Goal-Oriented Requirements Engineering

John Mylopoulos
University of Toronto/Trento

SIKS Masterclass on
"Requirements Engineering and Information Modelling",
VU, Amsterdam, May 30, 2006

Abstract

- We review some of the history of Software Engineering since 1968 and discuss some the underlying concepts of Structured and Object-Oriented Software Development, noting that they don't offer means of capturing and analyzing design spaces for the software-to-be.
- We then introduce Goal-Oriented Requirements Engineering concepts adopted from the KAOS and *i** modelling frameworks, noting key ideas. We also argue that they do offer the basic ingredients required for characterizing and analyzing design spaces.
- We then sketch three applications of Goal-Oriented Requirements Engineering. The first involves establishing an Agent-Oriented Software Development methodology called *Tropos*. The second addresses the design of high-variability software for applications such as home care and business process design. The third strives to extend database design with an extra step mapping stakeholder goals to a conceptual schema.
- The research reported in this presentation was conducted with colleagues at the Universities of Toronto (Canada) and Trento (Italy).

…1968…

- Herbert Simon (subsequent Nobel prize and Turing award winner) delivered the Karl Taylor Compton Lecture at MIT on April 2, 1968.
- In his lecture, Simon chose to take on the perception that Engineering is not as academically respectable as Science…
- In response to this perception, he sketched the key ingredients of a *Science of Design*.

Towards a Science of Design

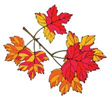
According to Simon, key ingredients would include:

- *Goals* -- what is the purpose for the artifact-to-be?
- A space of *alternatives (design space)* -- what are the different ways of meeting these goals?
- *Search* among the alternatives -- computational methods; optimal and sub-optimal search algorithms;
- *Evaluation* of alternatives with respect to *criteria*;
- A Logic for designs;
- Representation of design problems;
- …

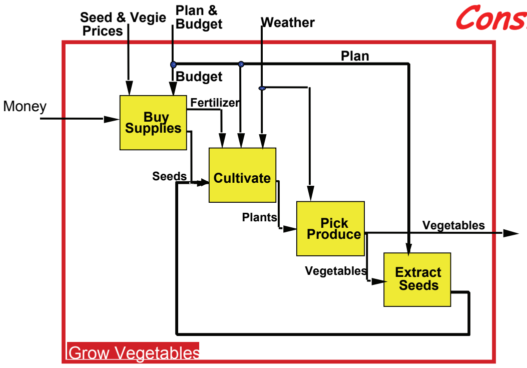
Software Engineering (~1968)

- Born the same year Simon delivered his lecture.
- Over the years, formulated software design paradigms (*methodologies*), such as Structured ('70s) and Object-Oriented ('80s) Software Development.
- But these have been largely recipe-oriented …

… cook-bookish!

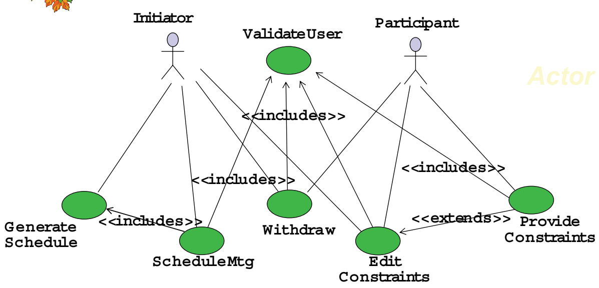


Consider…



…Where are the goals? …alternatives?

Consider…



…Where are the goals? …alternatives?

Software development methodologies are ***linear…*** *Alternatives are hidden, to be considered by the designer alone, rather than through a rational, formalizable, analyzable process…*

…Goal-Oriented Requirements Engineering could offer a foundation for changing all that!

This Talk

- *Goal-Oriented RE -- History and key ideas*
- *Moving forward -- Tropos*
- *Moving forward -- Designing high-variability software*
- *Moving forward -- From stakeholder goals to database conceptual schemas.*

Goal-Oriented Requirements Engineering (~1993)

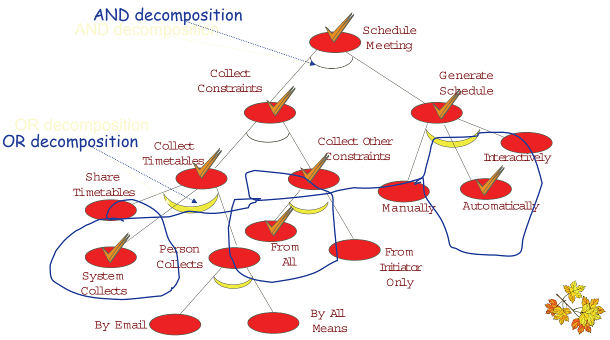
- Goal-oriented analysis focuses on early requirements, when alternatives are being explored and evaluated.
- During goal-oriented analysis, we start with initial stakeholder goals such as "Fulfill every book request", or "Schedule meeting" and keep refining them until we have reduced them to alternative collections of design decisions each of which can satisfy the initial goals.
- Initial goals may be contradictory, so the analysis must facilitate the discovery of tradeoffs and the search of the full space of alternatives, rather than a subset.

Goal-Oriented Analysis a la KAOS

- (Organizational) goals lead to requirements.
- Goals justify and explain the presence of requirements which are not necessarily comprehensible by clients.
- Goals provide basic information for detecting and resolving conflicts that arise from multiple viewpoints [Dardenne93].
- Example goal:

SystemGoal Achieve[BookRequestSatisfied]
InstanceOf SatisfactionGoal
Concerns Borrower, Book, Borrowing,...
Definition (∀bor: Borrower, b: Book, lib: Library)
(Requesting(bor, b) ∧ b.subject ∈ lib.coverageArea ⇒
□[(∃bc: BookCopy) (Copy(bc, b) ∧ Borrowing(bor, bc)))]

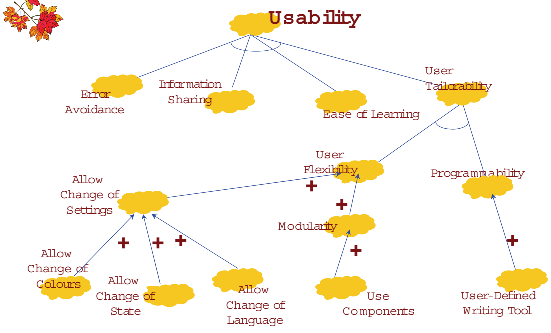
Goal Analysis Leads to Alternatives



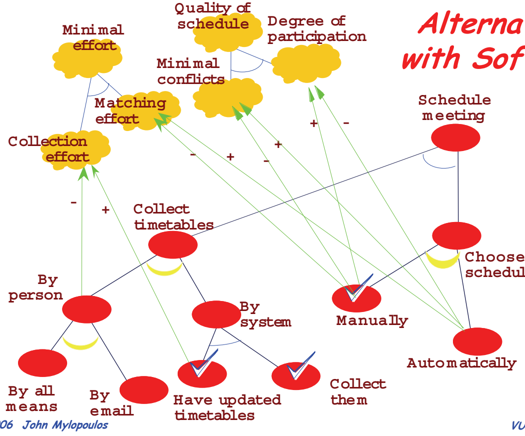
Softgoals

- Functional goals, such as "Schedule meeting" are well defined in the sense that they admit a formal definition.
- Non-functional goals, such "higher profits", "higher customer satisfaction" or "easily maintainable system" specify *qualities* the software system should adhere to.
- Such qualities usually admit no generally agreed upon definition, are inter-related and often contradictory.
- Such goals are represented as ***softgoals***.
- Softgoals can be thought as "fuzzy goals" with no clear-cut criteria for satisfaction; hence softgoals are *satisficed*, rather than satisfied (NFR framework, [Mylopoulos92], [Chung93]).

Softgoals for Representing Non-Functional Requirements



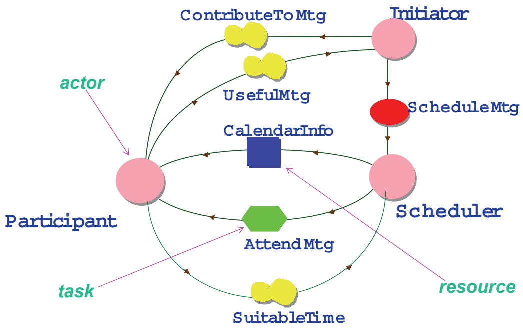
Evaluating Alternatives with Softgoals



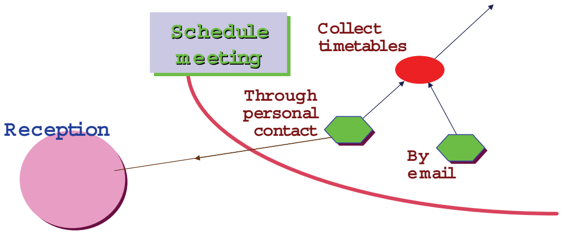
Stakeholders and Their Goals

- In KAOS, goals are global objectives for the system-to-be.
- In *i** [Yu93], goals are desired by *actors* and are *delegated* to other actors for fulfillment.
- In this framework then, early requirements involve identifying stakeholders and their goals, analyzing these goals, delegating them to other actors etc.
- The result of this process consists of *actor dependency* and *actor rationale* models.

An Actor Dependency Model



An Actor Rationale Model

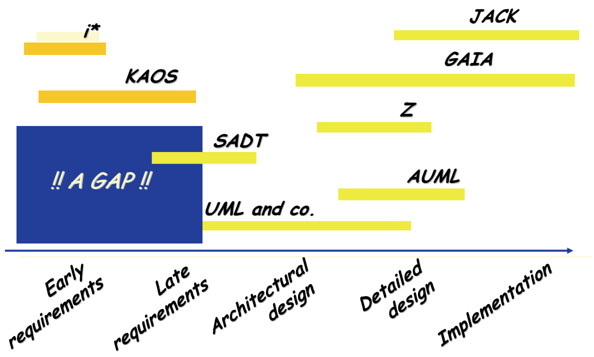


Actor dependencies are intentional: One actor ***wants*** something, another is ***willing*** and ***able*** to deliver.

Goals in Software Design

- KAOS, the NFR proposal, as well as *i** advocate the use of goals in designing software.
- KAOS uses goals to go from organizational objectives to functional requirements.
- NFR uses them to represent and analyze non-functional requirements. Non-functional requirements lead to criteria for evaluate functional alternatives.
- *i** relates goals to the actors who want them and keeps track of delegations.

So What? Early Requirements Phase



Credits

- Many researchers worked with goals a decade or more ago:
  - ✓ Martin Feather and Steve Fickas;
  - ✓ Colin Potts and Annie Anton;
  - ✓ Janis Bubenko;
  - ✓ Colette Rolland;
  - ✓ Periklis Loucopoulos and Evangelia Kavakli;
  - ✓ …

This Talk

- *GORE -- History and key ideas*
- *Moving forward -- Tropos*
- *Moving forward -- Designing high-variability software*
- *Moving forward -- From stakeholder goals to database conceptual schemas.*

…An Idea... (~2000)

- Software Engineering methodologies have traditionally come about in a “late-to-early” phase (or, “downstream-to-upstream”) fashion.
- In particular, Structured Programming preceded Structured Analysis and Design; likewise, Object-Oriented Programming preceded Object-Oriented Design and Analysis.
- In both cases, programming concepts were projected upstream to dictate how designs and requirements are conceived.

What would happen if we projected requirements concepts downstream to define software designs and even implementations?

The Tropos Methodology

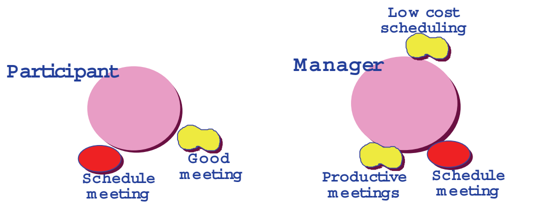
- Proposes a set of primitive concepts adopted from *i** (actor, goal, actor dependency,…) and a process for building agent-oriented software.
- Covers four phases of software development:
  - ✓ *Early requirements* -- identify stakeholders and their goals;
  - ✓ *Late requirements* -- introduce system-to-be as another actor who can accommodate some of these goals;
  - ✓ *Architectural design* -- more system actors are added and are assigned responsibilities;
  - ✓ *Detailed design* -- complete the specification of system actors.

Agent-Oriented Software Engineering

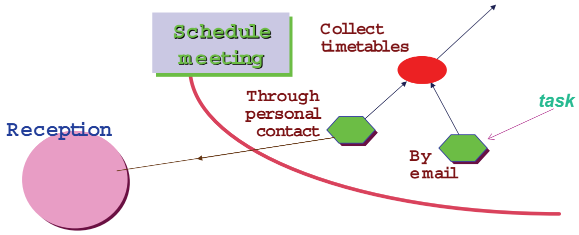
- Many researchers working on it.
- Research on the topic generally comes in two flavours:
  - ✓ Extend UML to support agent communication, negotiation etc. (e.g., [Bauer99, Odell00]);
  - ✓ Extend current agent programming platforms (e.g., JACK) to support not just programming but also design activities [Jennings00].
- All AOSE methodologies involve to a greater or lesser extend intentional concepts, analysis of alternatives, etc.

Early Requirements in Tropos: External Actors and their Goals

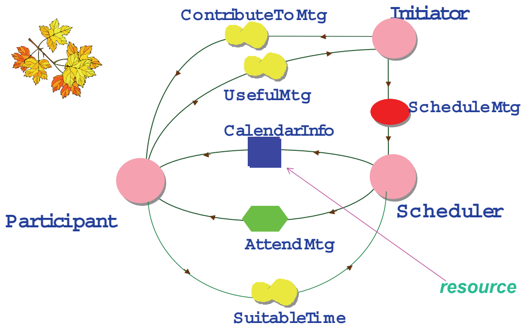
A social setting consists of actors, each having *goals* (and/or *softgoals*) to be fulfilled.



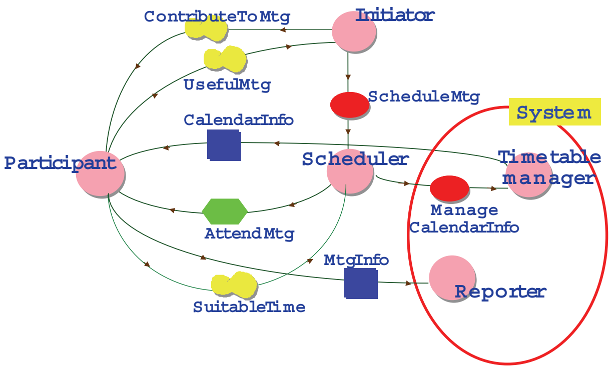
Actor Dependencies



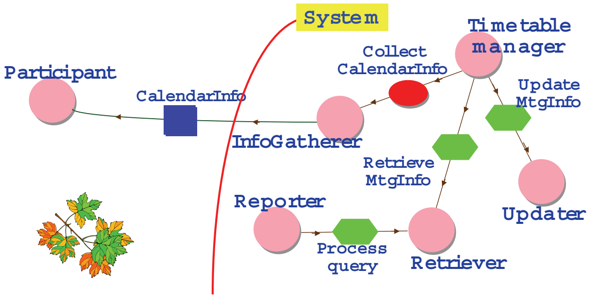
Actor Dependency Models



Late Requirements with i*



Software Architectures with i*



...Yet Another Software Development Process

- Initialization: Identify stakeholder actors and their goals;
- Step: For each new goal, the actor who owns it:
  - ✓ adopts it;
  - ✓ delegates it to another existing actor;
  - ✓ delegates it to a new actor;
  - ✓ decomposes it into new subgoals;
  - ✓ declares the goal "denied".
- Termination condition: All initial goals have been fulfilled (…to an acceptable degree), assuming all actors deliver on their commitments.

Analyzing Tropos Models

- Models are used primarily for human communication
- But, this is not enough! Large models can be hard to understand, or take seriously.
- We need analysis techniques which offer evidence that a model makes sense:
  - ✓ *Simulation* through model checking, to explore the properties of goals, entities, etc. over their lifetime;
  - ✓ *Goal analysis* uses a SAT prover to determine whether a goal can be fulfilled;
  - ✓ *Social analysis* uses a planner to explore alternative delegations for a given set of actors and goals.

The Tropos Project

- Project was launched in April 2000.
- Participating teams includes:
  - ✓ UToronto (Canada): Eric Yu, Alexei Lapouchnian, Sotirios Liaskos, Yijun Yu, Yiqiao Wang, Neil Ernst;
  - ✓ UTrento/IRST (Italy): Anna Perini, Angelo Susi, Loris Penserini, Paolo Giorgini, Fabio Massacci, Roberto Sebastiani, Nicola Zannone, Paolo Traverso,...;
  - ✓ RWTH Aachen (Germany): Matthias Jarke, Gerhard Lakemeyer,…;
  - ✓ FUPernambuco (Brazil): Jaelson Castro,…;
- Publications and other information about the project can be found at http://www.troposproject.org.

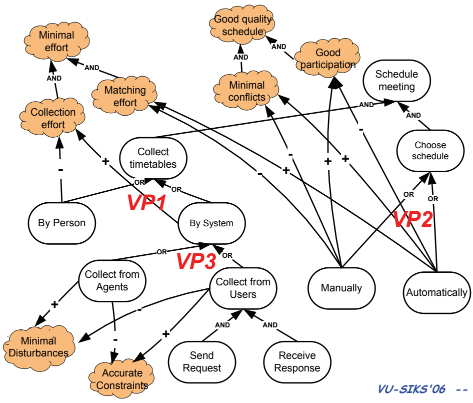
This Talk

- *GORE -- History and key ideas*
- *Moving forward -- Tropos*
- ***Moving forward -- Designing high-variability software***
- ***Moving forward -- From stakeholder goals to database conceptual schemas.***

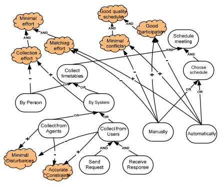
Designing for High Variability

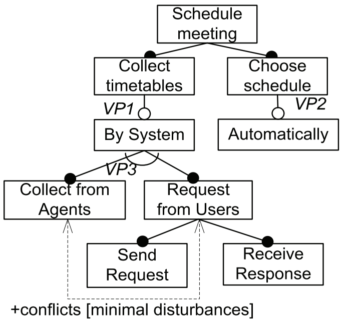
- Instead of choosing *one* solution for the fulfillment of a top-level goal, we could choose to support them *all*.
- This leads to software solutions that can be customized in many different ways, depending on stakeholder preferences and environmental parameters.
- Work by Alexei Lapouchnian, Sotiris Liaskos, Yiqiao Wang and Yijun Yu.

From a Goal Model to Architectural Views



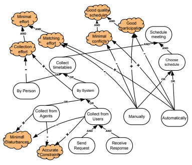
Feature Model

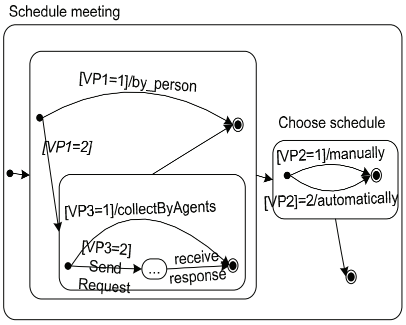




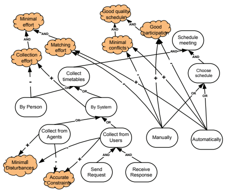
+conflicts [minimal disturbances]
+conflicts [accurate constraints]

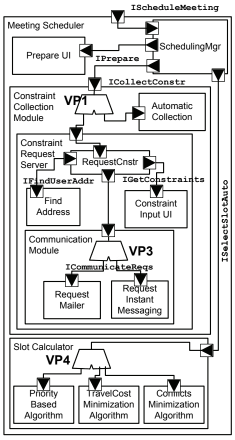
Statecharts





Software Architecture





High-Variability Designs

- → Enterprise Resource Planning (ERP) software is generic and can be customized in (very) many different ways.
- → But we don't have yet *systematic* ways of generating such designs.
- → Envisioned applications for high-variability software:
  - ✓ Business process design;
  - ✓ Home care software for the elderly.

Autonomic (Application) Software

- → (According to IBM) This is software that can self-configure, self-repair and self-optimize.
- → For us,

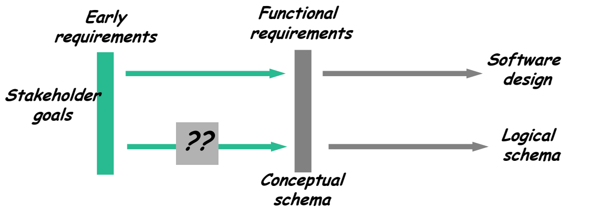
  Autonomicity =
  High-Variability+Monitoring+Diagnosis+Adaptivity
- → Our goal-oriented framework may not be appropriate for autonomic *system* software (e.g., an OS) or middleware (e.g., a DBMS).
- → But it certainly is for *application* software!
- → Different mechanisms required for
  - ✓ Self-repair -- real-time reconfiguration and recovery
  - ✓ Self-configuring and self-optimization -- off-line reconfiguration, no recovery

This Talk

- *GORE -- History and key ideas*
- *Moving forward -- Tropos*
- *Moving forward -- Designing high-variability software*
- ***Moving forward -- From stakeholder goals to database conceptual schemas.***

Database Design Revisited

- **The state-of-the-art in database design hasn't changed since the 80s.**
- **Let's fill in the gap between early requirements and conceptual schema design (work with Lei Jiang, Alex Borgida, Thodoros Topaloglou)**



What is Missing?

- **The (qualitative) models we built may be formalizable, BUT they are subjective.**
- **We need theories of measurement for cognitive/social phenomena.**
- **Our models are also often redundant, include too many variables [Menzies99].**
- **… Probably other things as well …**

Conclusions

- **We have argued that Goal-Oriented Requirements Engineering opens the way towards defining and analyzing *design spaces* for software.**
- **We also sketched three applications of goal-oriented RE concepts:**
  - ✓ **Designing agent-oriented software -- the Tropos project [JAAMAS04, JInformationSystems03];**
  - ✓ **Designing high-variability software [RE'06]**
  - ✓ **Designing conceptual database schemas [RE'06].**

References

- [Bauer99] Bauer, B., *Extending UML for the Specification of Agent Interaction Protocols.* OMG document ad/99-12-03.
- [Castro02] Castro, J., Kolp, M., Mylopoulos, J., "Towards Requirements-Driven Software Development Methodology: The Tropos Project," *Information Systems 27(2)*, Pergamon Press, June 2002, 365-389.
- [Chung00] Chung, L., Nixon, B., Yu, E., Mylopoulos, J., *Non-Functional Requirements in Software Engineering*, Kluwer Publishing, 2000.
- [Dardenne93] Dardenne, A., van Lamsweerde, A. and Fickas, S., "Goal–directed Requirements Acquisition", *Science of Computer Programming, 20*, 1993.
- [Fuxman01a] .Fuxman, A., Pistore, M., Mylopoulos, J. and Traverso, P., "Model Checking Early Requirements Specifications in Tropos", Proceedings Fifth International IEEE Symposium on Requirements Engineering, Toronto, August 2001.
- [Fuxman01b] Fuxman,A., Giorgini, P., Kolp, M., Mylopoulos, J., "Information Systems as Social Organizations", Proceedings International Conference on Formal Ontologies for Information Systems, Ogunquit Maine, October 2001.
- [Iglesias98] Iglesias, C., Garrijo, M. and Gonzalez, J., "A Survey of Agent-Oriented Methodologies", *Proceedings of the 5th International Workshop on Intelligent Agents: Agent Theories, Architectures, and Languages* (ATAL-98), Paris, France, July 1998.

...More References…

- [Jennings00] Jennings, N. "On Agent-Based Software Engineering", *Artificial Intelligence 117*, 2000.
- [Menzies99] Menzies, T., Easterbrook, S., Nuseibeh, B., and S.Waugh. "An Empirical Investigation of Multiple Viewpoint Reasoning in Requirements Engineering", In *RE'99*, 1999. Available from http://menzies.us/pdf/99re.pdf.
- [Mylopoulos92] Mylopoulos, J., Chung, L. and Nixon, B., "Representing and Using Non-Functional Requirements: A Process-Oriented Approach," *IEEE Transactions on Software Engineering 18(6)*, June 1992, 483-497.
- [Odell00] Odell, J., Van Dyke Parunak, H. and Bernhard, B., "Representing Agent Interaction Protocols in UML", *Proceedings 1st International Workshop on Agent-Oriented Software Engineering* (AOSE00), Limerick, June 2000.
- [Simon69] Simon, H., *The Sciences of the Artificial*, The MIT Press, 1969
- [Wooldridge00] Wooldridge, M., Jennings, N., and Kinny, D., "The Gaia Methodology for Agent-Oriented Analysis and Design," *Journal of Autonomous Agents and Multi-Agent Systems, 3(3)*, 2000, 285–312.
- [Yu95] Yu, E., *Modelling Strategic Relationships for Process Reengineering*, Ph.D. thesis, Department of Computer Science, University of Toronto, 1995.
- [Zambonelli00] Zambonelli, F., Jennings, N., Omicini, A., and Wooldridge, M., "Agent-Oriented Software Engineering for Internet Applications," in Omicini, A., Zambonelli, F., Klusch, M., and Tolks-Dorf R., (editors), *Coordination of Internet Agents: Models, Technologies, and Applications*, Springer-Verlag LNCS, 2000, 326–346.