Engenharia de Software Orientada a Agentes

WESAAC 2014

Anarosa Alves Franco Brandão
anarosa.brandao@usp.br
Escola Politécnica – Universidade de São Paulo

INTRODUÇÃO

Introdução

- Engenharia de software
  - Disciplina da engenharia que se ocupa dos aspectos relacionados à produção de software (Somerville, 2001)
  - Disciplina que estuda a criação e aplicação de abordagens sistemáticas, disciplinadas e quantificáveis para o desenvolvimento, operação e manutenção de software. (IEEE Computer Soc.)

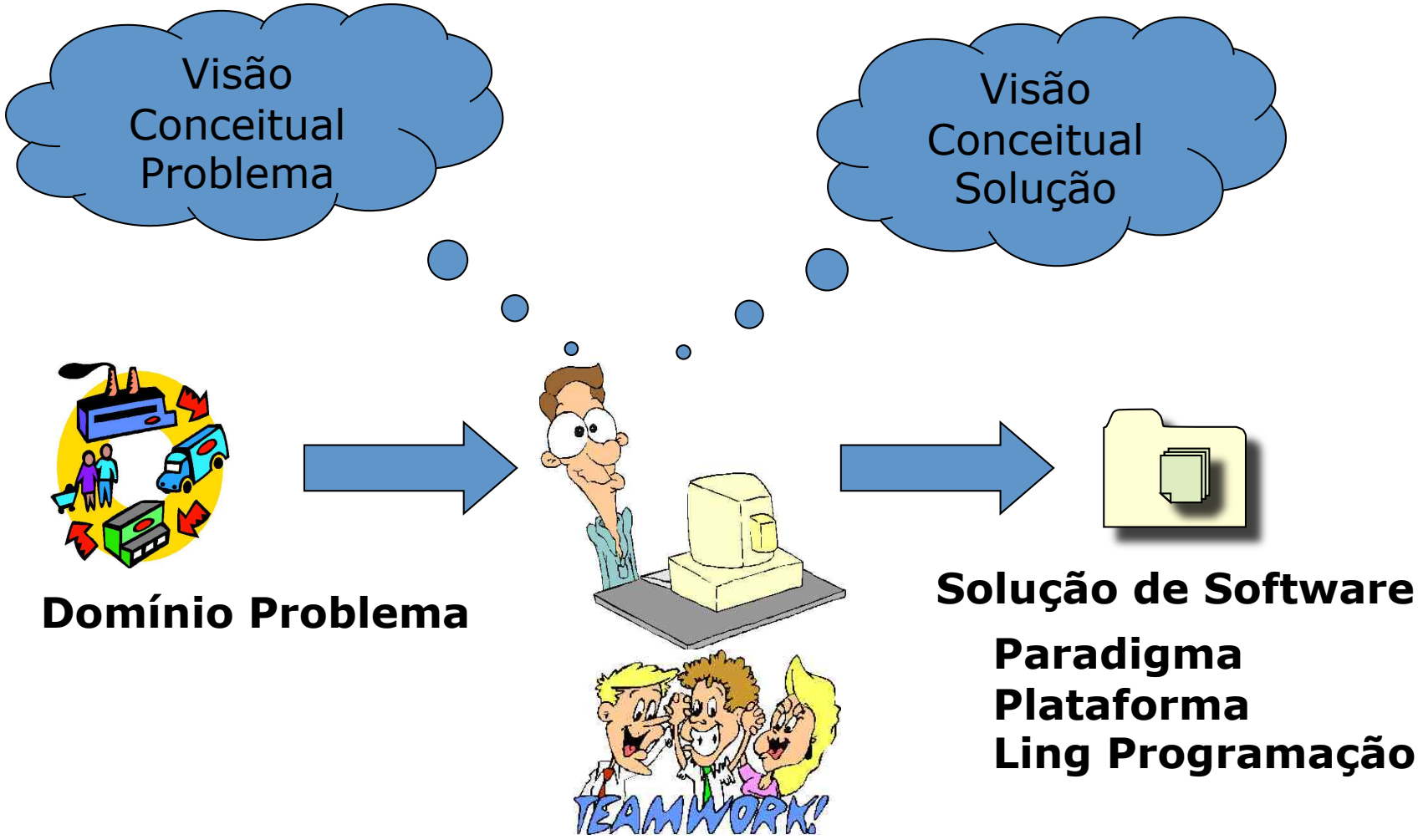
Engenharia de software em camadas (Pressman, 2005)

Introdução

- Software
  - Programas de computadores, documentação associada e dados de configuração necessários para fazer os programas funcionarem corretamente (Sommerville, 2001)
  - Produto da engenharia de software
    - Produto de engenharia é sempre resultado da condução de um processo

Introdução

- Produção de software

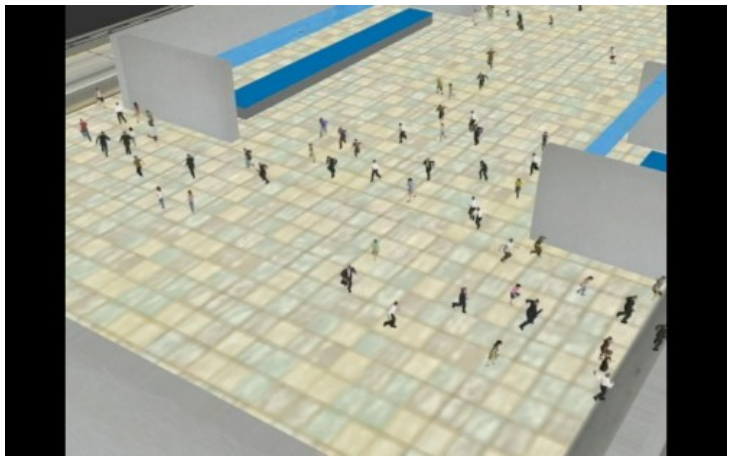


Introdução

- Engenharia de software orientada a agentes (AOSE)
  - Disciplina de engenharia de software que adota agentes de software como paradigma de desenvolvimento de software
  - Agente de Software
    - Sistema de computação que é situado num ambiente, onde atua de forma autônoma a fim de atingir seus objetivos. (Wooldridge, 2009)

Introdução

- Sistemas multiagentes (SMA)
  - Conjunto de agentes possivelmente organizados que interagem num ambiente comum. (Demazeau, 2000)
  - Produto da AOSE



Introdução

- Mas por que adotar uma solução de software que usa agentes?
  - Alguns domínios são adequados a ela!
  - Alguns domínios nos quais a abordagem de agentes já foi aplicada
    - Sistemas de manufatura e controle ambiental
    - Computação móvel e pervasiva
    - Internet
    - E-commerce

AGENTES X OBJETOS

Agentes X Objetos

- Considere o domínio da reengenharia de processos de negócios:
  - Desenvolver um sistema (software) para auxílio a reengenharia de processos de negócios
    - Criar modelos da organização e funcionamento de um negócio
    - Realizar simulações a partir dos modelos

Agentes X Objetos

- Solução orientada a objetos
  - Tudo é modelado como objeto
    - Entidade com identidade
  - Objetos são abstrações e/ou encapsulamentos
    - Dados
    - Realização de comportamentos
  - Objetos interagem
    - Interações entre objetos refletem dependências funcionais

Agentes X Objetos

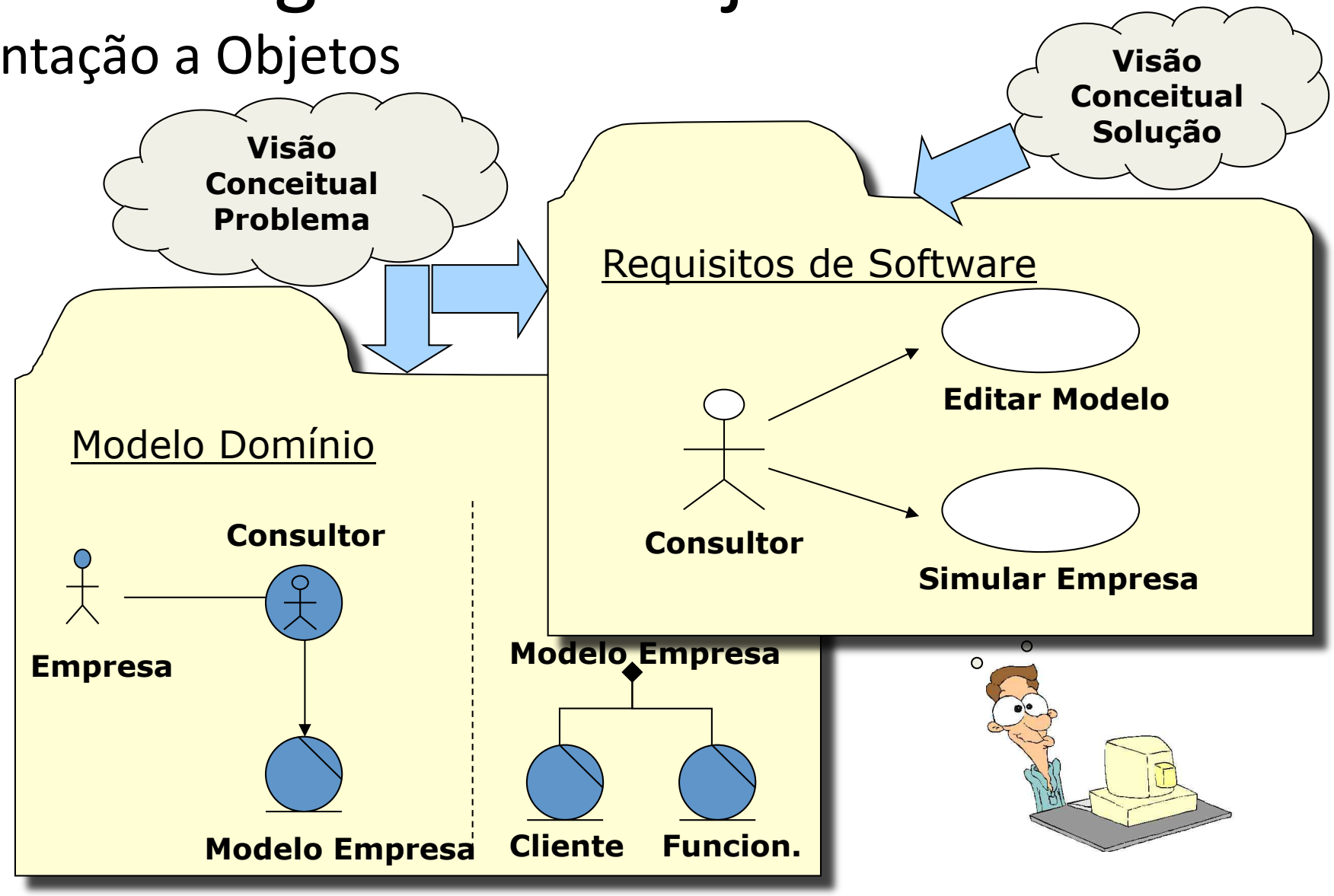
- Solução orientada a agentes
  - Além de objetos existem agentes, que são entidades
    - Com identidade
    - Situadas
      - Estão inseridas num ambiente onde percebem e atuam
    - Autônomas
      - Possuem controle sobre suas ações
    - Pró-ativas
      - Além de responder a estímulos também perseguem objetivos

Agentes X Objetos

- Solução orientada a agentes
  - Alguns agentes são sociais
    - Agentes sociais são capazes de se comunicar diretamente com outros agentes
  - Agentes, sociais ou não, formam sistemas multiagentes
    - População de agentes
    - Ambiente
    - Protocolos de Interação
    - Organização Social

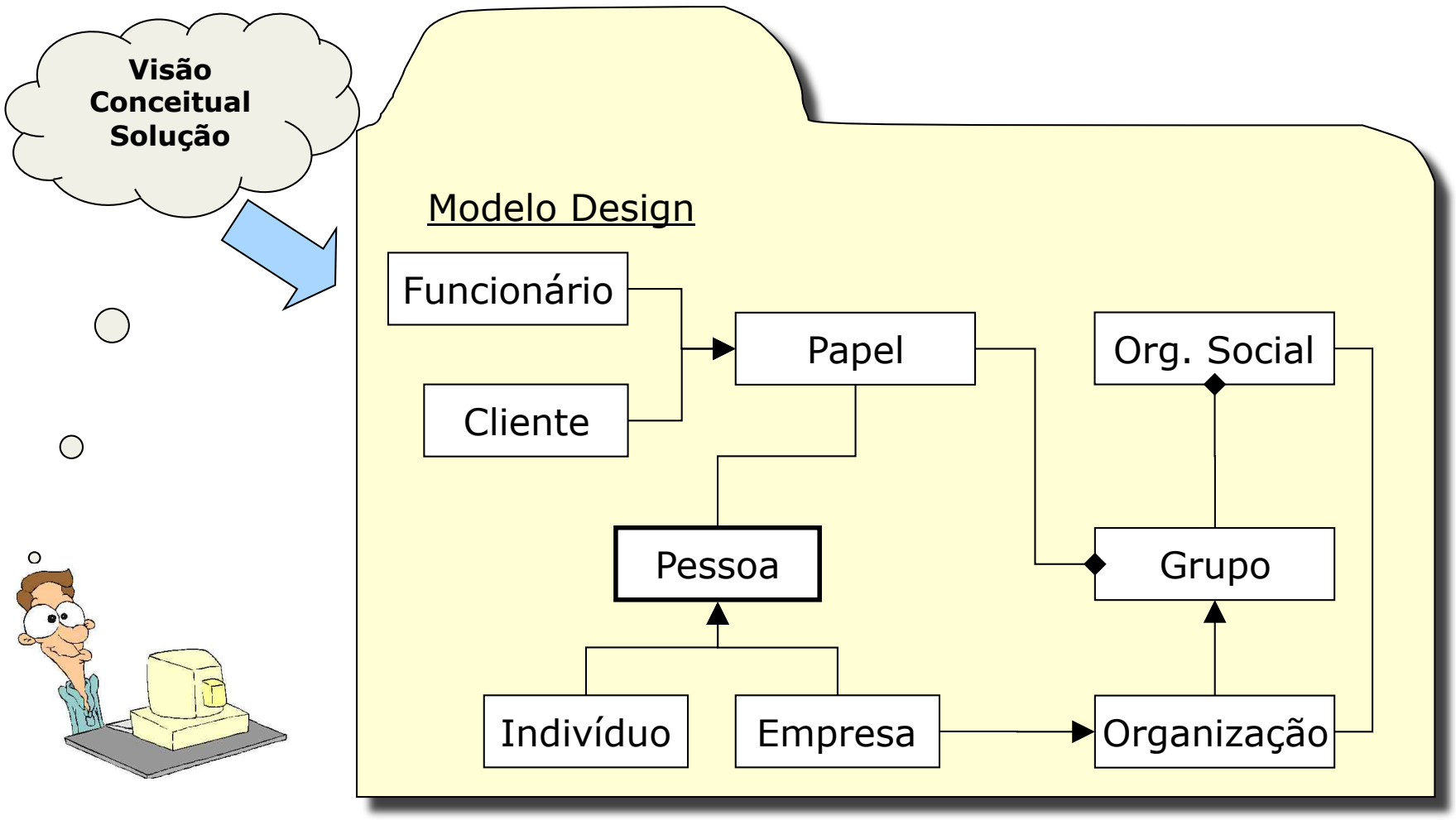
Agentes X Objetos

- Orientação a Objetos



Agentes X Objetos

- Orientação a Objetos



Agentes X Objetos

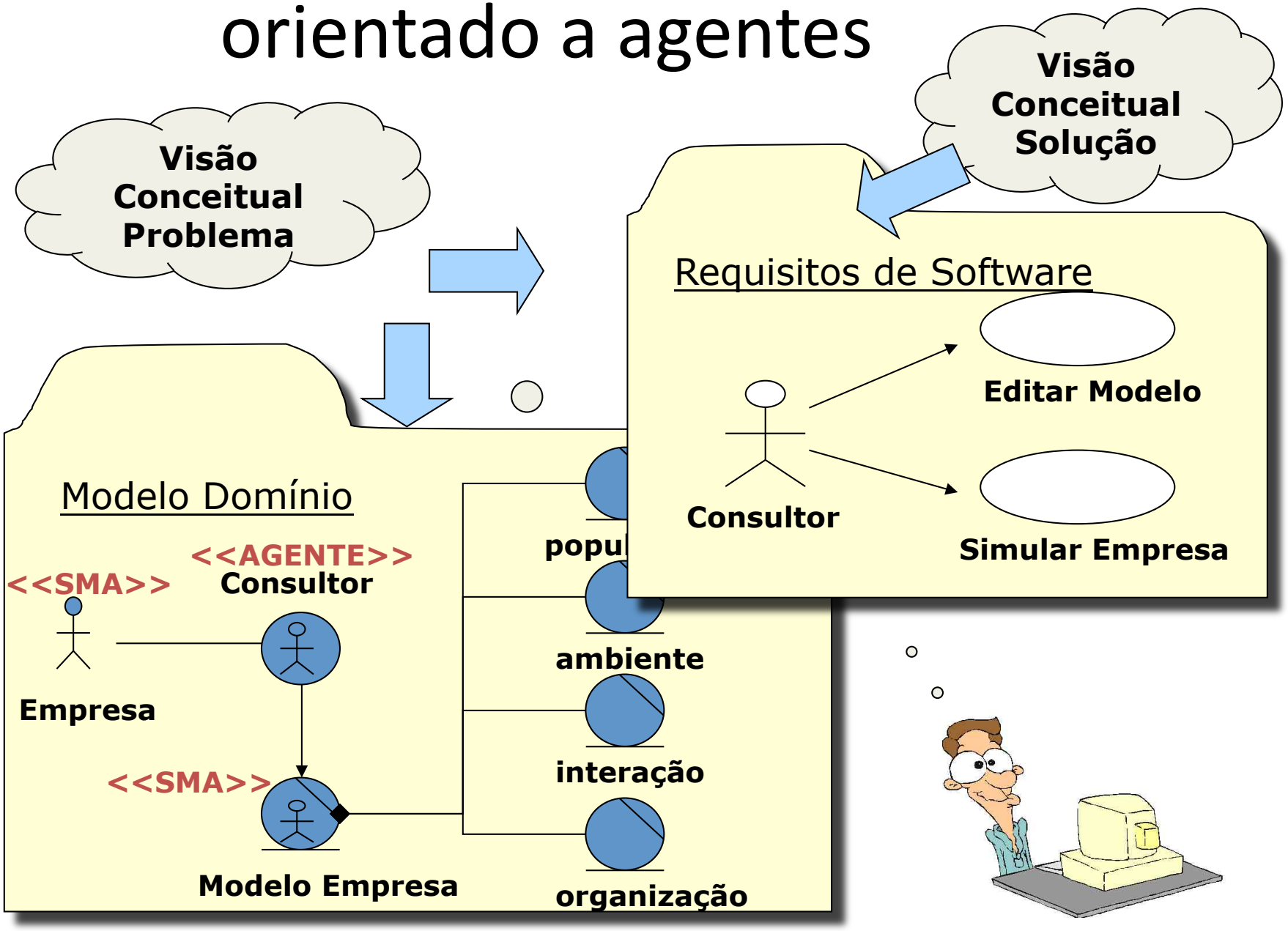
- Orientação a Objetos
  - Alguns Requisitos Problemáticos
    - Durante modelagem e simulação do funcionamento de uma empresa, o consultor deseja levar em consideração o fato de que as pessoas que a formam são autônomas, pró-ativas e sociais
      - Pessoas podem dizer não quando outra manda uma mensagem solicitando algum serviço
      - Elas agem ativamente seguindo seus próprios objetivos
      - Pessoas quando interagem seguem alguns protocolos socialmente estabelecidos
    - Ainda, o consultor deseja distinguir pessoas dos recursos e elementos que formam um ambiente físico no qual estão imersas
    - Por fim, o consultor deseja estudar como a organização de uma empresa leva a empresa a ser eficiente e como pessoas autônomas e pro-ativas se adaptam a organização e adaptam a organização a elas

Agentes X Objetos

- Orientação a Objetos
  - Observamos que:
    - o domínio de problema é altamente complexo
    - a orientação a objetos é um paradigma onde os conceitos básicos (objetos+ativação de métodos) se encontram em um nível de abstração muito baixo para o domínio de problema
    - a aplicação pode ser desenvolvida utilizando-se somente objetos mas o trabalho será enorme
  - Agentes surgem como alternativa...

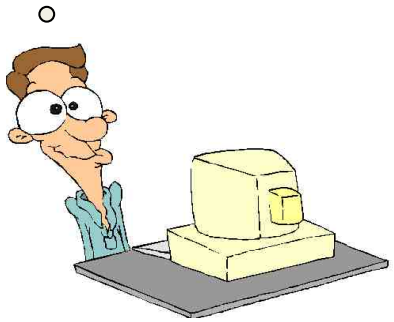
DESENVOLVIMENTO DE SOFTWARE ORIENTADO A AGENTES

Desenvolvimento de Software orientado a agentes



Desenvolvimento de Software orientado a agentes

Visão Conceitual Solução



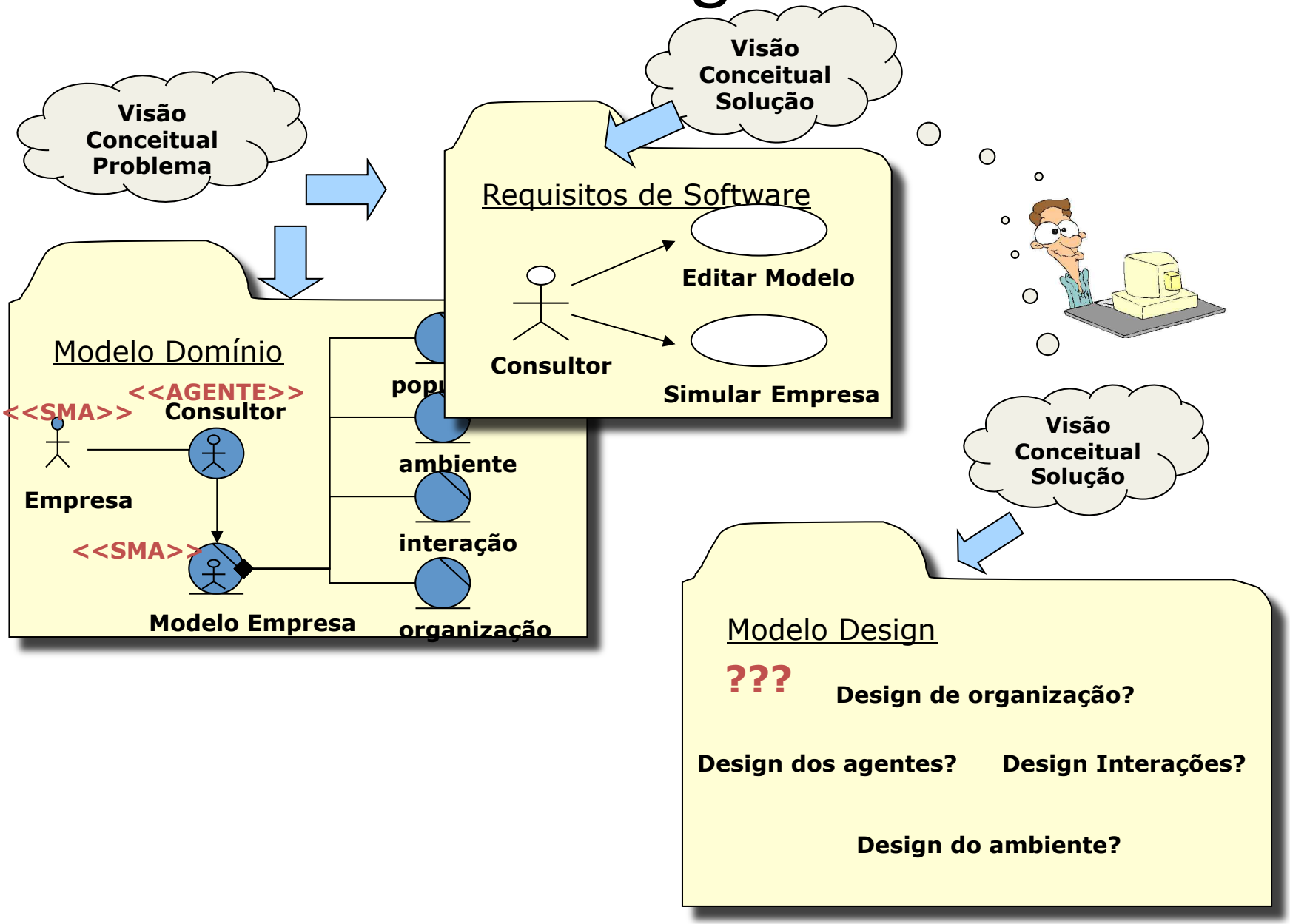
Modelo Design

??? **Design de organização?**

Design dos agentes? **Design Interações?**

Design do ambiente?

Desenvolvimento de software orientado a agentes



Desenvolvimento de software orientado a agentes

- Orientação a agentes (focar em quê?)

  "A key question …

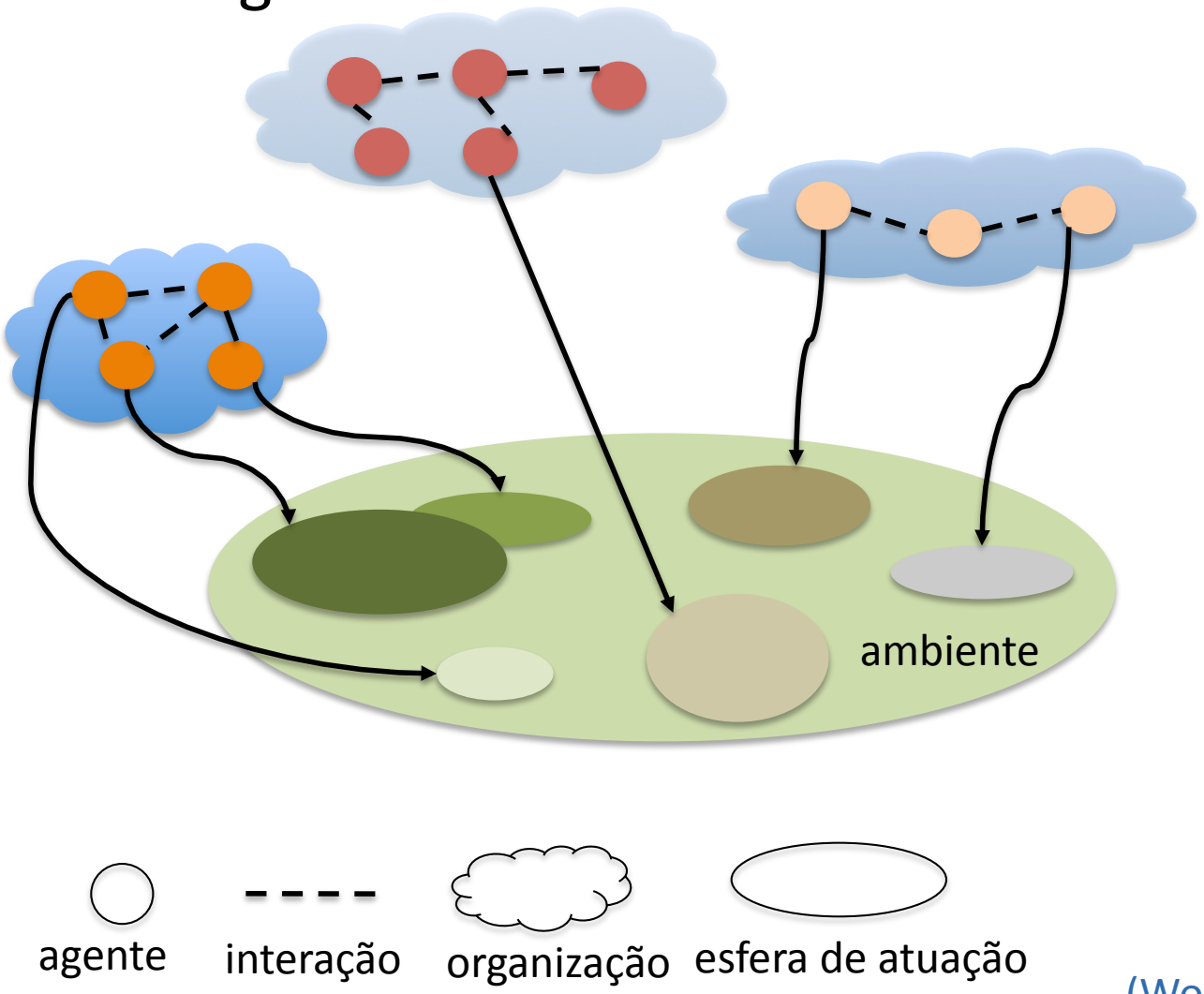
  ... what is the scale of observation which one should situate the engineering work?

  ..., any effective approach to AOSE requires fixing the scale of observation, to give meaning to the concept of software systems and to enable the identification of the issues that have to be faced in their engineering."

(Zambonelli & Omicini, 2004)

Desenvolvimento de software orientado a agentes

- Sistemas multiagentes



(Wooldridge, 2009)

Desenvolvimento de software orientado a agentes

- Orientação a agentes (diferentes abordagens)
  - Micro (centrado no agente)
    - Detalhar características de cada agente, mecanismos de interação agente-agente e agente-ambiente
    - Comportamento global emergente
  - Macro (centrado na organização)
    - O entendimento e controle do comportamento global do sistema é o interesse principal
  - Misto
    - Controle global X emergência

Desenvolvimento de software orientado a agentes

- Orientação a agentes (diferentes abordagens)
  - Como utilizar a tecnologia de agentes e SMA de forma
    - Efetiva
    - Controlável
    - Previsível
    - Econômica
    - …

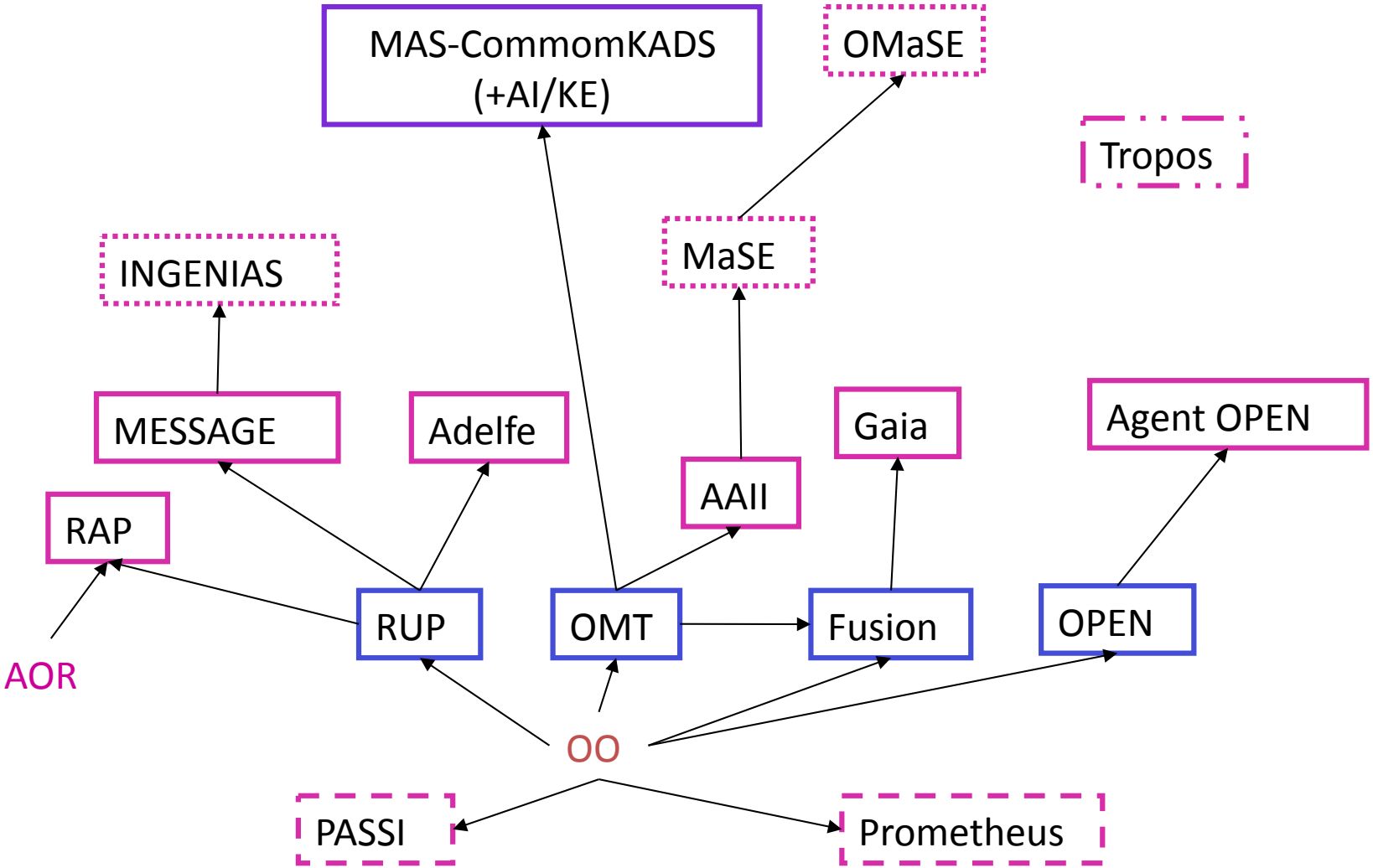
  para desenvolver aplicações no mundo real?

METODOLOGIAS AOSE

Metodologias AOSE

- Dados os conceitos importantes, vamos falar de AOSE
  - Disciplina recente da Engenharia de Software
  - Histórico começa nos anos de 1990
    - Foco na definição de metodologias para apoiar o desenvolvimento de SMA
    - Pouca ou nenhuma atenção aos aspectos organizacionais de SMA

Metodologias AOSE



Adaptada de (Henderson-Sellers & Giorgini, 2005)

Metodologias AOSE

- Caracterizam-se por:
  - conter abstrações necessárias para modelar agentes e SMAs; (metamodelo)
  - foco em sociedades (organizadas) de agentes;
  - agentes situados em ambientes;
  - agentes desempenhando papéis para atingir metas;
  - agentes interagindo seguindo protocolos de comunicação.

Metodologias AOSE

- Prescrevem atividades de (não necessariamente todas):
  - Requisitos
  - Analise
  - Design
  - Implementação
  - Teste

Metodologias AOSE

- Atividades de (elicitação de) requisitos
  - identificar, organizar e documentar o conjunto de capacidades que os usuários esperam que o software deve apresentar para resolver o problema deles ou atingir seus objetivos.

Metodologias AOSE

- Atividades de análise
  - Analisar os requisitos do sistema com a finalidade de descrever o modelo conceitual da solução do problema que o software deve resolver
  - Pode incluir (ou englobar) atividades de requisitos

Metodologias AOSE

- Atividades de design
  - Desenvolver a arquitetura do sistema em termos de seus subsistemas, componentes etc e refinar o modelo conceitual da solução para descrever uma especificação da solução a ser codificada.

Metodologias AOSE

- Atividades de implementação
  - Construir o software especificado através de sua codificação usando uma ou mais linguagens de programação, produzindo código fonte, executáveis, scripts, etc

Metodologias AOSE

- Atividades de teste
  - Relacionadas à verificação e validação do programa implementado

History of AOSE: Three Generations

1. mid to late 90s
   - Examples: DESIRE, AAII, MAS-CommonKADS, Gaia
   - Generally briefly described
   - Lacking tool support
   - May not cover full life cycle
2. late 90s to early 00s
   - Examples: MaSE, Tropos, MESSAGE, Prometheus
   - More detailed descriptions
   - Tend to have tool support
   - Tend to cover Requirements to Implementation
3. mid to late 00s
   - Examples: PASSI, INGENIAS, ADEM
   - Increased focus on UML and Model-Driven Development
   - Tend to be more complex

Year	Methodologies
1995	DESIRE
1996	AAII, MAS-CommonKADS
1999	MaSE
2000	Gaia (v1), Tropos
2001	MESSAGE, Prometheus
2002	PASSI, INGENIAS
2003	Gaia (v2)
2005	ADEM
2007	O-MaSE

Winikoff, 2013

Metodologias AOSE

- Menor foco no desenvolvimento de novas metodologias a partir de meados da década de 2000
  - Movimentação para padronização (IEEE-FIPA)
  - Reuso (de fragmentos) de metodologias existentes
    - Ex: Medee Framework

The IEEE FIPA Standards Committee has two kinds of groups that contribute to the standards process: working groups (WG) and Study Group (SG)

Working Groups

Working Groups are established to carry out standardization projects within the scope of the FIPA SC. Each Working Group shall be responsible for the definitive content of one or more standards projects and for responding to views and objections thereon. A working group is expected to produce its first PAR (Project Authorization Request) within six month of charter acceptance.

The following working groups have been formed and approved:

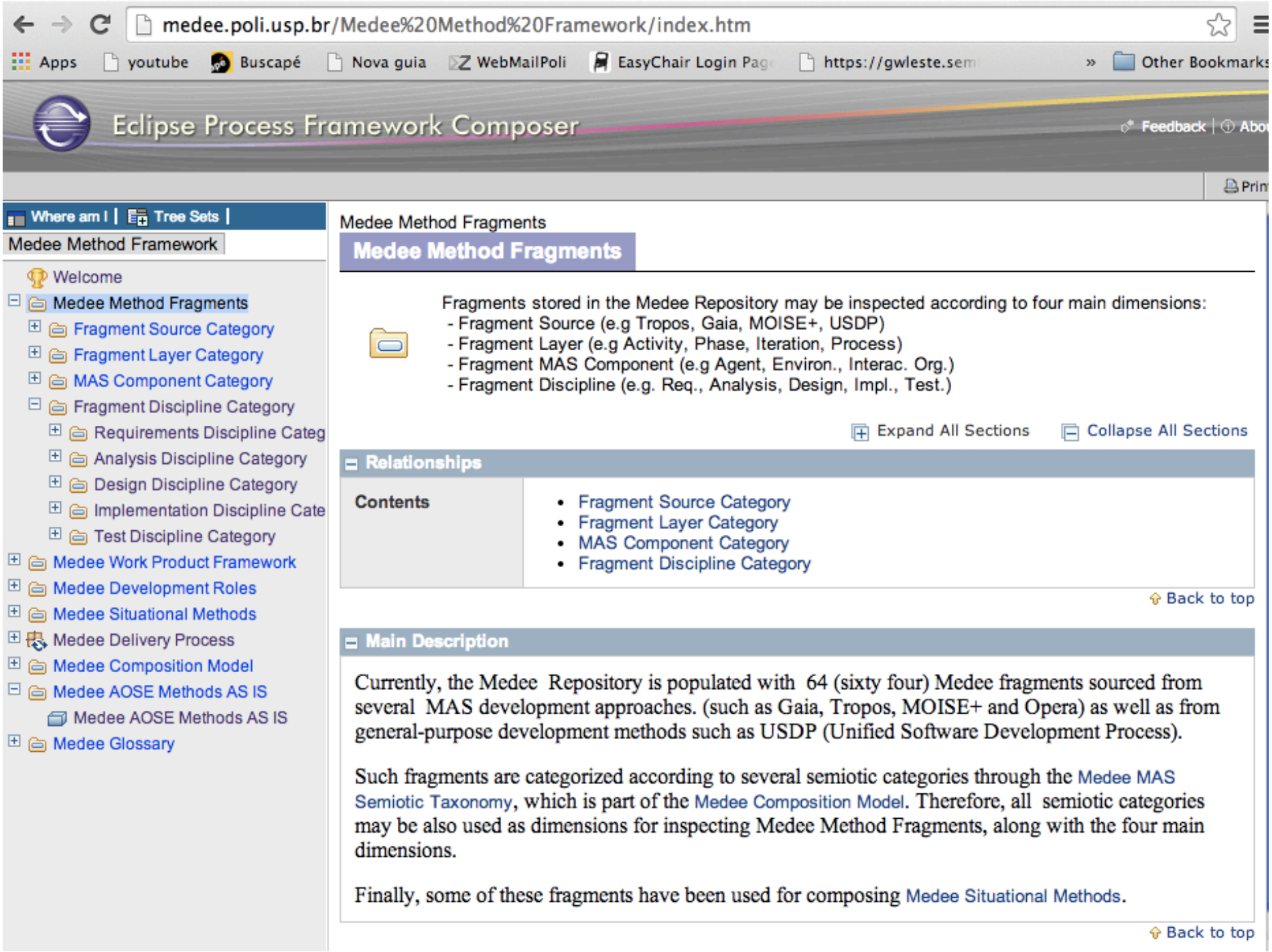
- Agents and Web Services Interoperability Working Group (AWSI WG)
- **Application Specifications Working Group (AS WG)** <-NEW
- **Design Process Documentation and Fragmentation (DPDF WG)** NEW
- Human-Agent Communications Working Group (HAC WG)
- Mobile Agents Working Group (MA WG)
- P2P Nomadic Agents Working Group (P2PNA WG) (closed)

Study Groups

The FIPA SC may form a Study Group for up to 12 months to foster usage standards or the appropriateness of standards or standards development projects. The SG should have a work plan which include specified deliverables. Study Groups do not produce standards specifications.

The following study group has been formed and approved:

- Review of FIPA Specification Study Group (ROFS SG)


medee.poli.usp.br/Medee%20Method%20Framework/index.htm
Apps
youtube
Buscapé
Nova guia
WebMailPoli
EasyChair Login Pag
https://gwleste.sem
Other Bookmarks
Eclipse Process Framework Composer
Feedback
Where am I | Tree Sets |
Medee Method Framework
Welcome
Medee Method Fragments
Fragment Source Category
Fragment Layer Category
MAS Component Category
Fragment Discipline Category
Requirements Discipline Categ
Analysis Discipline Category
Design Discipline Category
Implementation Discipline Cate
Test Discipline Category
Medee Work Product Framework
Medee Development Roles
Medee Situational Methods
Medee Delivery Process
Medee Composition Model
Medee AOSE Methods AS IS
Medee AOSE Methods AS IS
Medee Glossary
Medee Method Fragments
Medee Method Fragments
Fragments stored in the Medee Repository may be inspected according to four main dimensions:
- Fragment Source (e.g Tropos, Gaia, MOISE+, USDP)
- Fragment Layer (e.g Activity, Phase, Iteration, Process)
- Fragment MAS Component (e.g Agent, Environ., Interac. Org.)
- Fragment Discipline (e.g. Req., Analysis, Design, Impl., Test.)
Expand All Sections
Collapse All Sections
Relationships
Contents
Fragment Source Category
Fragment Layer Category
MAS Component Category
Fragment Discipline Category
Back to top
Main Description
Currently, the Medee Repository is populated with 64 (sixty four) Medee fragments sourced from several MAS development approaches. (such as Gaia, Tropos, MOISE+ and Opera) as well as from general-purpose development methods such as USDP (Unified Software Development Process).
Such fragments are categorized according to several semiotic categories through the Medee MAS Semiotic Taxonomy, which is part of the Medee Composition Model. Therefore, all semiotic categories may be also used as dimensions for inspecting Medee Method Fragments, along with the four main dimensions.
Finally, some of these fragments have been used for composing Medee Situational Methods.
Back to top

Tropos

- Tropos é uma metodologia de desenvolvimento dirigido a requisitos
- É fortemente baseada em conceitos usados na fase inicial de requisitos - “early requirements”.

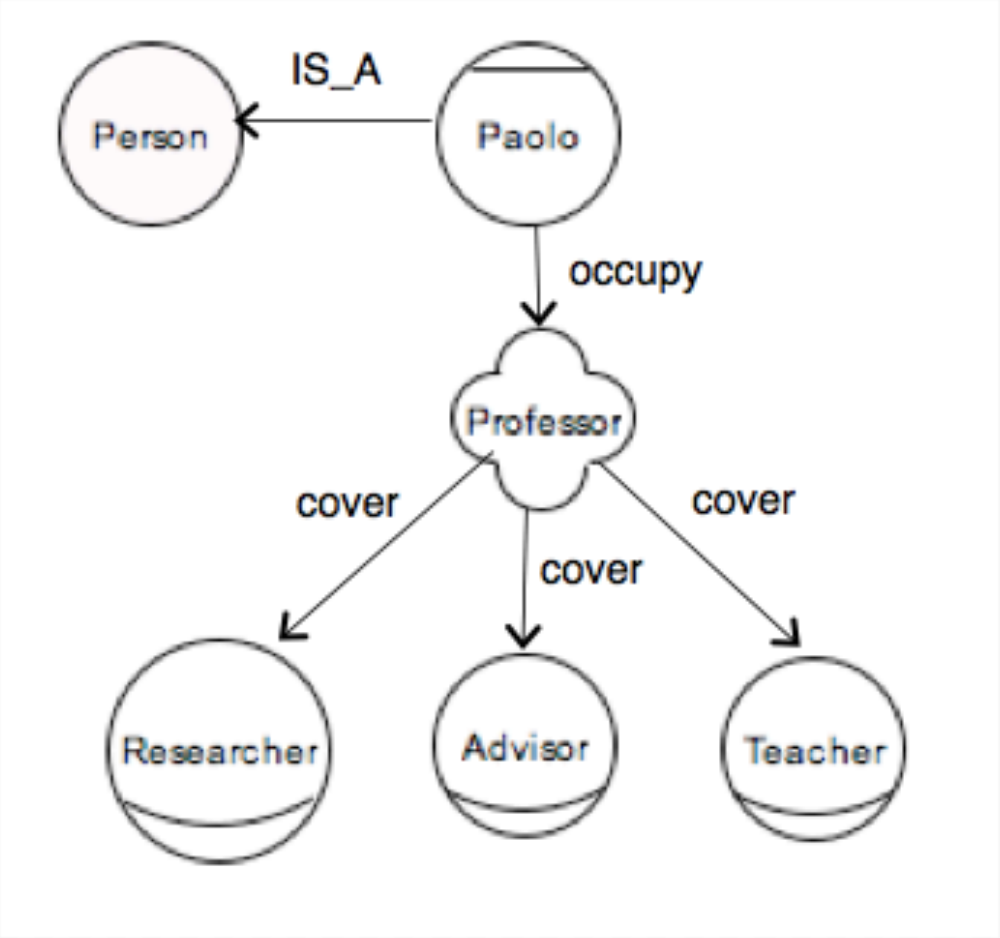
- Para descrever Topos usamos material preparado pelo prof. Paolo Giorgini

http://www.dit.unitn.it/~pgiorgio

Tropos language: basic concepts

- Actor
  - Intentional entity: role, position, agent (human or software)
- Goal (softgoal)
  - Strategic interest of an actor
- Task
  - Particular course of action that can be executed in order to satisfy a goal
- Resource
  - Physical or informational entity (without intentionality)
- Social dependency (between two actors)
  - One actor depends on another to accomplish a goal, execute a task, or deliver a resource

Actor: graphical representation



Paolo is a Person

Paolo occupies the position of Professor

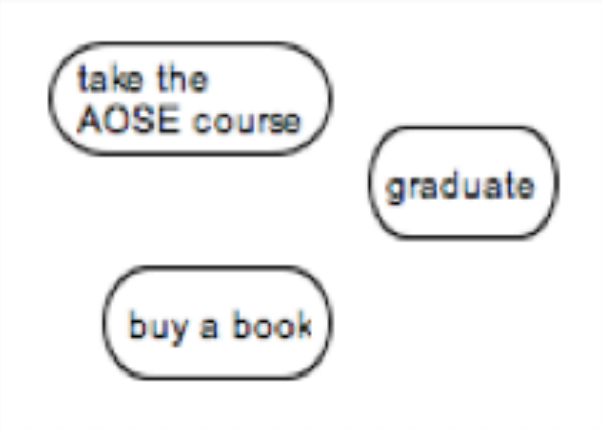
The position Professor covers the roles

- Researcher
- Advisor
- Teacher

---> Paolo plays the role of Researcher, Advisor, Teacher

Goal and softgoal

- A Goal is always associated to an actor
- Goal
  - Hard goal: we have a clear criteria to know when and whether the goal is satisfied
  - Soft Goal: not clearly defined and it is not possible to define a criteria of satisfiability (often used for qualities)

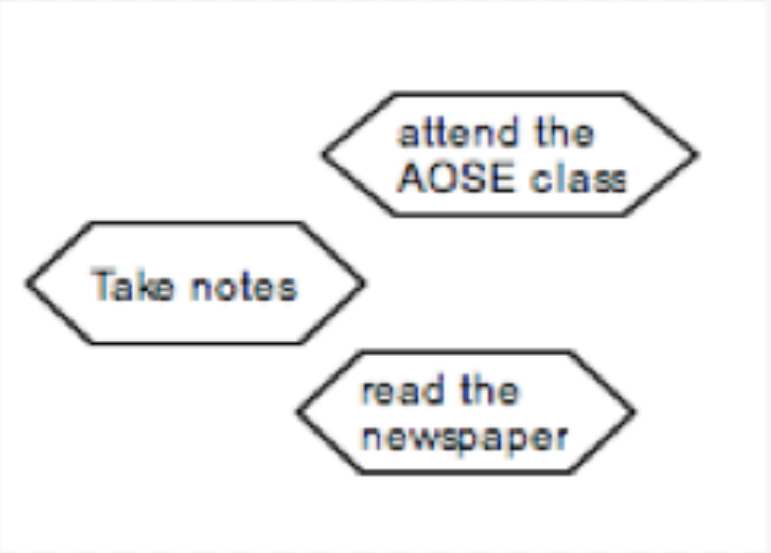


(hard) Goals

Softgoals

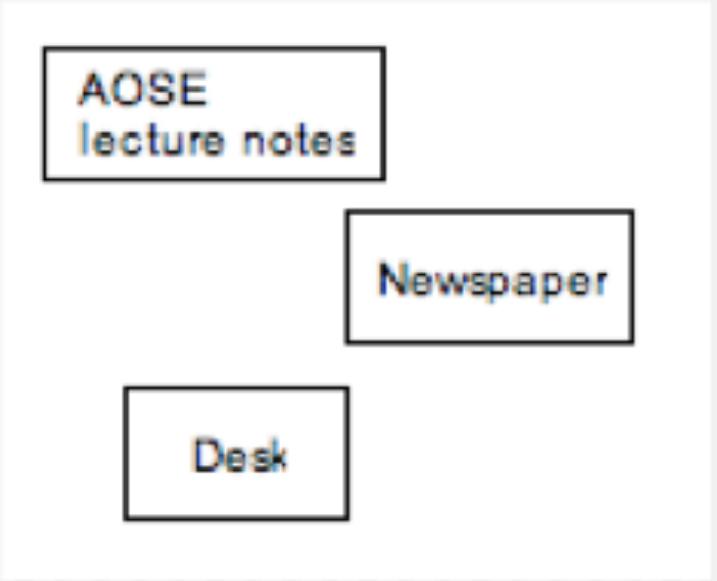
Task -- Plan

- A task is a particular course of action
  - Different levels of abstraction
    - Plan with atomic and/or composite actions
    - Business Process description
- A task is used to satisfy goals
  - always associated to one or more goals



Resource

- A non intentional entity
  - Physical (e.g., printer, car, desk, …)
  - Informative (e.g., AOSE lecture notes, web site, …)
- A resource can be used to
  - achieve goals
  - Perform tasks
- Can be produced by
  - achieving goals
  - executing tasks
- A resource can be shared
  - by agents for
    - Achieving goals and executing tasks



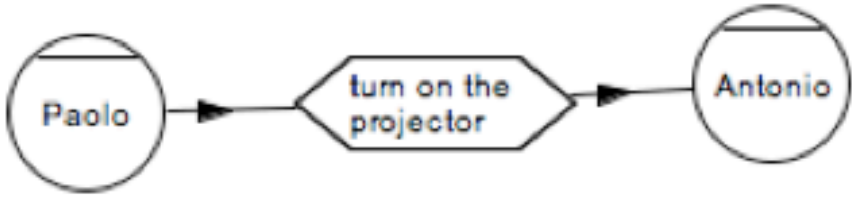
Social dependency

- Always between two actors
  - One actor depends on another to accomplish a goal, execute a task, or deliver a resource
- Content of the dependency
  - Goal
  - Task
  - Resource
- An example of goal dependency
  1. Paolo needs to book and hotel in Malaga
  2. Antonio is able to book the hotel
  3. Paolo ask Antonio to book the Hotel
  4. Antonio agrees to book the hotel for Paolo
  5. Paolo depends on Antonio to book the Hotel in Malaga

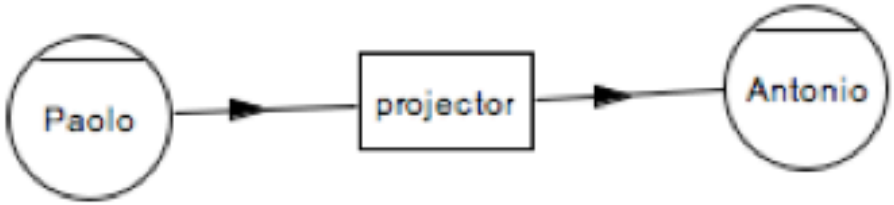
Social dependency: graphical rep.

Goal dependency

Softgoal dependency



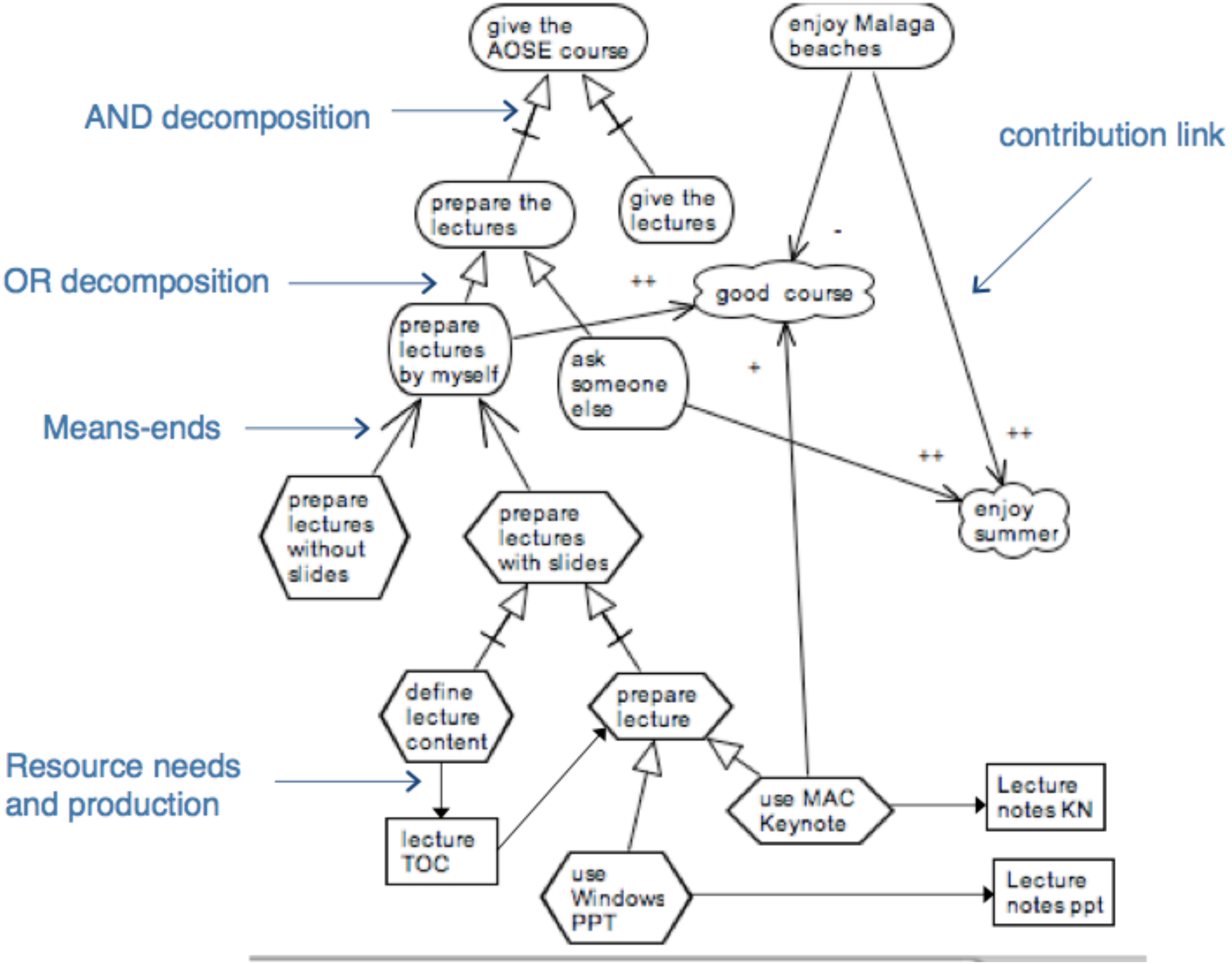
Task dependency



Resource dependency

Tropos relations

- Decomposition
  - AND decomposition
  - OR decomposition
  - A goal can be decomposed in subgoals
  - A task can be decomposed in subtasks
- Means-ends
  - A task (mean) can be used to achieve a goal (end)
- Contribution
  - A goal/task/softgoal can contributes to the satisfaction of a softgoal
    - +,++,-,-- (we will see more on this in the next lectures)
- Resource need
  - A task/goal needs a resource
- Resource production
  - A task/goal produces a resource


give the AOSE course
enjoy Malaga beaches
AND decomposition
contribution link
prepare the lectures
give the lectures
OR decomposition
good course
prepare lectures by myself
ask someone else
Means-ends
prepare lectures without slides
prepare lectures with slides
enjoy summer
define lecture content
prepare lecture
Resource needs and production
use MAC Keynote
Lecture notes KN
lecture TOC
use Windows PPT
Lecture notes ppt

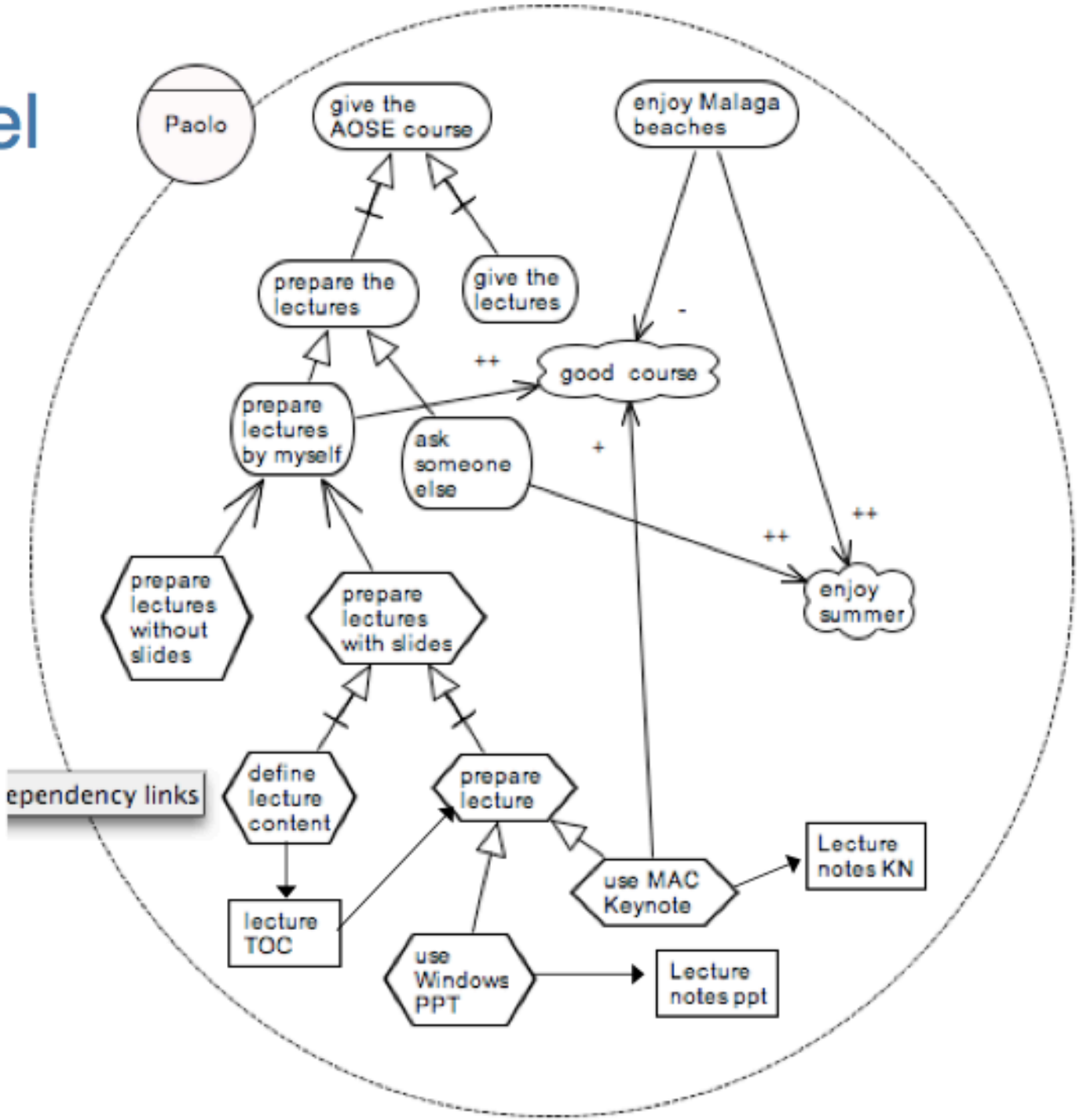
Tropos models

- Three main type of models
  - Actor Model
    - Goal Model
    - Plan Model
    - Capability Model (only for design)
  - Dependency Model
  - Mixed model
    - dependency+actor model

Actor model

- Allows to model a single actor
- Represent the local view
- Process
  1. Goal modeling
     - Main (top) goals and softgoals are identified
     - Goals AND/OR-decomposed
  2. Task modeling
     - For each leaf goal, means tasks are identified
     - Tasks are AND/OR-decomposed
     - Resources needs and production are identified
  3. Contribution analysis
     - For each softgoal, possible incoming and outgoing contributions are identified

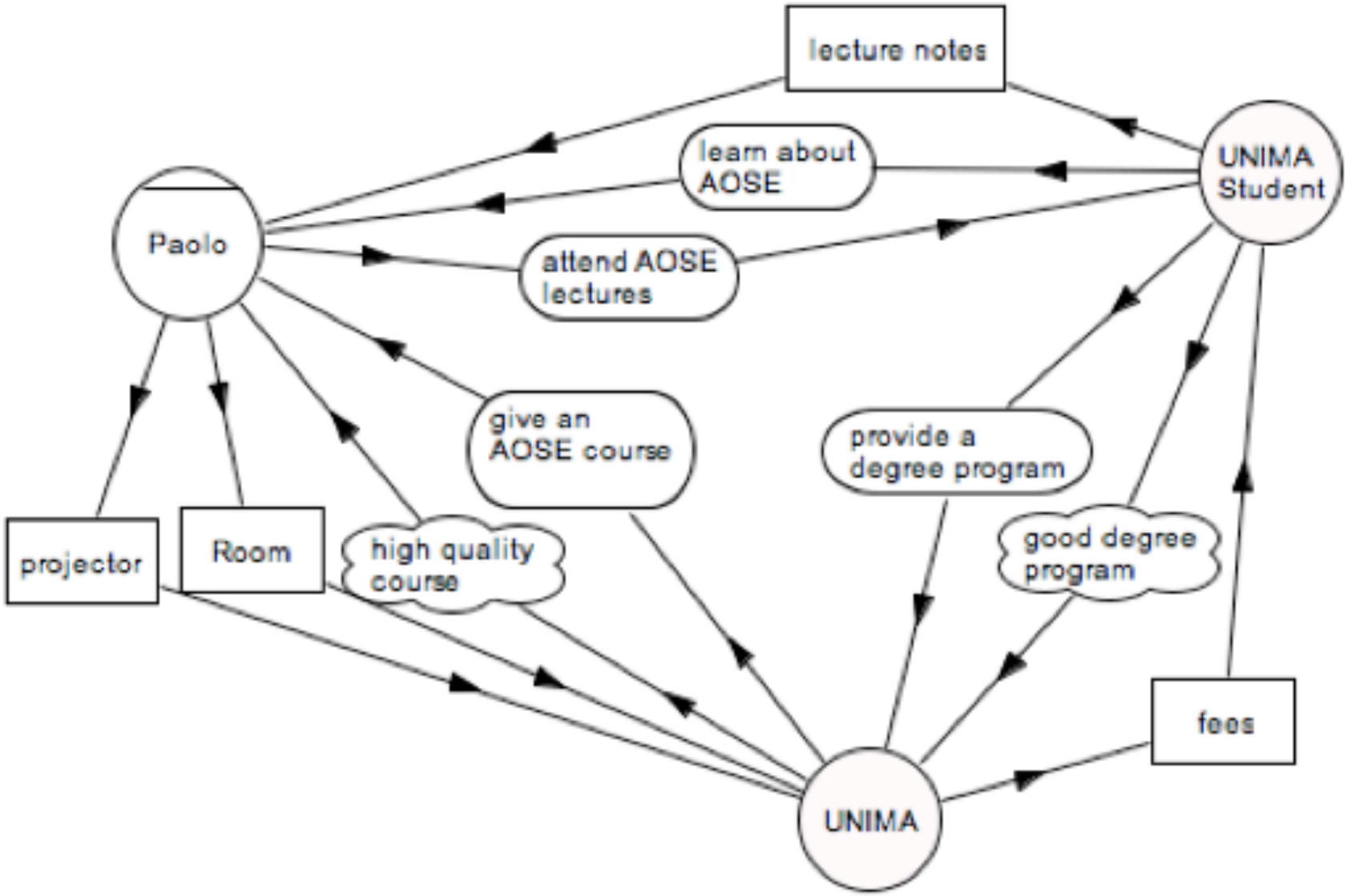
Actor model


Paolo
give the AOSE course
enjoy Malaga beaches
prepare the lectures
give the lectures
good course
prepare lectures by myself
ask someone else
prepare lectures without slides
prepare lectures with slides
enjoy summer
ependency links
define lecture content
prepare lecture
use MAC Keynote
Lecture notes KN
lecture TOC
use Windows PPT
Lecture notes ppt
-
++
+
++
++

Dependency model

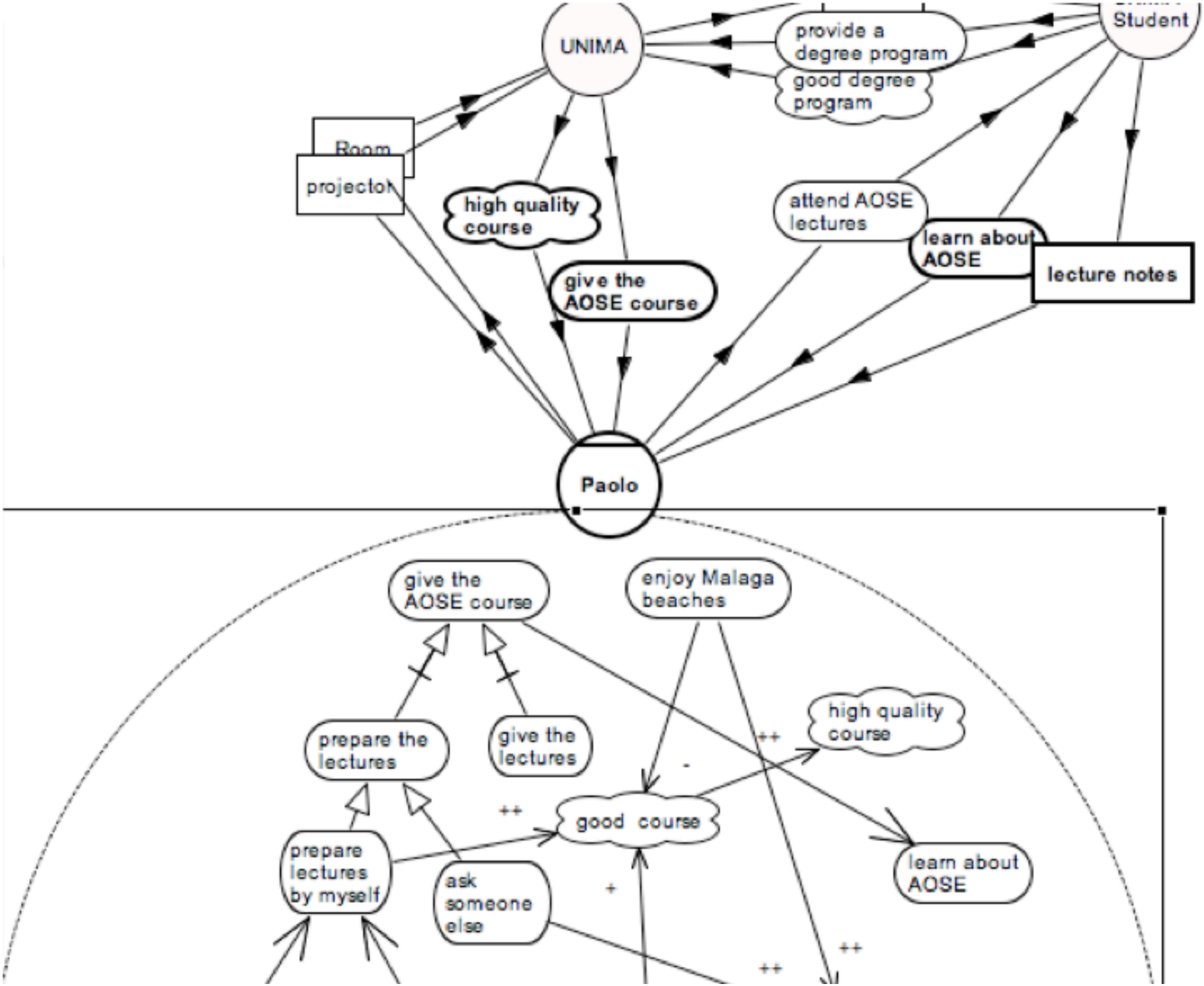
- Allows to build a social (organisational) view
- The focus is on social dependencies
- Process
  1. Actors identification
     - The relevant actors are identified (scope of the model)
  2. Social dependencies are identified
     - Each actor is analyzed identifying incoming and outgoing dependencies (goal/task/resource dependency)
     - Actor models are used for the analysis
  3. Model refinement
     - More actor can be identified (go back to 1)
- The level of abstraction depends on the objective of the model
  - No precise rules are given

Dependency model



Mixed model

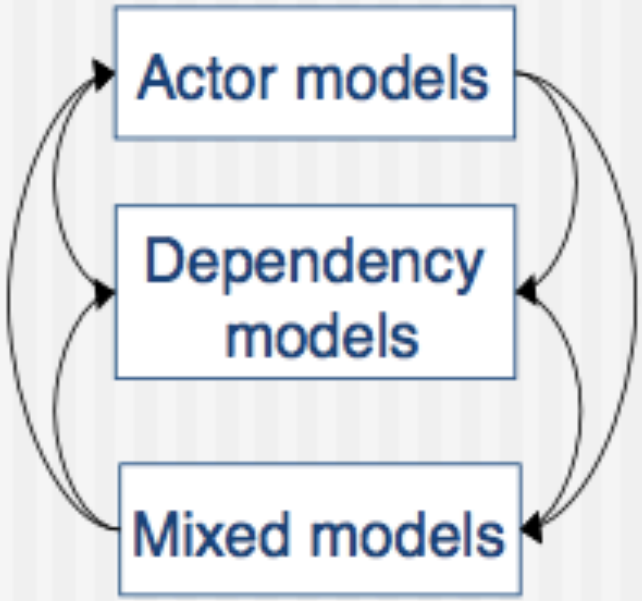
- Is used to have a mixed view between actor models and their dependencies model
  - Often used to build the dependency model or refine the actor model
  - Has to be used very carefully
    - can become very complicated and difficult to read
- Usual process
  1. From an initial actor diagram, (new) dependencies are identified
     - E.g., Actor A has a goal G that is decomposed in two subgoals G1 and G2, goal G2 is delegated to actor B
  2. The new dependencies introduce in the actor model(s) new goals/ tasks
     - Actor B has now goal G2 that can be analyzed in its actor model
  3. Go back to 1. or stop the process


UNIMA
provide a degree program
good degree program
Student
Room
projector
high quality course
attend AOSE lectures
learn about AOSE
lecture notes
give the AOSE course
Paolo
give the AOSE course
enjoy Malaga beaches
high quality course
prepare the lectures
give the lectures
++
-
++
good course
prepare lectures by myself
ask someone else
learn about AOSE
+
++
++

Tropos modeling

- It is up to the analyst/designer to decide
  - the most appropriate level of abstraction (details)
  - When to stop
  - How (and whether) verify the completeness of models
  - Split the modelling problem in subproblems
- The questions to be asked are (strictly ordered)
  1. WHAT
  2. WHY
  3. HOW

  always try to find the WHY

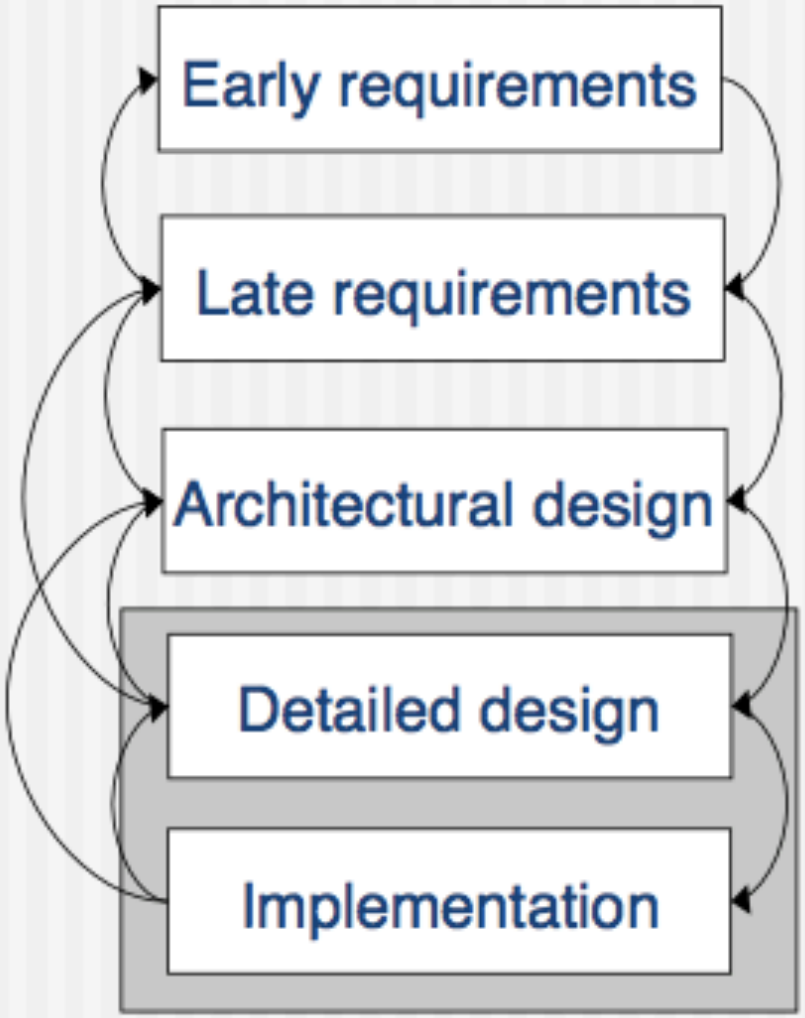


The Tropos phases

- Early requirements
  - The socio and organizational setting is analyzed
  - We are not interested in describing the system-to-be, but just the most relevant actors and their relationships in the domain where the system will operate
- Late requirements
  - The system-to-be is introduced as a new actor of the social domain analyzed in the previous phase
  - The system-to-be is analyzed in terms of Tropos concepts
- Architectural design
  - The Actor system-to-be is designed
  - Subactors are introduced and delegated of goals/task
  - Agents are identified
  - Agent capabilities are identified
- Detailed design
  - Capabilities, protocols, and agent's tasks/plan are specified in detail

The Tropos process

An iterative and transformational approach



Strongly dependent on the (A)PL

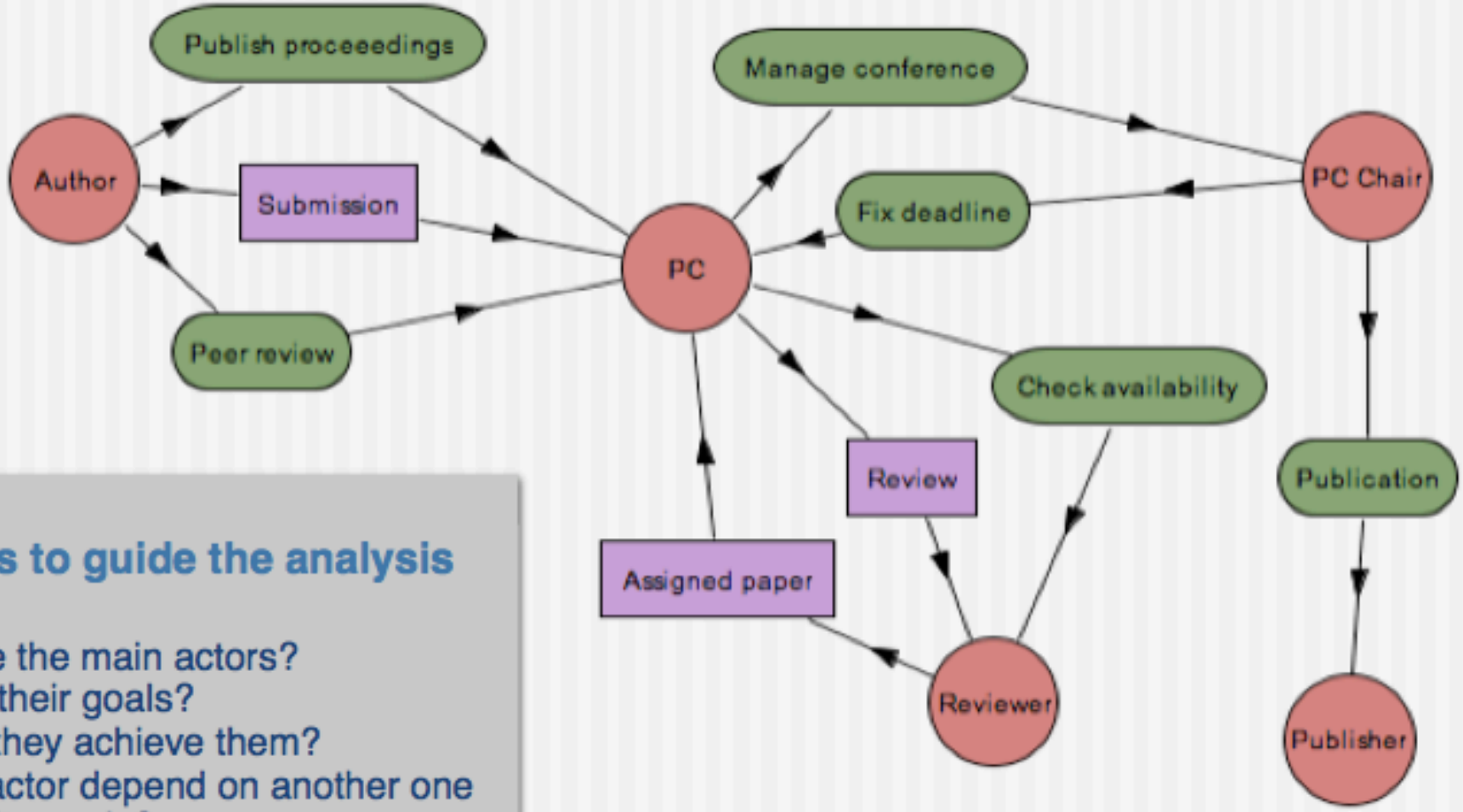
An Example (Thanks people from IRST)

- The Conference Management System (CMS)
  - Pay attention, in the next lecture you have to do it !!!!

- Setting up and running a conference it is a multi-phase process involving several individuals and groups [e.g.Zam01]
- During the submission phase, the authors need to be informed that their paper has been received (a submission number assigned)
- Once the submission deadline has been passed, the program committee (PC) has to manage the review of the papers: contacting potential reviewers; asking them to review a number of papers
- Reviewers can decide to accept or not to review a paper and in case of acceptance they have to produce a review by a given deadline. Reviews are collected and used to decide about paper acceptance by the PC
- Authors will be notified by the PC about paper acceptance/rejection. In case of acceptance they are requested to produce a revised version of the paper (camera-ready).
- Finally, the publisher has to collect these final version and print the proceedings.
- Some organizational rules apply to the actors in the CMS: scenario: e.g. there must be at least three reviewers for each paper; a paper author does not review his own paper; ….

Early requirements

We analyze the environment (i.e. organizational setting) and model it in terms of relevant actors and their respective dependencies



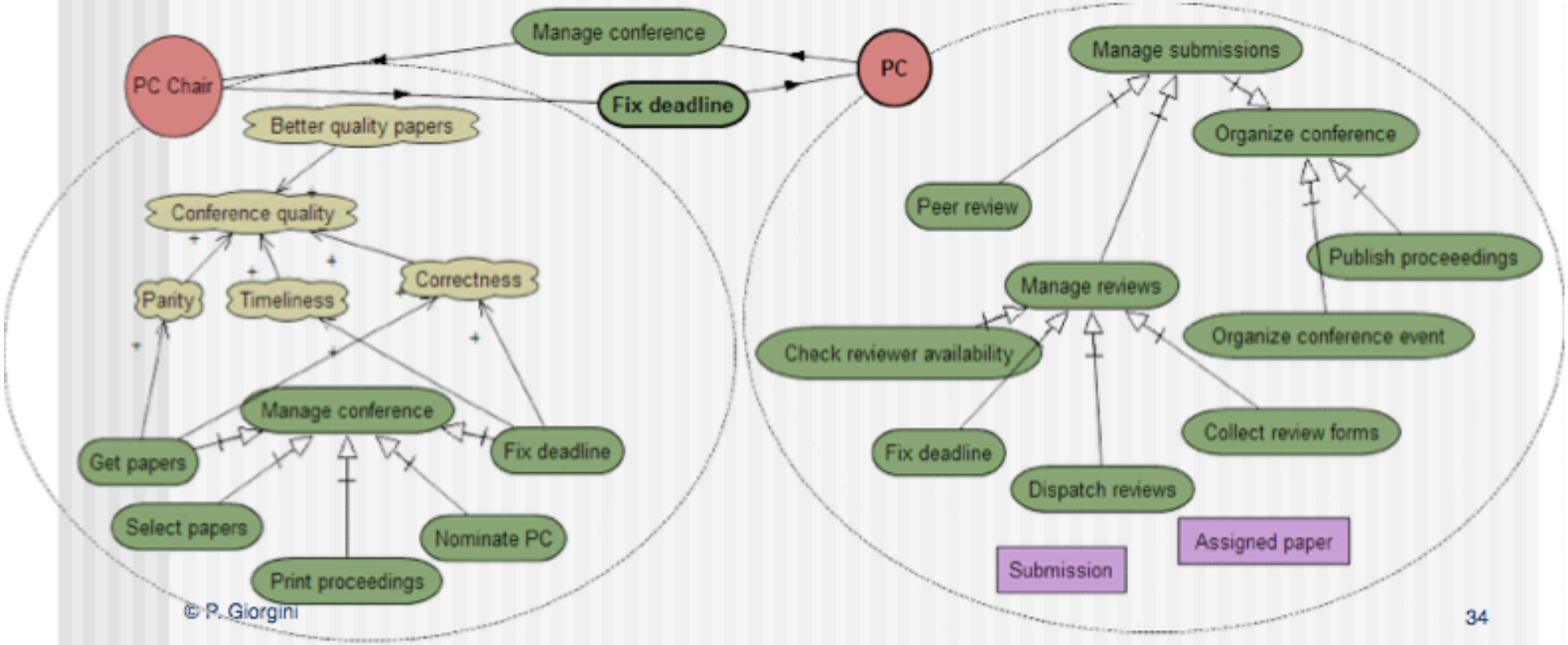
Questions to guide the analysis

- Which are the main actors?
- What are their goals?
- How can they achieve them?
- Does an actor depend on another one to achieve its goals?

Questions to guide the analysis

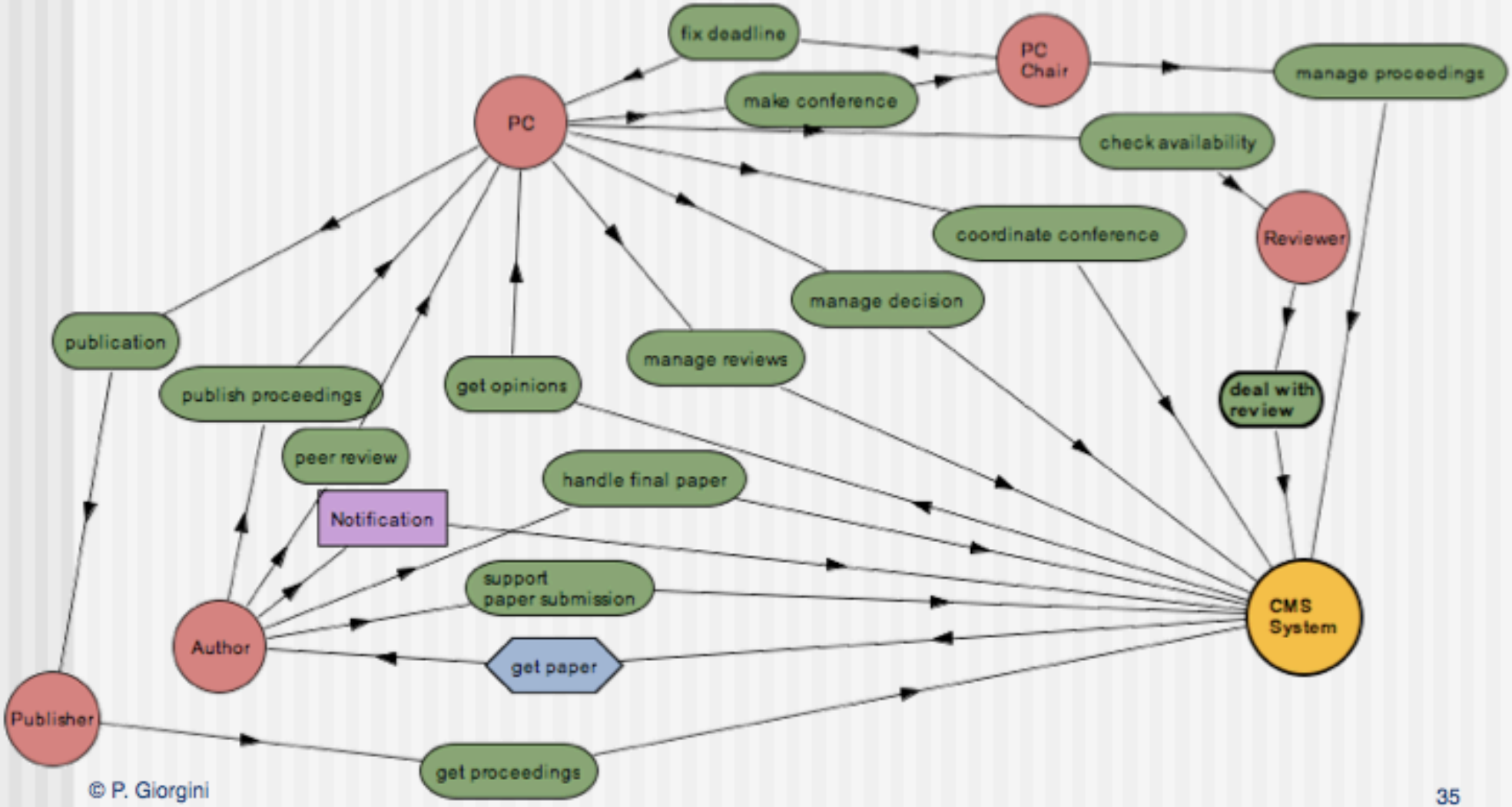
Taking the perspective of each single actor, analyze each goal:

- How can it be decomposed in sub-goals?
- Are there alternative ways to reach a goal? How to evaluate/choose an alternative?
- Are there means (e.g.plan /resources) to achieving them?
- Are there (soft) goals which may prevent the achievement of a goal?
- Are there (soft) goals which may contribute to the achievement of a goal?



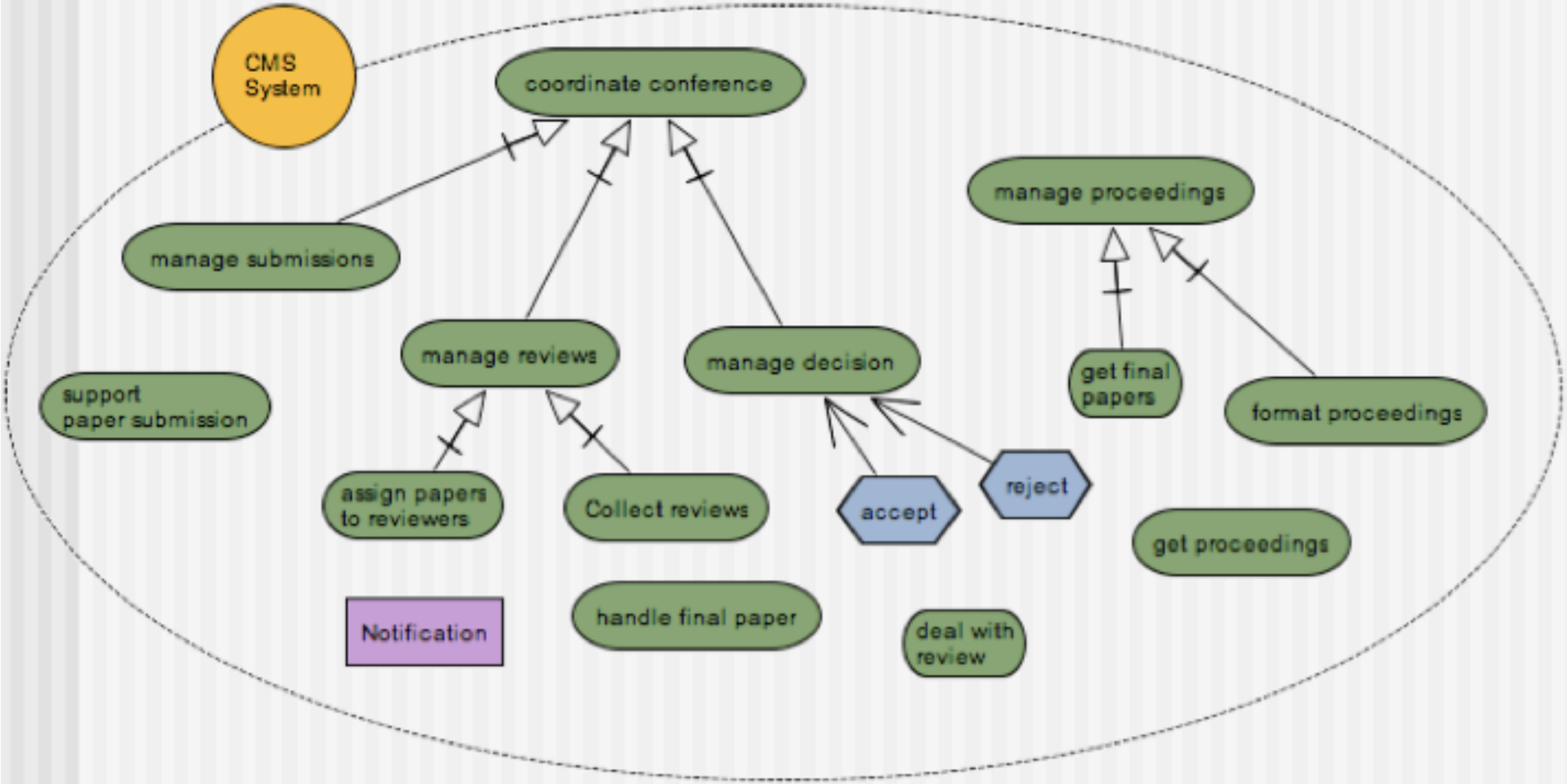
Late requirements

We introduce the system actor and analyze its dependencies with actors in its environment identifying system's functional and non-functional requirements



Late requirements

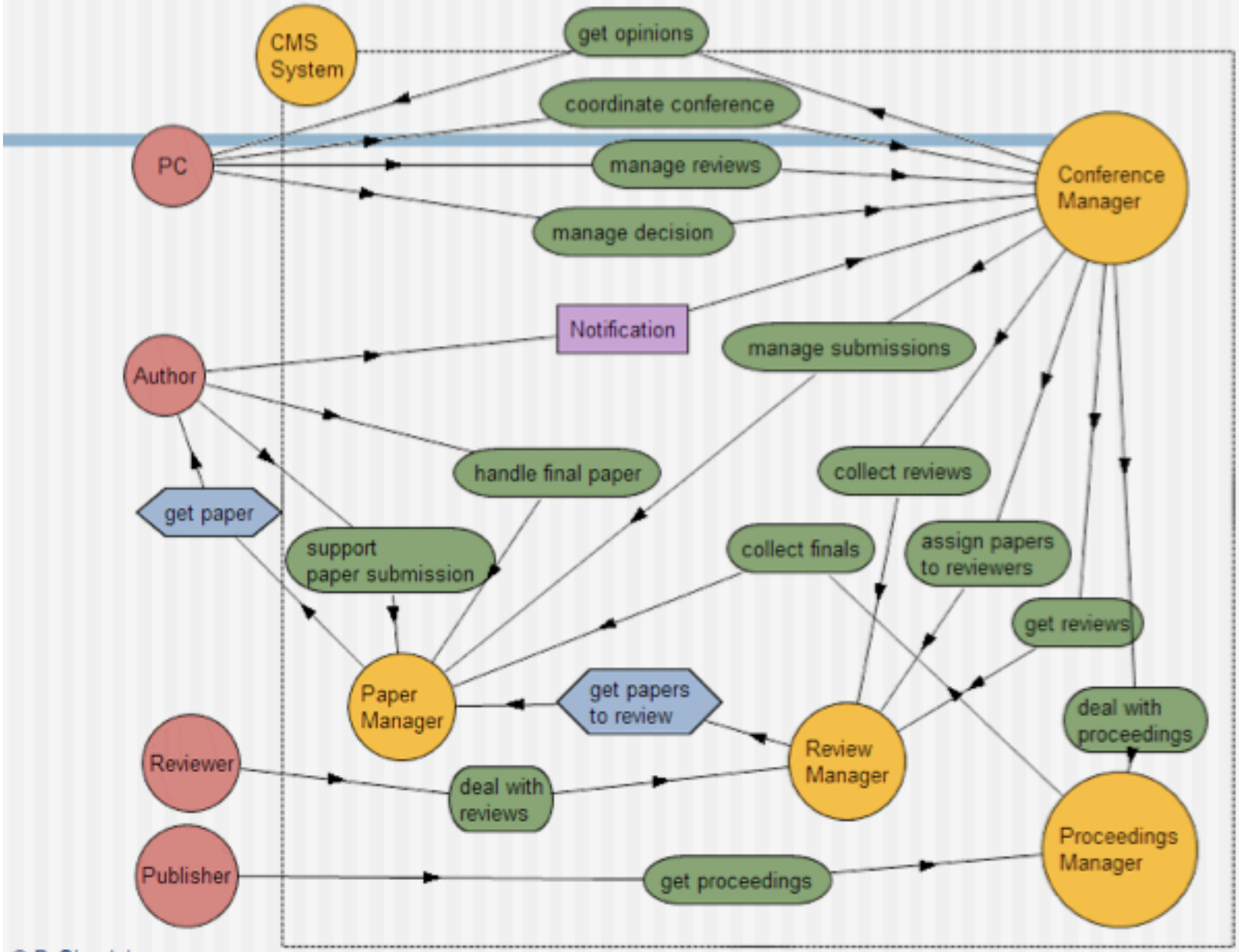
The goals *decomposition*, *means-end* and *contribution* analysis are performed on the system's goals



Architectural design: 3 activities

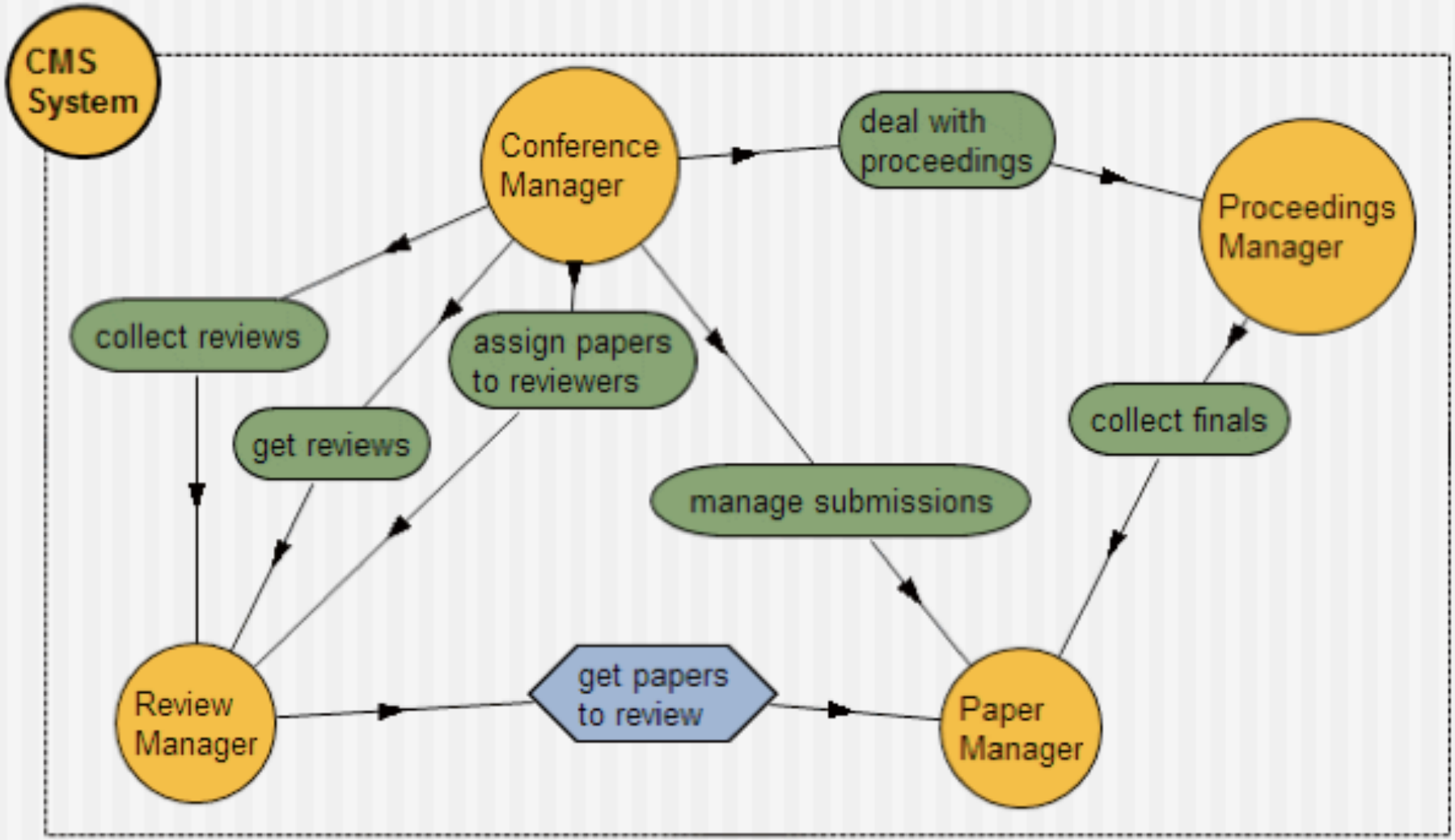
1. Decomposing and refining the system actor diagram
   - definition of the overall architecture (patterns -- if necessary)
   - inclusion of new actors due to delegation of subgoals upon goal analysis of system's goals
   - inclusion of new actors according to the choice of a specific architectural style (design patterns)
   - inclusion of new actors contributing positively to the fulfillment of some Non Functional Requirements
2. Identifying actors' capabilities
3. From *actors* to *agents*

Architectural design (step 1)

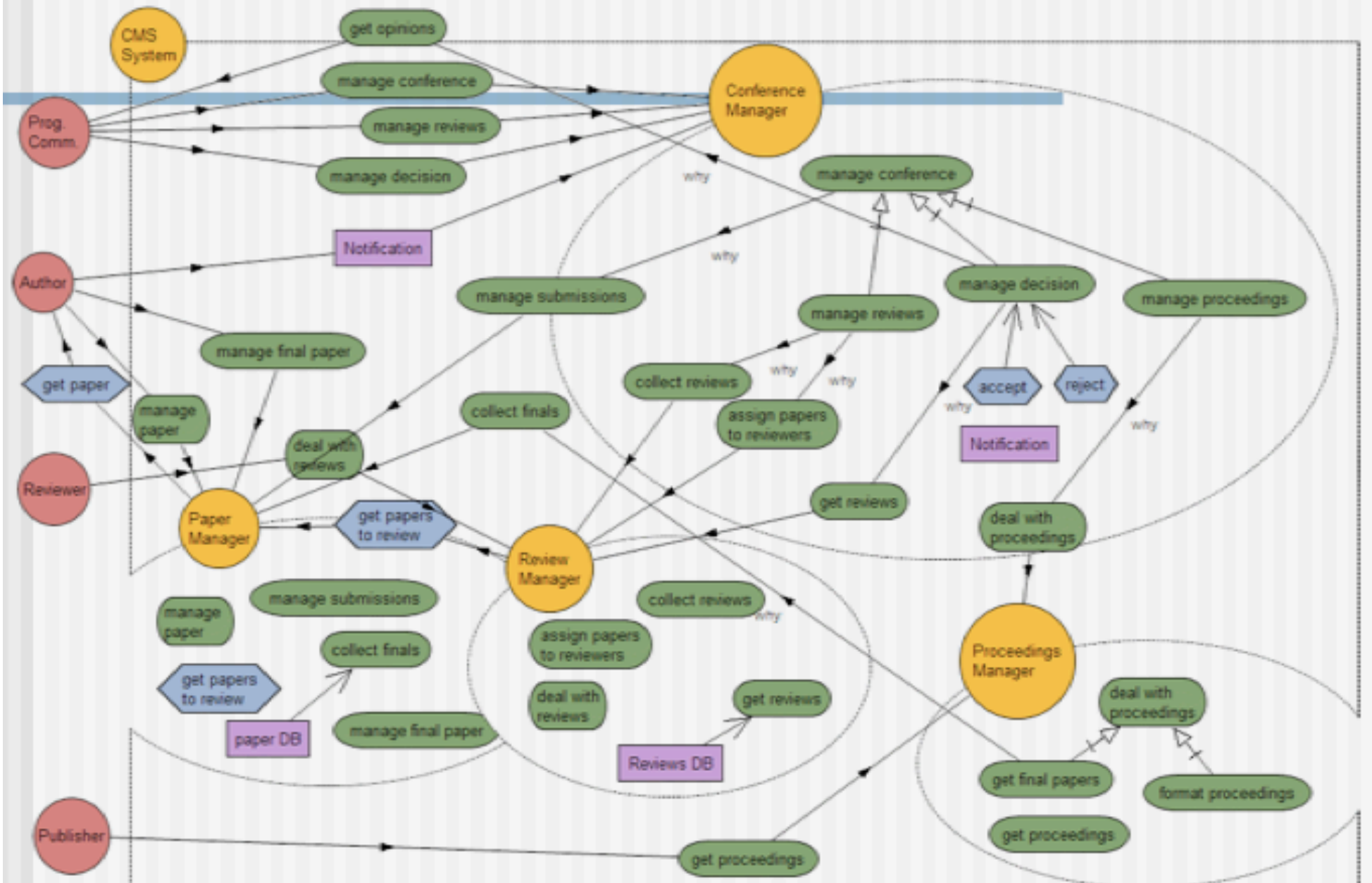


Architectural design (step 1)

Focusing on System Sub-Actor dependencies

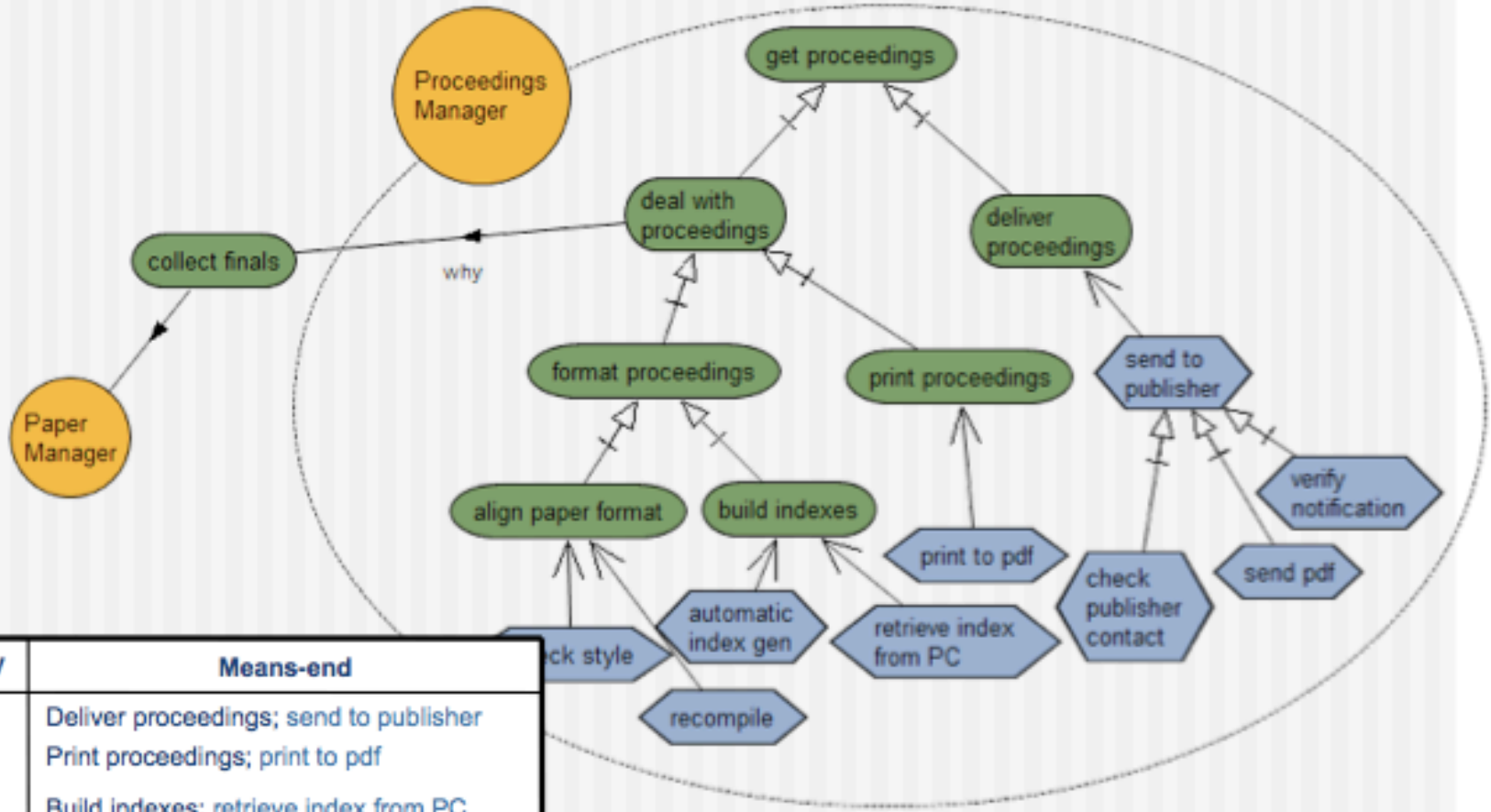


Architectural design (step 1 - with open balloon)



Architectural design (step 1)

Focusing on the the **`Proceedings Manager`** actor (agent)



Agent Name	Capability	Means-end
Proceedings Manager	***Cp1***	Deliver proceedings; send to publisher
	Cp2	Print proceedings; print to pdf
	Cp3	Build indexes; retrieve index from PC
	Cp4	Build indexes; automatic index gen
	Cp5	Align paper format; check style
	Cp6	Align paper format; recompile

Detailed design

- Specification of the agents' micro level taking into account the implementation platform.
- We model:
  - capabilities
    - *capability diagrams* `(currently UML activity diagrams)`
  - plans
    - *plan diagrams* `(currently UML activity diagrams`
  - agents interaction
    - *agent interactions diagrams* `(currently (A)UML sequence diagrams)`
- Automated transformations can be applied
  - we are not going to see this

Detailed design

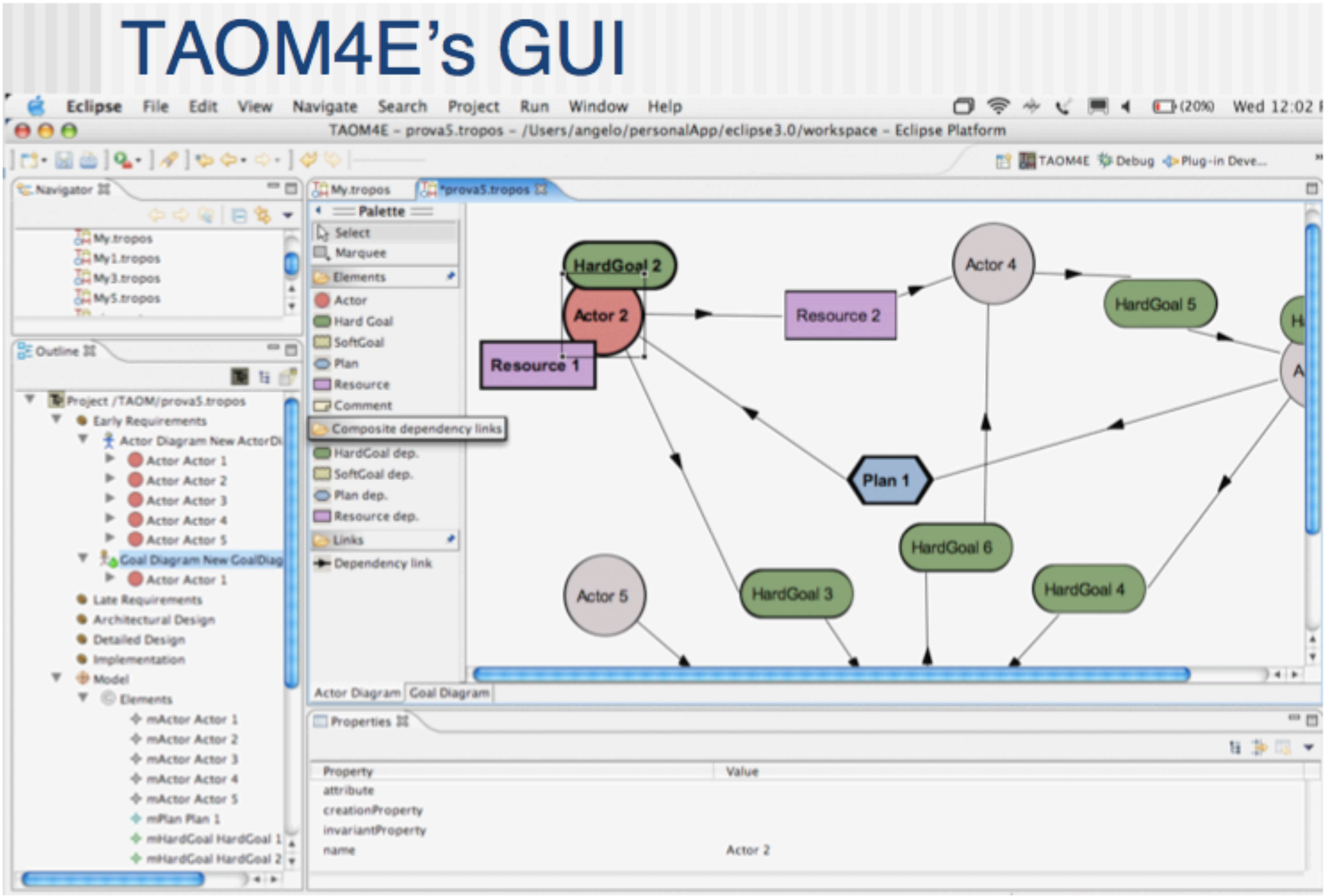
Capability diagram

Agent interaction diagram

TAOM4E

- TAOM4E : Tool for Agent Oriented visual Modeling for the Eclipse platform)
- Model-based approach
- Supports the *Tropos* methodology
- TAOM4E is based on the Eclipse Platform
  - offers a flexible solution to the problem of component integration.
- TAOM4E offers
  - a graphical interface for all the Tropos phases
  - Integration with
  - Automated jadex code generation
  - Access to reasoning tools (risk analysis, security analysis, goal analysis….)

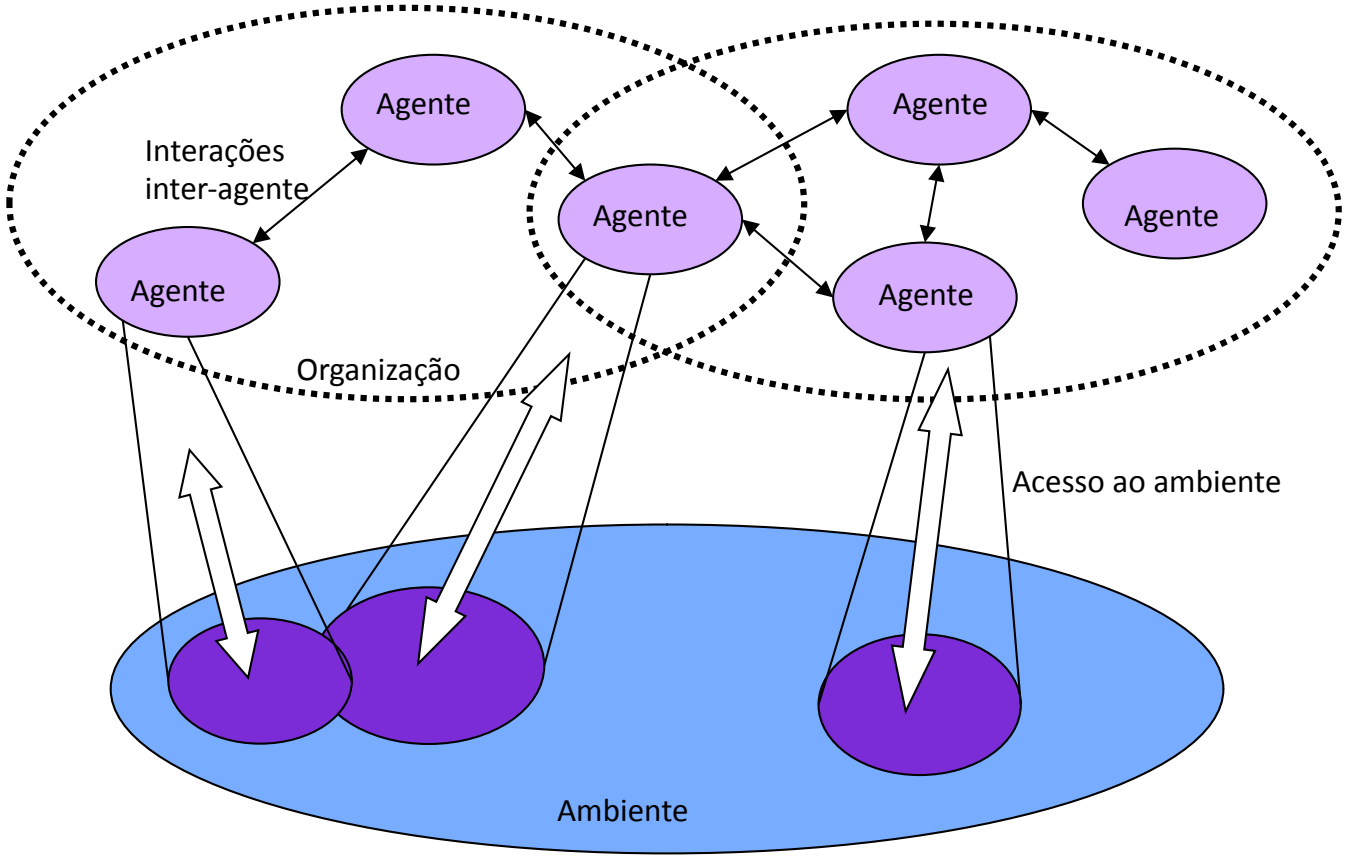
TAOM4E's GUI


Eclipse File Edit View Navigate Search Project Run Window Help
TAOM4E - prova5.tropos - /Users/angelo/personalApp/eclipse3.0/workspace - Eclipse Platform
Navigator
My.tropos
My1.tropos
My3.tropos
My5.tropos
Outline
Project /TAOM/prova5.tropos
Early Requirements
Actor Diagram New ActorDi
Actor Actor 1
Actor Actor 2
Actor Actor 3
Actor Actor 4
Actor Actor 5
Goal Diagram New GoalDiag
Actor Actor 1
Late Requirements
Architectural Design
Detailed Design
Implementation
Model
Elements
mActor Actor 1
mActor Actor 2
mActor Actor 3
mActor Actor 4
mActor Actor 5
mPlan Plan 1
mHardGoal HardGoal 1
mHardGoal HardGoal 2
My.tropos
*prova5.tropos
Palette
Select
Marquee
Elements
Actor
Hard Goal
SoftGoal
Plan
Resource
Comment
Composite dependency links
HardGoal dep.
SoftGoal dep.
Plan dep.
Resource dep.
Links
Dependency link
HardGoal 2
Actor 2
Resource 1
Resource 2
Actor 4
HardGoal 5
Plan 1
HardGoal 6
Actor 5
HardGoal 3
HardGoal 4
Actor Diagram
Goal Diagram
Properties
Property
Value
attribute
creationProperty
invariantProperty
name
Actor 2

METODOLOGIA GAIA

A metodologia GAIA

- Considera SMAs como organizações computacionais

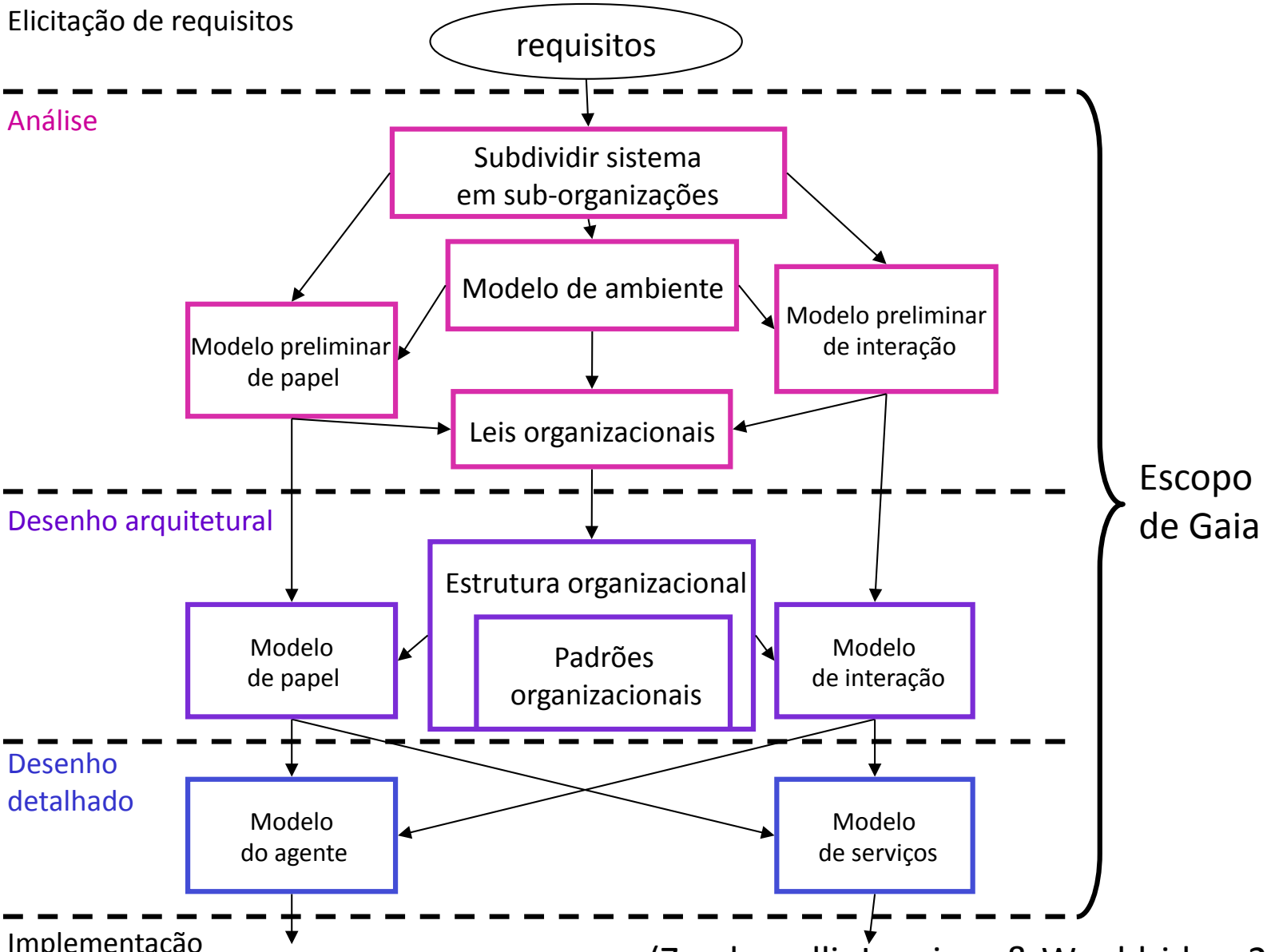


(Zambonelli, Jennings & Wooldridge, 2005)

A metodologia GAIA

- Abstrações básicas:
  - Ambiente
  - Papéis
  - Interações
  - Leis organizacionais e estruturas organizacionais
- não se compromete com notações nem com técnicas de modelagem
- provê suporte às fases de análise e desenho

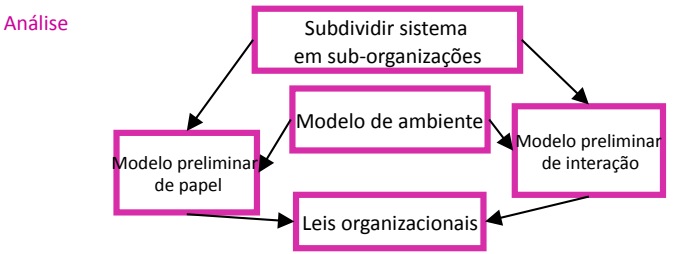
A metodologia GAIA



(Zambonelli, Jennings & Wooldridge, 2005)

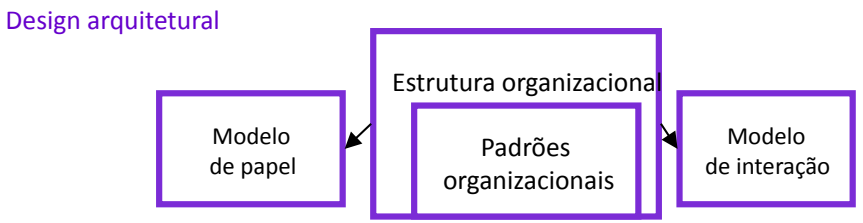
A metodologia GAIA: fase de análise

- Definição das organizações
- Modelo de ambiente
- Modelo preliminar de papéis
  - Permissões
  - Responsabilidades
- Modelo preliminar de interações
  - Protocolos de interação entre agentes
- Leis organizacionais
  - Definem a dinâmica da evolução das organizações (liveness)
  - Definem invariantes globais que devem ser respeitadas (safety)
- Saída:
  - Características funcionais (e algumas não-funcionais) do SMA
  - Características operacionais do ambiente do SMA



A metodologia GAIA: fase de design

- Desenho arquitetural
  - Modelo de papéis
    - Definição das atividades nas quais os papéis estão envolvidos
    - Definição de papéis organizacionais
  - Modelo de interações (ou protocolos)
    - Definição dos protocolos que o SMA necessita
    - Definição de protocolos organizacionais



A metodologia GAIA: fase de design

- Desenho detalhado
  - Definição do modelo do agente
    - Agente é uma entidade de software ativa que desempenha um conjunto de papéis
    - Diagrama ou tabela associando cada agente aos papéis que ele vai desempenhar
  - Definição do modelo de serviços
    - Identificação dos serviços associados a cada papel


tomasz bobrzyński

MODELOS ORGANIZACIONAIS

Modelos Organizacionais

- Adotam visão sociológica e organizacional para modelar SMA

- Oferecem framework conceitual, sintaxe e semântica para especificação da organização subjacente ao SMA
  - Tais especificações podem ser implementadas usando plataformas tradicionais de agentes ou um middleware organizacional

Modelo MOISE+

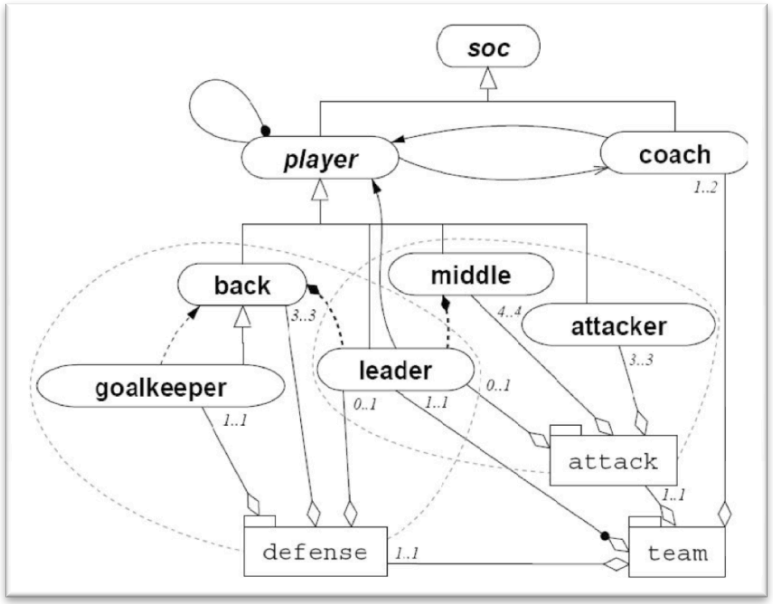
- *M*OISE+ (Model of Organization for Multiagent Systems)

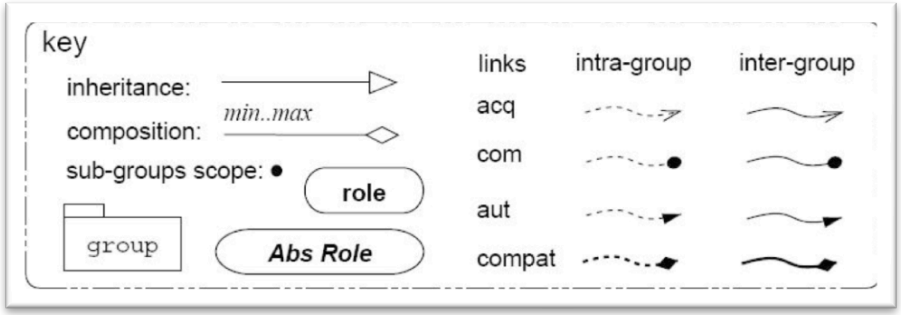
- Considera três dimensões para especificar uma organização subjacente a um SMA (aberto) centrado na organização
  - Especificação estrutural
  - Especificação funcional
  - Especificação deôntica/normativa

(Hubner et al, 2002)

Modelo MOISE+

- Especificação estrutural
  - Papéis
  - Relações entre papéis
  - Grupos

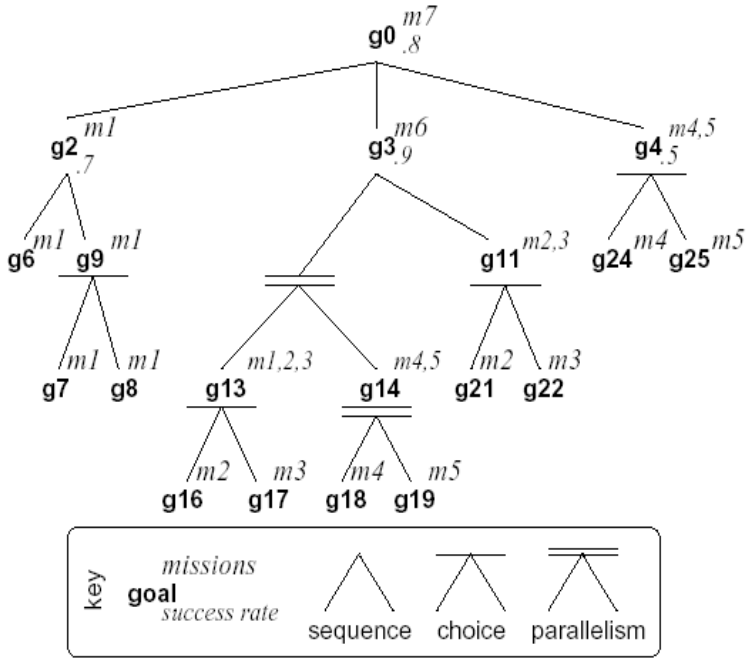




(Hubner et al, 2002)

Modelo MOISE+

- Especificação funcional
  - Metas
  - Missões
  - Planos
- Esquemas sociais



(Hubner et al, 2002)

Modelo MOISE+

- Especificação deôntica (ou normativa)
  - Obrigações
  - Permissões
- Nivel individual
  - Para cada papel em determinada missão
- Relaciona especificações estrutural e funcional
  - Possibilita reorganização dinâmica da organização

(Hubner et al, 2002)

Modelo MOISE+

- Infraestrutura/Plataforma de execução
  - S-MOISE+ (Hubner et al, 2007)
  - JaCaMo (Boissier et al, 2013)

Outros modelos organizacionais

- OperA (Dignum, 2004)
- AGR (Ferber et al, 2004)
- TAEMS (Lesser et al, 2004)
- ISLANDER (Esteva et al, 2002)

LINGUAGENS DE MODELAGEM

Linguagens de modelagem

- AUML
  - Uma proposta de extensão da UML para AOSE
    - FIPA modeling TC
    - Primeiras resultados
      - Protocolos de interação
      - Modelagens de organizações
    - Diagramas de classe
    - Diagramas de sequência

(Bauer, 2002)

Linguagens de modelagem

- AORML
  - Modelagem composta de dois tipos de modelos
    - Externo: um modelo conceitual, voltado para a análise do domínio da aplicação.
      - diagramas de agentes, diagramas de estrutura de interação, diagrama de sequência de interação e diagrama de padrão de interação
    - Interno: corresponde a um modelo de design da aplicação.
      - diagramas de estrutura de reação, diagramas de seqüência de reação e diagramas de padrão de reação

(Wagner, 2003)

Linguagens de modelagem

- MAS-ML
  - Estende UML com a adição de metaclasses ao metamodelo UML
  - Modelos estáticos
    - Diagrama de classe
    - Diagrama de organização
    - Diagrama de papel
  - Modelos dinâmicos
    - Diagrama de sequência
    - Diagrama de atividades

(Silva and Lucena, 2004)
(Silva, Choren & Lucena, 2005)

LINGUAGENS DE PROGRAMAÇÃO

Linguagens de programação AO

- Exemplos:
  - AgentSpeak(L)
    - [Rao, 1996], [Bordini et al, 2005]
  - 3APL
    - [Hindriks et al, 1999], [Dastani, et al, 2005]
  - GOAL
    - [de Boer et al, 2002]
  - IMPACT
    - [Subrahmanian et al, 2000]

PLATAFORMAS DE DESENVOLVIMENTO

Plataformas de apoio ao desenvolvimento de SMA

- JADE
  http://jade.tilab.com/
- JASON
  http://jason.sourceforge.net/Jason/Jason.html
- JACK
  http://www.aosgrp.com/
- JADEX
  http://jadex-agents.informatik.uni-hamburg.de/xwiki/
- JACAMO
  http://jacamo.sourceforge.net/

AOSE: PERSPECTIVAS E FUTURO

Perspectivas

- Workshop dedicado EMAS
  - 2ª edição em 2014 na AAMAS
  - Esforço conjunto de 3 comunidades relacionadas a Engenharia de SMA
    - AOSE; PROMAS; DALT