Software Architectures as Social Structures

John Mylopoulos
University of Toronto

First ICSE Workshop titled "From Software Requirements to Architectures" (STRAW'01)
Toronto, May 14, 2001

...An Idea...

- 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?***

Requirements Concepts?

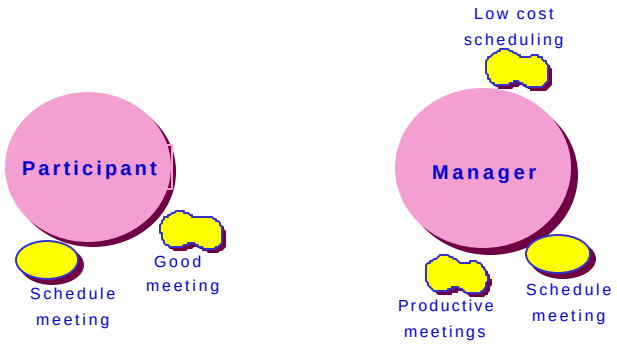
- We distinguish between early and late requirements.
- Early requirements are concerned with understanding the organizational context within which the system-to-be will eventually operate.
- Late requirements focus on identifying (and modeling) functional and non-functional requirements for the system-to-be.
- We turn to early requirements for inspiration... Late requirements are ***so*** 20th century...

Early Requirements Concepts?

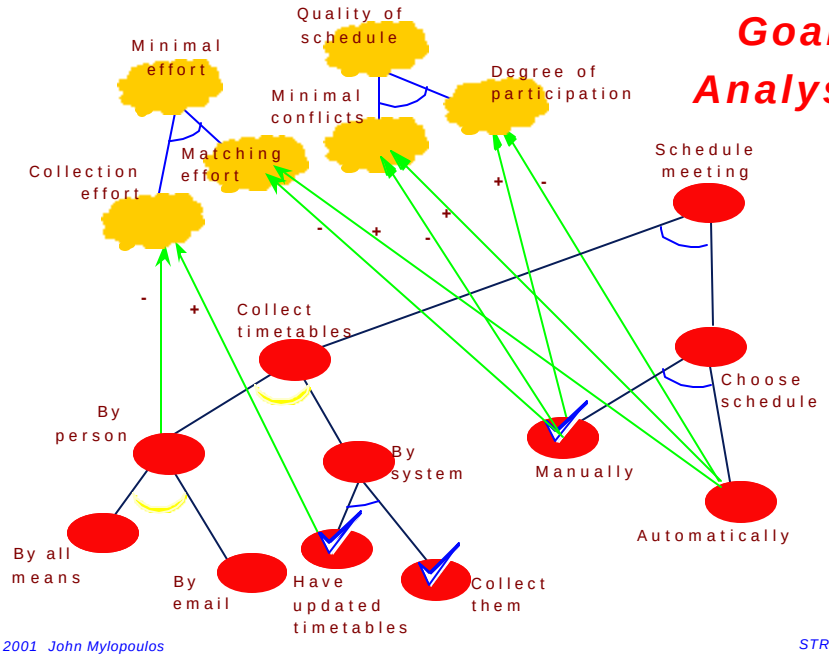
- The organizational environment of a software system can be conceptualized as a set of business processes, actors and/or goals.
- The KAOS project defines the state-of-the-art on modeling early requirements in terms of goals; also offers well-developed analysis techniques and tools for generating late requirements.
- We focus on actors and goals. In particular, we adopt the *i** framework of Eric Yu [Yu95].

Actors and their Goals

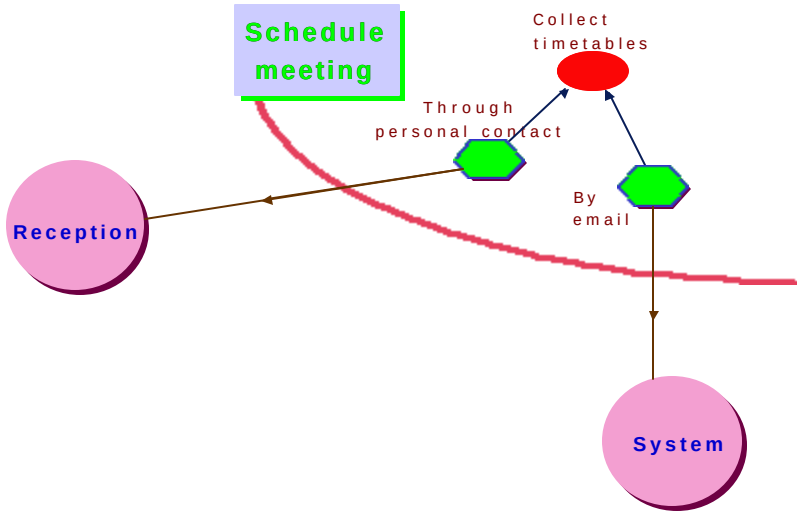
A social setting consists of actors, each having goals (and/or softgoals) to be fulfilled.



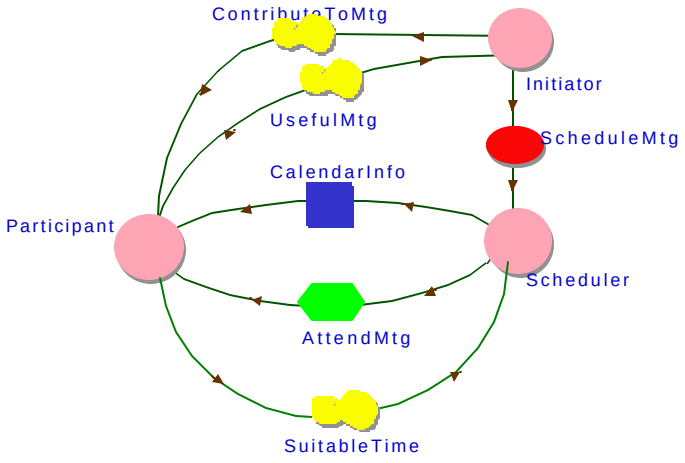
Goal Analysis



Actor Dependencies



Actor Dependency Models



Tasks vs Goals

- Tasks are processes for fulfilling goals.
- In general, there will be many tasks for fulfilling a single goal.
- When actors are assigned goals, they are (or, ought to be) equipped with means for fulfilling them by carrying out one or more tasks.

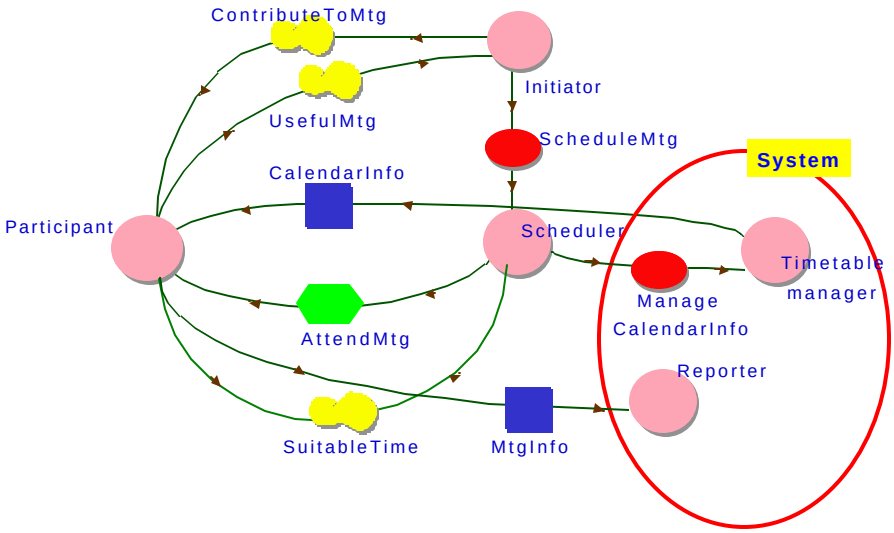
Goals vs Softgoals

- Goals come with a predicate which defines when they are fulfilled.
- Softgoals are "natural kinds" or "primitive terms" in the sense that they don't have such a predicate. However, they can still be analyzed and the degree of their fulfillment can be evaluated in terms of other goals.

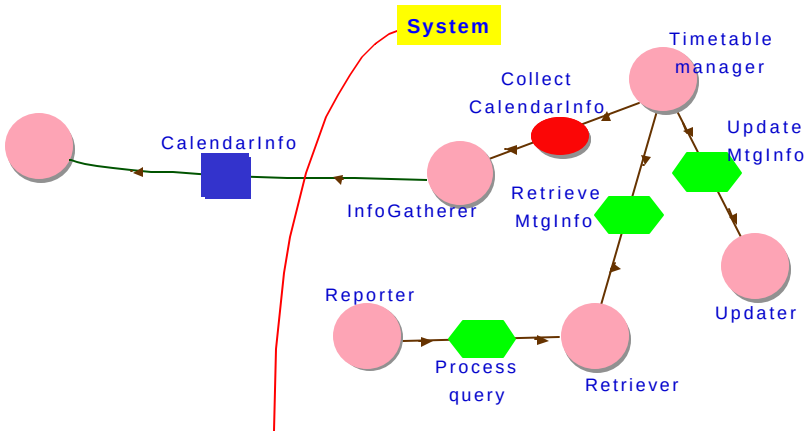
Using These Concepts

- During early requirements, these concepts are used to model external stakeholders (people, organizations, existing systems), their relevant goals and inter-dependencies.
- During late requirements, the system-to-be enters the picture as one or a few actors participating in *i** models.
- During architectural design, the actors being modelled are all system actors.

Late Requirements with i*



Software Architectures with i*



What is Different?

- Goal refinement extends functional decomposition methods in the sense that it explores alternatives.
- Actor dependency graphs extend object interaction diagrams in that a dependency is monitored and may be discarded, also it can be established at run-time.
- In general, an actor architecture is open and dynamic; evolves through negotiation, matchmaking and like-minded mechanisms.
- The distinction between design and run-time is blurred.
- So is the boundary between a system and its environment (software or otherwise.)

Why is this Better (…Sometimes…)

- Traditionally, goals (and softgoals) are operationalized and/or metricized before late requirements.
- This means that a solution to a goal is frozen into a software design early on and the architect has to work within the confines of that solution.
- This won't do in situations where the operational environment of a system, including its stakeholders, keeps changing.

Architectural Styles Revisited

- If the nature of the building blocks of software architectures changes, so should the relevant design styles
- Where do we turn for ideas?
  - ✓Try organizational theory for (more-or-less) tightly structured actor networks (see [Kolp, this workshop]);
  - ✓More generally, try social theory (social networks, communities and the like); see work by Jarke et al;

Potential Applications

- Consider agent-based systems and the foothold they are gaining in eCommerce software practice.
- Or, consider Application Service Providers and the idea that a software application is composed dynamically from a number of externally provided services.
- Or, consider peer-to-peer (P2P) computing where the stakeholders of a P2P network are defined dynamically and at run time, also services are developed bottom up by "factoring out" commonly performed tasks and common interests/goals among peers.

Tropos

- Adopt *i** to support modeling and analysis from early requirements to detailed design.
- We are developing a formal Tropos language (inspired by KAOS) to augment *i** diagrams.
- Developing a set of analysis techniques for Tropos, including temporal analysis (using model checking), goal analysis, and (...soon...) social structure analysis..
- Also developing a methodology which supports software development from early requirements to detailed design.
- This is a long-term project with most of the research work ahead -- rather than behind -- us.

Conclusions

- There is a fundamental difference between requirements -- which are basically intentional in nature -- and architectures -- which describe extensional structures and behaviours.
- Our proposal eliminates this difference altogether in the name of software malleability...

...we don't want to freeze any more solutions to a given requirement in the software designs we produce!!

What is Software?

- A mathematical abstraction, a theory, which can be analyzed for consistency and can be refined into a more specialized theory (***Mathematical perspective***)
- An engineering artifact, which is designed, tested and deployed using engineering methods; the methods rely heavily on testing and inspection for validation (***Engineering perspective***)
- A non-human agent, with its own personality and behavior, defined by its past history and structural makeup (***CogSci perspective***)
- A social structure of software agents, who communicate, negotiate, collaborate and cooperate to fulfil their goals (***Social perspective***)