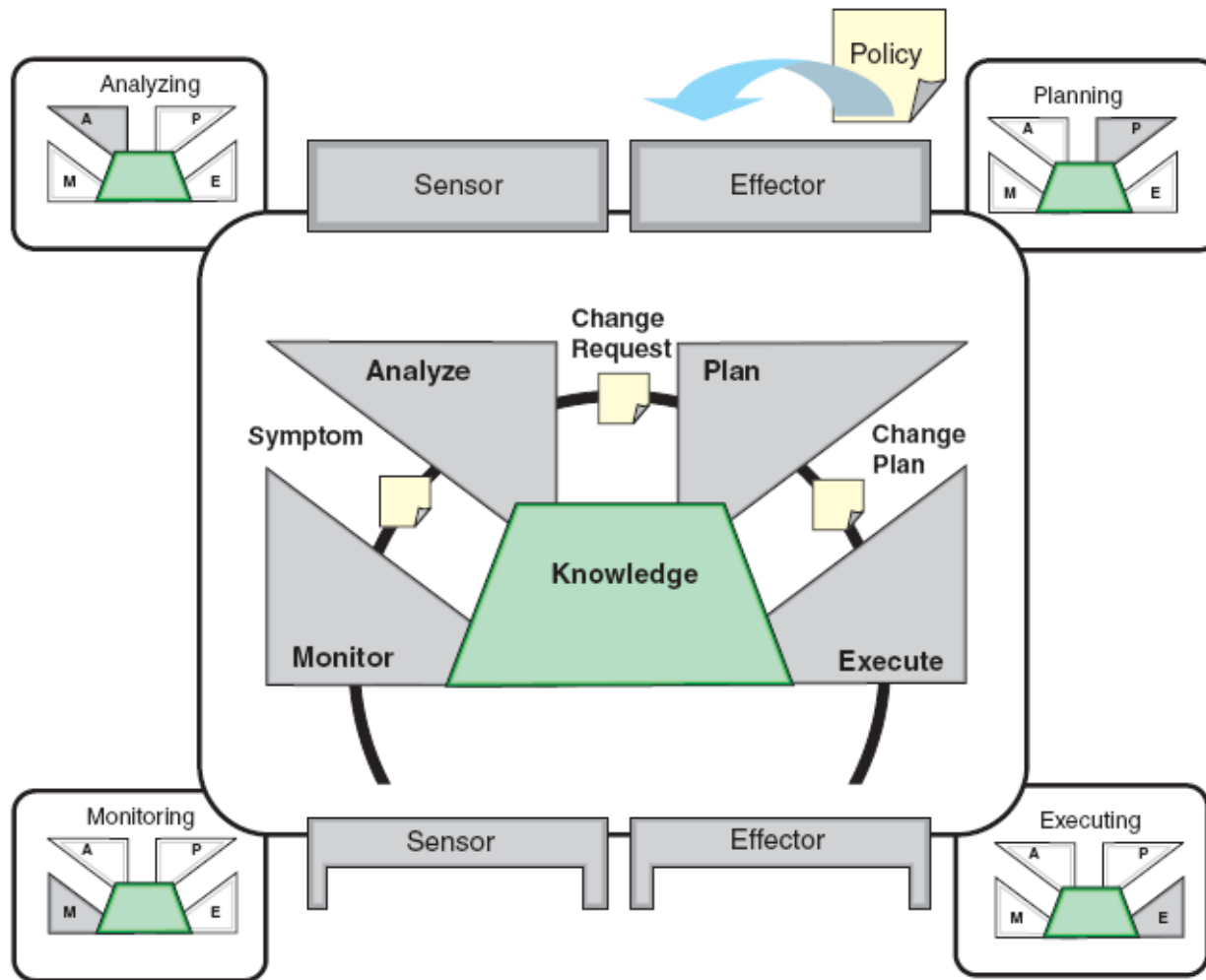


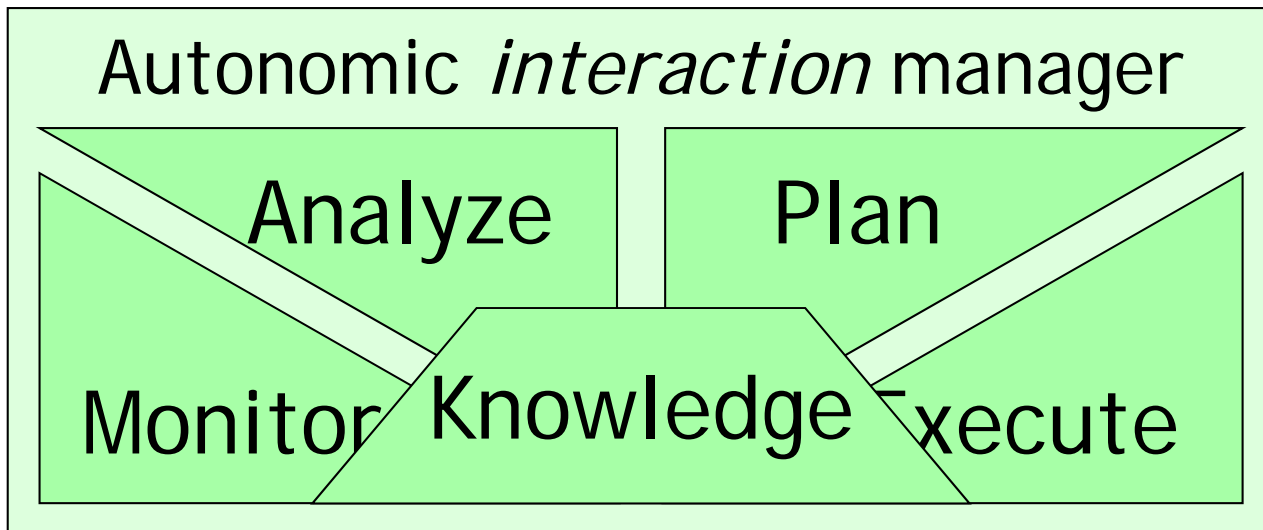
(Autonomic) Supervision

Luciano Baresi

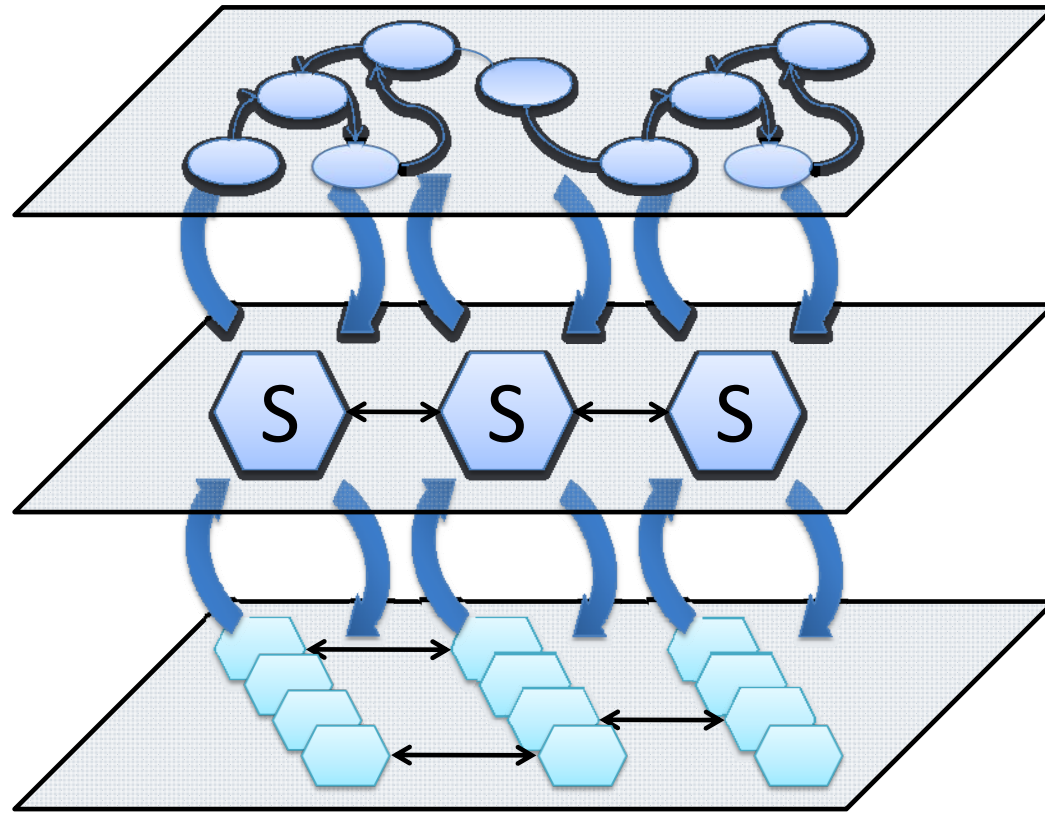
Autonomic systems



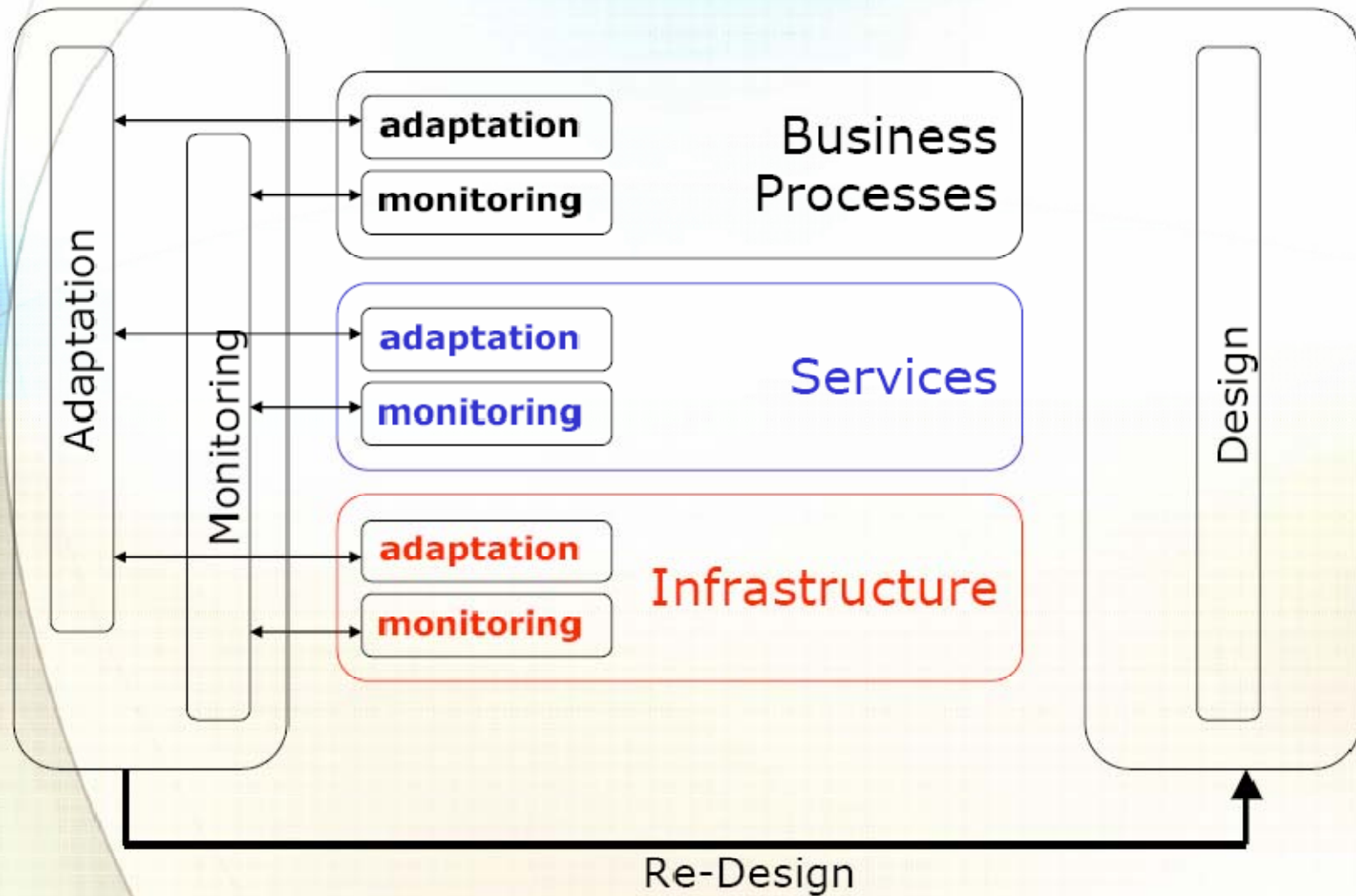
Human operators



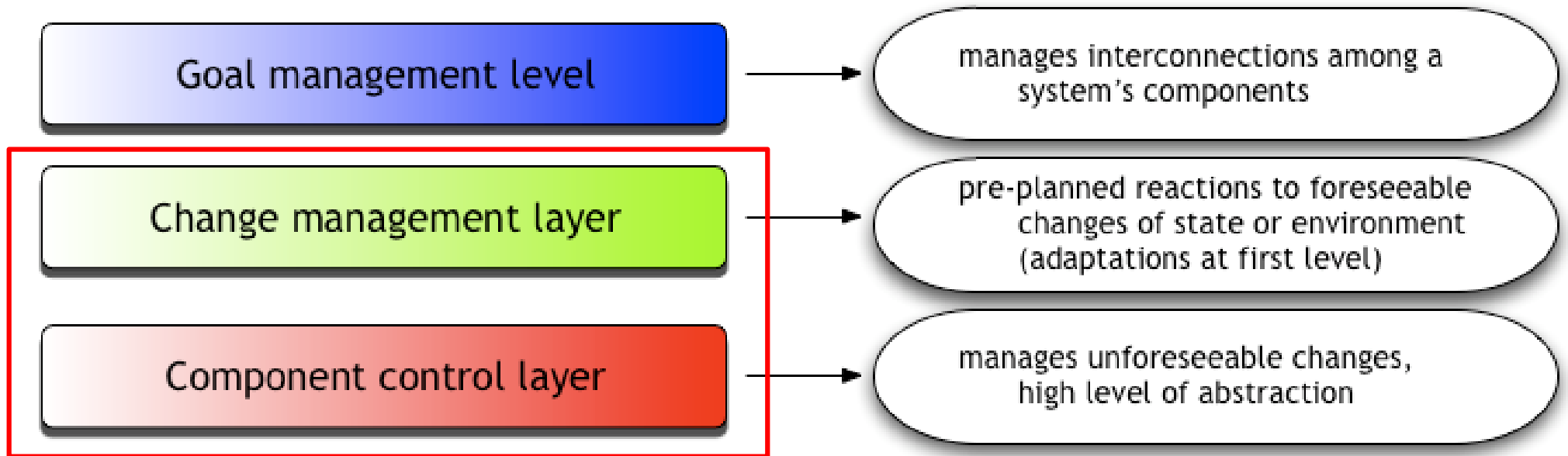
Supervision



A Framework for Monitoring & Adaptation



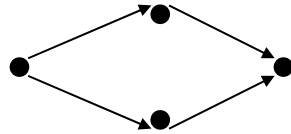
What can we do?



AWPs vs. Autonomic Computing

Autonomic Web Processes

Business Processes



- **Self Configuring:** Processes configured with respect to business policies.
- **Self Healing:** Quick responses to failures, leading to large savings in cost.
- **Self Optimizing:** Environment changes lead to reconfiguration to a lower cost process.

Autonomic Computing

Autonomic IT Infrastructure



Databases



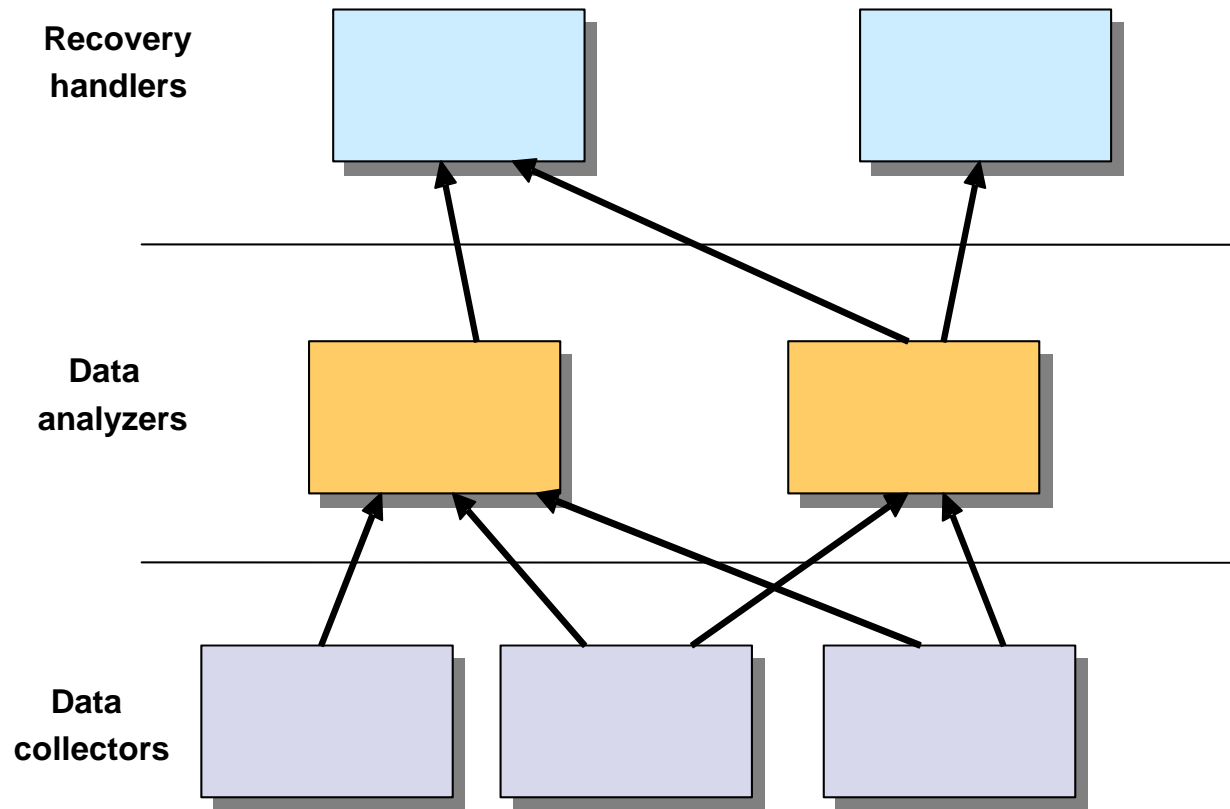
Networks



Servers

- **Self Configuring:** Lower IT cost on maintenance and deployment.
- **Self Healing:** Lower human involvement in problem detection, analysis and solving.
- **Self Optimizing:** Better SLAs to customers of the IT infrastructure.

Three key layers



Loose and strict monitoring

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

Monitoring

Approach	Language		Abstraction		Properties		Directives			Timeliness		
	Logic	HL/WHL	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	

Example

- Assumptions

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

$$\mathbf{Happens}(ir:FindAvail(l,veh),t1,\mathcal{R}(t1,t1)) \wedge \mathbf{Happens}(as:A1(veh),t1,\mathcal{R}(t1,t1)) \wedge \neg \mathbf{HoldsAt}(Available(v),t1-t_u) \Rightarrow \neg \mathbf{Initiates}(as:A1(veh),equalTo(veh,v),t1)$$

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

$$\mathbf{Happens}(rc:RelKey(v,c,l),t1,\mathcal{R}(t1,t1)) \wedge \mathbf{Happens}(rc:RetKey(v,l),t2,\mathcal{R}(t2,t2)) \wedge (t1 \leq t3) \wedge (t3 \leq t2) \Rightarrow \neg \mathbf{HoldsAt}(Available(v),t3)$$

Recovery

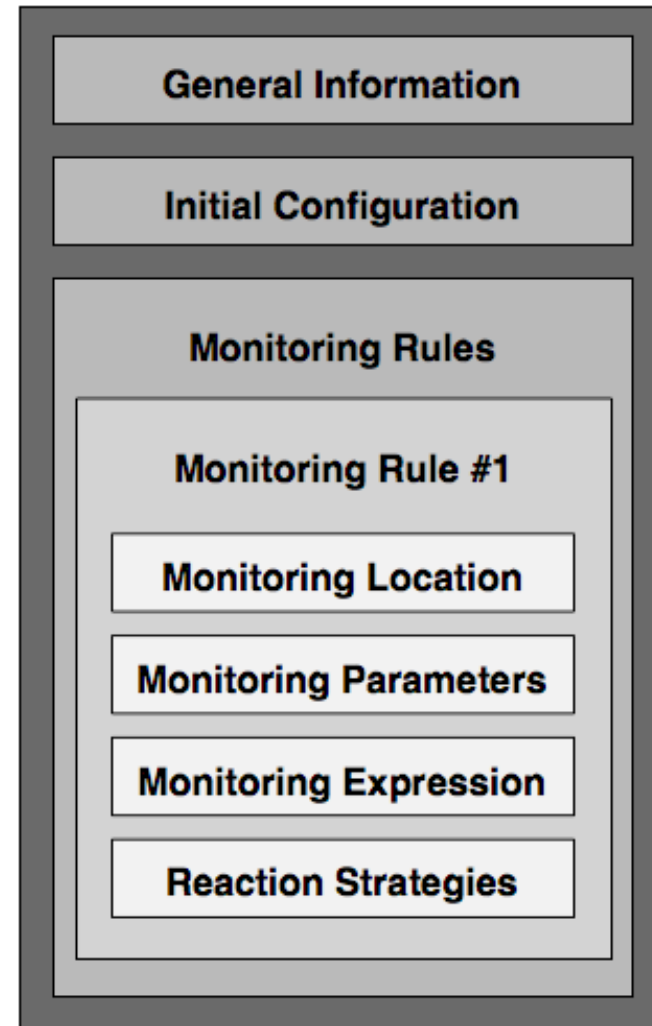
Approach	Language			Location		Actions					Data Source	
	Programming	Logic	HL/VHL	Instance	Proxy	Retry	Substitute	Compensate	Restore	Others	Process	External
Ardagna et al.			x		x	x	x	x			x	
Colombo et al.	x		x		x		x				x	
Moser et al.			x		x		x				x	
Charfi et al.			x	x				x	x		x	
Dynamo		x		x		x	x	x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

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.

WSReL

- 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
 - ...

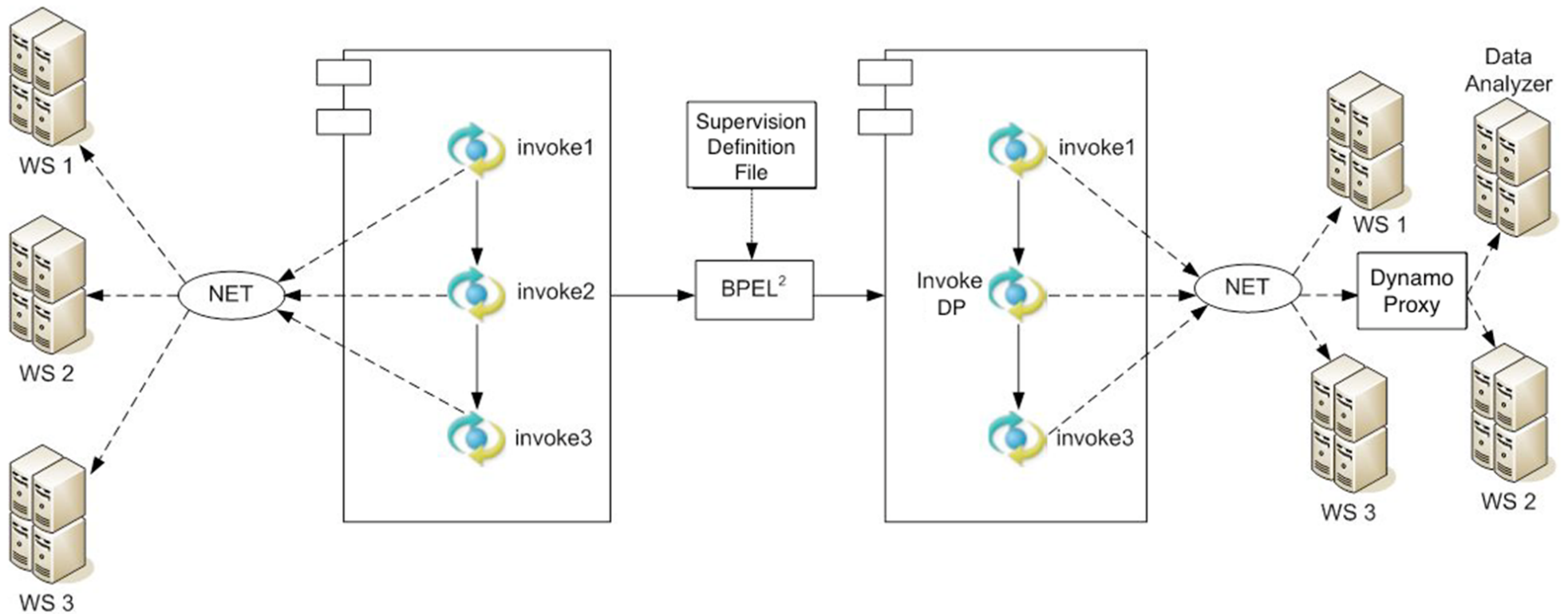
Example: rebinding

- 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

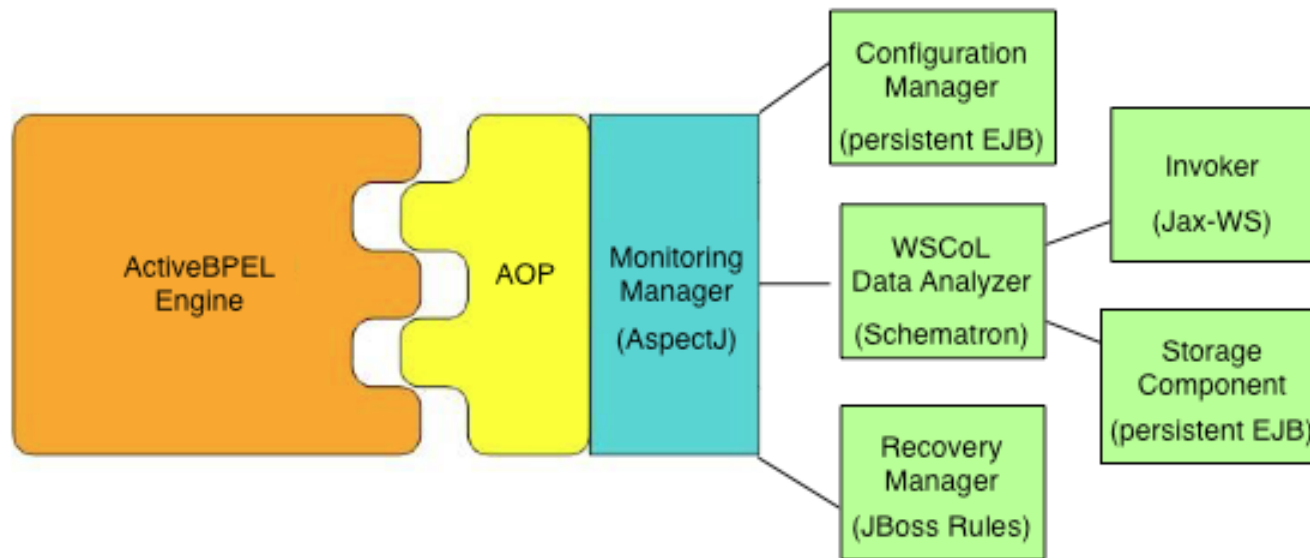
Rebinding may fail...

- 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

Proxy-based solutions



Our AOP-based infrastructure



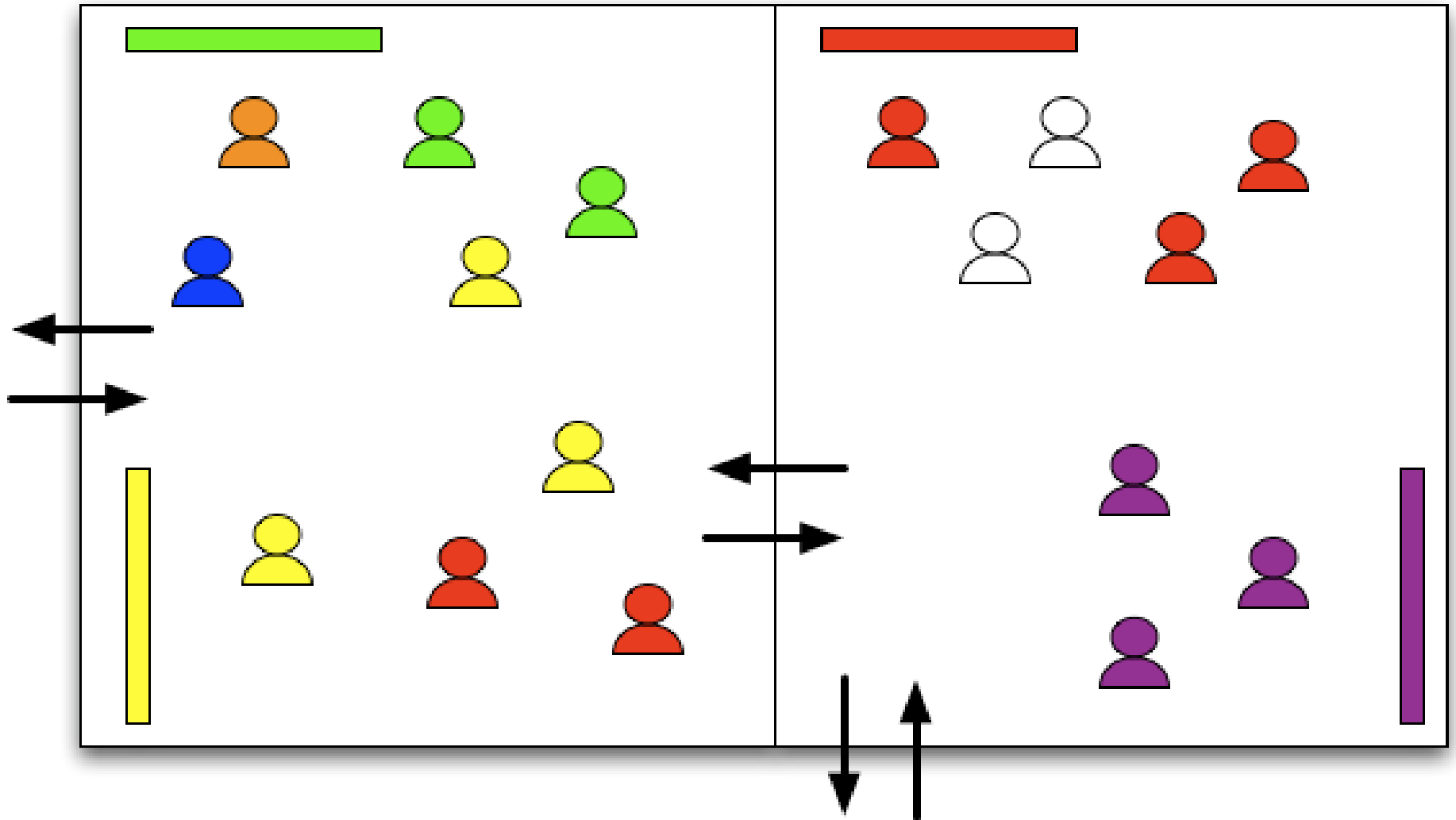
- Main components:
 - Monitoring Manager: manages the overall supervision process (hooks into ActiveBPEL)
 - Configuration Manager: contains all the defined supervision rules
 - Data Analyzer: responsible for analysis
 - Recovery Manager: responsible for taking action

More flexibility

New hypotheses

- Peer to peer systems
 - Hundreds of different elements
- Extrinsic supervision
 - Decentralized control
- Transparent supervision
 - No fat special-purpose autonomic components
- Fine-grained control

An example

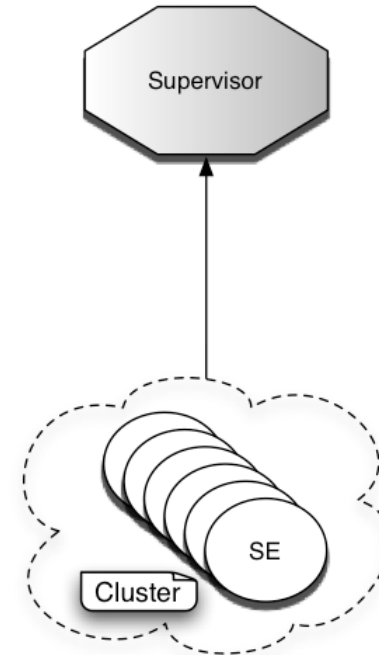
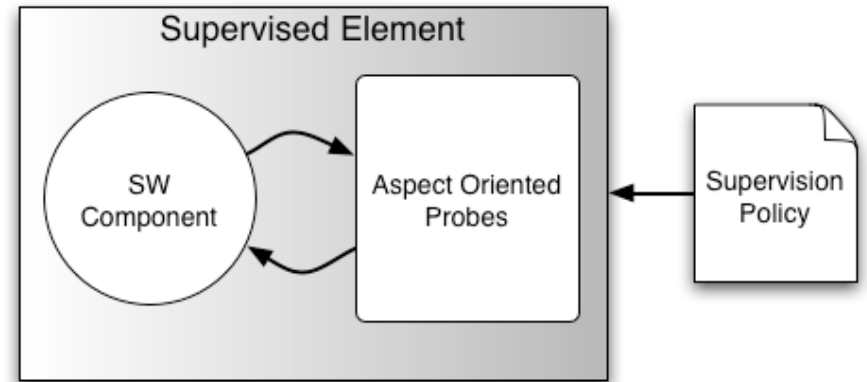


Adaptation levels

- Three levels of adaptation
 - component adaptation
 - group-wide adaptation
 - system-wide adaptation
- Local reaction
 - Each component reacts in isolation
 - No need for interactions and cooperation
- Autonomic reaction
 - System-or group-wide reactions
 - Components must exchange information and synchronize

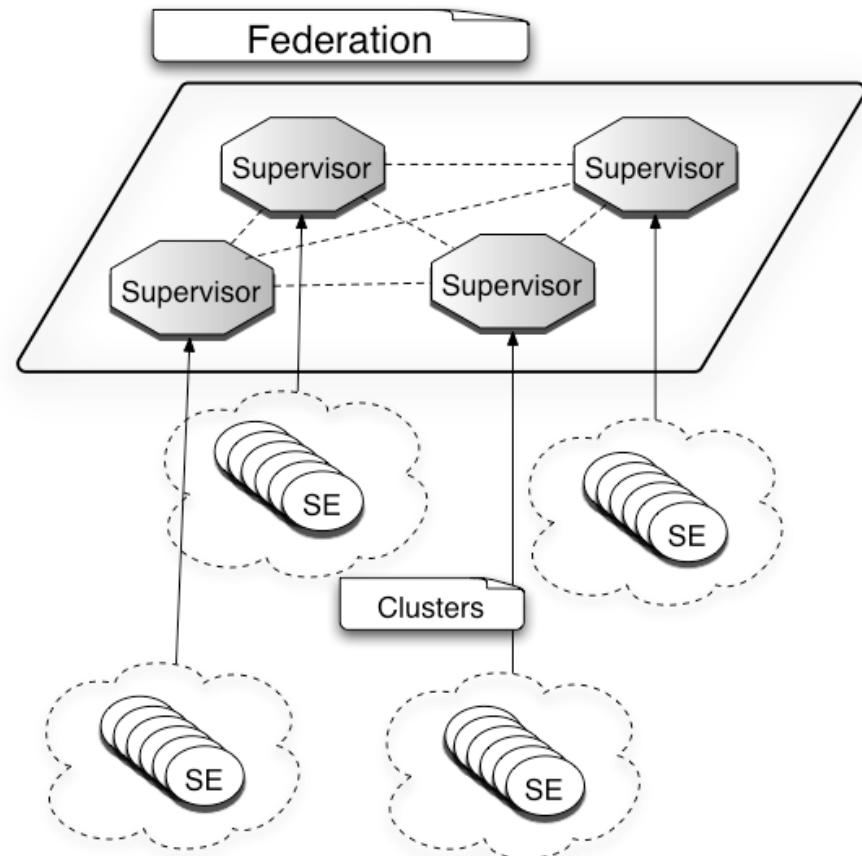
Our model – part I

- Supervised elements (SE)
 - in charge of executing business logic
 - augmented with AOP
 - sufficient for local adaptation
- Grouping (clustering)
 - event-based communication solves membership issues
- Supervisors
 - special-purpose components
 - subscribe to receive events from supervised elements

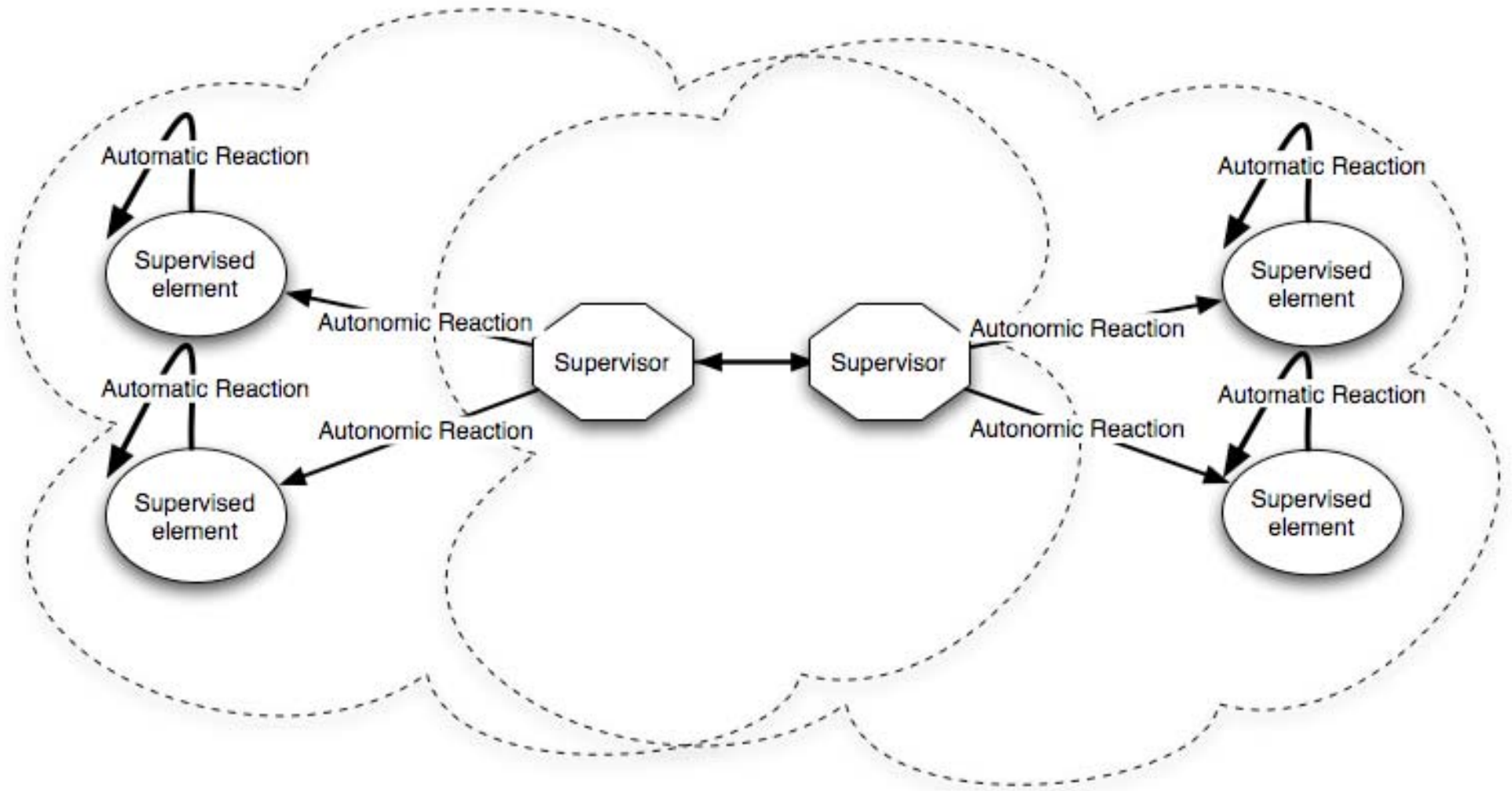


The Model - part II

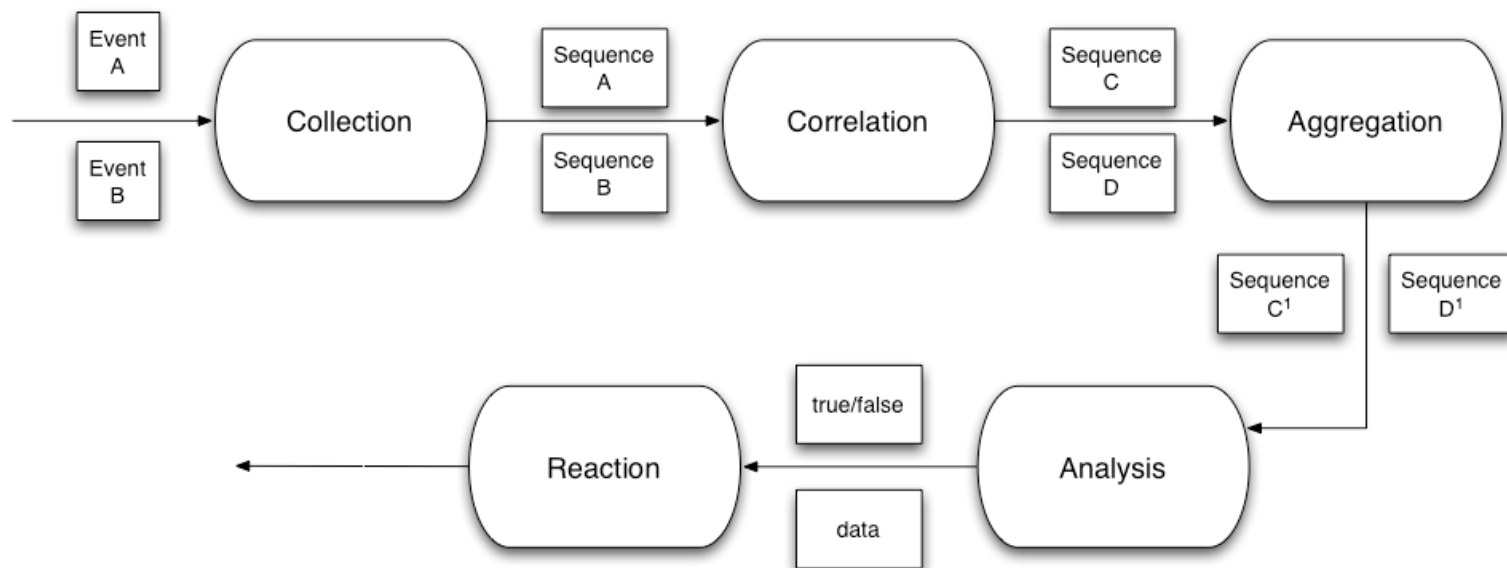
- Supervisors
 - Communicate through federation
 - Other supervisors are seen as other supervised elements



A different view



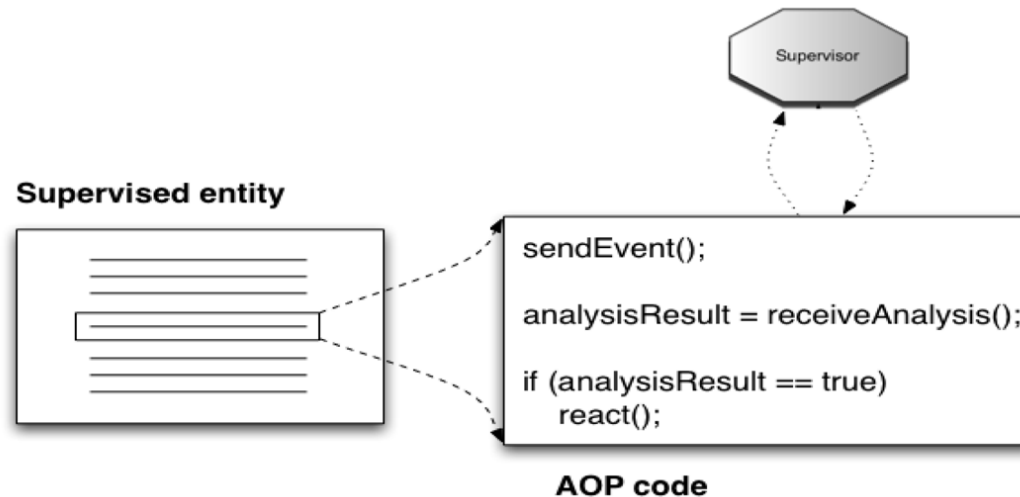
The Control Loop (CEP)



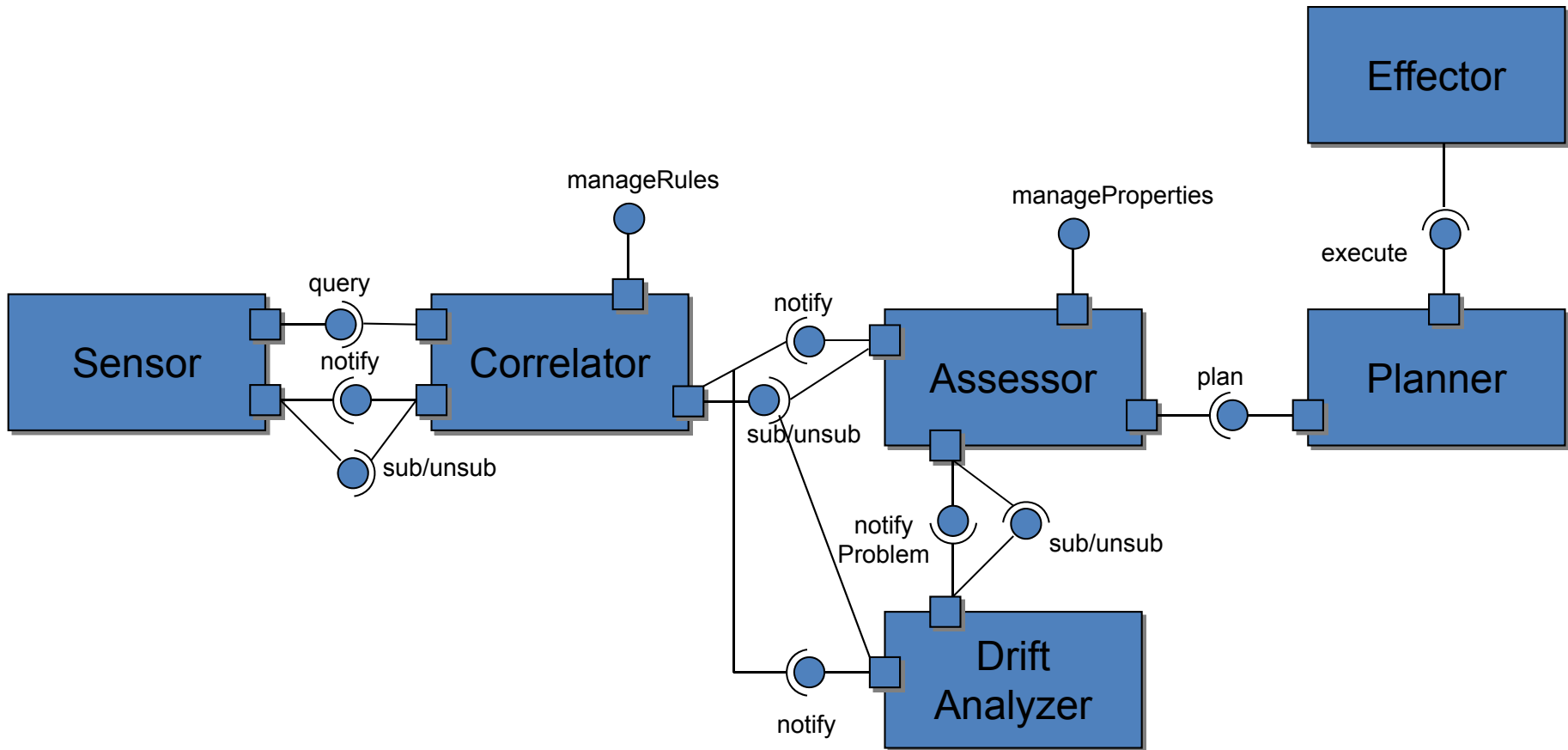
- Collection - receives events and creates sequences of events
- Correlation - events from different sequences are logically grouped together using a correlation property
- Aggregation - data aggregation, elimination, or re-arrangement before analysis
- Analysis - checks to see whether the sequences of correlated events satisfy an overall property
- Reaction - pushes reaction back to the components

Reactions

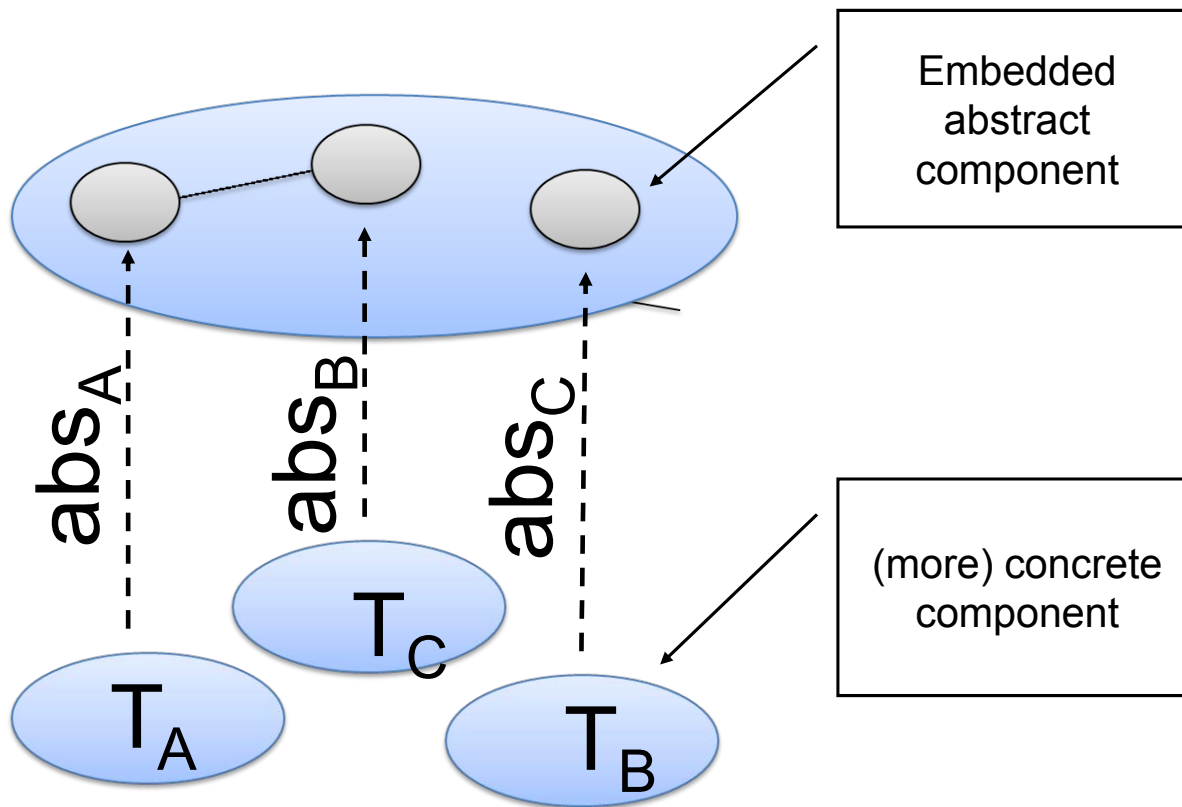
- Pre-defined plans distributed across the supervisors and their elements
 - Fragments of (Java) code
- Synchronous reactions (Intrusive approach, guarantees critical properties)
 - SE waits for a response from the control loop
 - Supervision code executed before allowing business logic to resume



Cascadas



Zooms and Model Trees



Planning for Autonomic Systems

- Reactive planning:
 - If an „undesired“ system state is encountered, find a way to guide the system back into a desired state
- Pro-active planning:
 - Try to determine whether the supervised system will enter an undesired state

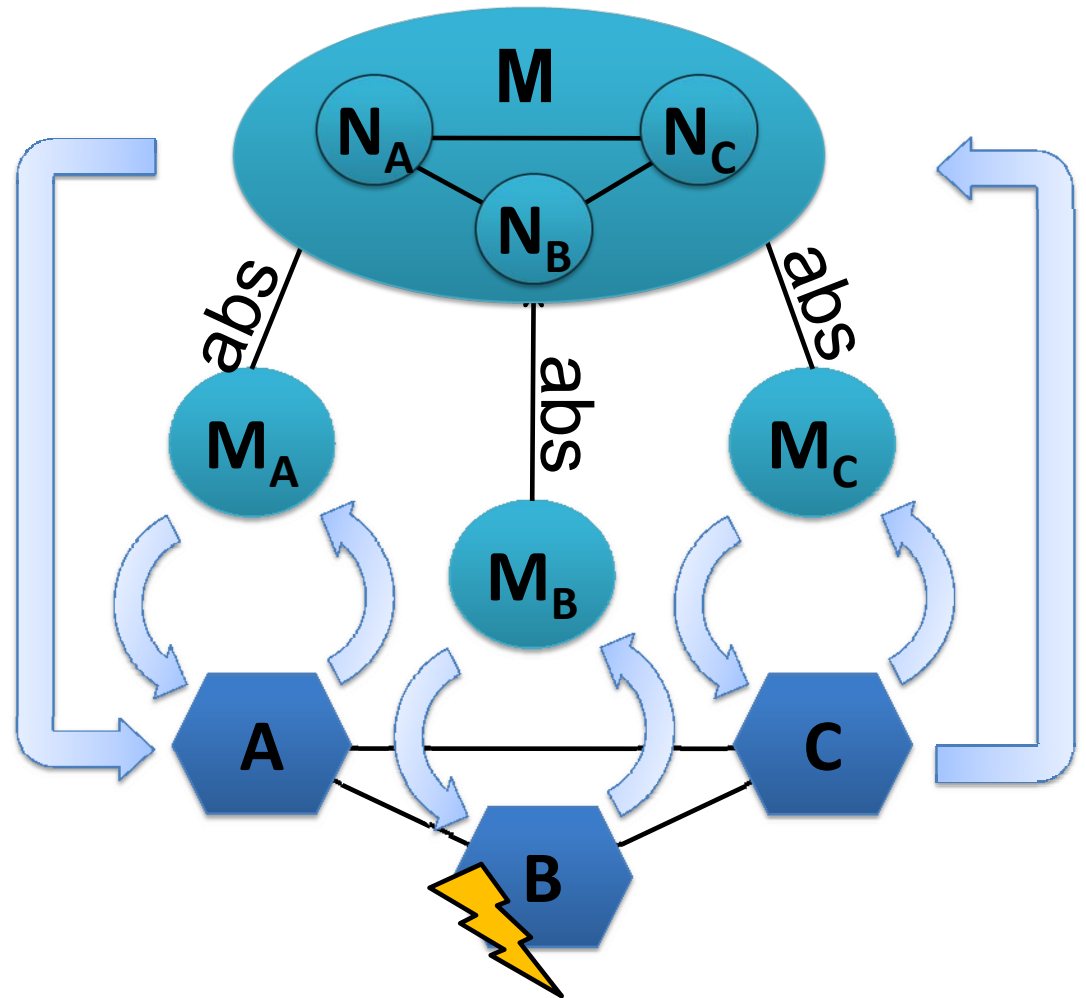
Planning

- Plans
 - are expressed by „decision trees“ (pomtrees)
 - computed by „symbolic“ execution of models
- Problems
 - managed system may not completely controllable (actions may be issued by the environment of a system)
 - planning may be not feasible (complexity / timeliness)
- Naïve solution
 - Build an „overall“ global model
 - THIS WILL FAIL DUE TO COMPLEXITY ISSUES

Planning Algorithm

If a problem situation is detected (say in B)

1. perform planning on subsystem / component level ($B \leftrightarrow M_B$)
2. If this fails, construct the composed model using
 - abstractions $M_x \rightarrow N_x$ and embeddings $N_x \rightarrow M$
3. Perform planning in M
the resulting plan defines a **coordinated** course of action of A, B, C
4. Use zooms to concretize the plan for A, B, C
(prune the decision tree by non-executable alternatives)



Drift Detection Framework

Monitoring

Event Based

Relevant information are sent to a centralised monitoring unit at pre-defined intervals.

Request Based

A centralised monitoring unit requests relevant information at pre-defined intervals.

Embedded

Individual components provide the functionality to monitor themselves. (decentralised)

Analytics

Boundaries

$\alpha \rightarrow (\beta^-, \beta^=, \beta^+)$?

Current State

Analysing the current state of α with respect to its boundaries specified by $(\beta^-, \beta^=, \beta^+)$.

Forecasting

Analysing IF and at what time t α reaches the boundaries specified by $(\beta^-, \beta^=, \beta^+)$.

Prediction

Predicting α for time $t + n$ based on θ or on other variables.

Reaction

Direct action vs. separation of concern

Descriptive Report

Detailed summary of all observed values, possible alerts and recommended corrections.

Centralised Reaction

Direct intervention centralised supervision system.

Embedded Reaction

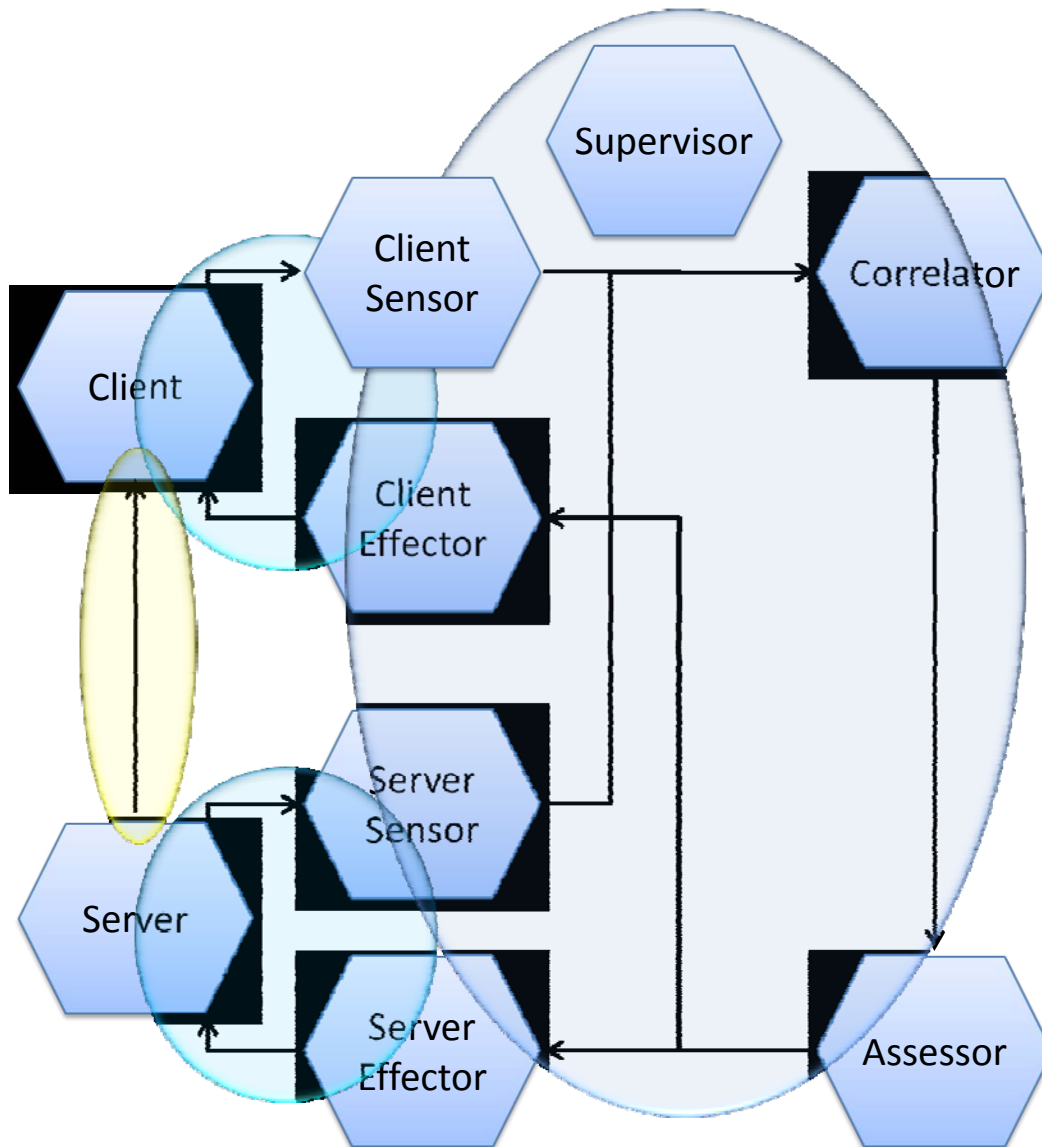
Individual components have full control to execute corrective measures.

$\omega = (\alpha, t)$ observation; $\theta = \omega_1, \omega_2, \omega_3, \dots$ history

Correlation/Assessment Specs

- Event Correlation Language
 - Language capable of considering time and of predicating on complex internal data
 - Reference to event history is possible
- Specs consists of two parts:
 - Temporal Correlation Analysis
 - Predicates on appearance of events and/their internal data
 - Aggregation
 - to build new events
- Temporal Analysis
 - Capable of considering time and of predicating on complex internal data.
 - Temporal predicates
 - Becomes, Until, Between, Within
 - Temporal functions
 - Count, elapsed, past...
 - Aggregate functions (sum, avg, min, max...)
 - Universal/Existential quantifiers
 - Relational operators
 - Arithmetic operators

Contract Structure



service contract

supervision contract

supervision service access

internal control contract

for internal communications

external supervision contract

to interact with the supervisables

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) bus

Questions?



Thank you !!!

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

Albert Einstein