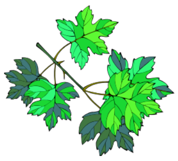
Goal-Oriented Requirements Engineering: Part II

John Mylopoulos
University of Toronto/Trento

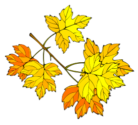
14th IEEE Requirements Engineering Conference
Minneapolis, September 15, 2006



Abstract

→ We briefly review the history and key ideas in Goal-Oriented Requirements Engineering research. We then sketch two applications of these ideas. The first involves establishing an Agent-Oriented Software Development method called *Tropos* which covers not only requirements but also design phases. The second addresses the design of high-variability software for applications such as home care software and business process design.

→ The research reported in this presentation was conducted with colleagues at the Universities of Toronto (Canada) and Trento (Italy).



GORE, Parts I & II

- → GORE = Goal-Oriented Requirements Engineering
- → Axel van Lamsweerde delivered a keynote talk with this title at RE'04 in Kyoto.
- → His talk reviewed the history and key ideas of GORE and described some of the on-going research and industrial experiences of the KAOS project.
- → The talk made an elegant case for GORE as an important edition to SE practice.
- → In this presentation, I review some of the research results and on-going work of the i*/Tropos project.
- → I'll also make the case for GORE as a foundation for a Theory of Software Design.

This Talk

- → *Goal-Oriented RE -- History and key ideas*
- → *Moving forward -- Tropos*
- → *Moving forward -- Designing high-variability software*
- → *Afterthoughts and conclusions*

Goal-Oriented Requirements Engineering (~1993)

- Goal-oriented analysis focuses on early requirements, when problems are identified, and alternative solutions are 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 functional requirements 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:

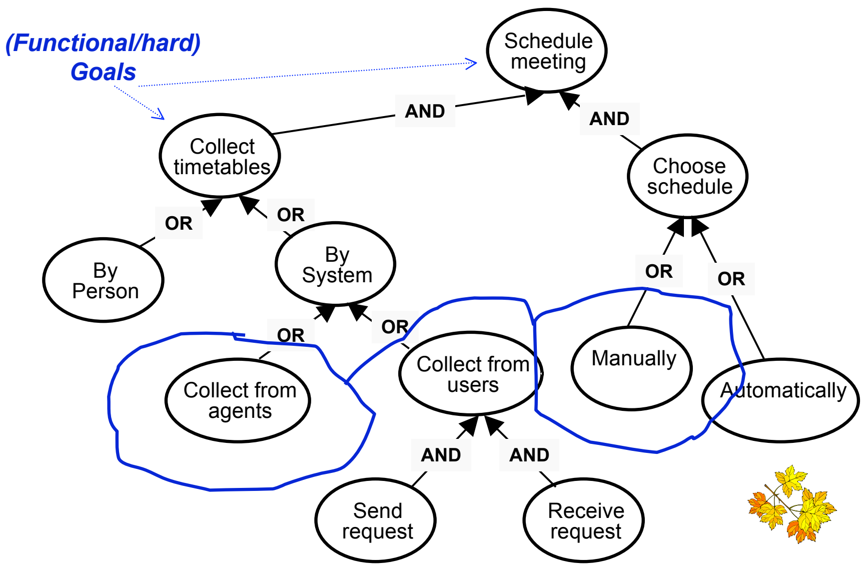
  SystemGoal Achieve[BookRequestSatisfied]

  InstanceOf SatisfactionGoal

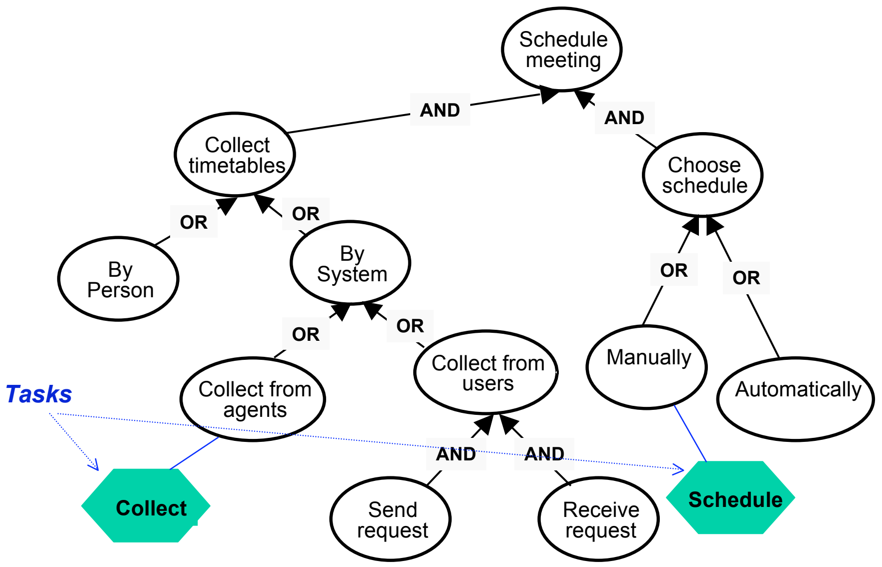
  Concerns Borrower, Book, Borrowing,...

  Definition ($\forall$bor: Borrower, b: Book, lib: Library)
  (Requesting(bor, b) $\wedge$ b.subject $\in$ lib.coverageArea $\Rightarrow$
  F[($\exists$bc: BookCopy) (Copy(bc, b) $\wedge$ Borrowing(bor, bc))]]

Goal Analysis Leads to Alternatives



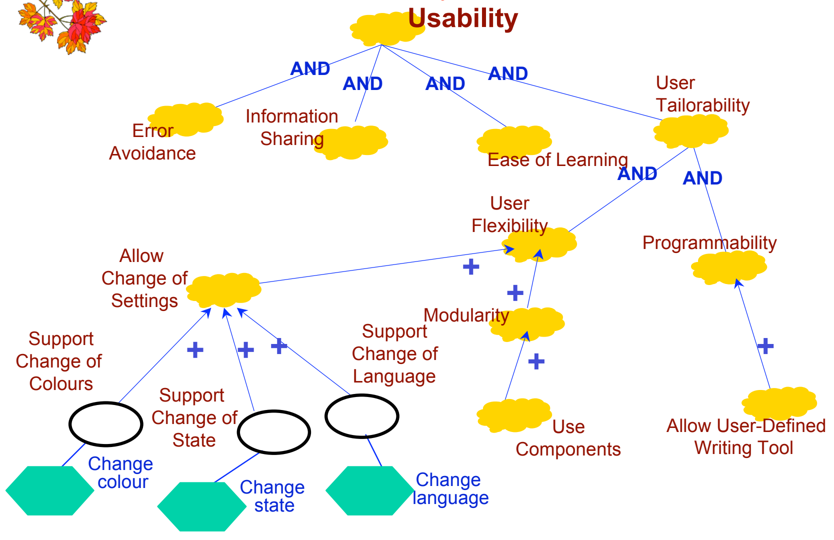
Alternatives Lead to Designs/Plans



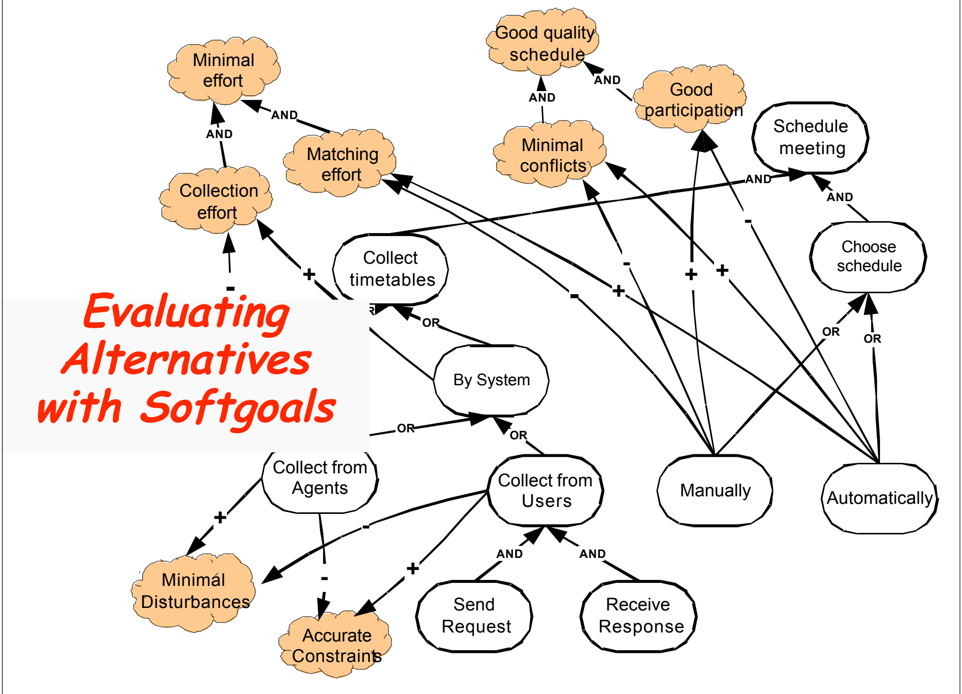
Softgoals

- 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* a socio-technical system should adhere to.
- Such qualities usually admit no generally agreed upon definition, are inter-related and often contradictory.
- Such qualities 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



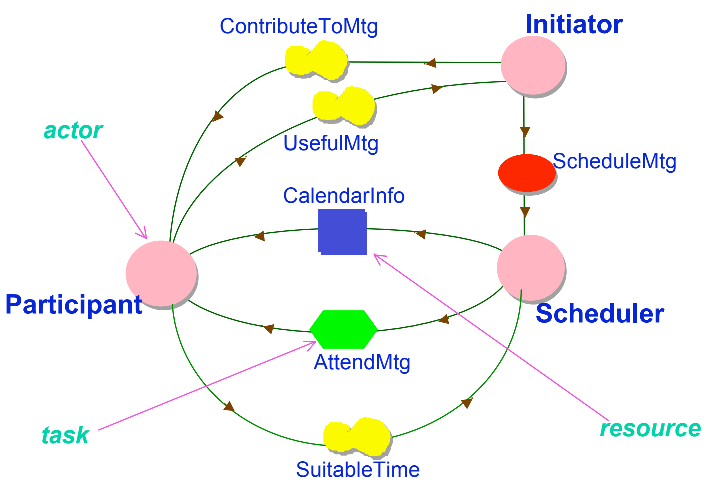
Evaluating Alternatives with Softgoals



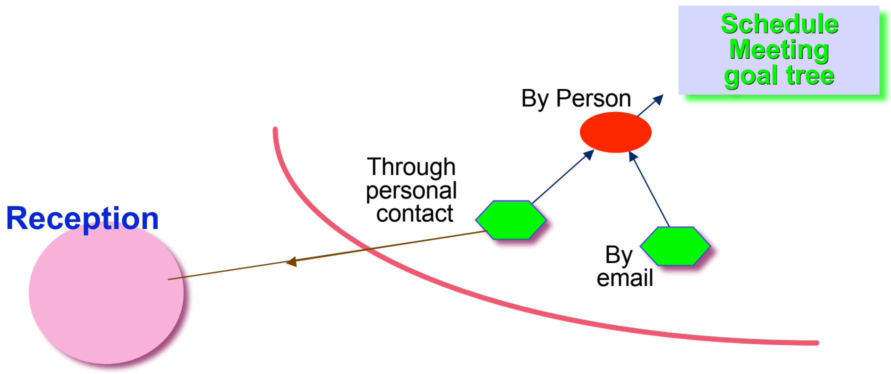
Stakeholders and Their Goals

- 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

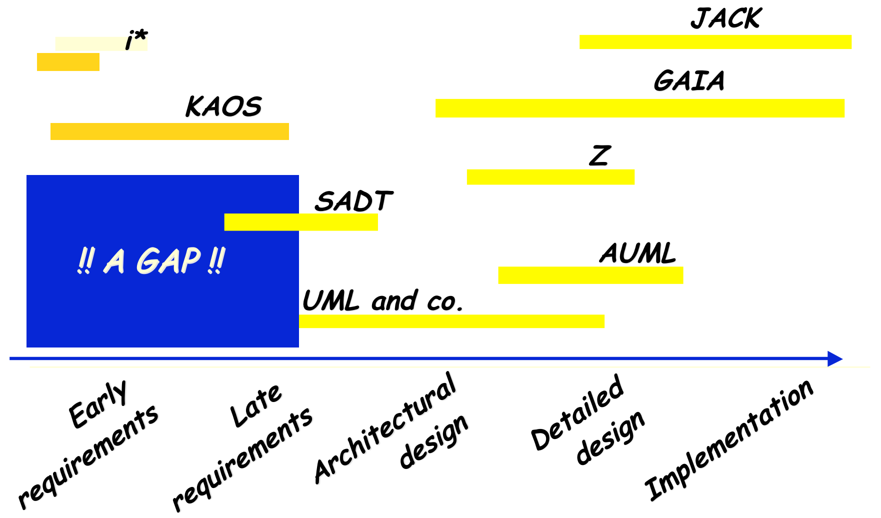


Actor dependencies are intentional: One actor *wants* something, another is *willing* and *able* to deliver.

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 (… AND functional requirements).
- *i** relates goals to the actors who want them and keeps track of delegations.

So What? Early Requirements Phase



Credits

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

Goals, Intentions and Tasks

- Goals are desired states of affairs, e.g., "I want to be president of the USA" (…)
- Intention = goal + commitment
- Actions are things agents can perform to change the state of affairs.
- Task = action + intention
- We use "intention" and "goal" synonymously; likewise for "task" and "plan".

Expressiveness of Goal Models

- → Different researchers use different notations and different primitive goal relationships.
- → In our work, we adopted a goal model that includes:
  - ✓ Goal types: (hard)goals, softgoals
  - ✓ Relationship types: AND (n-ary), OR (n-ary), ++ (makes, binary), -- (breaks, binary), + (helps, binary), - (hurts, binary)
- → This model without +, - is expressively equivalent to Propositional Calculus (PC).
- → The problem of deciding if given root-level goals can be fulfilled can be reduced to the satisfiability problem in PC.

This Talk

- → *GORE -- History and key ideas*
- → ***Moving forward -- Tropos***
- → ***Moving forward -- Designing high-variability software***
- → ***Afterthoughts and conclusions***

…An Idea... (~2000)

→ Software Engineering methods 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., AUML [Bauer99, Odell00]);

  ✓ Extend current agent programming platforms (e.g., JACK) to support not just programming but also design activities, e.g., GAIA [Jennings00].

→ All AOSE methods involve to a greater or lesser extend intentional concepts, analysis of alternatives, etc.

Software Development as Multi-Agent Planning

→ 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;

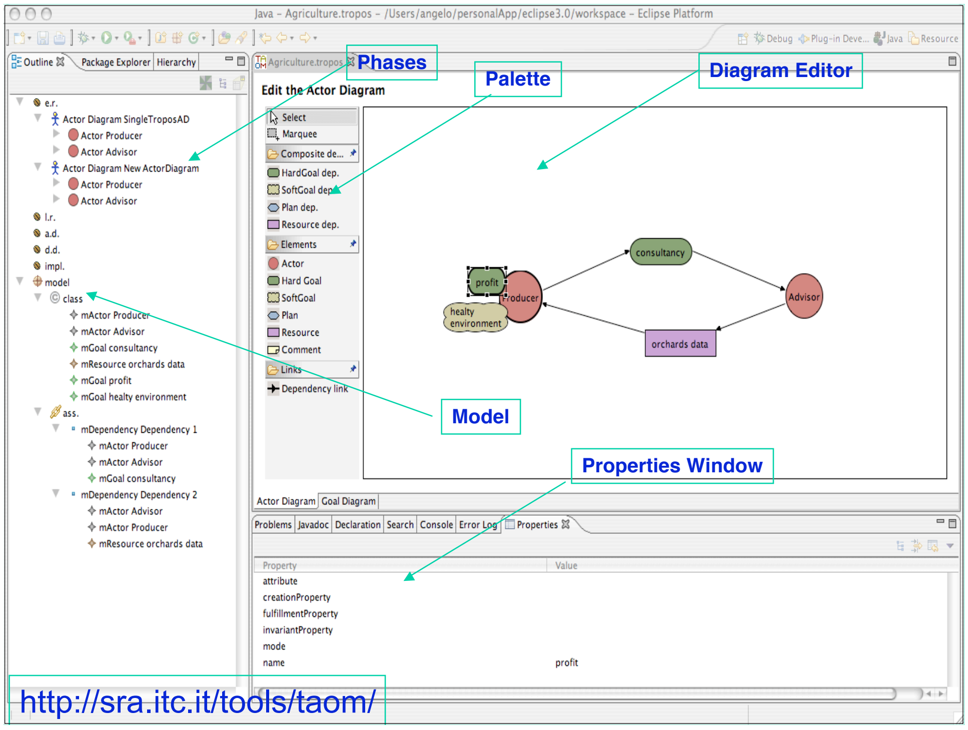
  ✓ decomposes it into new subgoals;

  ✓ 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

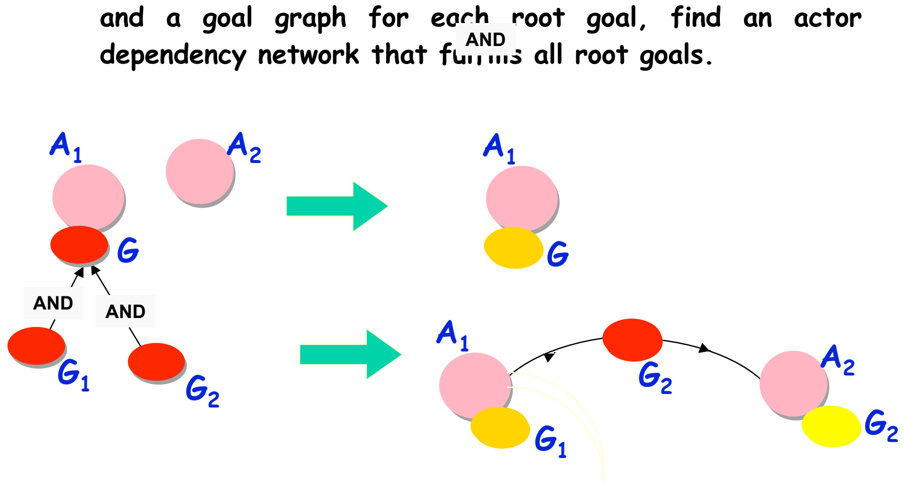
- Models are used primarily for human communication
- But, this is not enough! Large models can be hard to understand, or take seriously.
- We need analysis techniques which offer evidence that a model makes sense:
  - ✓ *Simulation* through model checking, to explore the properties of goals, entities, etc. over their lifetime [RE'01, RE'03, REJ];
  - ✓ *Goal analysis* uses a SAT prover to determine whether a goal can be fulfilled [ER'02, JoDS'03, CAiSE'04];
  - ✓ *Social analysis* uses a planner to explore alternative delegations for a given set of actors and goals.
- The tools we have developed use off-the-shelf inference engines (respectively nuSMV, MinWeight solver, LPG-td).



http://sra.itc.it/tools/taom/

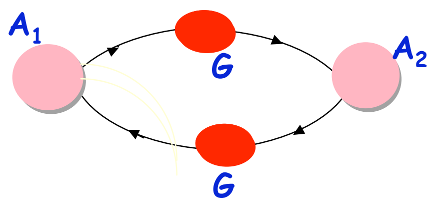
Social Analysis

→ Given a set of actors, each with associated root goals, and a goal graph for each root goal, find an actor dependency network that fulfills all root goals.



Well-Formed Dependency Networks

→ Some dependency graphs don't make sense...



→ What is a "good" dependency network, assuming that we are interested in:
- ✓ minimizing dependence;
- ✓ distributing work;
- ✓ network stability

→ PhD thesis by Volha Bryl (Trento).

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, Yudis Asnar, Volha Bryl, Paolo Traverso,...;
  - ✓ Elsewhere: Jaelson Castro (Brazil), Matthias Jarke (Germany), Manuel Kolp (Belgium), Julio Leite (Brazil), Gerhard Lakemeyer (Germany), Lin Liu (China);
- Publications and other information about the project can be found at http://www.troposproject.org.

This Talk

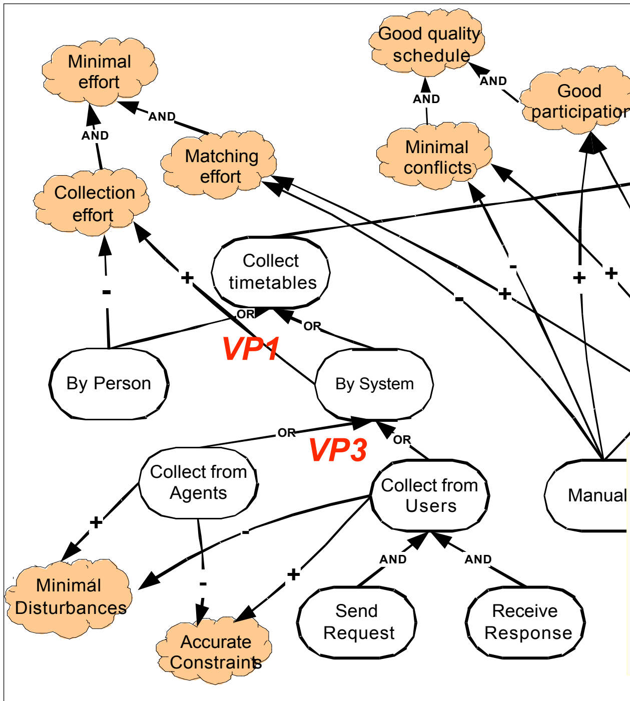
- GORE -- History and key ideas
- Moving forward -- Tropos
- *Moving forward -- Designing high-variability software*
- *Afterthoughts and conclusions*

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.

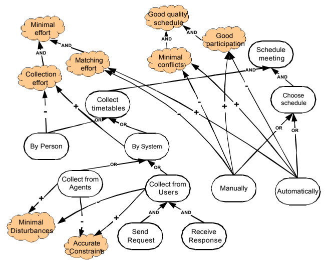
On-Going Research

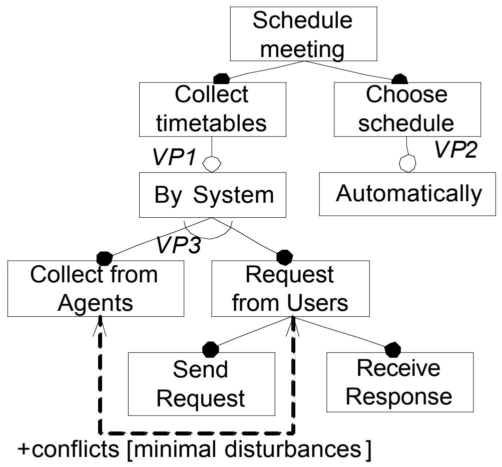
- → From goals to generic designs: Develop a tool-supported method for generating different design views from a given goal model; in our work we have focus on the generation of a feature model, a statechart model and a software architecture.
- → Characterize variability: Goal models constitute one source of variability in design, but there are also others. These may be dependent on what is the design about (e.g., software, business process, database) [RE'06a, RE'06b].
- → PhD theses by Sotiris Liaskos, Alexei Lapouchnian, Lei Jiang (Toronto).



From a Goal Model to Design Views

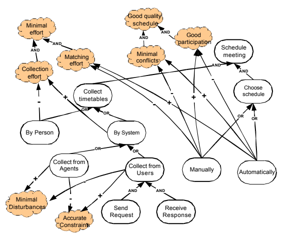
… to a Feature Model

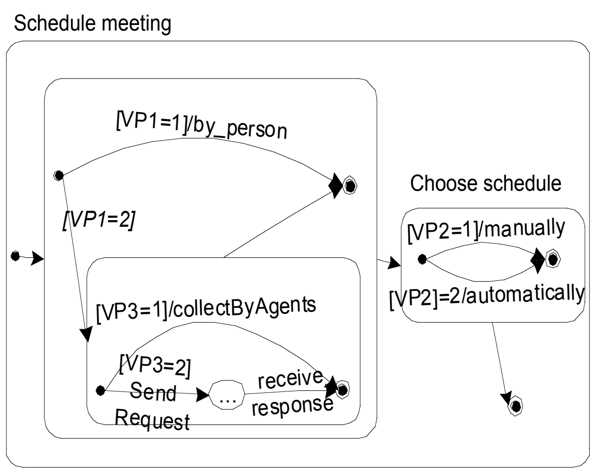




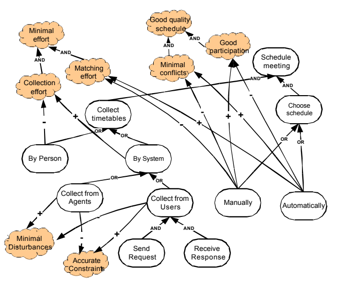
+conflicts [minimal disturbances]
+conflicts [accurate constraints]

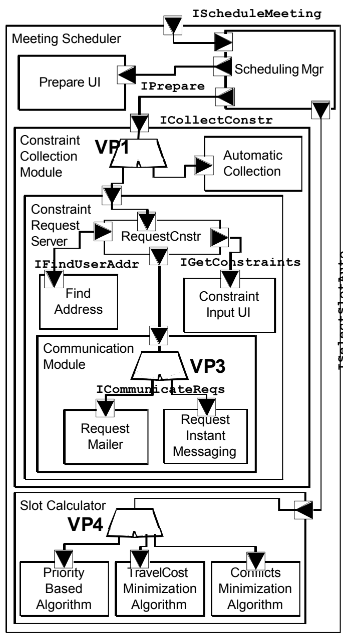
… to a Statechart





… to a Software Architecture

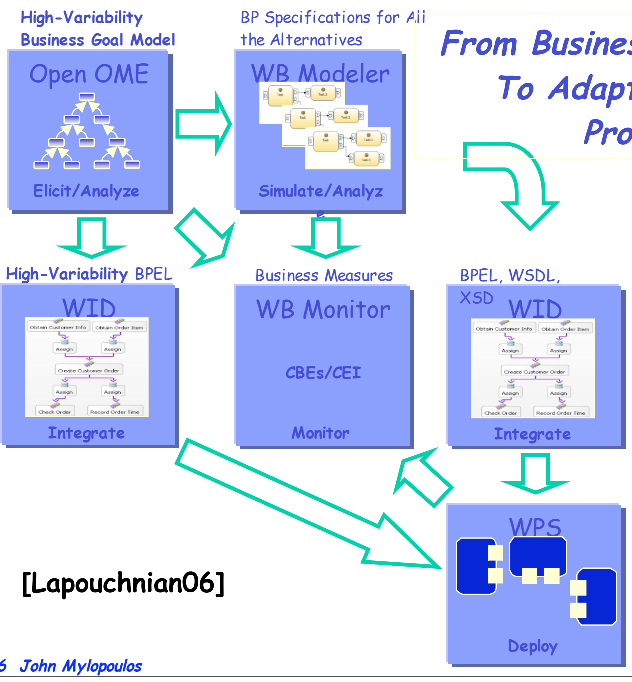




Why is this Problem Important?

- 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 (Alexei Lapouchnian);
  - ✓ Home care software for the elderly (Sotiris Liaskos).

From Business Requirements To Adaptive Business Processes



[Lapouchnian06]

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
- PhD thesis by Yiqiao Wang (Toronto) is looking at the problem of designing monitoring and diagnostic mechanisms for autonomic software.

Other Threads

- [Security] Extend Tropos to support concepts of ownership, permission and trust; this leads to models where you can check whether every actor has the permissions she needs to carry out her obligations [RE'05]; PhD thesis by Nicola Zannone (Trento).
- [Risk Management] Extend the DDP risk management framework [Feather05] to allow hierarchical goal/requirement and risk decompositions; PhD thesis by Yudis Asnar (Trento).

This Talk

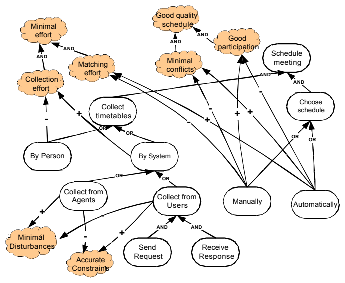
- → GORE -- History and key ideas
- → Moving forward -- Tropos
- → Moving forward -- Designing high-variability software
- → *Afterthoughts and conclusions*

Research on Modelling

- → Much research in RE involves building and analyzing models.
- → Not surprisingly, the quality of such research depends critically on (a) the modelling language used, (b) the reasoning facilities provided.
- → Two pitfalls to avoid:
  - ✓ Treating variations of modelling languages as contributions;
  - ✓ Proposing new modelling languages, instead of building on what already exists.
- → There is a very rich theory of (formal) models and reasoning support to be found in Knowledge Representation research.

Formal vs Informal

- Many of the modelling language we (in the RE community) use are formal. There are great (and known) pitfalls in using informal ones.
- Most of the models we built are semi-formal; this means that the models capture and formalize some aspects of the domain, but not all. This is the nature of the SE enterprise …
- For example,



The Big Picture …

- We are working on characterizations of design spaces for software, business processes, databases, ….
- These spaces are partly defined by variations in goal fulfillment and actor delegation strategies, but also on other dimensions.
- We are also working on a theory for evaluating design alternatives with respect to a set of criteria consisting of stakeholder needs and preferences.
- These together can form the basis for a theory of design, along the lines of Herbert Simon's vision for a Science of Design [Simon69].

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 often under-constrained, include too many variables [Menzies99].
- To have autonomic/adaptive systems, models and their implementations will need to evolve at run-time.
- … For sure, other things as well …

Conclusions

- GORE introduces *new concepts* in software development processes.
- We have argued that GORE shows a path towards defining and analyzing *design spaces* for software.
- We also sketched two applications of GORE concepts:
  - ✓ Designing agent-oriented software -- the Tropos project [JAAMAS04, JInformationSystems03];
  - ✓ Designing high-variability software [RE'03, RE'06a, RE'06b].

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.