Trust and Commitments as Unifying Bases for Social Computing

Munindar P. Singh

North Carolina State University

August 2013

Abstractions for Social Computing

- Today, social computing is viewed at a low level
  - In an ad hoc manner, in specific applications
  - Via statistical models of networks
  - Without regard to the nature of the relationships
- Proposal: model the contents of the relationships
  - Trust
  - Commitments
  - Other normative relationships, as needed

This presentation emphasizes trust

Example of Social Computing: Scientific Collaboration

Global Hybrid Profile Mooring Being Launched; Credit: Tom Kleindinst, WHOI

Example of Social Computing: Finance

Bilateral Price Discovery (*BPD*)

What do these messages mean?

- Must the *Maker* trade at the quoted price if the *Taker* accepts the price?
- Must the *Taker* trade at an accepted price even without confirmation?
- Why is it that multiple RFQs are OK but multiple accepts are not?

Example of Social Computing: Service Ecosystems

Regulation versus Regimentation

Amish rumspringa

- Regimentation: preventing bad behavior
  - Fits a closed system
  - Reflects a pessimistic stance
  - Presumes a regimenting infrastructure
- Regulation: discouraging and correcting—though *allowing*—bad behavior
  - Fits an open system
  - Reflects an optimistic stance
  - Presumes a regulating social system

Applying Trust for Social Computing

Thesis: Trust underlies *all* interactions among autonomous parties

- Trust reflects the truster's *dependence* on the trustee
  - For a purpose
  - In a context
- Currently, trust is applied
  - Embedded into each specific application
  - Not reusable
- Many types of social relationships, each nuanced
  - Casual (acquaintanceship or friendship)
  - Familial
  - Communal
  - Organizational
  - Practical (task-specific)
- How may we abstract out trust to apply it as a basis for social computing applications?

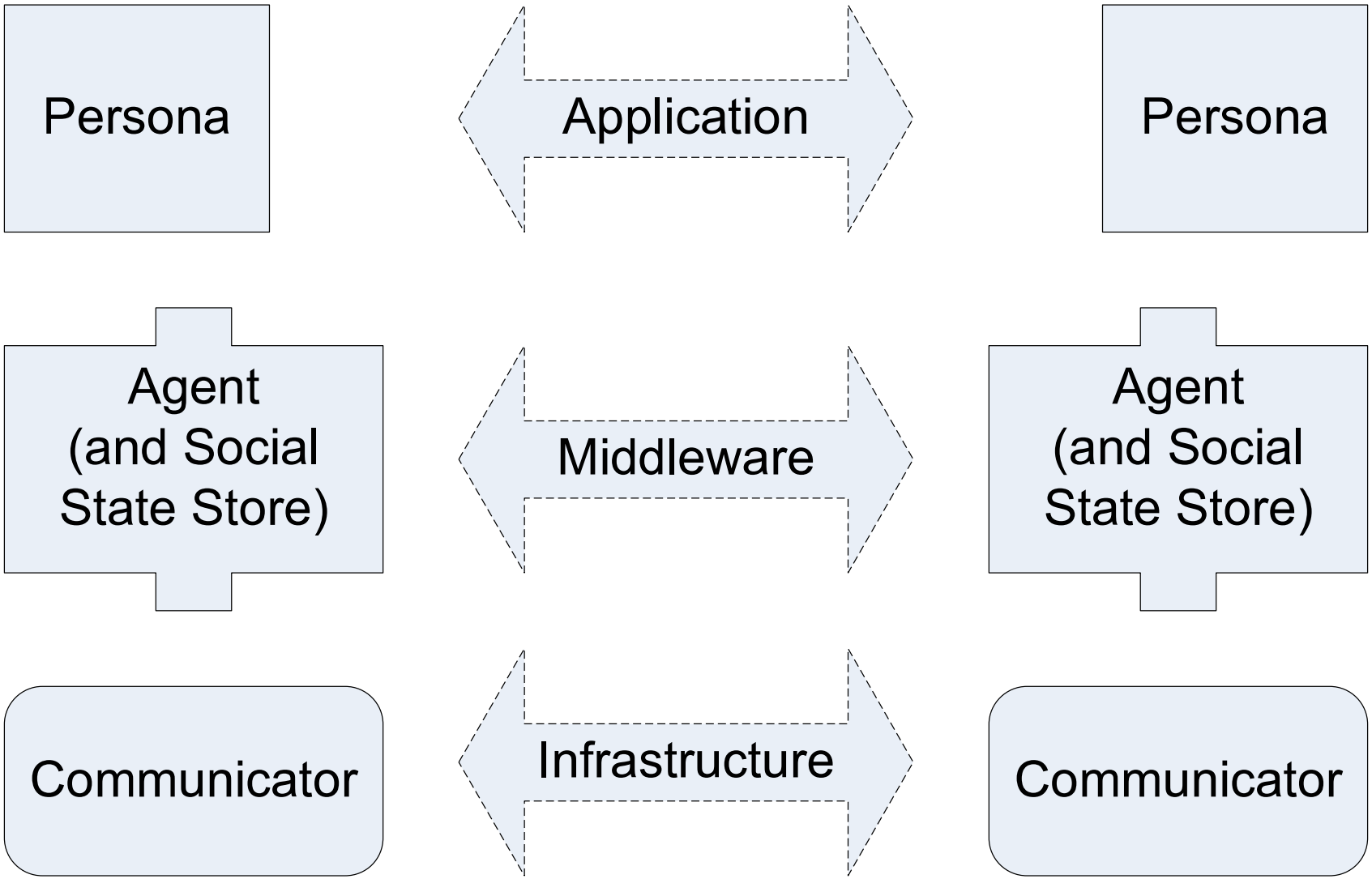
Notions of Trust

Existing literature

- *Subjective*
  - As a conglomerate of mental attitudes
- *Social*
  - Based on social relationships
- *Distributed*
  - Based on certificate chains
- *Measured* applied to any of the above
  - Based on heuristics, probabilities, utility, . . .

Traditional approaches emphasize estimation over meaning

Social Middleware to Support Social Applications



Architecture Conceptually

How a system is organized

- Primarily its ingredients
  - Components
  - Connectors
- But ideally reflecting an *architectural style*
  - Constraints on components and connectors
  - Patterns on components and connectors

Architecture: Electrical System

Components; connectors; constraints; patterns

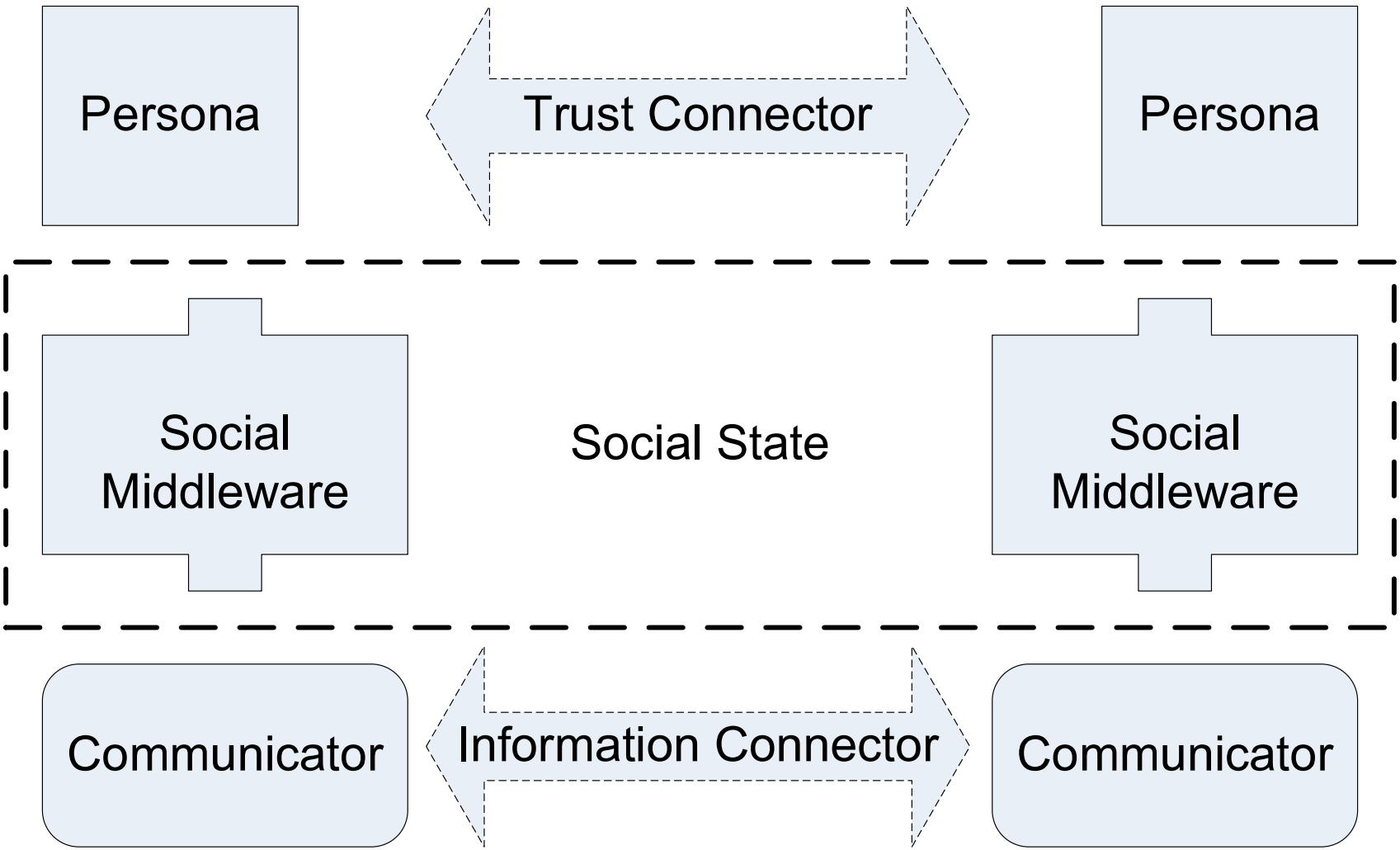
- Key elements
  - Components: power elements, i.e., sources and sinks
  - Connectors: conductors
- Styles based on
  - Constraints: no short circuits; (on contents) Kirchhoff's laws; . . .
  - Patterns: star; hierarchical separated by circuit breakers; . . .
- How do we characterize the elements and conductors logically?
  - Current is what flows over a conductor
  - Current drawn, voltage expected, impedance offered is how we characterize a power element

Architecture: Social System

Components; connectors; constraints; patterns

- Key elements
  - Components: individuals
  - Connectors: social relationships
- Styles based on
  - Constraints: reciprocal (Facebook), ...
  - Patterns: clique; group; ...
- How do we characterize the individuals and their relationships?
  - *Claim:* Trust is what flows over a relationship
  - Can we characterize relationships in a reusable manner, even though not domain-independent?

Social Middleware Related to Architecture



Realizing Social Applications

Modeling and programming interactions among autonomous parties

- Specify and configure
  - Roles
  - Social interactions
    - Their effects on social states
  - Any additional constraints
- Realize over middleware that offers primitives for social interactions
  - Communicating
  - Maintaining social state
  - Computing trust on behalf of a participant

Envisioned Usage of the Middleware, Toto

Just a possible scenario

- Toto defines one or more notions of assessment
- Configure a new application (persona) agent: generate events and assessments produced by the persona in the application
- Application agent
  - Queries Toto regarding trust to place in another persona with respect to an application event
  - Reports user assessment to middleware
- Toto incorporates such assessment into its models

Metadata on Interactions

- Toto would support defining a variety of application events
  - Challenge: how to characterize relevance of some events to others
- We may design such an event to capture the metadata we want: easy to process if limited to builtin types
- An application may generate an assessment based on an explicit or an implicit user action

Explicitness

- Toto would support user access to its models
  - Initialization
  - Inspection
  - Alteration
  - Deletion
- Potentially, Toto could generate explanations

Understanding Trust in Architectural Terms

General model of trust

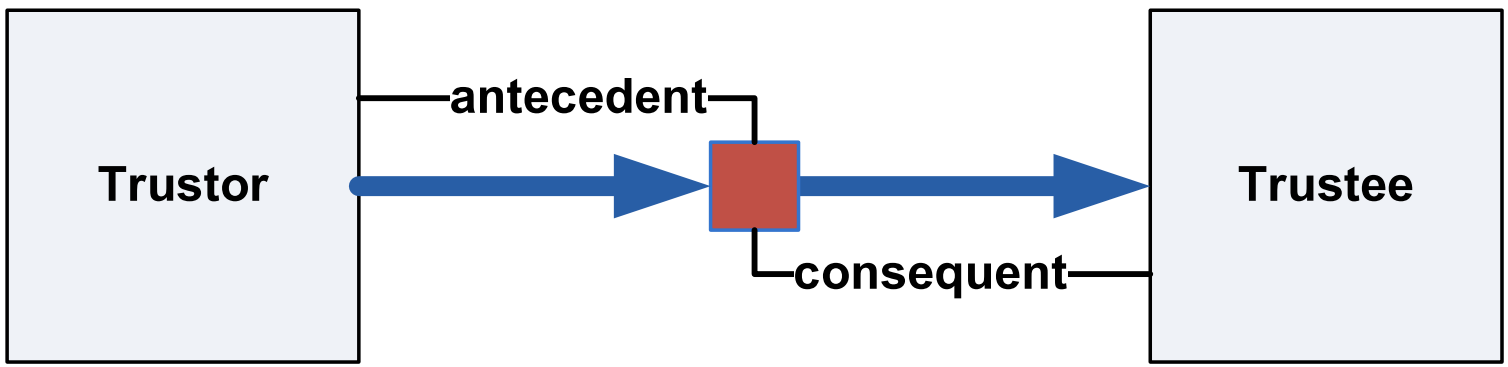
- Notions of dependence
- Conditional
- Compositional
- Semantic
- General

Trust from a Logical Standpoint

- $\mathsf{T}_{truster,trustee}(antecedent, consequent)$
  - $\mathsf{T}_{Alice,Bob}$(raise alert, send warning)
  - $\mathsf{T}_{truster,trustee}(\top, consequent)$: unconditional trust
- ACTIVATE: $\mathsf{T}_{x,y}(r,u) \wedge r \rightarrow \mathsf{T}_{x,y}(\top,u)$
  - $\mathsf{T}_{Alice,Bob}$(raise alert, send warning) $\wedge$ raise alert $\Rightarrow \mathsf{T}_{Alice,Bob}(\top$, send warning)
- COMPLETE: $u \rightarrow \neg\mathsf{T}_{x,y}(r,u)$
  - send warning $\Rightarrow \neg\mathsf{T}_{Alice,Bob}$(raise alert, send warning)
  - send warning $\Rightarrow \neg\mathsf{T}_{Alice,Bob}(\top$, send warning)

A formal semantics underlies the above notion

Schematic of an Architectural Connector as Trust



Postulates for Trust

Active trust basics

(Omitting *truster* and *trustee* when they are the same throughout)

- Complete a connector: dependence has been fulfilled
  - $u \rightarrow \neg \mathsf{T}(r, u)$
- Activate a connector: make dependence stronger (strongest when $r = \top$)
  - $\mathsf{T}(r \wedge s, u) \wedge r \rightarrow \mathsf{T}(s, u)$
- Partition a connector: a dependence for two things is a dependence for each separately (if it isn't already done)
  - $\mathsf{T}(r, u \wedge v) \wedge \neg u \rightarrow \mathsf{T}(r, u)$

Postulates for Trust

Connector integrity

- Avoid conflict: dependence must be internally consistent
  - $\mathsf{T}(r, u) \rightarrow \neg \mathsf{T}(r, \neg u)$
- Nonvacuity: dependence must be for something tangible
  - From $r \vdash u$ infer $\neg \mathsf{T}(r, u)$
- Tighten: if a dependence holds then a narrower dependence also holds
  - From $\mathsf{T}(r, u), s \vdash r, s \nvdash u$ infer $\mathsf{T}(s, u)$

Postulates for Trust

Connector structure

- Combine antecedents: two connectors with the same consequent (fulfillment condition) yield a broader connector
  - $\mathsf{T}(r, u) \wedge \mathsf{T}(s, u) \rightarrow \mathsf{T}(r \vee s, u)$
- Combine consequents: two connectors with the same antecedent (trigger condition) yield a stronger connector
  - $\mathsf{T}(r, u) \wedge \mathsf{T}(r, v) \rightarrow \mathsf{T}(r, u \wedge v)$
- Chain: two chained dependencies yield a combined dependence
  - From $\mathsf{T}(r, u), u \vdash s, \mathsf{T}(s, v)$ infer $\mathsf{T}(r, v)$

Postulates for Trust

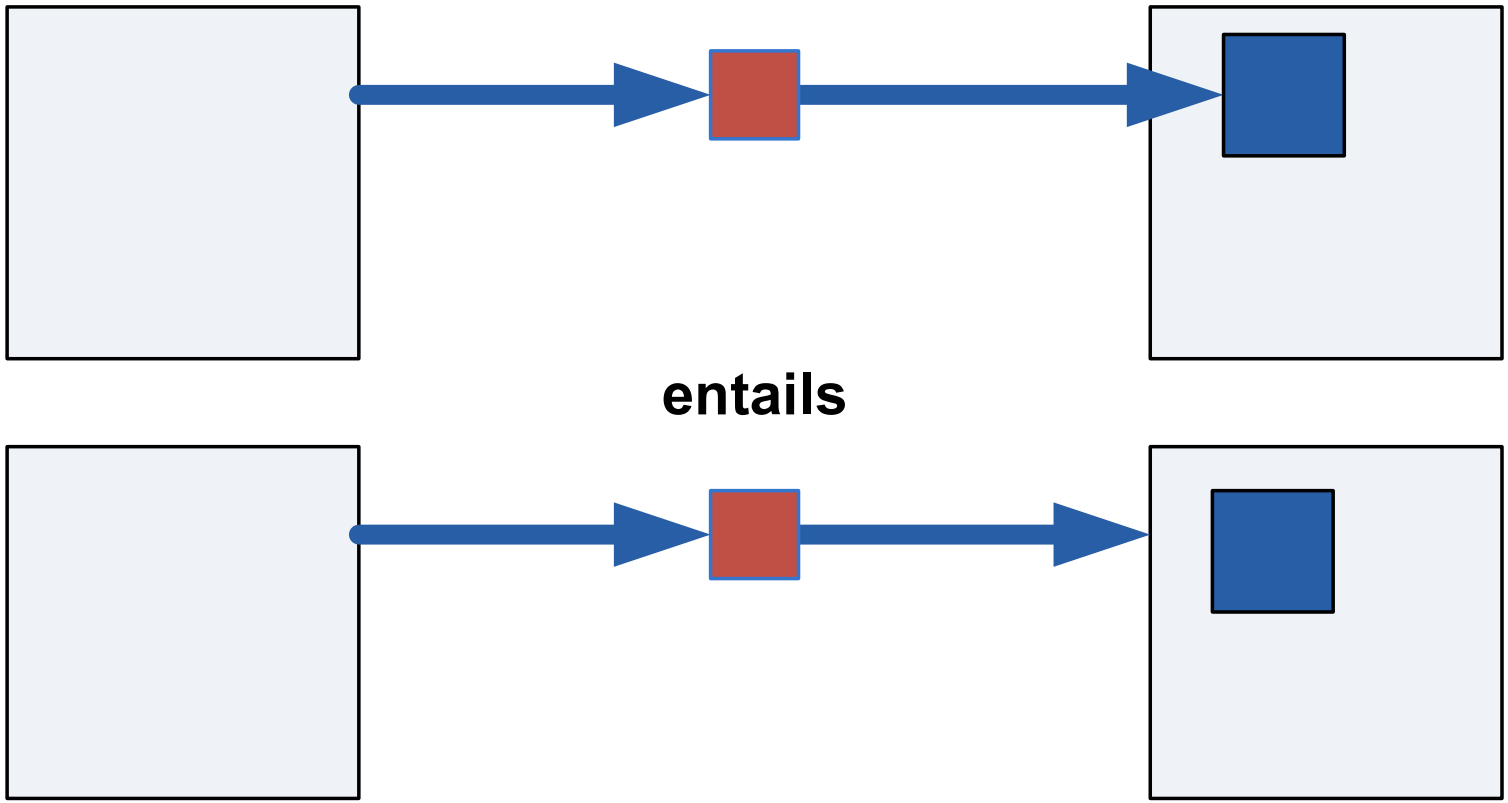
Connector meaning

- Exposure: the trustee's commitment is its level of exposure if the truster trusts it for it
  - $\mathsf{C}_{x,y}(r,u) \rightarrow \mathsf{T}_{y,x}(r,u)$
- Transient alignment: when the trustee commits to support the dependency
  - $\mathsf{T}_{x,y}(r,u) \rightarrow \mathsf{C}_{y,x}(r,u)$
- Well-placed trust: when trust is fulfilled in the actual execution
  - $\mathsf{T}_{x,y}(\text{true},u) \rightarrow \mathsf{R}u$
- Whole-hearted alignment: when trust is backed by a steady commitment until success
  - $\mathsf{T}_{x,y}(s,v) \rightarrow \mathsf{R}(s \rightarrow (\mathsf{C}_{y,x}(s,v)\mathsf{U}v))$

(Above, $\mathsf{C}_{x,y}(r,u)$ refers to a commitment from x to y; R indicates "on the real execution path"; and $p\mathsf{U}q$ means p holds until q does)

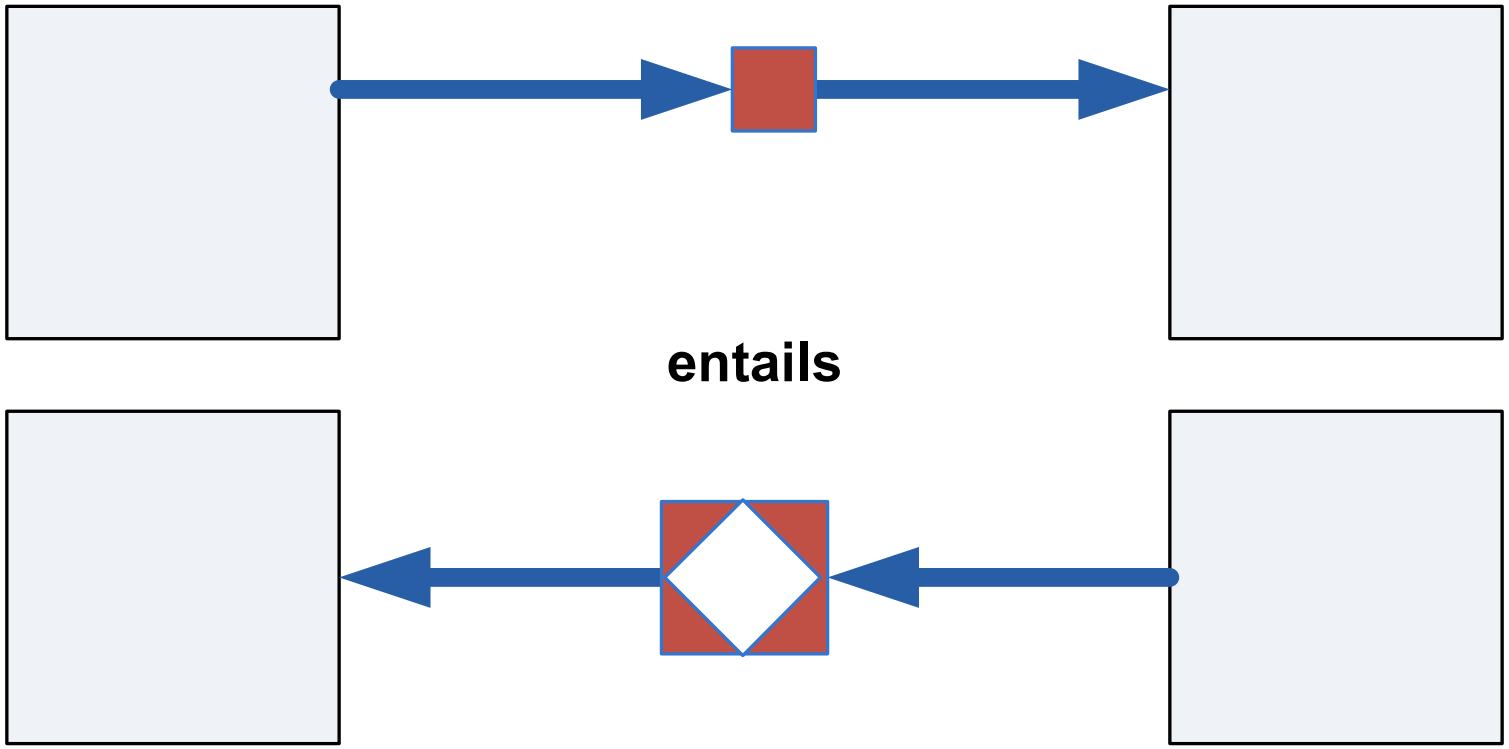
TRUSTEE'S TEAM, Schematically

If you trust a team member, you trust the team



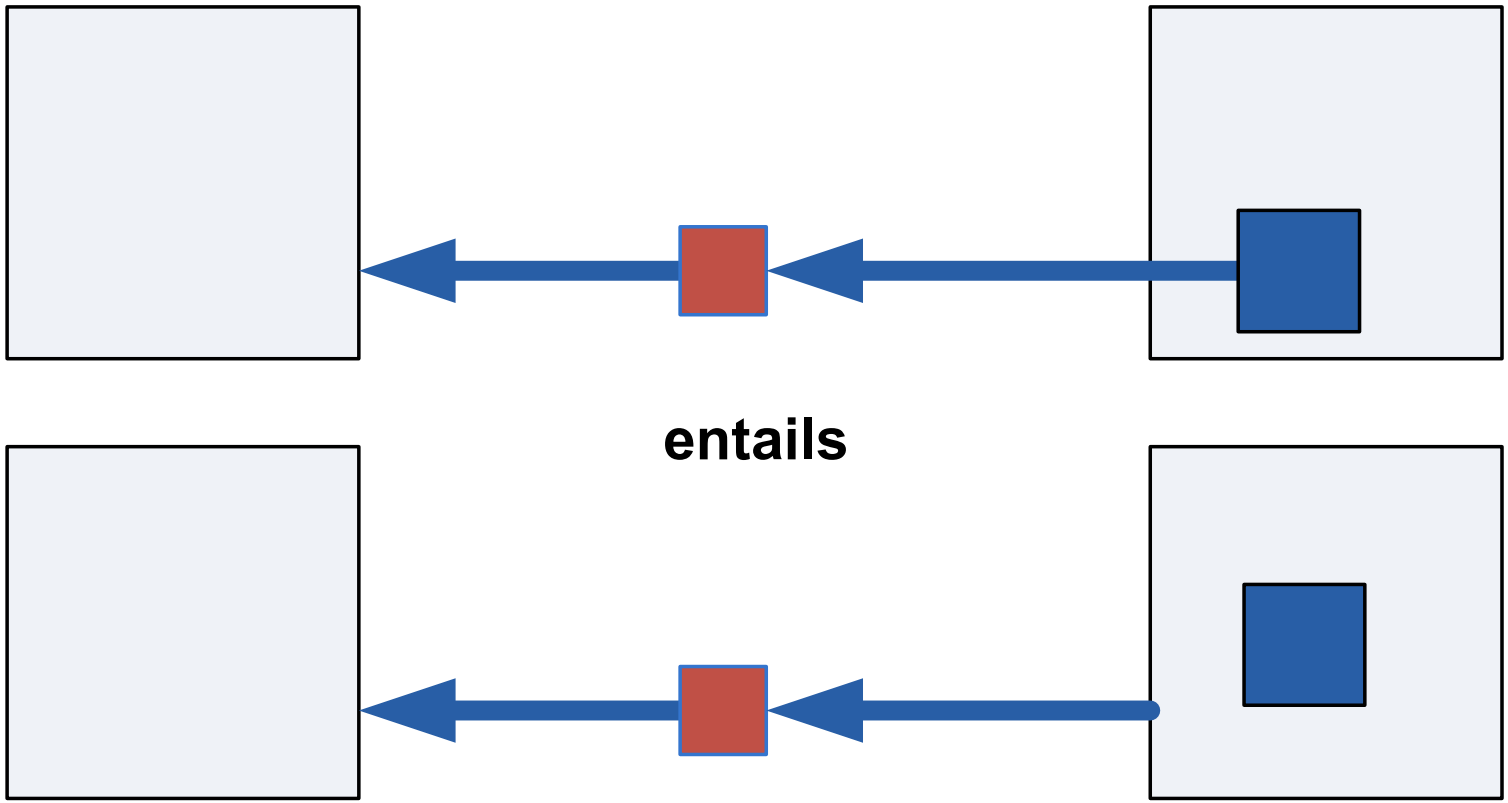
TRANSIENT ALIGNMENT, Schematically

The trustee is committed to what you trust them for



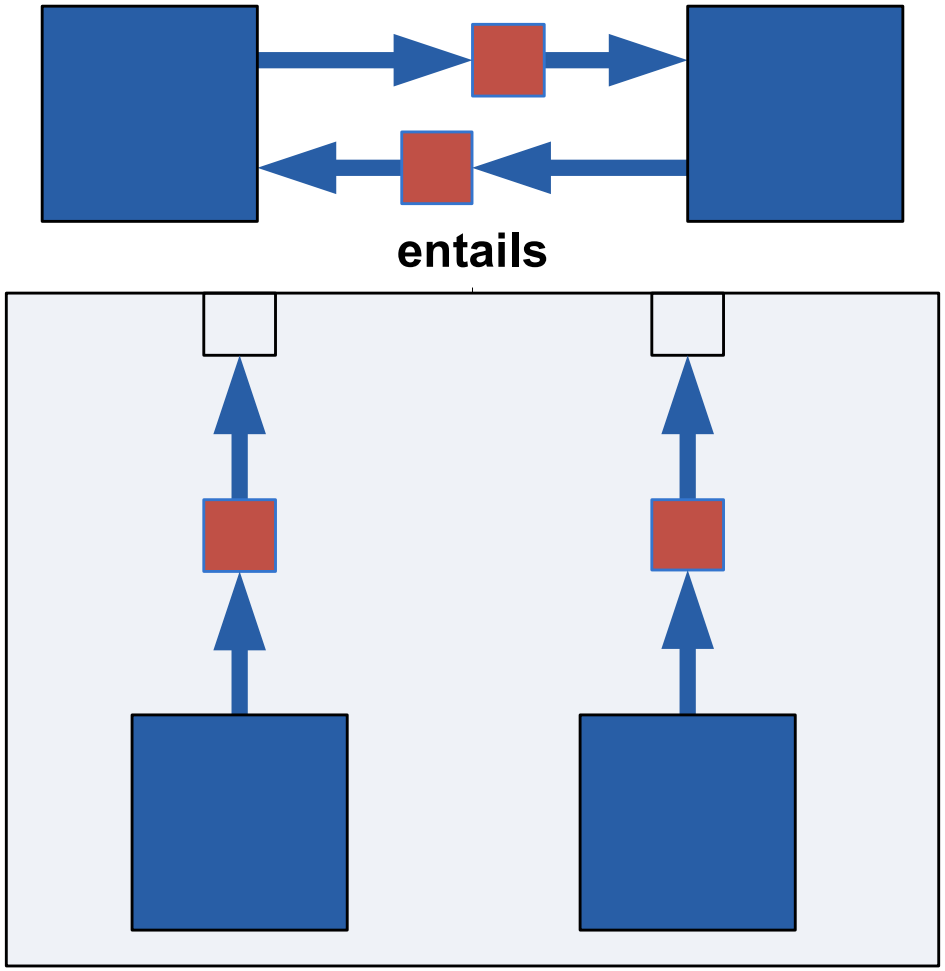
TRUSTER'S TEAM, Schematically

Your team trusts whom you trust



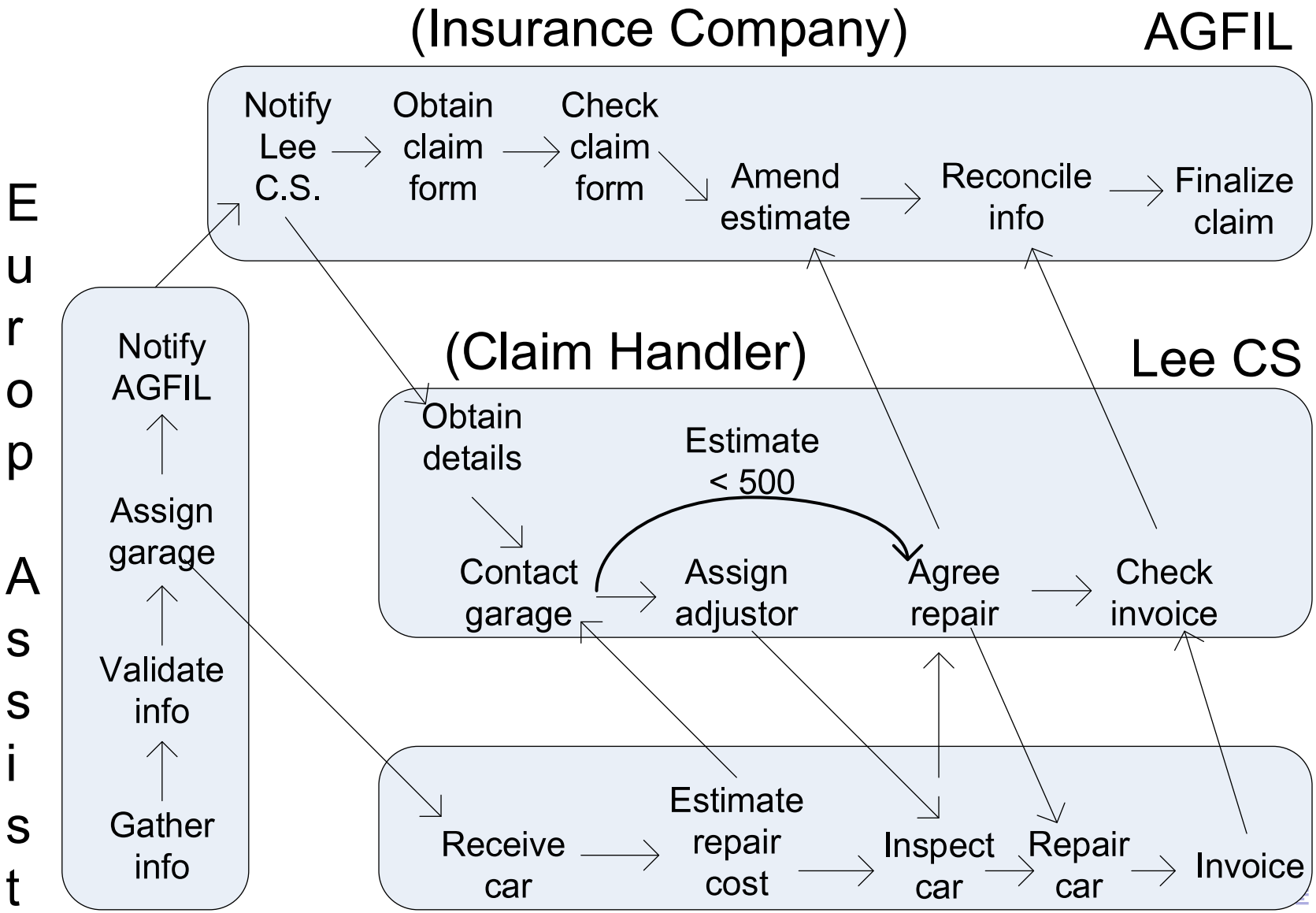
PARALLEL TEAMWORK, Schematically

If you trust each other, you are part of a team



Cross-Organizational Business Process Example

Insurance scenario modeled operationally



Applying the Postulates

- Doe would ACTIVATE his dependence on the mechanic
- The mechanic would COMPLETE the dependence by repairing the car
- The mechanic gives Doe a loaner car for a week: the loaner is PARTITIONED from the repair itself
- Doe can COMBINE his dependence on the mechanic to trust the mechanic to repair the car whether Doe brings it in or asks the mechanic to tow it to his shop
- Under PERSISTENCE, the mechanic holds his trust in being paid in a timely fashion by AGFIL until he submits a bill or gets paid
- Doe and the mechanic demonstrate WHOLE-HEARTED ALIGNMENT because the mechanic remains committed to completing the repairs until he does so
- Doe applies PARALLEL TEAMWORK to place his trust in the team consisting of AGFIL, Lee CS, and the mechanic to process his claim

Formal Model

- Possible moments
  - Partitioned into (disjoint) paths
- Propositions map to sets of moments
- Trust and commitments are conditioned by antecedent propositions
  - Map each antecedent proposition to a set of consequent sets, each of the latter is a set of paths, namely those
    - The debtor commits to bringing about
    - The truster trusts the trustee to bring about

Correspondence Theory

- State constraints on the above model in a *modular* manner
  - One constraint for each axiom
  - Interestingly, many constraints can be stated simply in terms of sets
- Yields a number of sound and complete axiomatizations, for each subset of the reasoning postulates

Conclusions and Directions

- Formalizing architectures for social computing based on trust
  - How can trust fit into an overall system architecture?
- Identifying suitable architecture styles
  - What are suitable patterns for different types of social applications?
- Mapping effectively to existing representations and estimation techniques
  - Computation paths can be used as a basis for judging probabilities and expected utilities
- Semantics
  - Already available: Montague-Scott models
  - Planned: Kripke models assuming some postulates
- Notation to facilitate modeling
  - Graphical or textual

Thanks!

http://www.csc.ncsu.edu/faculty/mpsingh/