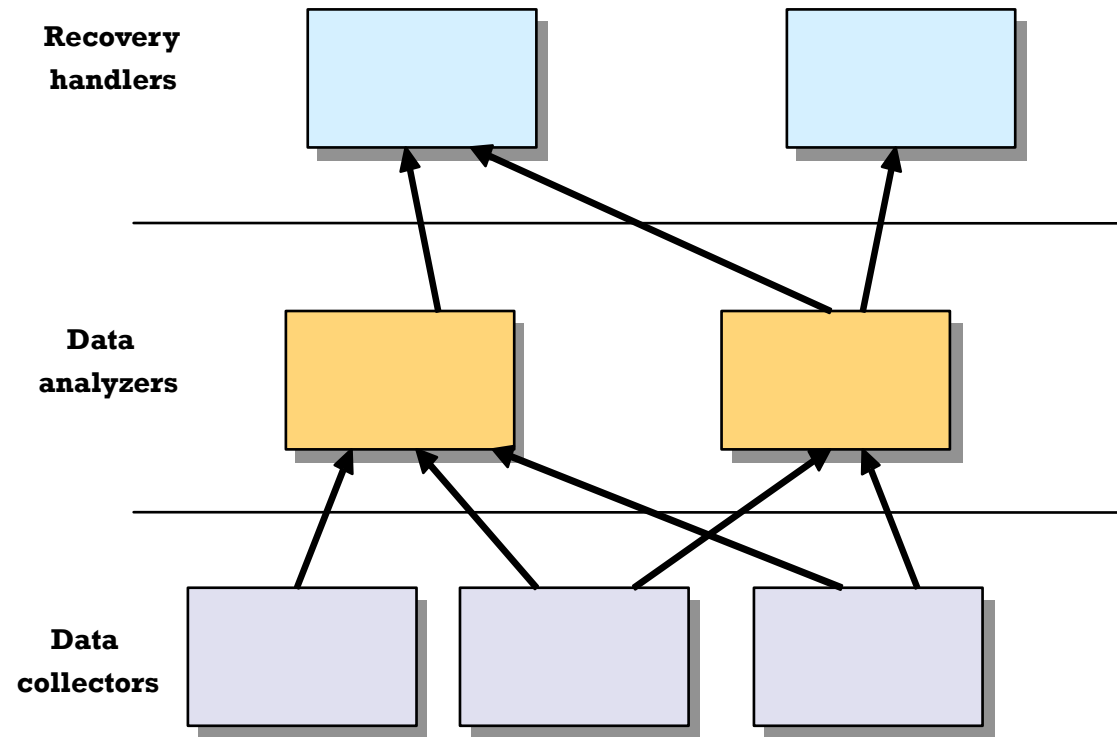
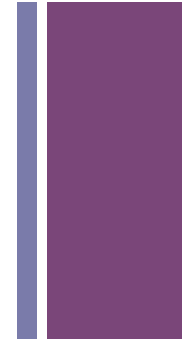


# + My recipe

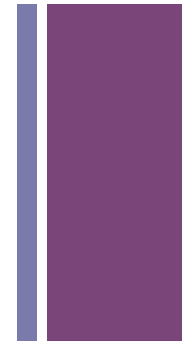
- Self-adaptability implies supervision
- The infrastructure must offer
  - Sufficient probes and analysis capabilities
  - Means to keep the execution on track in case of problems



# + Three key layers

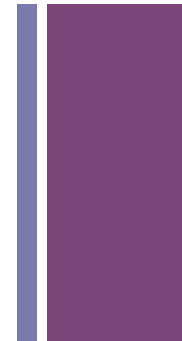


# + Monitoring



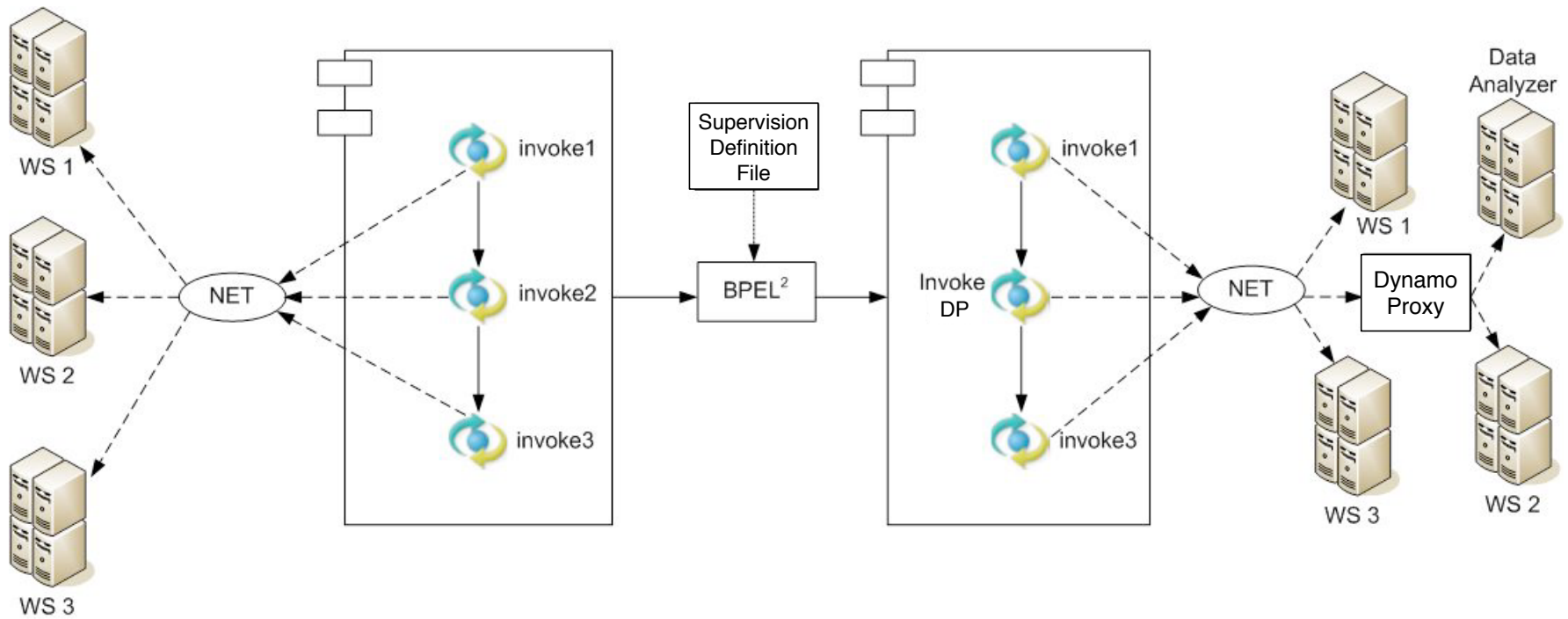
Approach	Language		Abstraction		Properties		Directives			Timeliness		
	Logic	HL/VHL	Domain	Implementation	Safety	Temporal	Process	Activity	Event	Post-Mortem	Synchronous	Asynchronous
Sahai et al.		x	x			x	x					x
Keller and Ludwig		x	x			x	x					x
Skene et al.		x	x			x	x					x
Erradi et al.		x		x	x	x		x			x	x
Mahbub and Spanoudakis	x			x	x	x			x	x		
Moser et al.		x	x		x			x				x
Dynamo	x	x		x	x			x			x	

# + Recovery



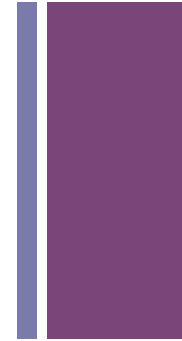
Approach	Language			Location		Actions					Data Source	
	Programming	Logic	HL/VHL	Instance	Proxy	Retry	Substitute	Compensate	Restore	Others	Process	External
Ardagna et al.			x		x	x	x	x			x	
Colombo et al.	x		x		x		x				x	
Moser et al.			x		x		x				x	
Charfi et al.			x	x				x	x		x	
Dynamo		x		x		x	x	x	x	x	x	x

# + Proxy-based solutions



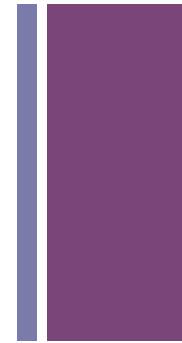
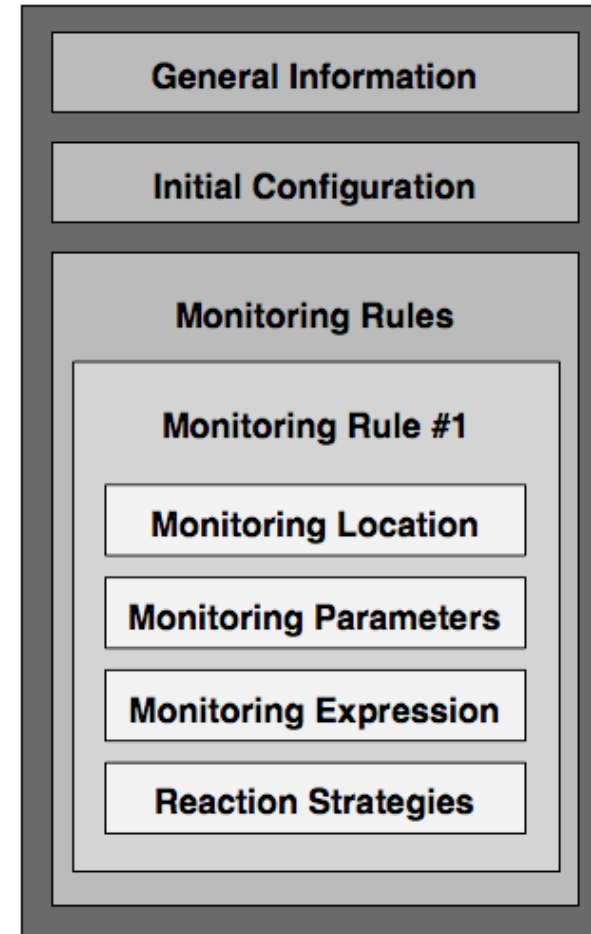
# + Our solution

- Design by contract
- Separation of concerns
  - The business logic is defined separately from supervision
  - supervision is a cross-cutting concern
- Monitoring:
  - Assertion-based
    - The functionalities and QoS needed by the process are defined as pre- and post-conditions on the interaction with the outside world
- Recovery
  - ECA rule based

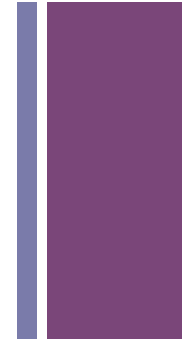


# + Supervision rules

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

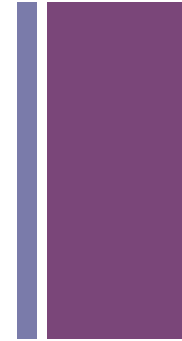


# + WSCoL



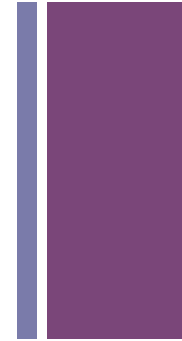
- Declarative specification of the behavioral properties (things we want to look out for)
- Mixes JML (lightweight version) and XML technology
- Two main activities:
  - Data Collection:
    - internal, external, and historical variables
    - Variable aliasing
  - Data Analysis: relationships between data
    - Typical boolean operators (and, or, not, implies, if and only if)
    - Relational operators (<, >, ==, <=, >=)
    - Typical mathematical operators (+, -, \*, /, %)
    - Quantifiers - forall, exists
    - Data computation - max, min, avg, sum, product
    - Data type specific functions - length, starts-with, etc.

# + WSR<sub>e</sub>L



- Event
  - Monitoring has signaled an error
- Condition
  - Discriminates between different recovery strategies depending, for example, on the extent of the error
  - Uses WSCoL to define the condition
- Action
  - A recovery strategy
  - Made up of different recovery steps
    - Step\_A || Step\_B || Step\_C
  - Each step is made up of a number of atomic recovery actions
    - Action\_A && Action\_B
- They do not have access to the process internals

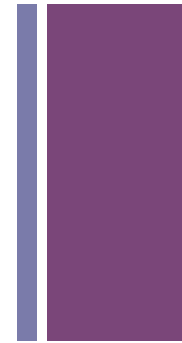
# + Reaction strategies



- Built-in solutions
  - Retry
  - Change monitoring rule
  - Change monitoring parameters
  - Call handlers provided by services
  - Warn and stop
- Third-party solutions
  - Rebind
  - Reorganize
  - Renegotiate
  - ...

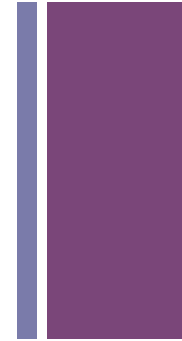
# + Example: rebinding

- A service may not be available
  - or, better services can be available
- QoS values deviate from the estimation
  - 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

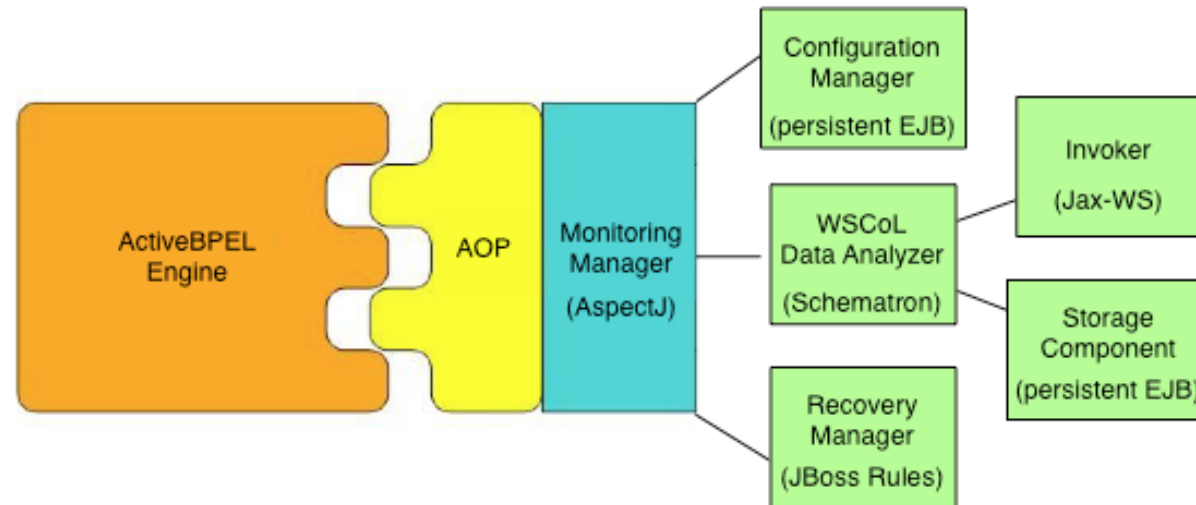


# + Rebinding may fail...

- 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

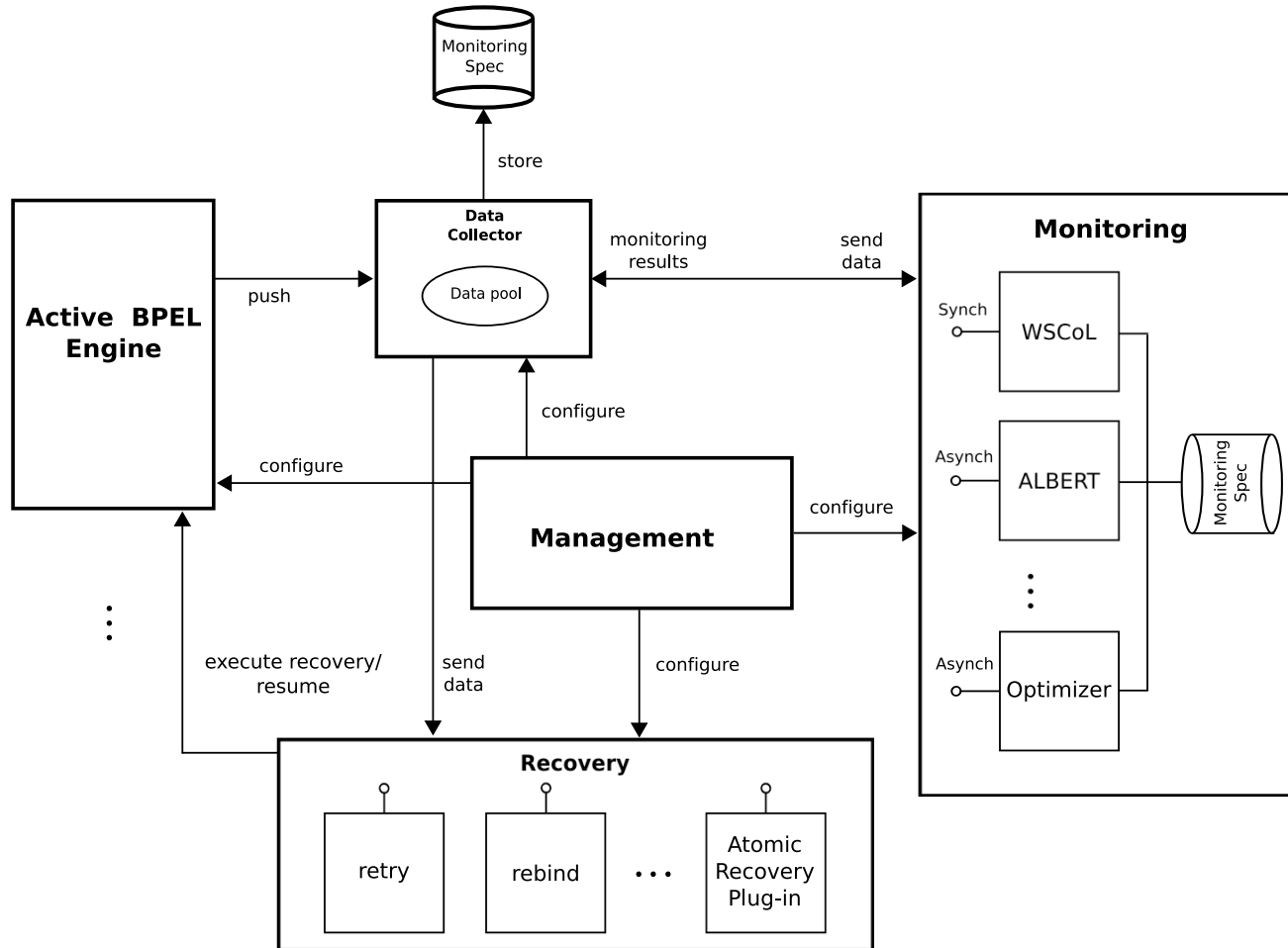


# + Our AOP-based infrastructure



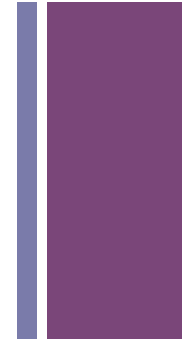
- Main components:
  - **Monitoring Manager:** manages the overall supervision process (hooks into ActiveBPEL)
  - **Configuration Manager:** contains all the defined supervision rules
  - **Data Analyzer:** responsible for analysis
  - **Recovery Manager:** responsible for taking action

# + More flexibility



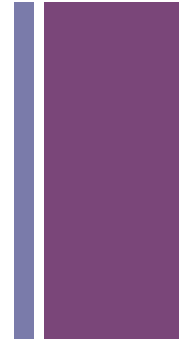
# + Some challenges

- Engineering challenges:
  - Life cycle of self-adaptive elements
  - Relationships among self-adaptive elements
  - Security, privacy, and trust
  - Goal specification
- Scientific challenges:
  - Behavioral abstractions and models
  - Robustness theory
  - Learning and optimization theory
  - Negotiation theory
  - Automated statistical modeling



# + Further challenges

- Semantic approaches
- Dynamic infrastructures
  - Clouds
- Advanced (enterprise) service buses





Questions?



**Thank you !!!**

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

Albert Einstein