

Un' introduzione al Progetto SOCS: formalizzazione e verifica di protocolli di comunicazione e sue applicazioni.

Paola Mello - DEIS

SOCS:

a computational **logic model** for the description,
analysis and verification of global and open
Societies Of heterogeneous Computees

SOCS home page:

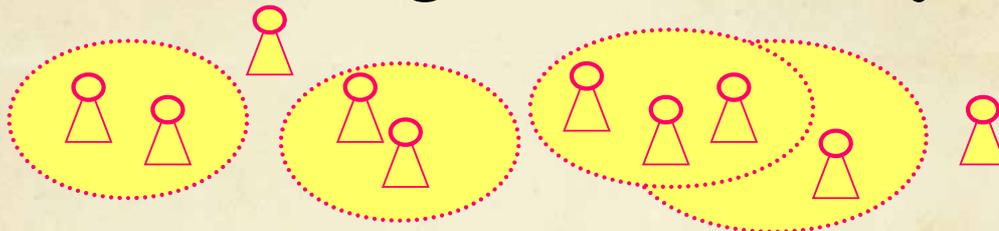
<http://lia.deis.unibo.it/research/socs/>

Societies Of Computees (SOCS)

- 3 years project (end June 2005)
- Funded by EU
- Partners:
 - University of Bologna
 - University of Ferrara
 - University of Pisa
 - University of Cyprus
 - Imperial College London
 - City University London



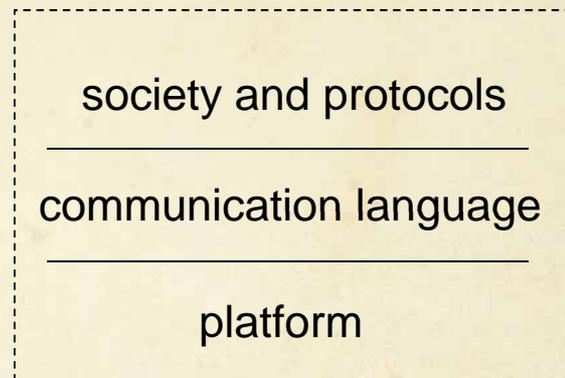
SOCS: Agents in a society



- Societies are groups of interacting agents (possibly with a common goal)
 - interactions are unconstrained (openness)
 - the semantics of interaction is defined in terms of protocols and expectations
 - the 'social' behaviour of computees can be observed
 - to give it an institutional meaning
 - to verify compliance to the protocols
 - to raise expectations, violations, sanctions
- We will focus now on social aspects

Basic architecture

- Data structures:
 - *SOKB* (Social Organization Knowledge Base)
 - *SEKB* (Social Environment Knowledge Base)
 - Social Integrity Constraints
 - Goals
- Roles (duties and capabilities)
- Entry / exit rules
- Semantics of interaction
 - at the protocol level
 - at the communication level
- Verification of interaction



layered architecture

Social events

Happened events come from the outer world.

Happened events that are “socially relevant” are recorded by the society infrastructure

$H(Event[, Time])$

Once they are recorded, they become part of the SEKB

The history of happened events is the set

$HAP = \{ H(Event, Time) \}$

Expectations

- Events can give raise to expectations in the society:

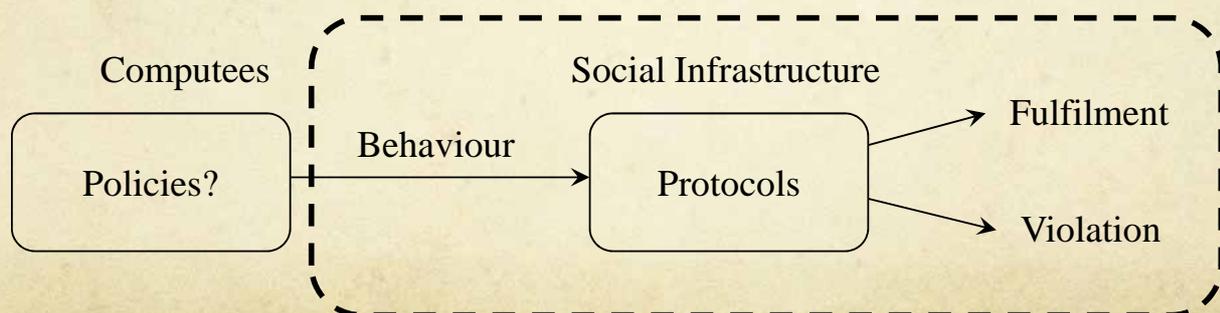
$E(Event[, Time])$

$EN(Event[, Time])$

- *Positive* expectations: events that are expected to occur
- *Negative* expectations: events that are *expected not to occur*
- The state of expectations is a conjunction EXP containing
 - literals of the form $(\neg)E(Event, Time)$
 - literals of the form $(\neg)EN(Event, Time)$
 - *Constraint Formulas* on the variables occurring in the other literals present in EXP

Protocols

- Computees behave according to their own policies
- Social expectations can be used:
 - to check the correct functioning of the society
 - to suggest to the computees a course of actions
- Protocols are defined through Social Integrity Constraints:
- The society generates expectations out of protocols & events



Social Integrity Constraints (SICs)

○ SICs ::= $[\chi \rightarrow \varphi]^*$

χ ::= $(\neg)\mathbf{H}(\text{Event } [, \text{Time}])$

φ ::= $\vee \{ \wedge (\neg)\mathbf{E}/\mathbf{NE}(\text{Event } [, \text{Time}]) / \text{constraints} \}$

SICs Examples

“If I make you an offer, you must answer me accepting or refusing before a deadline d”

$\mathbf{H}(\text{tell}(\text{Me}, \text{You}, \text{offer}(\text{Item}, \text{Price}), \text{T}), \text{T}) \rightarrow$

$\mathbf{E}(\text{tell}(\text{You}, \text{Me}, \text{accept}(\text{Item}, \text{Price}), \text{T}'), \text{T}' \leq \text{T} + \text{d}) \vee$

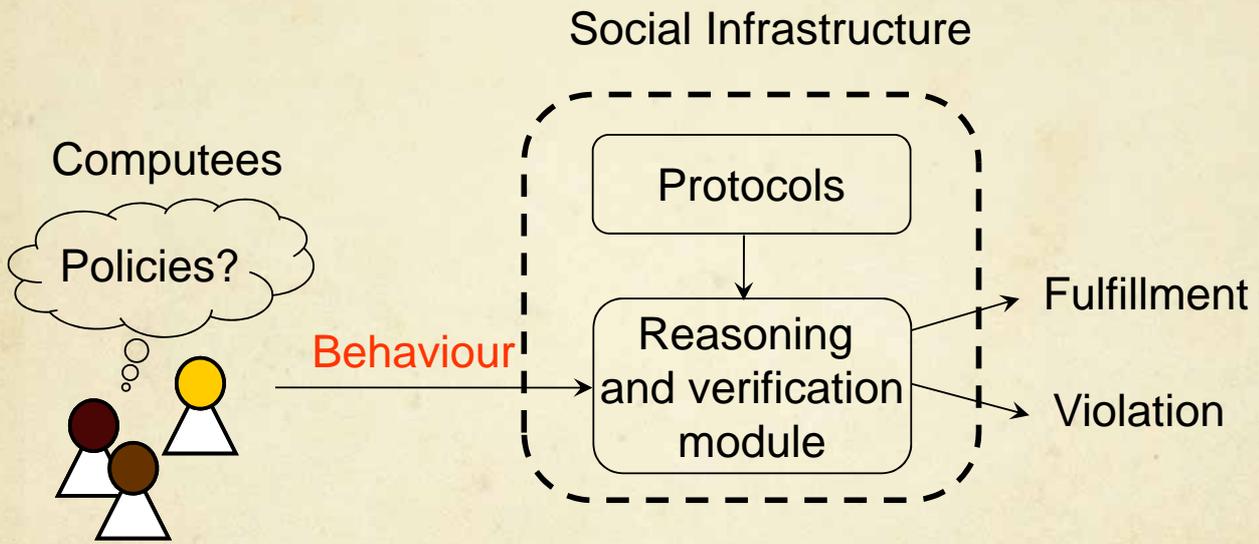
$\mathbf{E}(\text{tell}(\text{You}, \text{Me}, \text{refuse}(\text{Item}, \text{Price}), \text{T}'), \text{T}' \leq \text{T} + \text{d})$

“If you accept my offer, you cannot refuse it later”

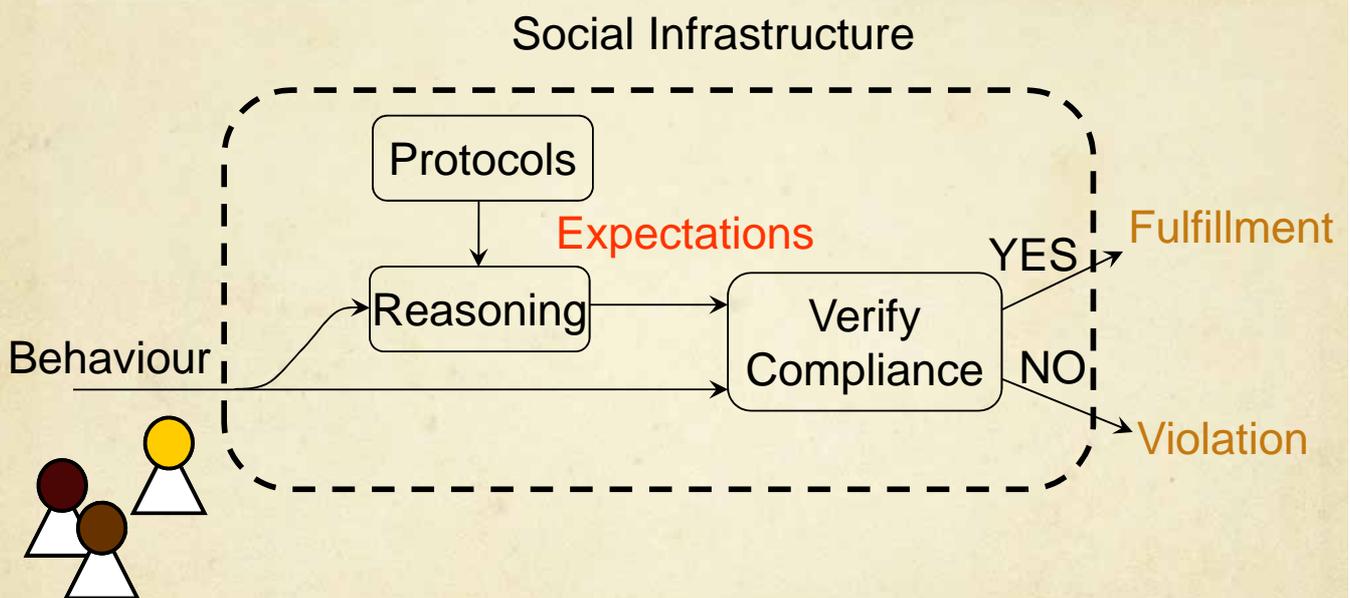
$\mathbf{H}(\text{tell}(\text{You}, \text{Me}, \text{accept}(\text{Item}, \text{Price}), \text{T}), \text{T}) \rightarrow$

$\mathbf{EN}(\text{tell}(\text{You}, \text{Me}, \text{refuse}(\text{Item}, \text{Price}), \text{Tr}), \text{Tr} \geq \text{T})$

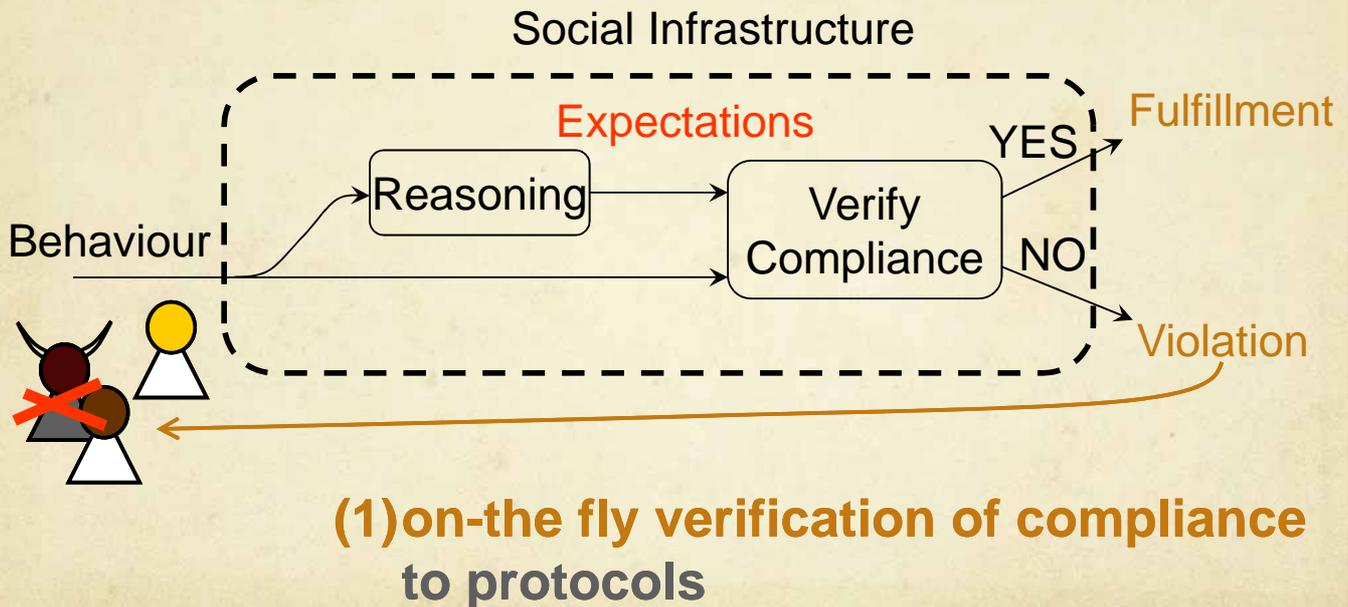
Compliance Verification



Social infrastructure



Social infrastructure



Social Integrity Constraints (ICs)

- Example of Social Integrity Constraint

Society where agents can **exchange resources**:

*If I make you an offer, you are expected to answer to me by **either accepting or refusing** before a deadline d*

$H(\text{tell}(\text{Me}, \text{You}, \text{offer}(\text{Item}, \text{Price}), T), T) \rightarrow$

$E(\text{tell}(\text{You}, \text{Me}, \text{accept}(\text{Item}, \text{Price}), T'), T' \leq T + d) \vee$

$E(\text{tell}(\text{You}, \text{Me}, \text{refuse}(\text{Item}, \text{Price}), T'), T' \leq T + d'$

If you accept my offer, you are expected to not refuse it later

$H(\text{tell}(\text{You}, \text{Me}, \text{accept}(\text{Item}, \text{Price}), T), T) \rightarrow$

$EN(\text{tell}(\text{You}, \text{Me}, \text{refuse}(\text{Item}, \text{Price}), T_r), T_r \geq T$

Example (fulfilment)

yves



→ H(tell(yves,thomas,offer(scooter,10\$),1)

thomas



E(tell(thomas,yves,accept(scooter,10\$),T'), T' < 7

∨

E(tell(thomas,yves,refuse(scooter,10\$),T'), T' < 7

H(tell(thomas,yves,accept(scooter,10\$),5) ←

fulfillment!

Example (violation)

yves (bidder)



→ H(tell(yves,thomas,bid(scooter,10\$),1)

thomas
(auctioneer)

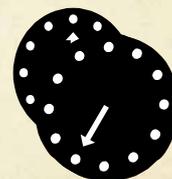


E(tell(thomas,yves,win(scooter,10\$),T'), T' < 7

∨

E(tell(thomas,yves,lose(scooter,10\$),T'), T' < 7

violation!



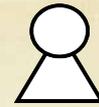
Example (violation)

yves



→ H(tell(yves,thomas,offer(scooter,10\$),1)

thomas



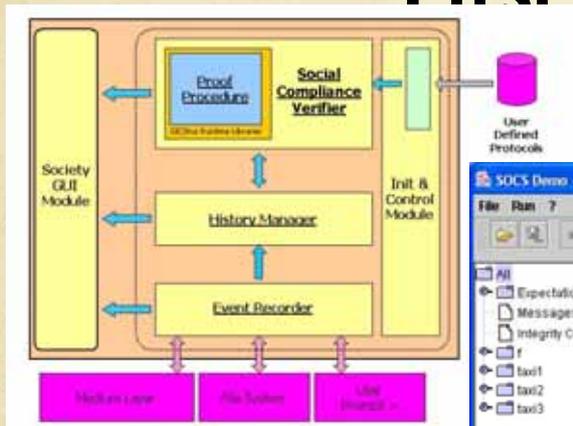
H(tell(thomas,yves,accept(scooter,10\$),5) ←

H(tell(thomas,yves,accept(Item,Price), T) ∨
EN(tell(thomas,yves,refuse(Item,Price), Tr), Tr>=T

H(tell(thomas,yves,refuse(scooter,10\$),8) ←

violation!

First Prototype



SocIDs	contextId	Sender	Receivers	Performative	Content	Time	Local Time
s0	action1	f	taxi1	current_time	taxi2station, 10	4:1073926617967	
s0	action1	f	taxi2	openauction	taxi2station, 10	5:1073926617997	
s0	action1	f	taxi3	openauction	taxi2station, 10	5:1073926618038	
s0	action1	f	taxi1	current_time	taxi2station, 3	5:1073926618058	
s0	action1	taxi1	f	bid	taxi2station, 3	6:1073926618068	
s0	action1	taxi2	f	current_time	taxi2station, 5	7:1073926618069	
s0	action1	taxi2	f	bid	taxi2station, 5	7:1073926618098	
s0	action1	taxi2	f	current_time	taxi2station, 5	8:1073926618099	
s0	action1	taxi2	f	current_time	taxi2station, 5	8:1073926618108	

Pointers to SOCS

- SOCS home page:

[SOC] <http://lia.deis.unibo.it/research/socs/>

- Publications:

- **SOCS deliverables** (contact me)
- Conferences: JELIA'02, UKMAS'02, CEEMAS'03, AAMAS'03, IJCAI'03, **AI*IA'03** (*Friday, Session 11, 10.45-13.20*) ecc.
- Workshops: DALT'03, CLIMA'02, ESAW'03, LCMAS'03 (see LNAI e ENTCS), FAMAS'03, MFI'03, PSE'03, ESAW'05 ecc.

- Projects: National, Spinner, PRITT ...

SOCS & SOCS-SI

- Sito del progetto: <http://www.lia.deis.unibo.it/Research/Projects/SOCS/>

- SCIFF Proof Procedure:

<http://lia.deis.unibo.it/research/sciff/>

- Applicativo SOCS-SI:

http://www.lia.deis.unibo.it/research/socs_si/socs_si.shtml

- Alcuni protocolli disponibili su web:

<http://www.lia.deis.unibo.it/research/socs/partners/societies/protocols.html>

Tesi e collaborazioni

- Possibili sotto forma di:
 - Tesi
 - Tirocini

Sviluppo e messa a punto del prototipo,

Scrittura di protocolli

Dimostrazione di proprietà di protocolli

Applicazioni al campo medico, sicurezza, e-learning, TCP/IP,
Composizione di Web Servers, traduzione di WS-CDL,
BPEL.



Applicazioni di SCIFF

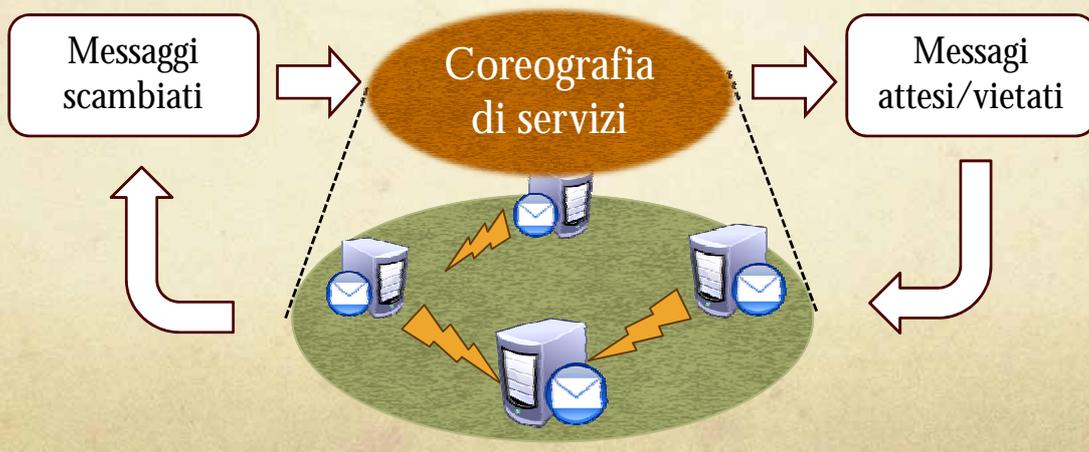
L'idea di fondo

- SCIFF astrae dai singoli partecipanti focalizzandosi sull'interazione e sugli eventi osservabili e rilevanti
- Quindi può essere applicata, in generale, a “sistemi basati su eventi”



L'idea di fondo

- SCIFF astrae dai singoli partecipanti focalizzandosi sull'interazione e sugli eventi osservabili e rilevanti
- Quindi può essere applicata, in generale, a “sistemi basati su eventi”



Applicazioni e linee di ricerca

1/2

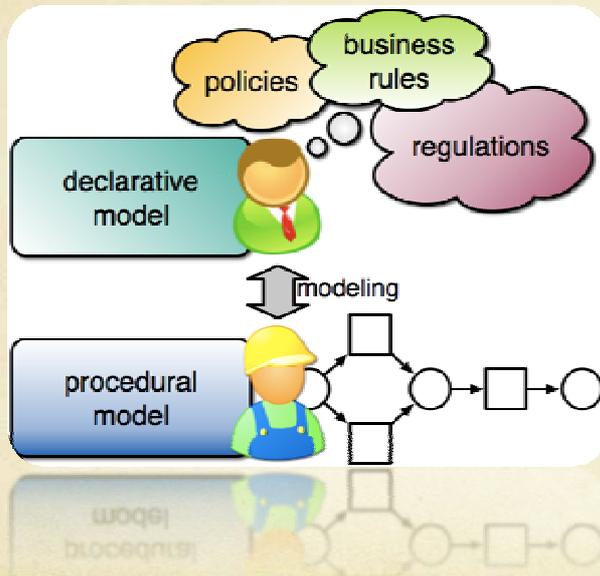
- Modellazione e verifica di processi aziendali
 - Specifica di processi flessibili e loosely-coupled (basati su vincoli/regole/politiche)
 - Monitoring e verifica di compliance
 - Verifiche statiche (consistenza/feasibility dei modelli)
 - Applicazioni in campo medico (linee guida)
 - Mining di processi loosely-coupled
- Formalizzazione e verifica di modelli per l'ingegneria dei requisiti

Applicazioni e linee di ricerca

2/2

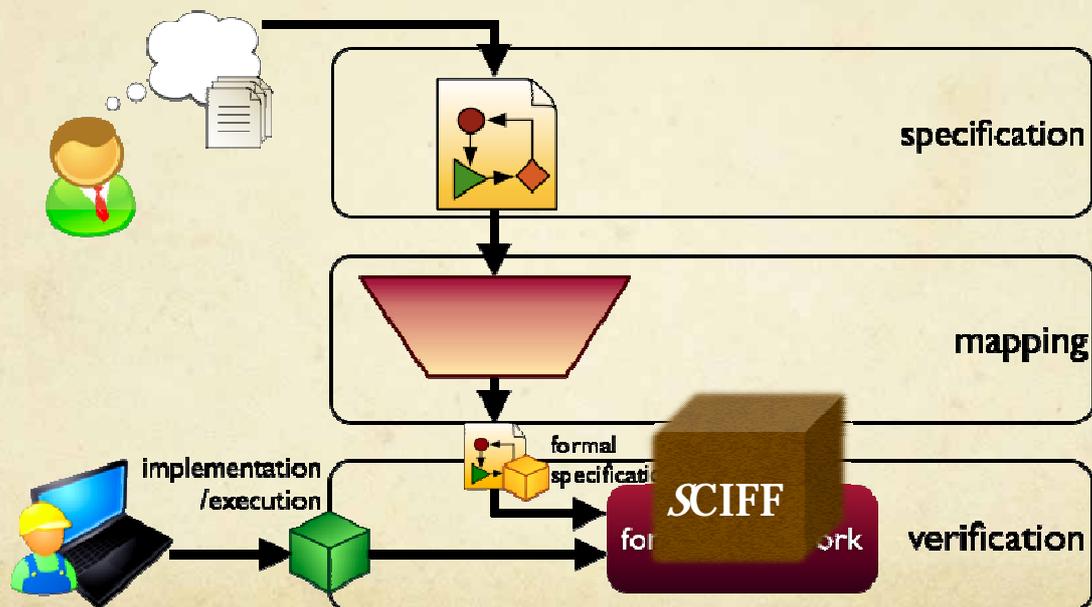
- Service Oriented Computing
 - Modellazione e verifica di coreografie (vedi sopra)
 - Discovery di servizi
 - Argumentation per la contrattazione tra servizi
 - Integrazione con i commitments
 - Integrazione con ontologie (web semantico)
- Model transformation per passare da linguaggi di specifica alla corrispondente formalizzazione SCIFF
- Linee di ricerca con aspetti fondazionali
 - Integrazione del calcolo degli eventi in SCIFF
 - E, di conseguenza, dei social commitments!
 - Confronto con model checking e logiche temporali

Modello inteso vs realizzazione



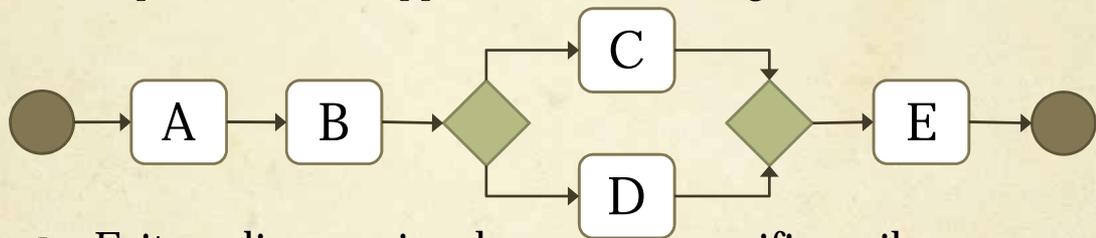
- Dato un problema...
- Modello procedurale: mira a sintetizzare una specifica soluzione
- Modello dichiarativo: si focalizza sul problema (COSA) senza specificare COME risolverlo
 - Algoritmo general-purpose!

Specifica e Verifica



Processi basati su vincoli

- Problema: i classici “production workflows” tendono a costringere troppo gli utenti (esperti del dominio)
 - Soprattutto nelle applicazioni di ultima generazione

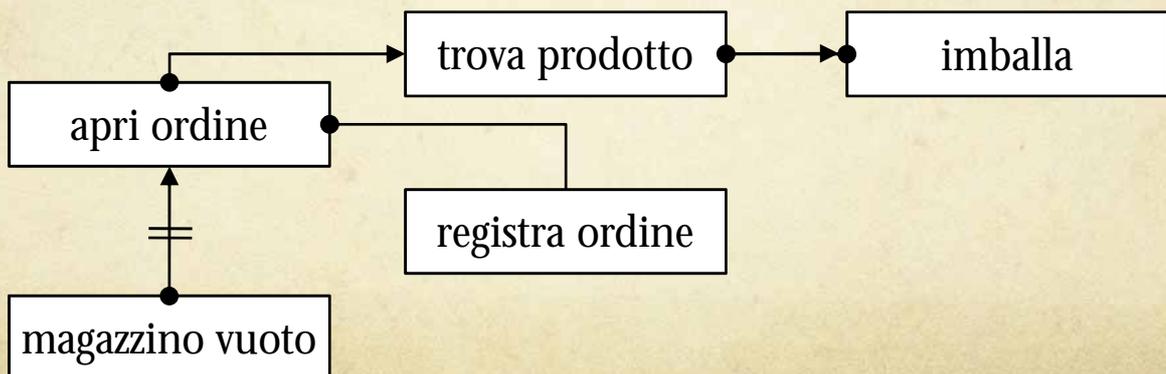


- Evitare di sovra-vincolare e sovra-specificare il processo è fondamentale
- Possibile: adottare linguaggi (e motori di esecuzione) più flessibili

ConDec/DecSerFlow

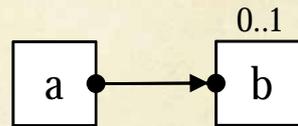
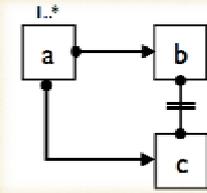
[van der Aalst and Pesic, 2006]

- Linguaggio grafico DICHIARATIVO per la modellazione di processi flessibili
- Non più un flusso, ma una serie di vincoli sui flussi accettati/vietati



Problematiche

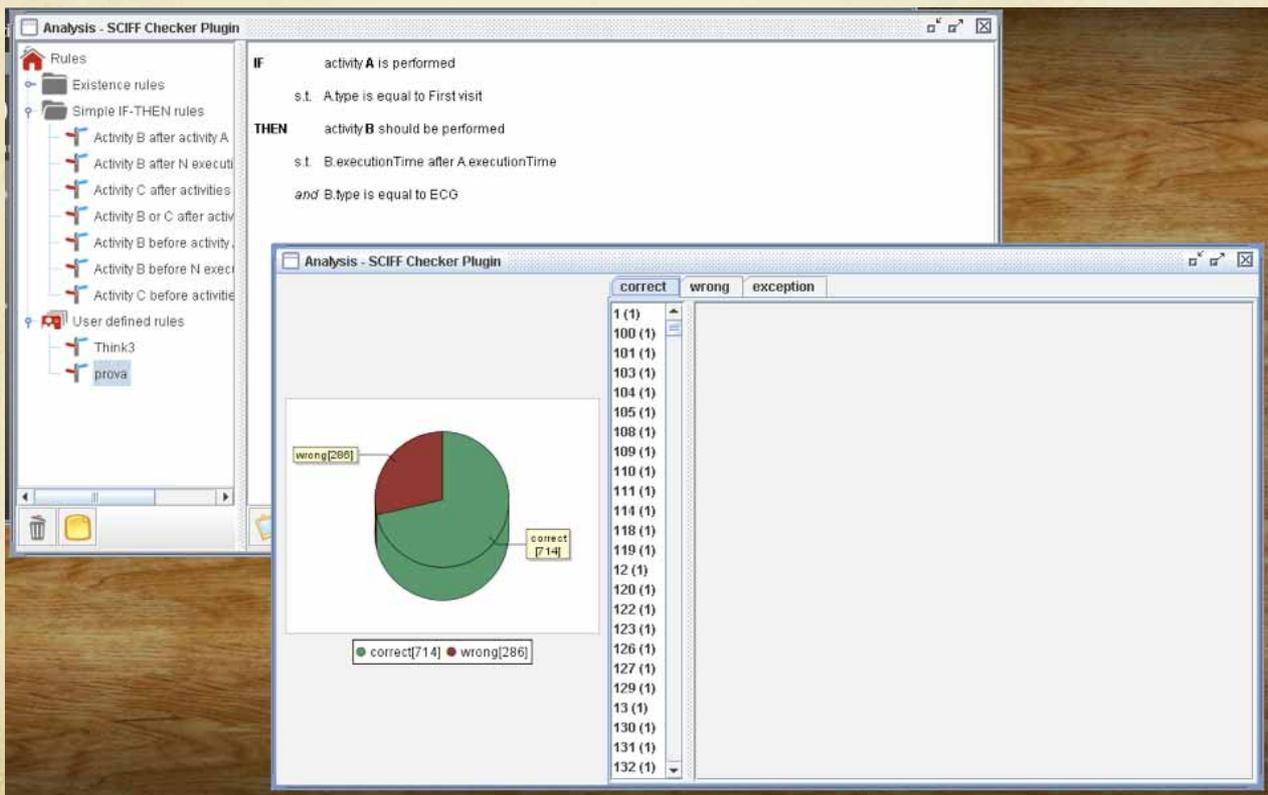
- Verifica dei modelli prodotti
 - Il modello che ho creato è consistente?
- Verifica di esecuzioni rispetto al modello
 - Una specifica esecuzione del processo sta rispettando i vincoli?
- Esecuzione dei modelli
- Estensioni (vincoli temporali, dati, ...)
- **Necessità di formalizzare il linguaggio (prima di tutto!)**
 - ConDec è traducibile in logica temporale e SCIFF



ConDec e SCIFF

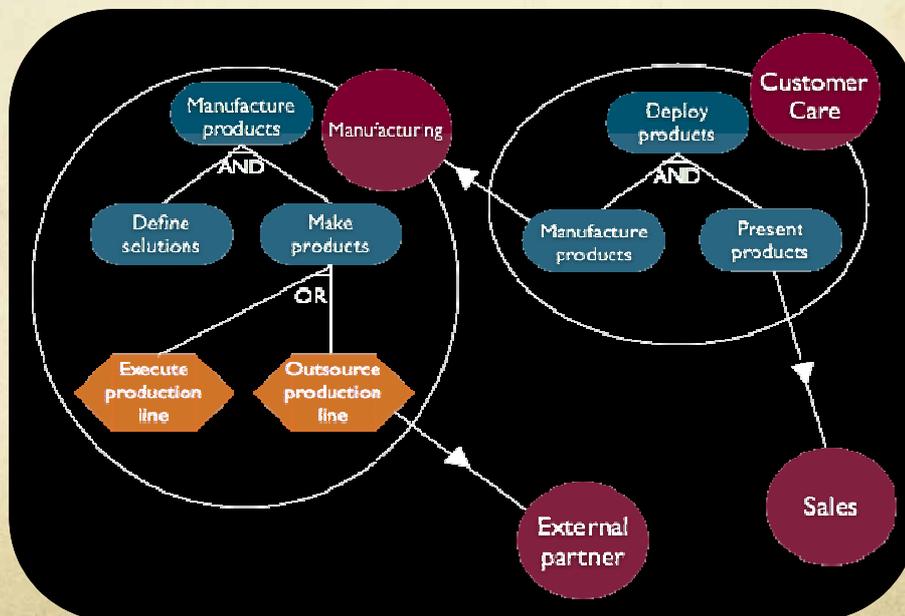
- SCIFF è capace di formalizzare tutti i vincoli ConDec, e di estenderli con vincoli temporali quantitativi
- Attività di ricerca (e di tesi)
 - Diverse tipologie di verifica (statica, dinamica, a posteriori per classificare tracce di esecuzione)
 - Model transformation per automatizzare la traduzione
 - Relativo editor (in ECLIPSE?)
 - SCIFF come motore di esecuzione
 - Confronto SCIFF – logica temporale
 - Mining di modelli ConDec a partire da tracce di esecuzione, utilizzando tecniche di ILP

Analisi dei log in ProM



Ingegneria dei requisiti

- Tropos: linguaggio di modellazione orientato agli agenti con particolare enfasi sui requisiti



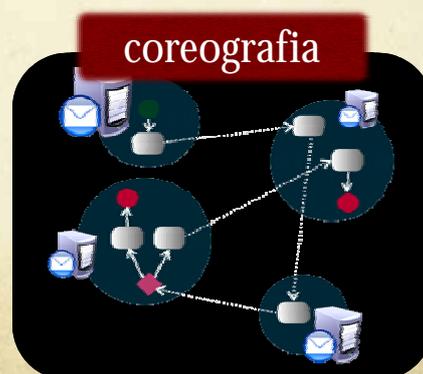
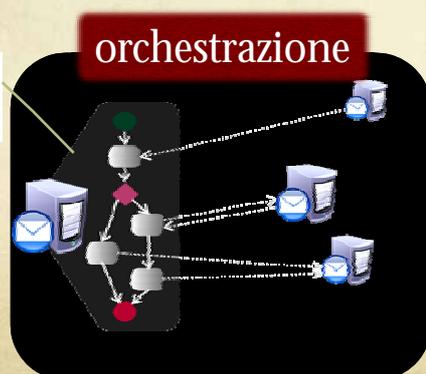
Tropos e SCIFF

- Gli alberi degli obiettivi si possono modellare come programmi Prolog (sono alberi AND/OR), le deleghe come regole reattive SCIFF
- Si aprono varie possibilità
 - Verifica della consistenza di un modello, magari estendendolo con durate temporali, vincoli di processo, eventi, ...
 - Realizzare meccanismi di ragionamento per scoprire se la specifica di un attore “specializza” una specifica astratta

Service Oriented Computing



Behavioral interface



SCIFF e Coreografie

- Specifica e monitoring di coreografie utilizzando DecSerFlow e SCIFF
 - E relativi meccanismi di intercettazione dei messaggi
- Definizione e verifiche di diversi criteri interoperabilità e conformance
 - L'interfaccia comportamentale di questo servizio può ricoprire un certo ruolo nella coreografia secondo il concetto di conformance definito?

SCIFF e discovery di servizi

- “Scoperta” di servizi
 - Al fine di coprire un certo ruolo in una coreografia o sintetizzare una composizione
 - Al fine di soddisfare i desideri/requisiti di un utente (search engine)
- Diversi aspetti
 - Semantici (sto cercando “libri” e un servizio vende “romanzi”...)
 - Comportamentali
 - Prendiamo in considerazione come il servizio si comporta, quali sono le politiche dell'utente (sono disposto a comprare con carta di credito a patto che tu sia un venditore certificato...)
- SCIFF Discovery Engine
 - Esplora questi diversi aspetti per realizzare un motore di ricerca “intelligente” per servizi
 - Aspetti sia formali (integrazione SCIFF-ragionamento ontologico) che pratici (implementazione concreta del motore)

Argumentation e servizi

- Argumentation: framework per ragionare ad alto livello su dialoghi, tesi contrapposte, argomentazioni che le sostengono e refutazioni
- Se immaginiamo servizi dotati di capacità di ragionamento
 - Possiamo immaginare le interazioni come “dialogo”
 - I servizi “discutono” per contrattare reciprocamente bisogni, desideri, requisiti
 - Quindi adattano le proprie politiche di esecuzione in base al proprio interlocutore

Aspetti fondazionali

- Attività di ricerca su DecSerFlow e ConDec
 - Formalizzabili in LTL e SCIFF
 - Quali sono i pregi e difetti dei due approcci?
 - Quando conviene utilizzare il primo, quando il secondo?
 - Che relazioni intercorrono in termini di espressività, capacità di ragionamento, complessità?
- Realizzazione del calcolo degli eventi in SCIFF
 - Event Calculus: permette di ragionare sugli effetti degli eventi (fluenti)
 - SCIFF: permette di ragionare in tempo reale sugli eventi che accadono e le relative aspettative
 - Applicazioni nel campo dei servizi: monitoring, integrazione di modelli a là DecSerFlow con i social commitments

Per info...

- Contattare
 - Prof. Paola Mello (paola.mello@unibo.it)
 - Prof. Paolo Torroni (paolo.torroni@unibo.it)
 - Dott. Federico Chesani (federico.chesani@unibo.it)
 - Dott. Marco Montali (marco.montali@unibo.it)
- Maggiori informazioni sulla SCIFF all'URL: <http://lia.deis.unibo.it/research/sciff/>
- Maggiori informazioni sulla parte relativa a processi e coreografie di servizi anche all'URL: <http://www-lia.deis.unibo.it/research/climb/>

Altre Applicazioni di AI per possibili tesi

- Applicazioni in campo medico (Sistemi basati sulla conoscenza), verifica di linee guida (anche in collaborazione con Noemalife S.p.A, Bologna/Anastasis);
- Sistemi di planning/scheduling per verniciatura (con CEFLA S.p.A. Imola);
- Sistemi per la verifica di log (anche con Think3 Casalecchio (BO) e CEFLA, progetto TOCAI);
- Sistemi per interpretazione di flussi turistici (progetto ministeriale PRIN07);
- Sistemi per il controllo di impianti di depurazione (ENEA);
- Semantic-Web e Semantic web/services;
- Business process, verifica e run-time monitoring.