# ICSOC 2006 - Chicago, IL

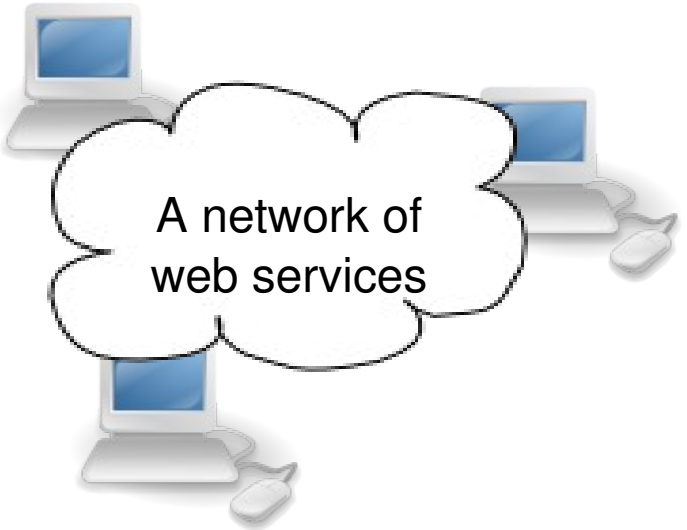
# A Priori Conformance Verification for Guaranteeing Interoperability in Open Environments

M. Baldoni, C. Baroglio, A. Martelli, V. Patti

Dipartimento di Informatica – Università degli Studi di Torino
c.so Svizzera, 185, Torino (Italy)

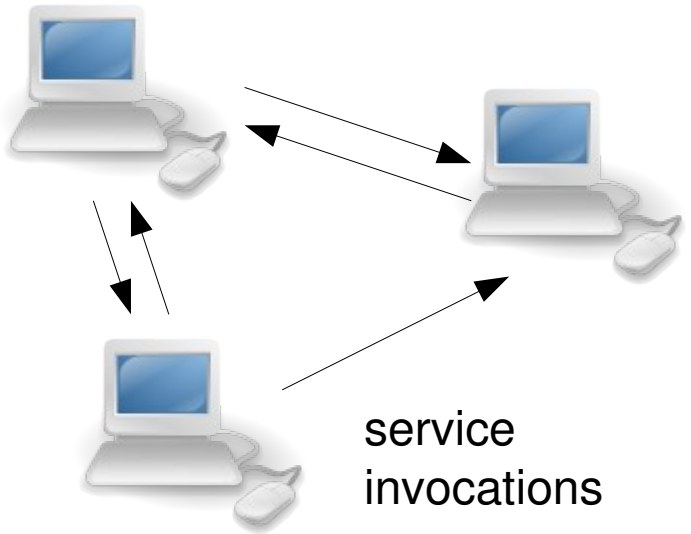
# Web Services

- Web services are heterogeneous devices that can be ***invoked*** over the web
- Executable description of their business process (especially the interactive behavior)
- Tasks: composition, selection, ...
- *Dynamic dimension* due to the openess of the web



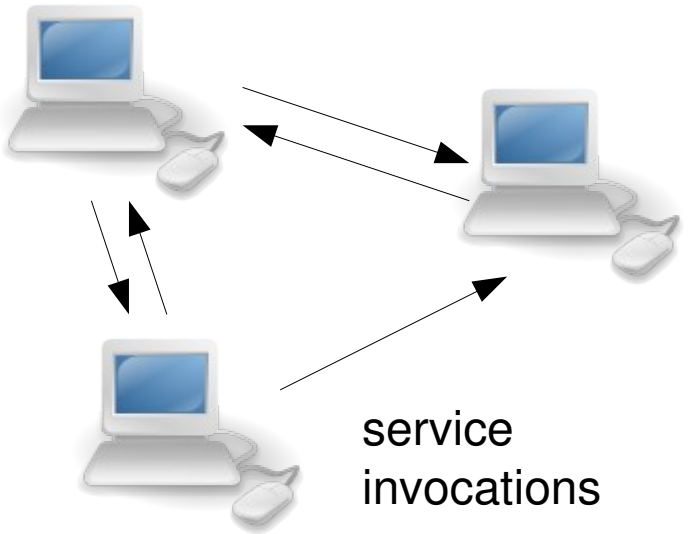
# WS interoperability

- How to guarantee the *interoperability* of a dynamically composed set of WS?
- Interoperability is the capability of a peer of *interacting* with others
- This means they will actually produce a *successful* “conversation” (sequence of message invocation)



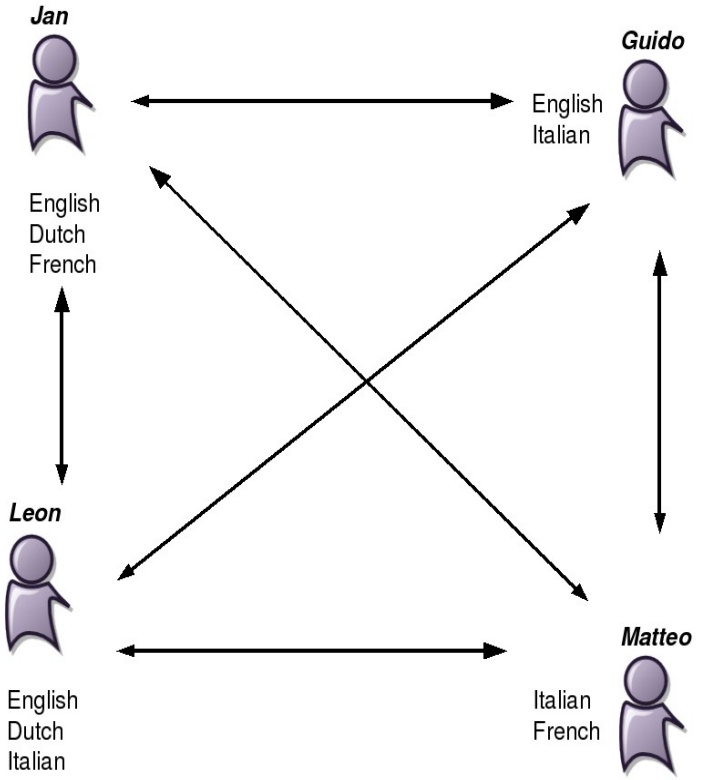
# WS interoperability

- Verification of the properties of the interactions among a set of processes: widely studied in the literature
- The web adds for Web Services a ***dynamic feature***: the set of available processes evolves in time, they can be ***reused***, they can be ***assembled*** dynamically and they are, in principle, unknown to each other



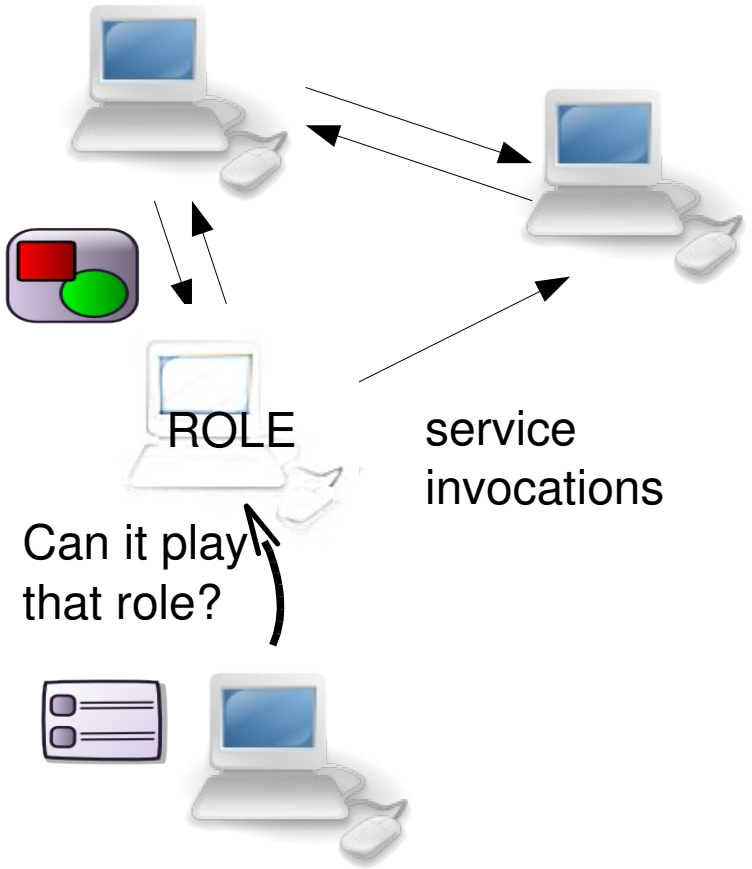
# The summer school example

- In open enviroments services are identified and composed on demand:
  - retrieval done component by component
  - components behaviour could be private, not accessible for inspection



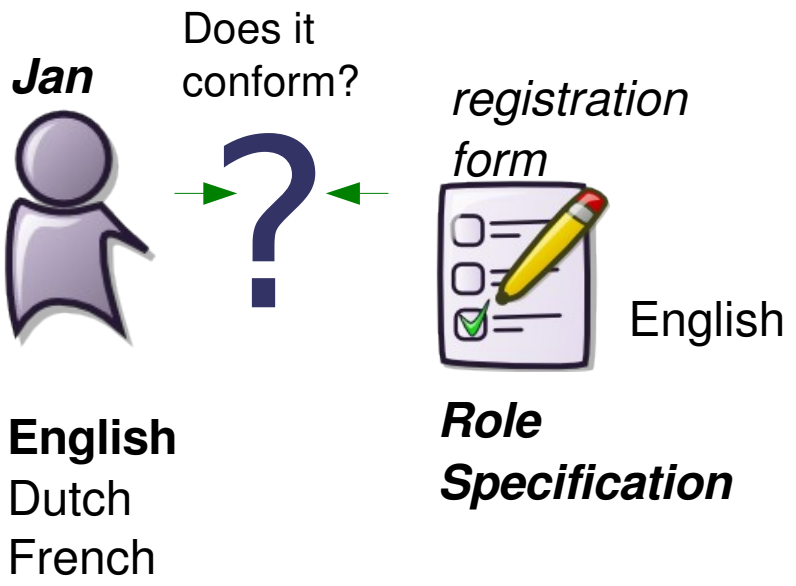
# Choreographies and protocols: public specifications

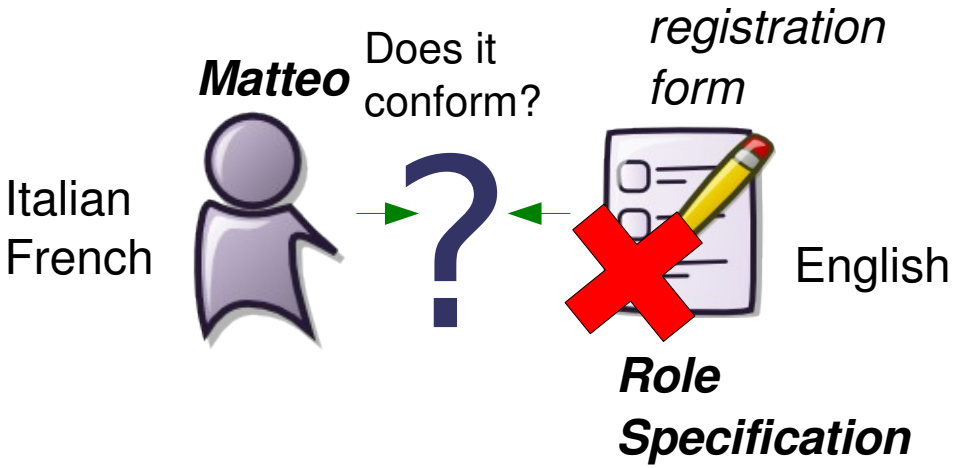
- Need for a “distributed” verification of interoperability
- Define and make public the set of interaction rules that the group should follow (***protocol or choreography***), eg. by means of a standardized language such as WS-CDL



# A-priori interoperability verification

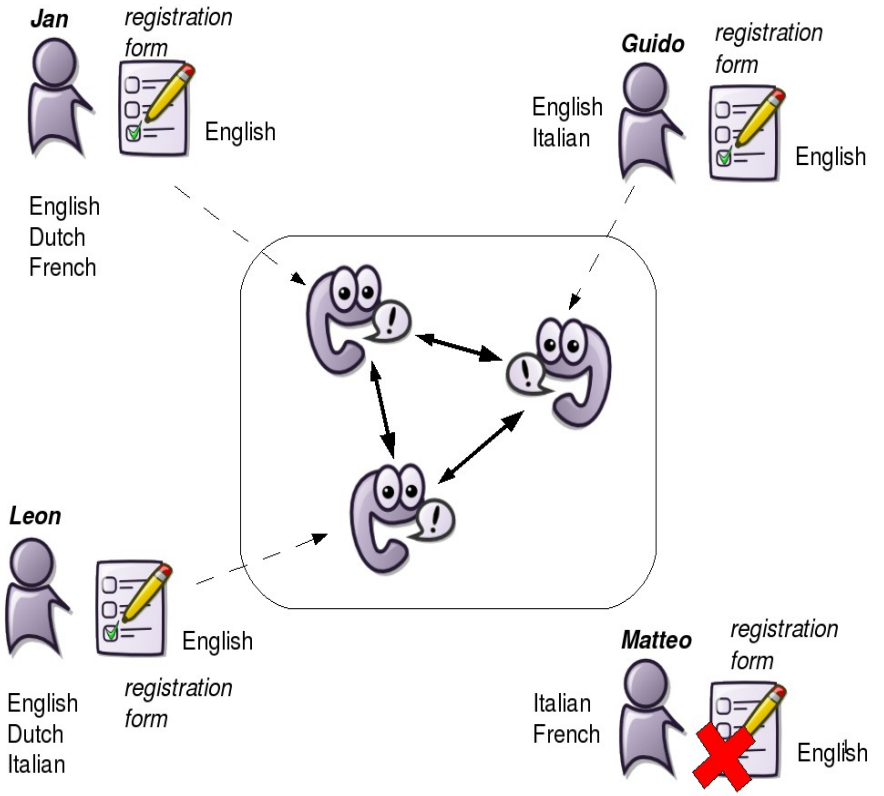
- To perform the conformance test against the role specification
  - ***"a-priori" interoperability test***
  - ***static interoperability cheking***
- ***Verify-once run-always***



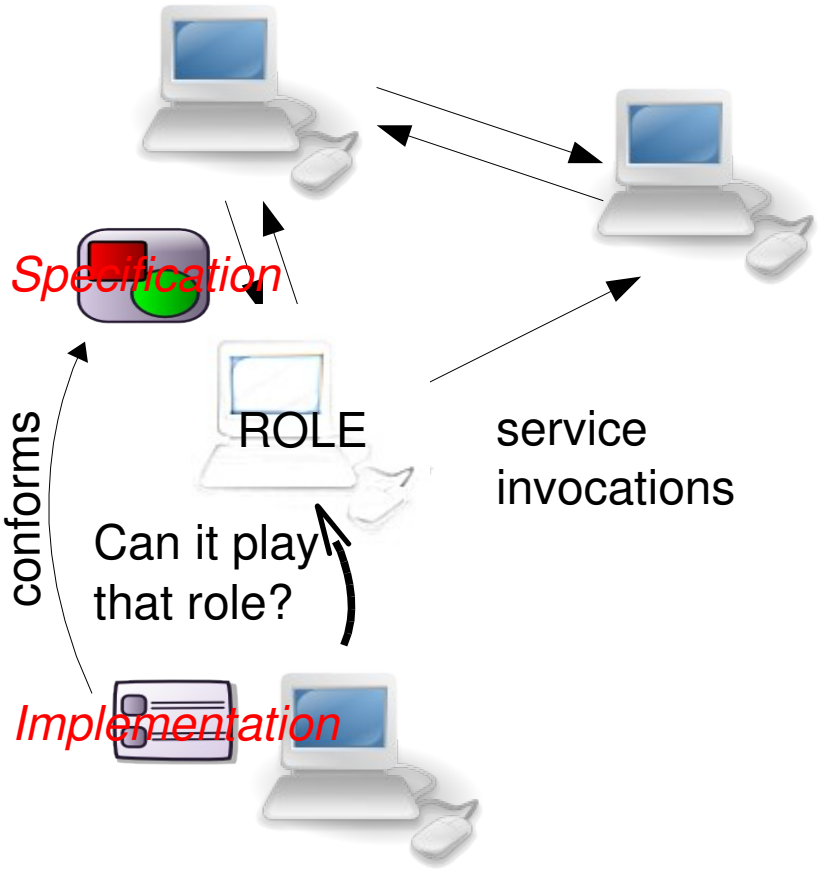
# The summer school example

- *Desire*: to check interoperability ***without any knowledge of the services that will play other roles***
- *Solution:* ***a-priori verification*** of interoperability



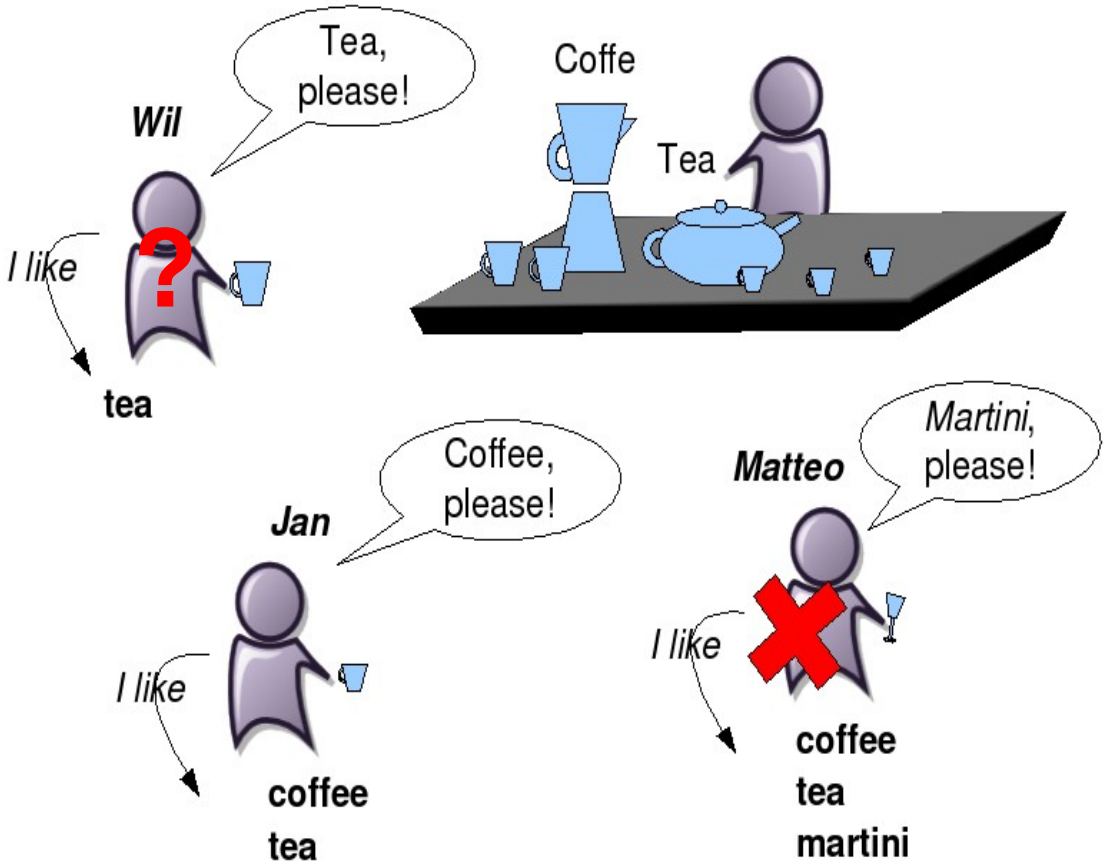
# Conformance test w.r.t. a protocol

- A service can enter the society only if its interactive behavior **conforms** to the communication protocol
- *The **conformance test w.r.t. a protocol** guarantees a-priori the interoperability and that the generated conversations will be "legal" w.r.t. the protocol*



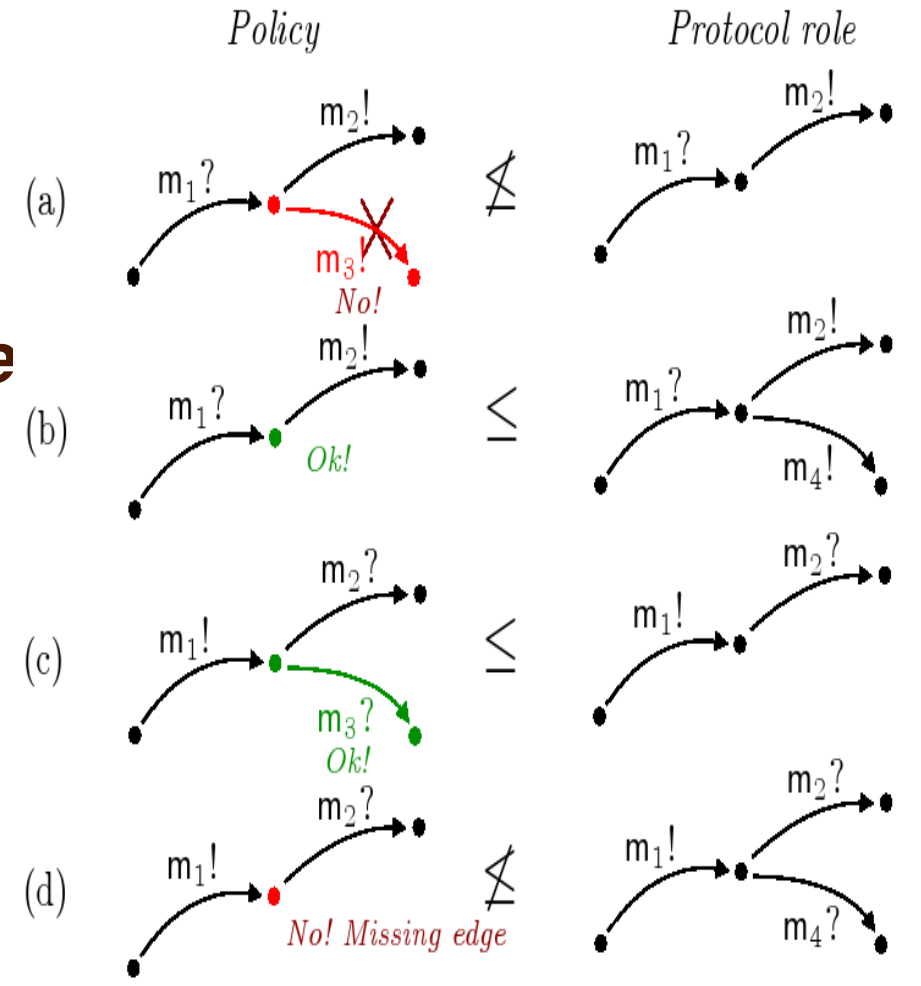
# Reusing components: the coffee break example

- Existing components ***might slightly differ*** from the specification ...
- ... but *when interacting* they could anyway behave as desired
- Conformance "***at design time***" is ***too strong*** to allow them to be chosen



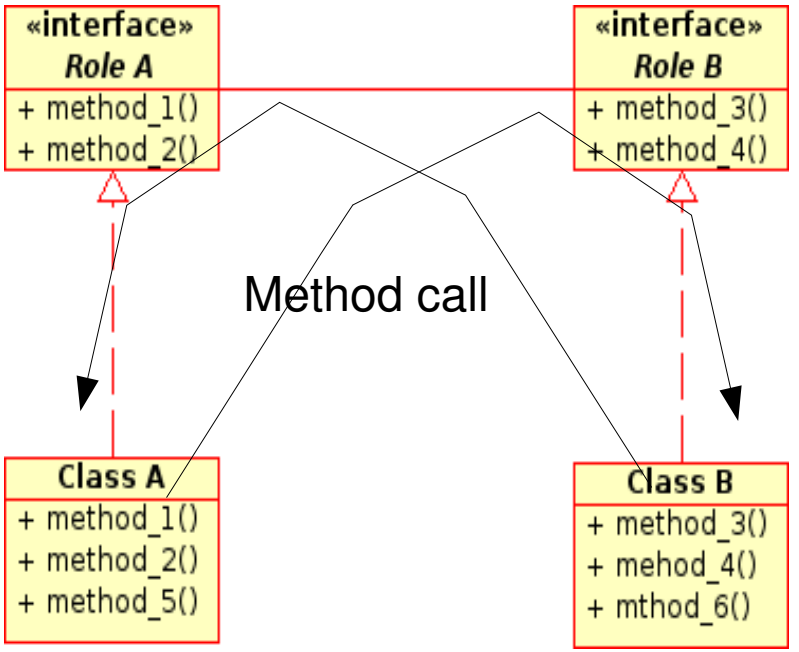
# Conformance test w.r.t. a role specification

- (b) and (c) do not compromise the interoperability
- A conformant policy ***never utters speech acts that are not expected***, according to the protocol, and it should be able to ***handle any message that can possibly be received***, according to the protocol
- ***=> It is not conformance w.r.t. design***



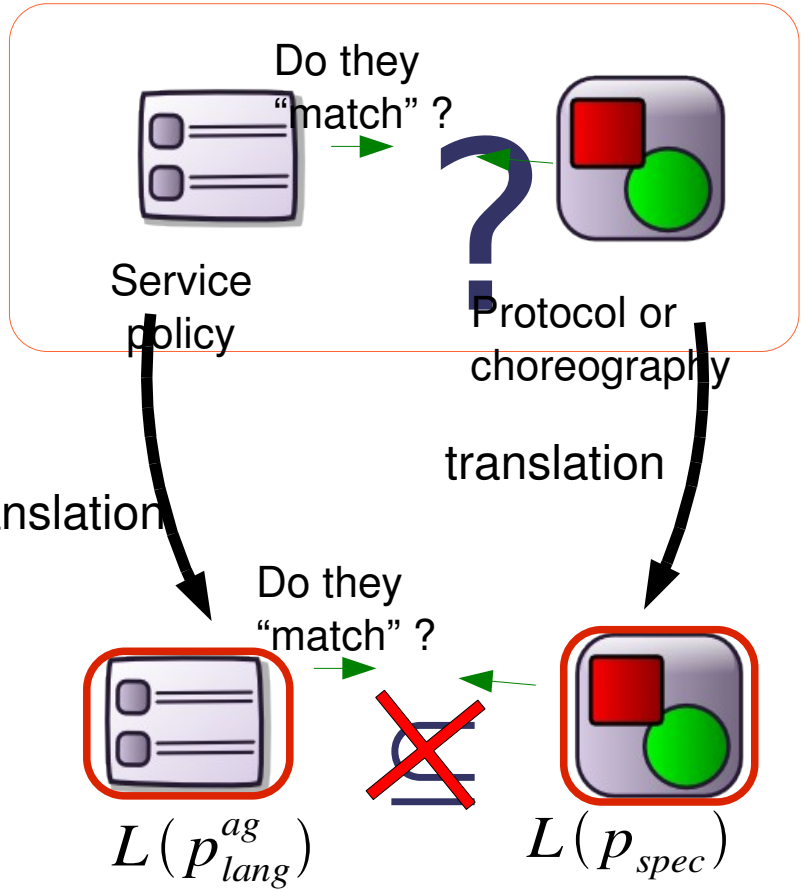
# Message invocation paradigm

- Similarity with method invocation over objects
  - An object must necessarily be able to handle messages sent by other objects in the context of its public interface
  - An object is not obliged to use all methods offered in the public interface
  - An object can have more methods than the interface requires

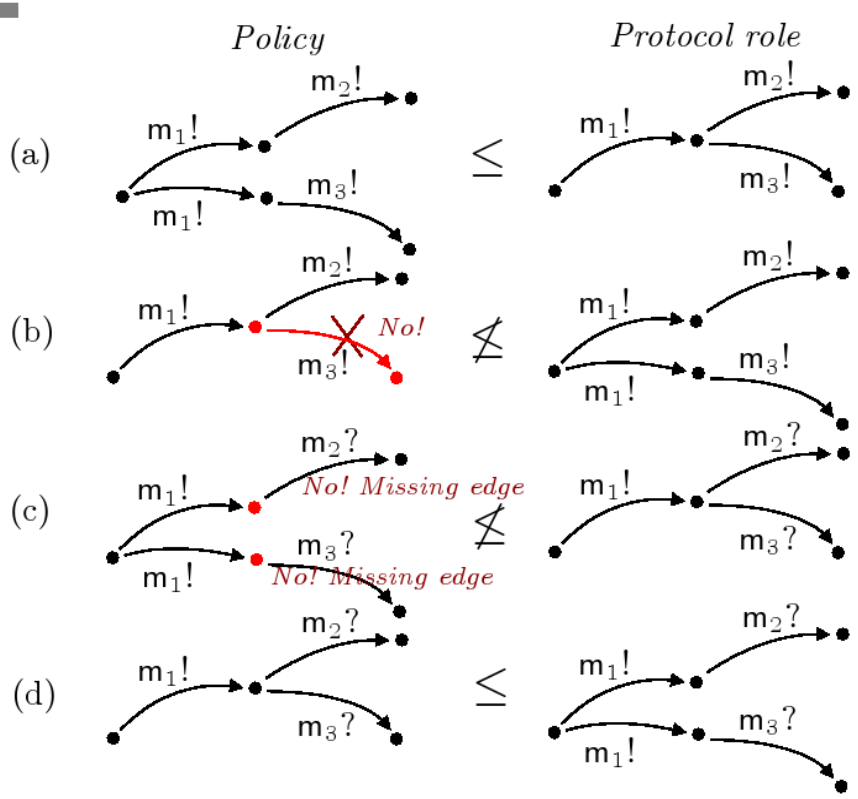


# Conformance test is not an inclusion test

- Protocols and conversation policies represented as *FSA*
- *Conformance test:* could not be a simple inclusion test based on the set of possible execution traces [CLIMA VI, WS-FM 05]



# Conformance test is not a bisimulation test




- The role description and the policy allow ***the same conversations but the structure differs***
- Only (b) and (c) compromise the interoperability
- *(bi)simulation is too strong*: *all cases are not bisimilar*
- Bisimulation in [Busi et al. 05, Zhao et al. 06], execution traces [Foster et al. 06, Alberti et al. 06]

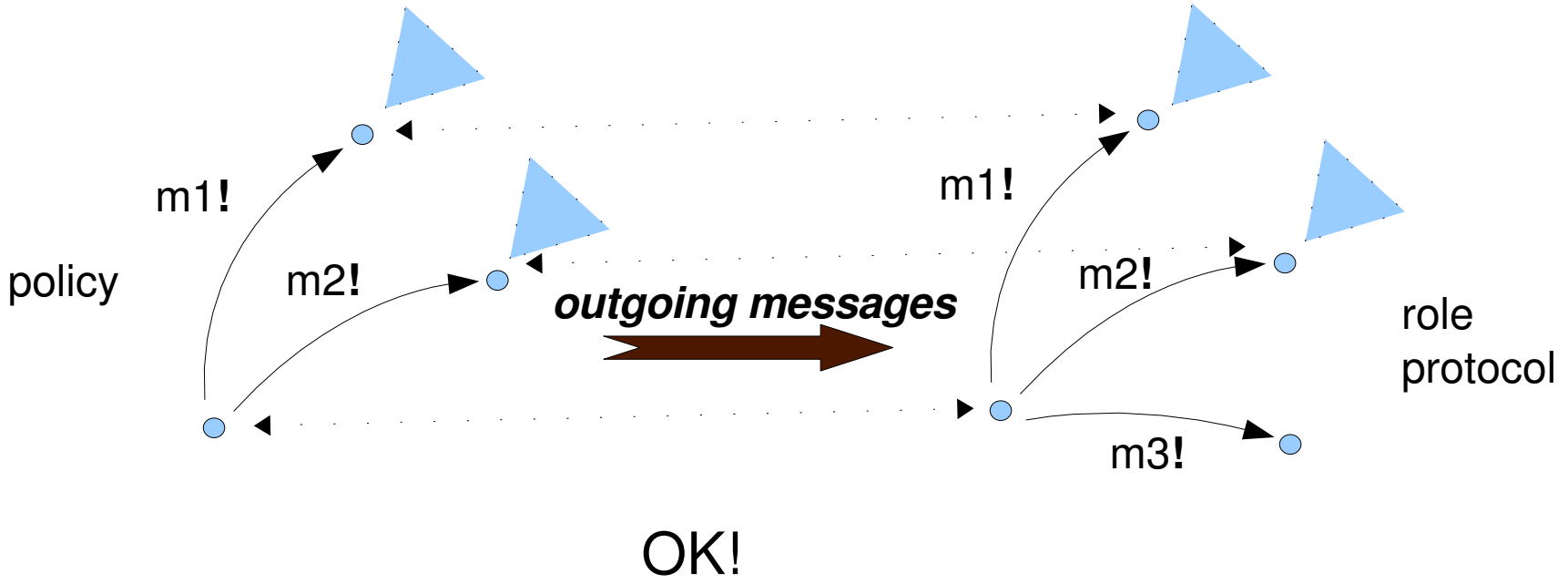
# Action and Re-Action

- An outgoing message is interpreted *an action* while an incoming message is interpreted as *a re-action*
- Sending a message implies a decision to perform an action while receiving a message does not
- Sending a message means *to request for an execution* of a task while receiving a message means *executing a task*

- *Asymmetric behaviour* w.r.t. the standard interpretation for sending and receiveing messages in process algebra and (bi)simulation techniques

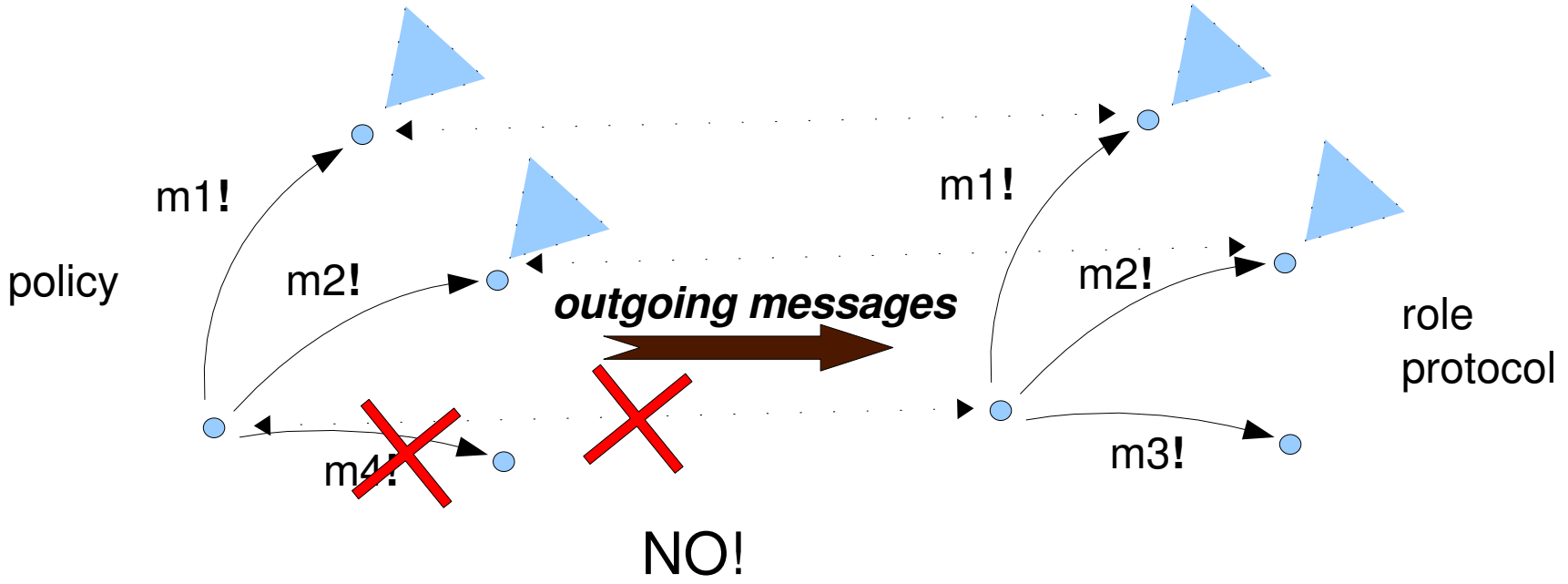
# Conformant simulation

- ***Our proposal***: an *asymmetric simulation* for dealing with outgoing and incoming messages in a different way



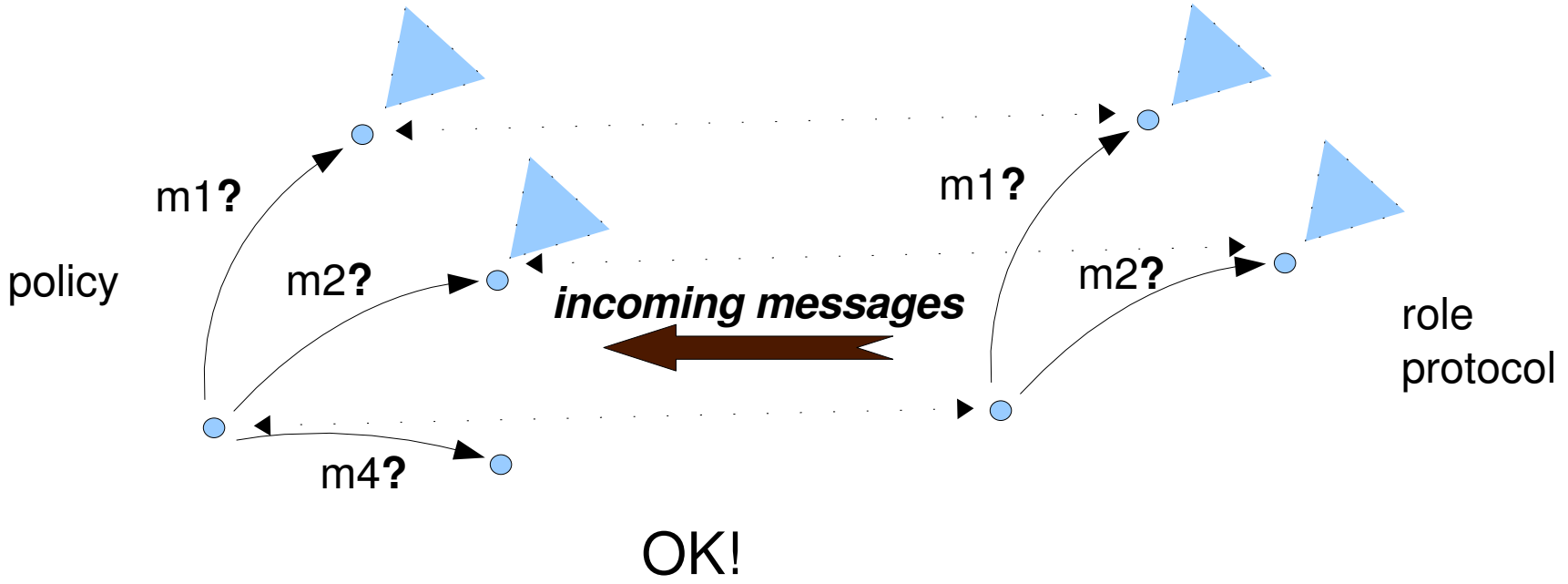
# Conformant simulation

- ***Our proposal***: an *asymmetric simulation* for dealing with outgoing and incoming messages in a different way



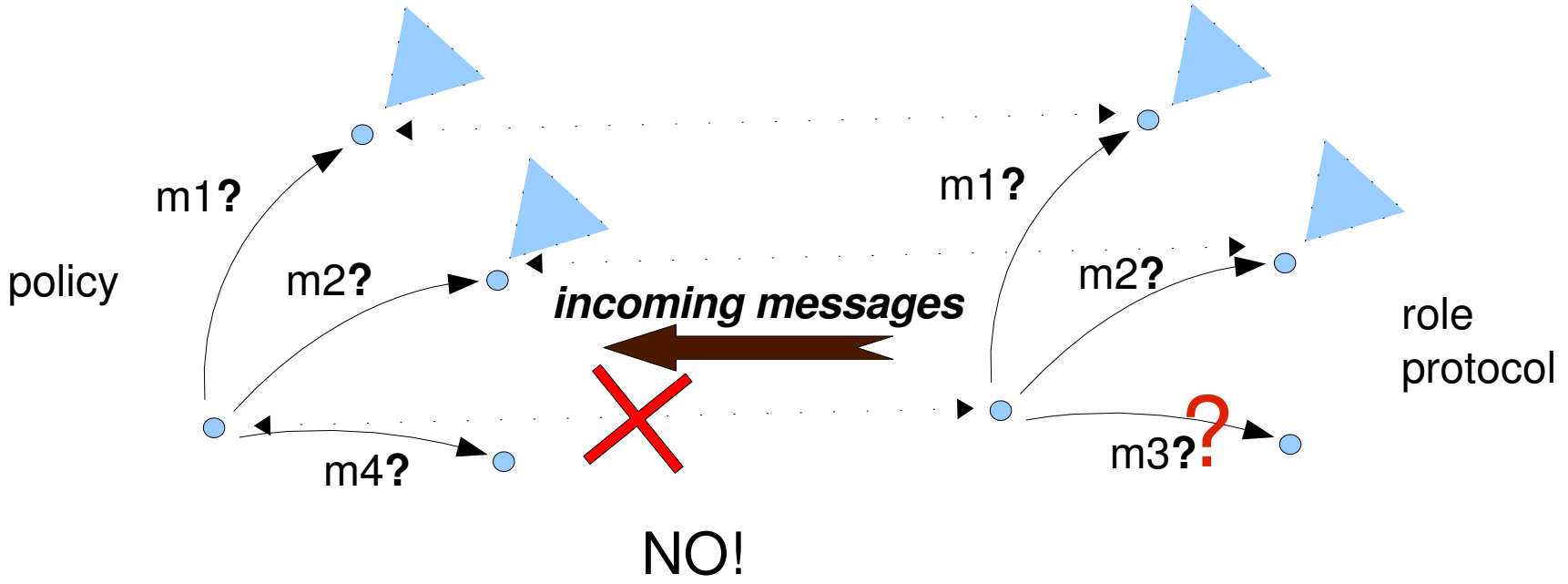
# Conformant simulation

- ***Our proposal***: an *asymmetric simulation* for dealing with outgoing and incoming messages in a different way



# Conformant simulation

- ***Our proposal***: an *asymmetric simulation* for dealing with outgoing and incoming messages in a different way



# Conformant simulation

- ***Our proposal***: an *asymmetric simulation* for dealing with outgoing and incoming messages in a different way
- ***Conformance simulation***

  *Given two FSA's $A_1$ and $A_2$. $A_1$ is confromant simulation of $A_2$, $A_1 \leq A_2$ iff there is a binary relation R between $A_1$ and $A_2$ such that*

  - $A_1.s_0$ *R* $A_2.s_0$
  - *for every outgoing message m! in $A_1.L$ and for every state $s_i$ in $A_1.S$, for every $s_j$ in $A_2.S$ s.t. $s_i$ R Sj and $(s_i, m!, s_{i+1})$ in $A_1.T$, then there is a state $s_{j+1}$ in $A_2.S$ s.t. $(s_j, m!, s_{j+1})$ in $A_2.T$ and $s_{i+1}$ R $s_{j+1}$*
  - *for every incoming message m? in $A_2.L$ and for every state $s_j$ in $A_2.S$, for every $s_i$ in $A_1.S$ s.t. $s_i$ R sj and $(s_j, m?, s_{j+1})$ in $A_2.T$, then there is a state $s_{i+1}$ in $A_1.S$ s.t. $(s_i, m!, s_{i+1})$ in $A_1.T$ and $s_{i+1}$ R $s_{j+1}$*

# Conformant simulation

**"$A_1 \leq A_2$" *does not entail* "$L(A_1) \subseteq L(A_2)$"**

- However, with "$A_1 \leq A_2$" we capture the fact that A1 will actually produce *a subset of the conversations foreseen by the role, when interacting with entities that play the other roles in the protocol*

- ***Proposition***
  Let $A_1 \otimes \ldots \otimes A_n$ be a protocol, and let $A_1', \ldots, A_n'$ be n policies s.t. $A_i' \leq A_i$, for i=1,...,n, then
  $A_1' \otimes \ldots \otimes A_n' \subseteq A_1 \otimes \ldots \otimes A_n$

# Complete conformance simulation

- We assume that in a protocol it is always possible to conclude a conversation whatever the point at which the interaction arrived. We expect a similar property to hold also for a set of conformant policies

**Definition 8 (Complete conformant simulation).** *Given two FSA's $A_1$ and $A_2$ we say that $A_1$ is a* complete conformant simulation *of $A_2$, written $A_1 \trianglelefteq A_2$, iff there is a $A_1$ is a conformant simulation of $A_2$ under a binary relation $\mathcal{R}$ and*

- *for all $s_i \in A_1.F$ such that $s_i \mathcal{R} s_j$, then $s_j \in A_2.F$;*
- *for all $s_j \in A_2.S$ such that $s_j$ is alive and $s_i \mathcal{R} s_j$, $s_i \in A_1.S$, then $s_i$ is alive.*

**Theorem 1 (Interoperability).** *Let $A_1 \otimes \cdots \otimes A_n$ be a protocol and let $A'_1, \ldots, A$ be $n$ policies such that $A'_i \trianglelefteq A_i$, for $i = 1, \ldots, n$. For any common string $\overline{\sigma'}$ of $A'_1 \otimes \cdots \otimes A'_n$ and $A_1 \otimes \cdots \otimes A_n$ there is a run $\sigma'\sigma''$ such that $\overline{\sigma'\sigma''}$ is an accepted string of $A'_1 \otimes \cdots \otimes A'_n$.*

***They will be able to conclude their interaction producing a legal accepted run***

# Conclusions and future work

- We face the problem of verifying the interoperability of a set of peers by exploiting an abstract description of the desired interaction: *static interoperability checking*

- Connections to standards like WS-CDL and BPEL
- A calculus for the “a-priori” interoperability test
- Can an agent/peer, having its own strategies, profitably take part to a “game”? In other words, does an agent/peer have the capabilities that allow it ***to reach its own goal by playing a given role***? [PPSWR 06, SOT-WEB 06]

# Conformant simulation

**Definition 1 (Finite State Automaton).** *A finite state automaton is a tuple* $(S, s_0, L, T, F)$, *where* $S$ *is a finite set of states,* $s_0 \in S$ *is a distinguished initial state,* $L$ *is a finite set of labels,* $T \subseteq (S \times L \times S)$ *is a set of transitions,* $F \in S$ *is a set of final states.*

**Definition 2 (Runs and strings).** *A run* $\sigma$ *of a FSA* $(S, s_0, L, T, F)$ *is an ordered, possibly infinite, set of transitions (a sequence)* $(s_0, l_0, s_1), (s_1, l_1, s_2), (s_2, l_2, s_3), \ldots$ *such that* $\forall i \geq 0$, $(s_i, l_i, s_{i+1}) \in T$, *while the sequence* $l_0 l_1 \ldots$ *is the corresponding string* $\overline{\sigma}$.

**Definition 3 (Acceptance).** *An accepting run of a finite state automaton* $(S, s_0, L, T, F)$ *is a finite run* $\sigma$ *in which the final transition* $(s_{n-1}, l_{n-1}, s_n)$ *has the property that* $s_n \in F$. *The corresponding string* $\overline{\sigma}$ *is an accepted string.*

# Conformant simulation

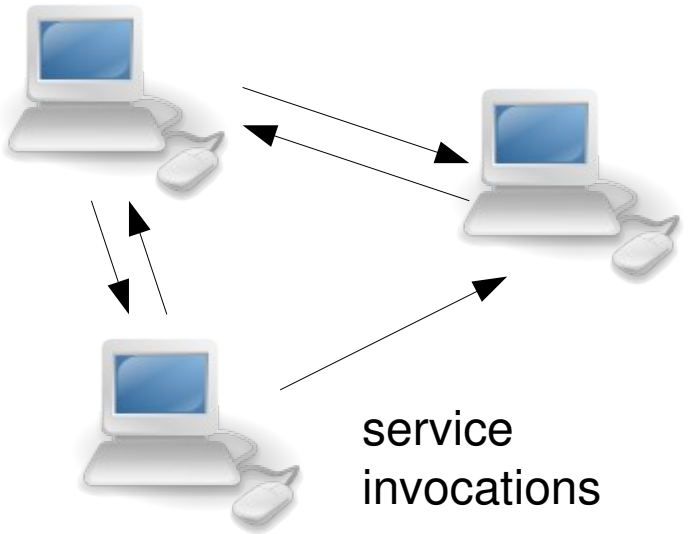
**Definition 4 (Free product).** *Let $A_i$, $i = 1, \ldots, n$, be $n$ FSA's. The* free product $A_1 \times \cdots \times A_n$ *is the FSA* $A = (S, s_0, L, T, F)$ *defined by:*

- $S$ *is the set* $A_1.S \times \cdots \times A_n.S$;
- $s_0$ *is the tuple* $(A_1.s_0, \ldots, A_n.s_0)$;
- $L$ *is the set* $A_1.L \times \cdots \times A_n.L$;
- $T$ *is the set of tuples* $((A_1.s_1, \ldots, A_n.s_n), (l_1, \ldots, l_n), (A_1.s'_1, \ldots, A_n.s'_n))$ *such that* $(A_i.s_i, l_i, A_i.s'_i) \in A_i.T$, *for* $i = 1, \ldots, n$; *and*
- $F$ *is the set of tuples* $(A_1.s_1, \ldots, A_n.s_n) \in A.S$ *such that* $s_i \in A_i.F$, *for* $i = 1, \ldots, n$.

**Definition 5 (Synchronous product).** *Let $A_i$, $i = 1, \ldots, n$, be $n$ FSA's. The* synchronous product *of the $A_i$'s, written $A_1 \otimes \cdots \otimes A_n$, is the FSA obtained as the free product of the $A_i$'s containing only the transitions* $((A_1.s_1, \ldots, A_n.s_n), (l_1, \ldots, l_n), (A_1.s'_1, \ldots, A_n.s'_n))$ *such that there exist $i$ and $j$,* $1 \leq i \neq j \leq n$, $l_i = \mathsf{m}!$, $l_j = \mathsf{m}?$, *and for any $k$ not equal to $i$ and $j$,* $l_k = \varepsilon$.

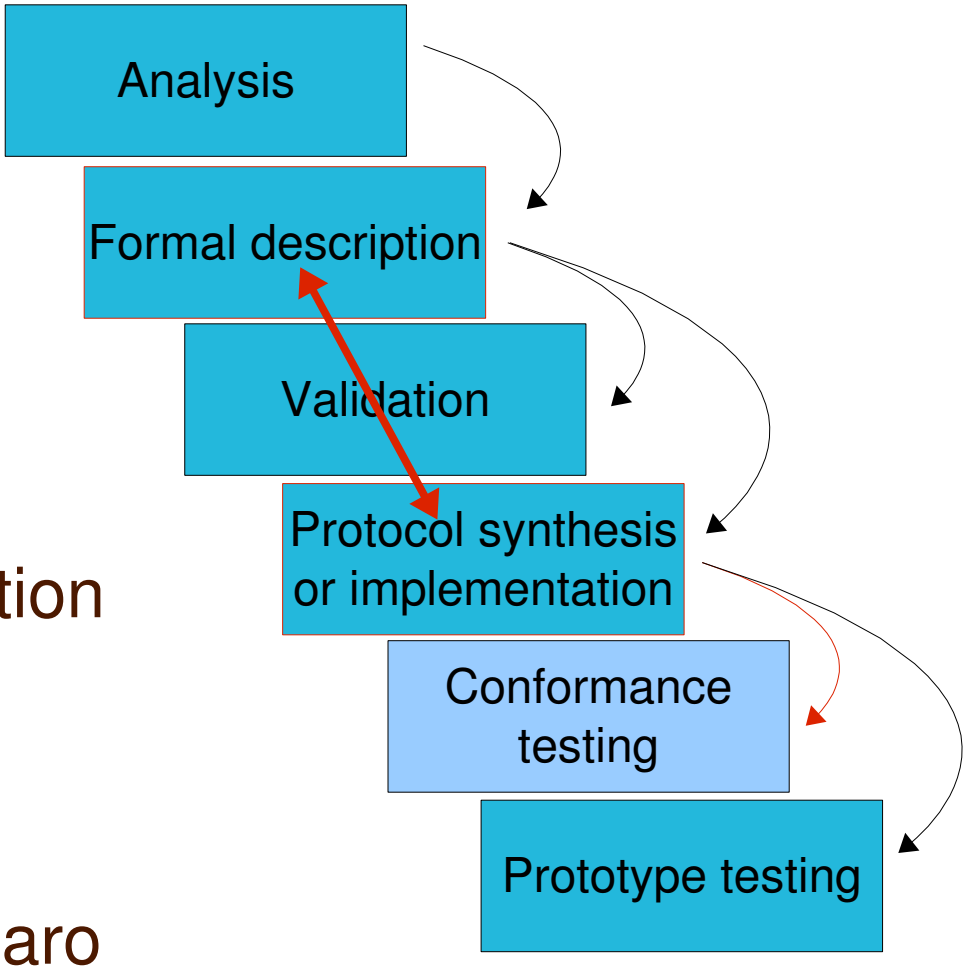
# WS orchestration and choreography

- Choreography: global point of view/abstract protocol, eg. *WS-CDL language*
- Behavioral interface: local point of view/policy, eg. *BPEL abstract process*
- Orchestration: describes both communicative and non-communicative behavior allowing execution, eg. *BPEL executable processes*



# Conformance in interaction protocols engineering

- To check if a policy (protocol subjective implementation) verifies the AUML specification, the FSM specification, the Social Commitments specification, etc.
- It is analogous to the validation phase but it concerns the program and not the formal description
- *Automatic synthesis* [Zavattaro et al., van der Aalst et al.]



# Conformance test: *our expectations*

- Any message that can be sent, at any point of the execution, will be handle by one of its interlocutors
- Whatever point of conversation has been reached, there is a way to bring it to the end
- A conversation is *legal w.r.t. a protocol* if it respects the specifications given by the protocol (it is an execution trace allowed by the protocol)