Tools and Methods based on Task Models

Fabio Paternò *(ISTI*-C.N.R.)

f.paterno@isti.cnr.it - http://giove.cnuce.cnr.it/~fabio/

ISTI-CNR

Structure of the Tutorial

- Introduction to task models, ConcurTaskTrees and CTTE
- Break
- Exercise with CTTE
- Lunch
- From the task model to the UI and viceversa
- Break
- Automatic support for usability evaluation using empirical information and task models

Why Model-based approaches?

- Highlight important information
- Help to manage complexity
- Useful to support methods

Models

- What are the properties of a model?
- A model should
  - Focus on one particular aspect of the real world (here, the UI, the Interactive Application) to be represented and emphasized
  - Raise the abstraction level by promoting appropriate abstractions of the real world (multiple and ample possibilities)
  - Be declarative, rather than procedural

Model-Based Interface Design and Development

- Goals:
  - To provide comprehensive development environments (i.e., design and implementation phases)
  - To improve usability and portability of interfaces
  - To integrate usability analysis with interface development
  - To promote declarative UI knowledge (rather than imperative, procedural)

How can we reach them?

- By using a new paradigm: model-based interface development involving 3 facets:
  - Models: explicitly capture knowledge about UI and Interactive Applications with appropriate abstractions
  - Methods: structure the definition and use of underlying models and related transformations
  - Supporting tools: support the use of the method by providing tools for models and their related transformations.

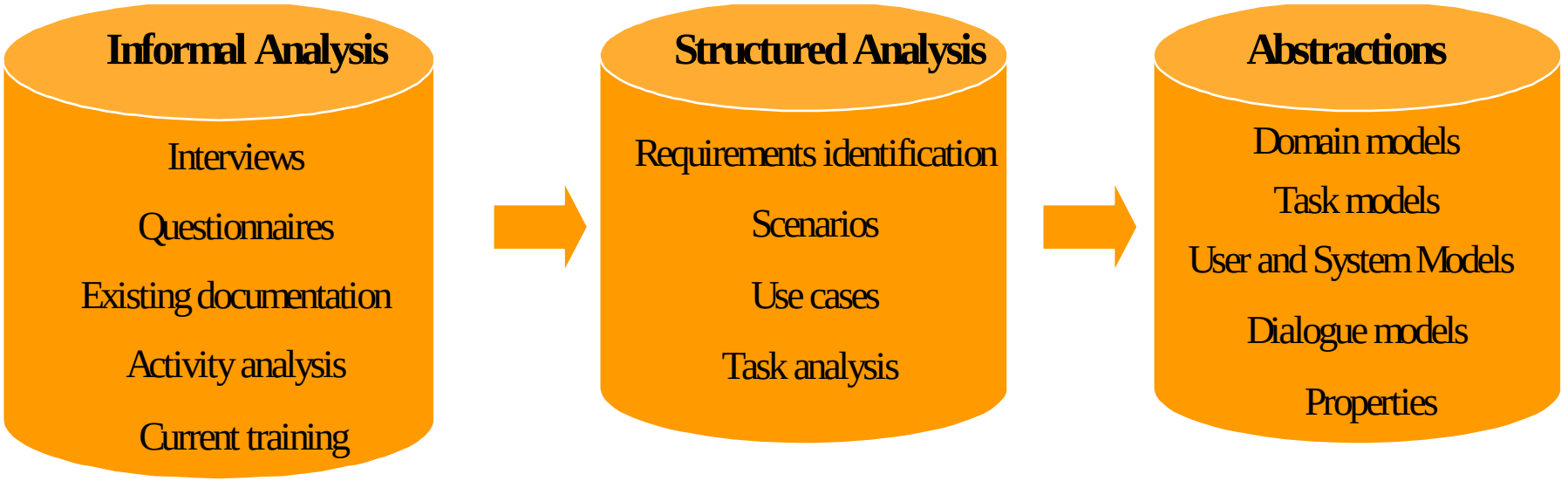
Significant Models in HCI

- Task models
- Cognitive architectures
- User models
- Domain Models
- Context Models
- Presentation Models
- Dialogue models

Definitions

- Task – activity that has to be performed to reach a goal
- Goal
  - desired modification of state
  - Attempt to receive state information
- Each task is associated with one goal
- Each goal is associated with one or multiple tasks
- Multiple abstraction levels - Basic task
- Task Analysis
- Task Models

Moving from informal to structured representations



Scenarios

- Informal,compact description of:
- one (or multiple) specific user
- Who interacts with a specific interface
- To reach a specific goal
- In a specific environment

Example of scenario

Silvia is looking for interesting papers on patterns. She makes a request to the on-line library by giving the name of the topic as one of the parameters of her request, and indicating that she is interested in papers written in English. The order of providing these two parameters is not important. She receives a long list of references. As she is interested in recent contributions she adds a further constraint in the request so that she receives information only on papers published in the last five years. The new list of publications is more manageable. She understands that the works by Gamma are very relevant. She would like to have them grouped so that they are presented together. Thus she makes a new request adding the constraint that the author has to be Gamma. The result is the information that she was looking for. Now she can move to another request for another topic.

Use of Scenarios

- Capture the context where the application is used
- Elicit requirements
- Identify important episodes from the user behaviour
- To provide a context for performing evaluation
- Ability to highlight issues and stimulate discussion while requiring limited effort to develop

Claim analysis (Carroll)

- Some design feature
- + causes (desirable consequences)
- - causes (undesirable consequences)

- Video information
- + is a very rich, intrinsically appealing medium
- - is difficult to search, and must be viewed linearly in real time

Use Cases

- Purpose
- Content
- Plurality
- Structure

Use Cases – Example UML

- Purpose - Requirements
- Content – Consistent prose + diagrams
- Plurality – Multiple scenarios
- Structure – Semi-formal

Task analysis

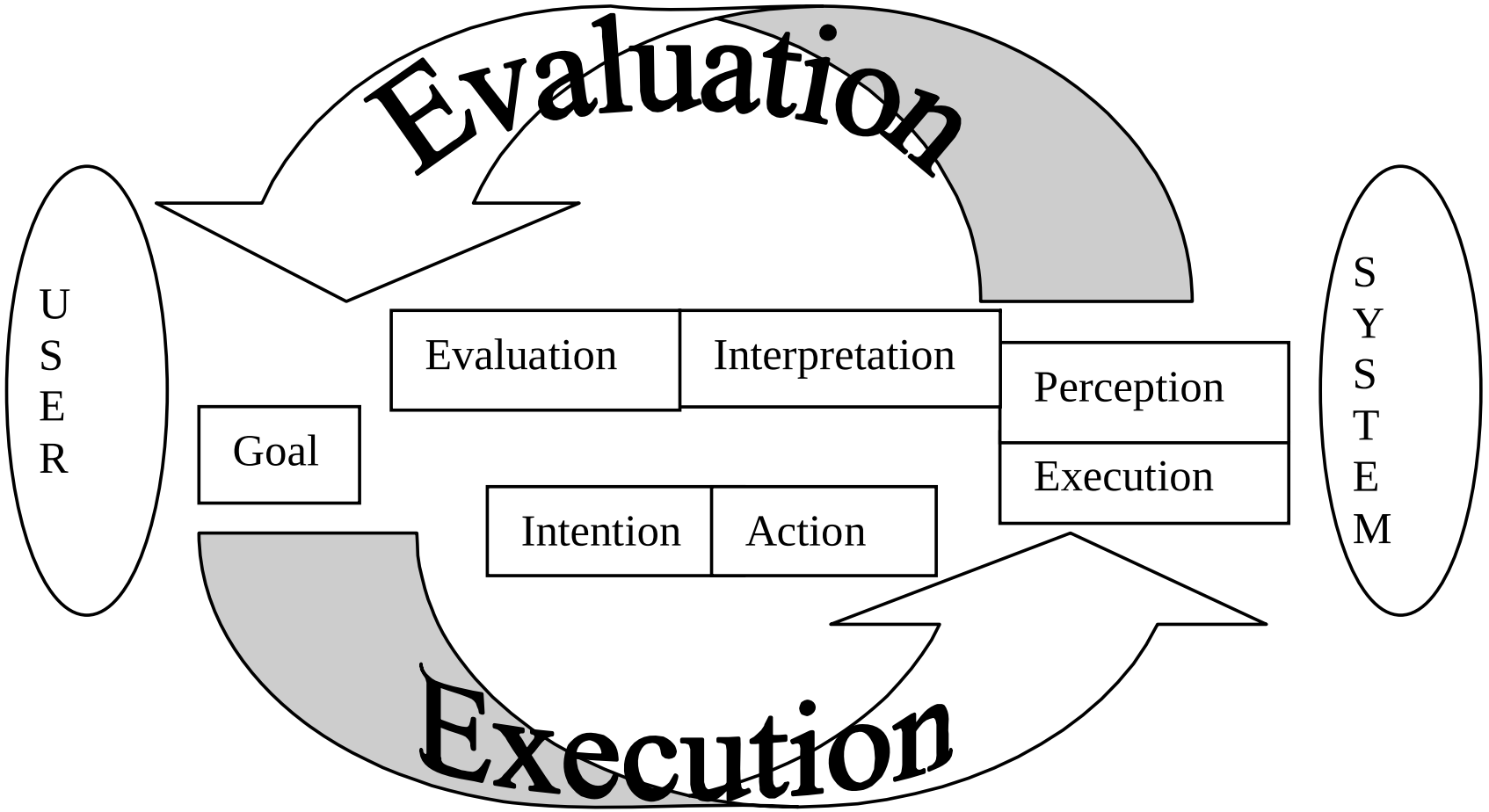
Example: Task analysis of tourists visiting a virtual museum application

- Tourists are characterised by a low average knowledge of the topics considered. Usually they prefer to have guided tours through the rooms of the museum and the town with pictures and information about the works of art. However linear pre-defined tours alone would be too restrictive so some degree of navigational freedom is important. Access to the information is provided with the support of spatial representations: the museum and town maps. This allows users to have immediate information about the locations of the works.
- Tourists want general information on the artistic works, and this information has to be presented clearly and in a limited amount because it has to be interpreted easily. Thus a work will be presented by an image, the title, a short description, the name of the author, the material and technique used for its creation, and when it was made. Additional information about the museum and the town can be provided on request, such as the path to get to the museum from the closest railway station or airport, information (title, data, location) on further exhibitions, and historical information on the town and the museum.

Task Analysis (task list)

- Access to guided tours through the museum and the town
- System enable some degree of navigational freedom
- Access to the information through spatial representations.
- Access to general information on the artistic works
- System presents information clearly and in a limited amount
- System presents a work by an image, the title, a short description, the name of the author, the material and technique used for its creation, and when it was made.
- Additional information about the museum and the town can be provided on request.

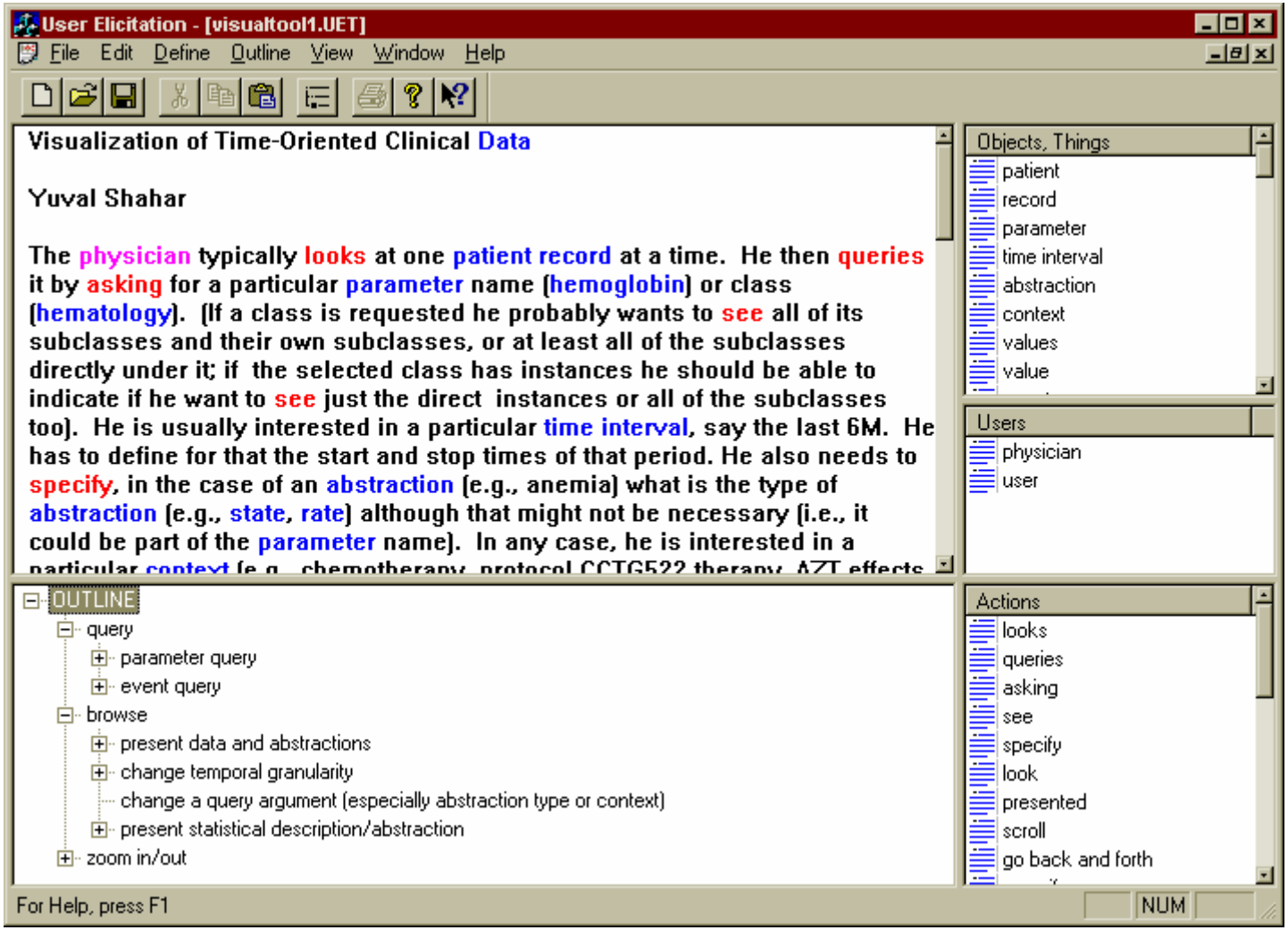
Norman's cycle of interaction



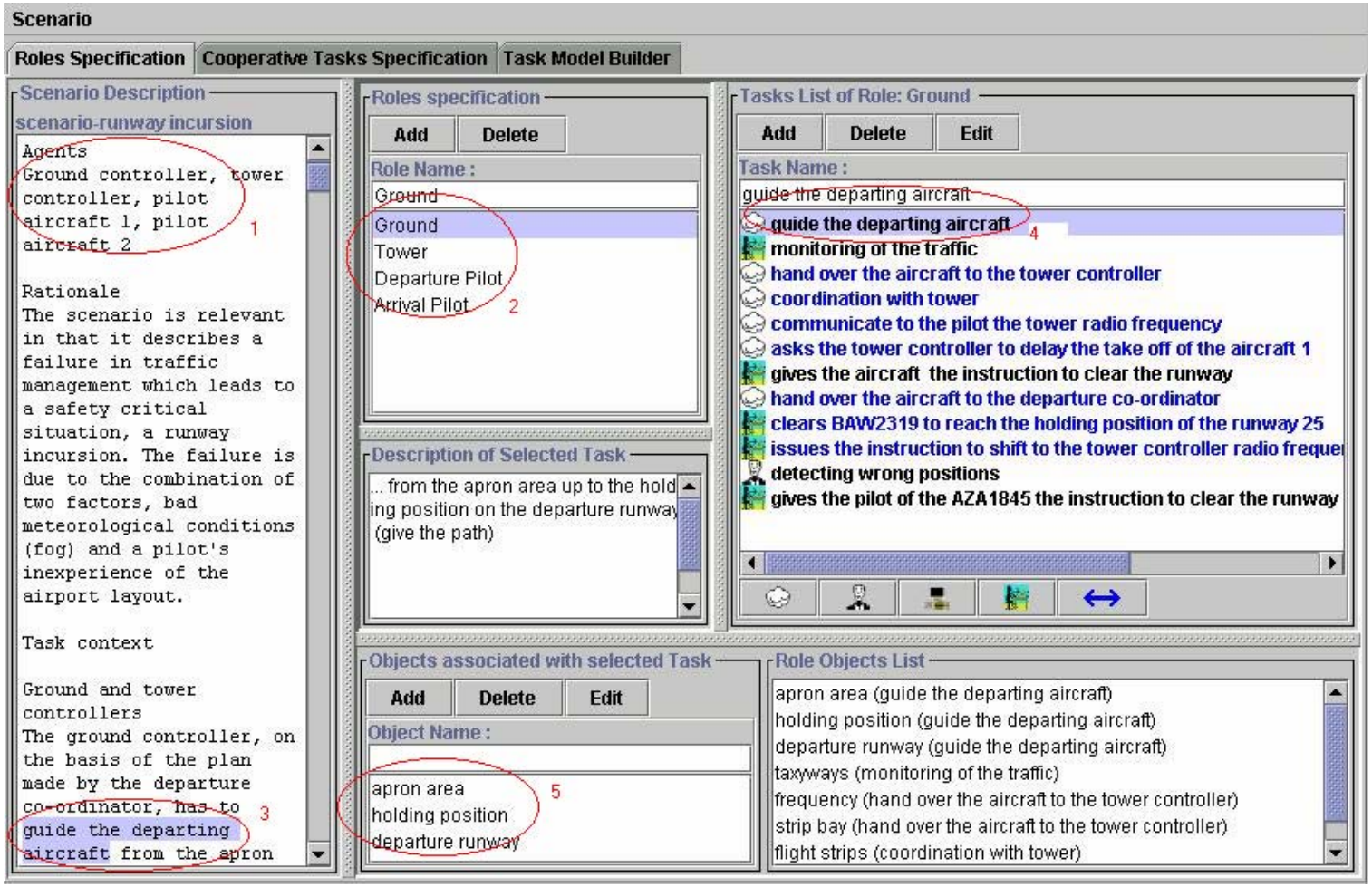
Detectable problems

- Lack of correspondence between user intentions and actions supported by the interface
- Lack of correspondence between representations provided by the system and those expected by the user
- The best interface is that invisible that does not provide obstacles when users perform their tasks

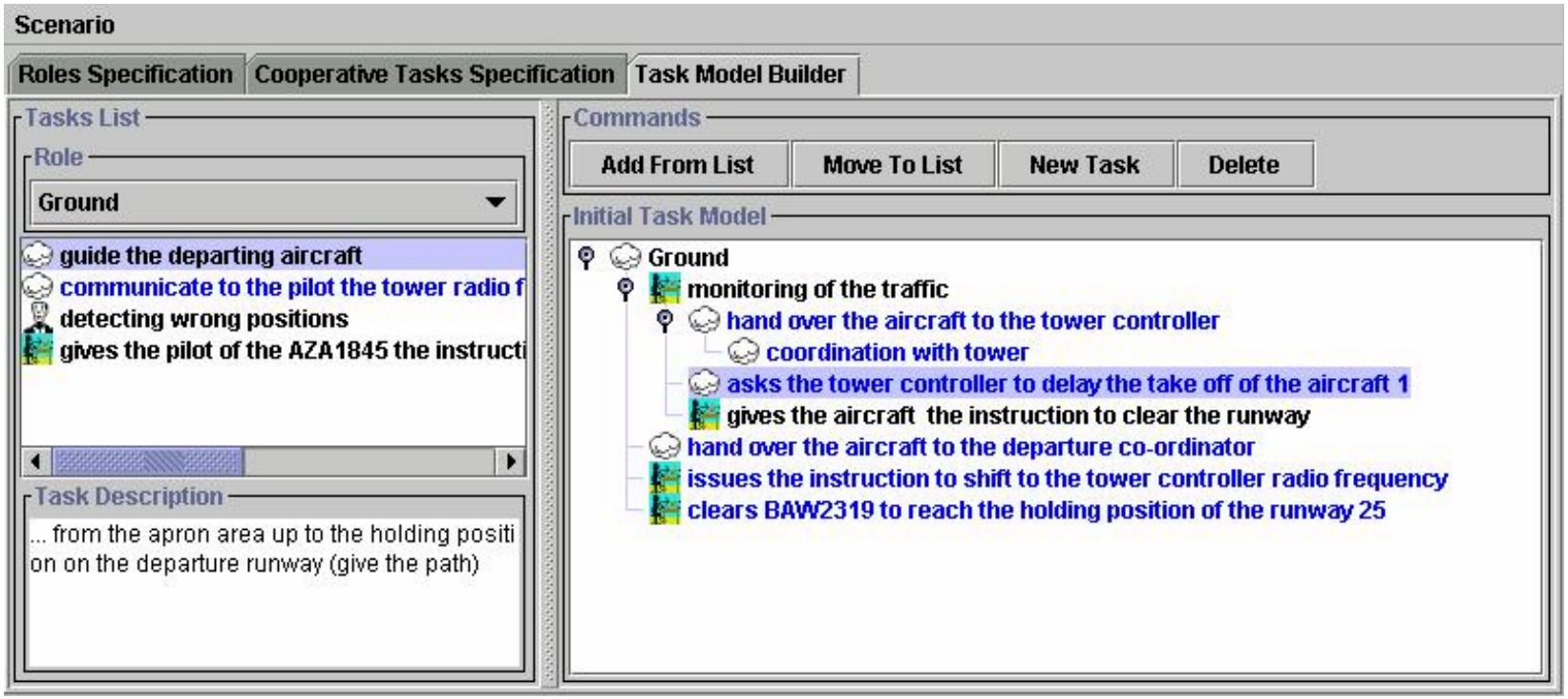
User-Elicitation Example for a Medical-Domain Interface



The environment supporting task identification



Tool support to structure the task model



Engineering task models

- Flexible and expressive notations with precise semantics
- Systematic methods able to indicate how to use information in the task models
- Availability of automatic tools to use such information efficiently

Advantages of Task-based approaches

- For the designer: high-level, structured approaches which allow integration of both functional and interactional aspects
- For the end user: support the generation of more understandable systems

The many possible task models

- (Describe) Existing System
- (Define) Envisioned System
- User

Use of Task Models

- Improve understanding of the application domain
- Record the result of interdisciplinary discussion
- Support effective design
- Support usability evaluation
- Support the user during a session
- Documentation

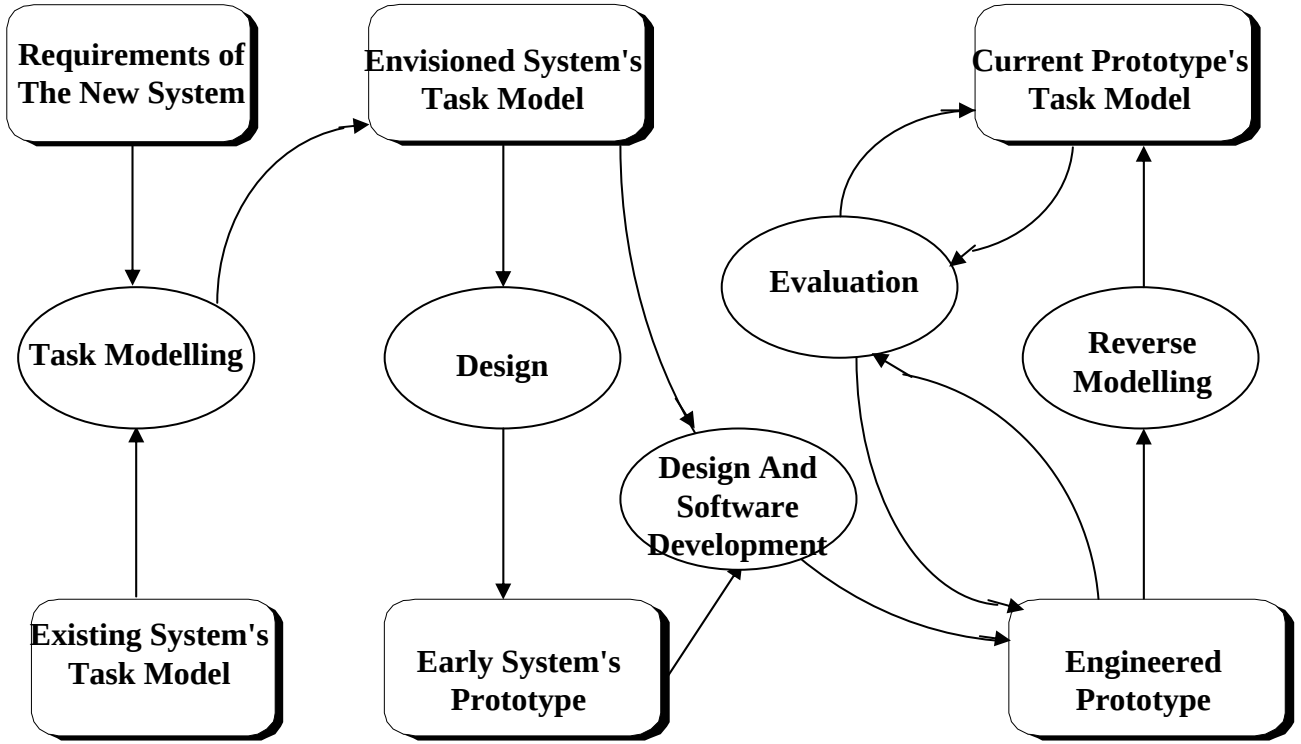
Task Models vs Scenarios

- Scenarios are informal descriptions of a specific use in a specific context
- Task models describe the main possible activities and their relationships
- Scenarios can support task model development
- Task models can support scenarios identification

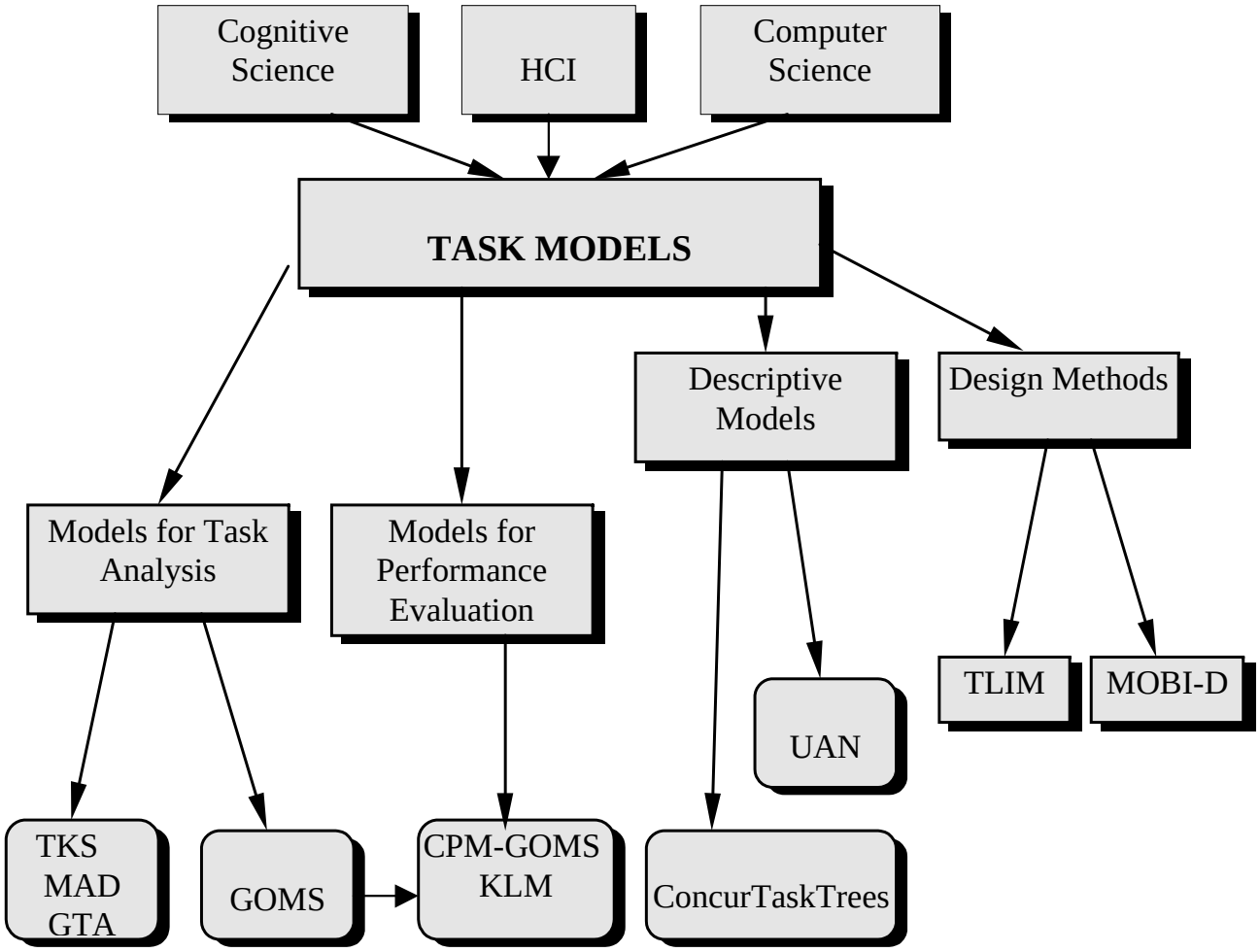
Representations of Task Models

- Hierarchical task analysis
- GOMS family
- UAN
- Different syntax (textual vs graphical)
- Different level of formality
- Different set of operators for task composition

Use of Models in the Life Cycle



Approaches to task models



Tools and Methods based on Task Models

Representations of Task Models

- Hierarchical task analysis
- GOMS family
- UAN
- Different syntax (textual vs graphical)
- Different level of formality
- Different set of operators for task composition

GOMS Example

```
GOAL: EDIT-MANUSCRIPT
        GOAL: EDIT-UNIT-Task repeat until no more unit tasks
                GOAL: ACQUIRE-UNIT-TASK
                        GET-NEXT-PAGE if at end of manuscript
                        GET-NEXT-TASK
                GOAL: EXECUTE-UNIT-TASK
                        GOAL:LOCATE-LINE
                                [select:  USE-QS-METHOD
                                          USE-LF-METHOD]
                        GOAL: MODIFY-TEXT
                                [select:  USE-S-METHOD
                                          USE-M-METHOD]
                                VERIFY-EDIT
```

Limitations of GOMS

- It does not consider user errors
- It does not consider the possibility of interruptions
- It considers only sequential tasks
- It can be inadequate for distributed applications (such as web-based applications)

UAN - User Action Notation

- The user interface is represented by a hierarchy of asynchronous tasks
- user action and system feedback are specified at a low level
- textual notation

Example of UAN specification

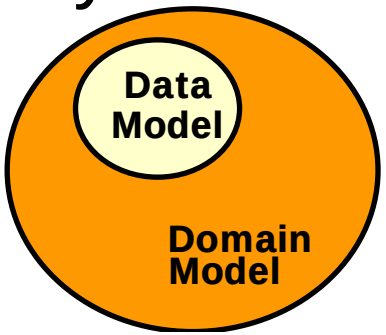
Task: BuildRequest:
*((SelR | ClearR | IconifyR)**
--> SpecField+)

Task: SelApplication

User Action	Interface Feedback	Interface State
~[x,y in AppICON] ∨∧(t<$t_{doubleClick}$) ∨∧)	w'!: w'-! UnMap(PrevAppliMenu) Map(AppMenu) UnMap(AppICON)	CurAppli=App CurMenu=AppMenu

Domain model

- Definition
  - A domain model defines the objects that a user can view, access, and manipulate through a user interface
  - A domain model represents objects of the domain with their relationships
  - Historically, data models have been considered for a while, but they are only a subset of domain models



Domain Model

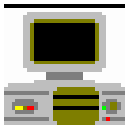
- Domain models extend data models
- Relationships among objects are made explicit and declarative
- Data models are useful only for automatic layout generation
- Domain models are can help to identify effective layout and user interface behavior

ConcurTaskTrees

- Focus on Activities
- Hierarchical Structure
- Graphical Syntax
- Rich set of temporal operators
- Task allocation
- Objects and task attributes

Categories of tasks

interaction



application

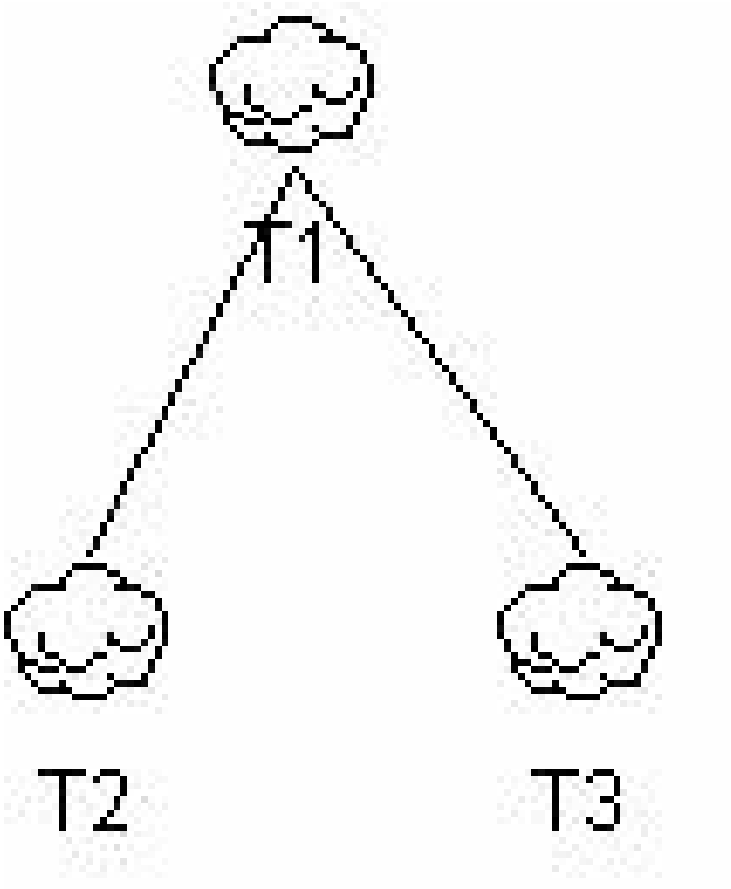
user

abstract

Temporal operators

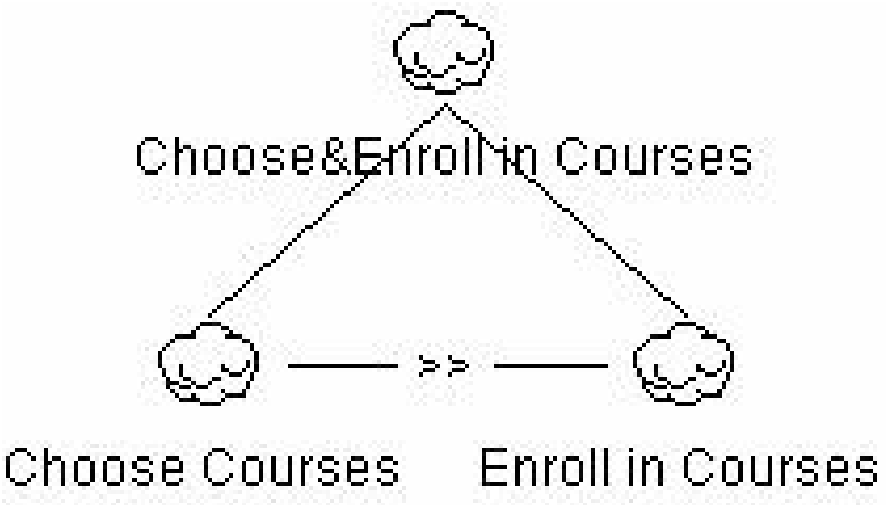
Enabling	**T1 >> T2 or T1 [ ]>> T2**
Disabling	**T1 [> T2**
Interruption	***T1* |> *T2***
Choice	**T1 [ ] T2**
Iteration	**T1* or $T1^{\{n\}}$**
Concurrency	**T1 ||| T2 T1 |[]| T2**
Optionality	**[T]**
Order Independency	**T1 |=| T2**

Hierarchy



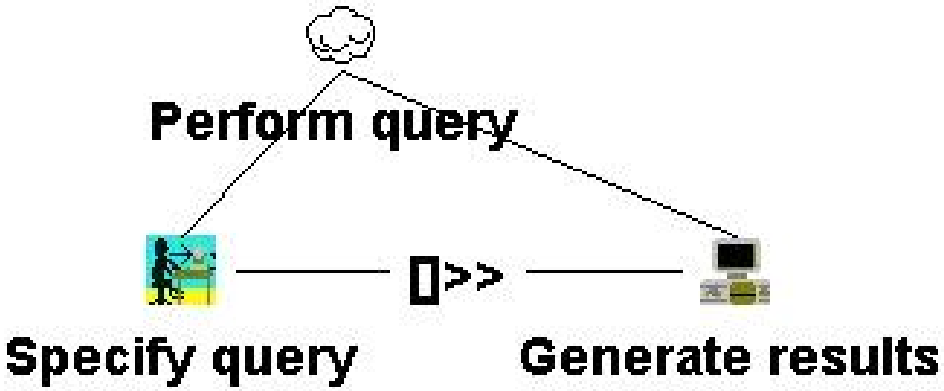
- Tasks at same level represent different options or different tasks that have to be performed
  - Read levels as “In order to do T1, I need to do T2 and T3”, or “In order to do T1, I need to do T2 or T3”

Enabling



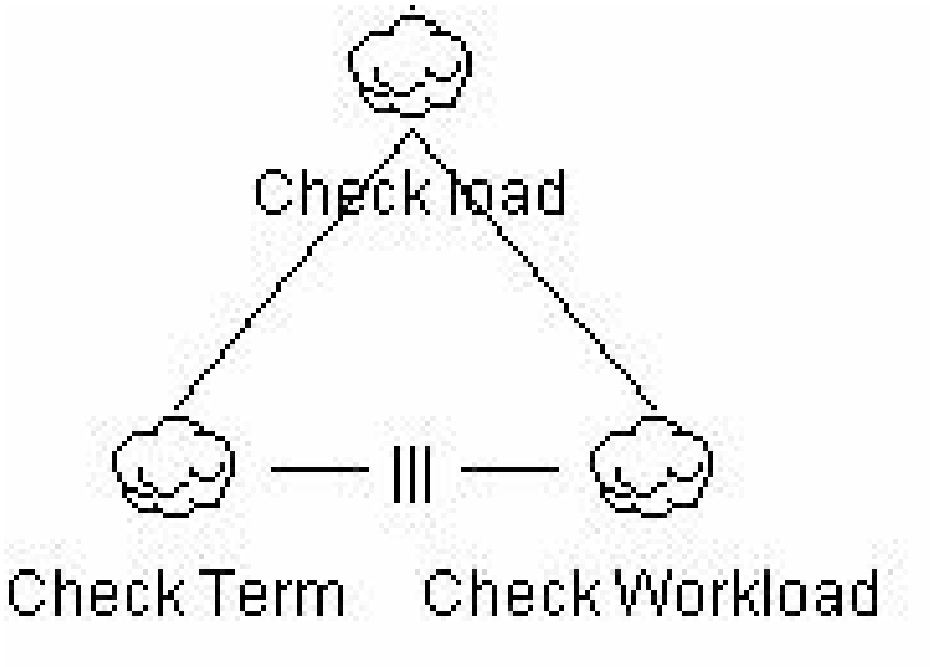
- Specifies second task cannot begin until first task performed
- I.e., I cannot enroll at university before I've chosen which courses to take

Enabling with Information Flow



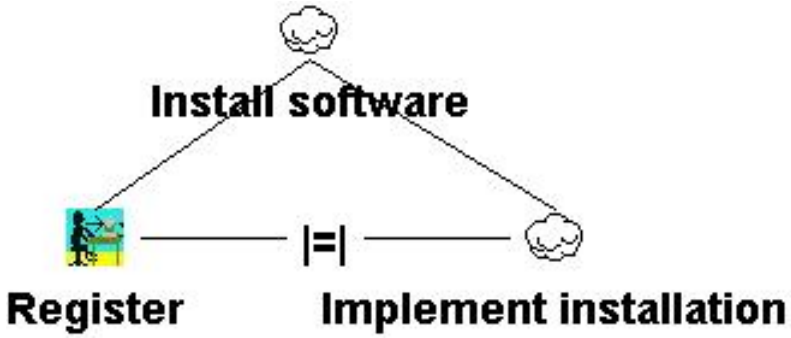
- Specifies second task cannot be performed until first task is performed, and that information produced in first task is used in second

Interleaving



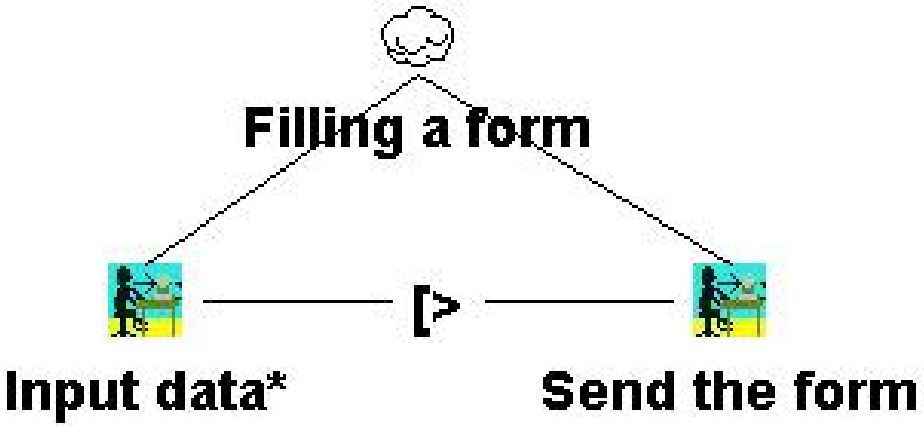
- Tasks can be performed in any order, or at same time
- In order to check the load of a set of courses, I need to consider what terms they fall in and to consider how much work each course represents
- I can do this in any order

Order Independence



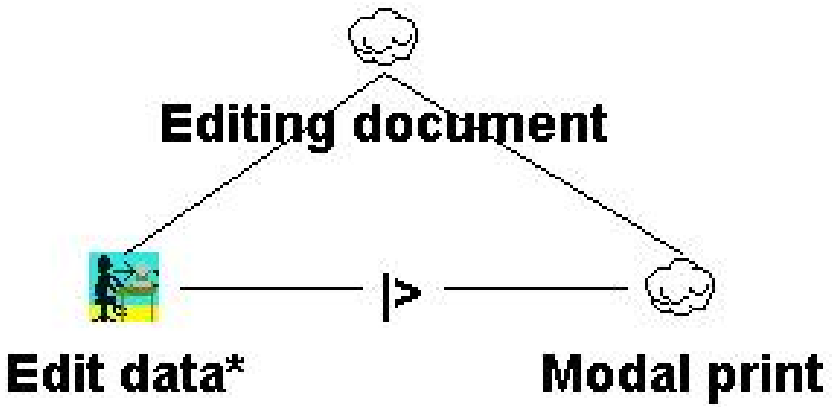
- Tasks can be performed in any order, but when one starts then it has to finish before the other one can start
- When I install new software I can start by either registering or implementing the installation but if I start one task I have to finish it before moving to the other one

Disabling



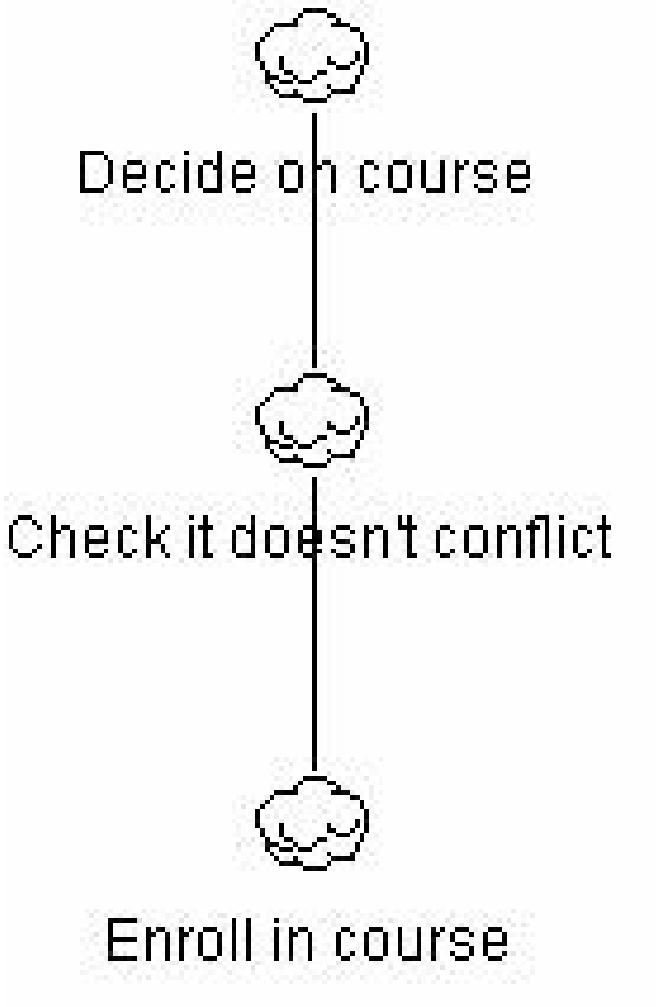
- The first task (usually an iterative task) is completely interrupted by the second task

Suspend-resume



- First task can be interrupted by the second one
- When the second terminates then the first one can be reactivated from the state reached before

Common Errors



- Hierarchy does NOT represent sequence
  - "In order to check a course doesn't conflict, I have to enroll in the course"

Task and attributes

Interaction tasks

Selection

Edit

Control

…

Application task

Overview

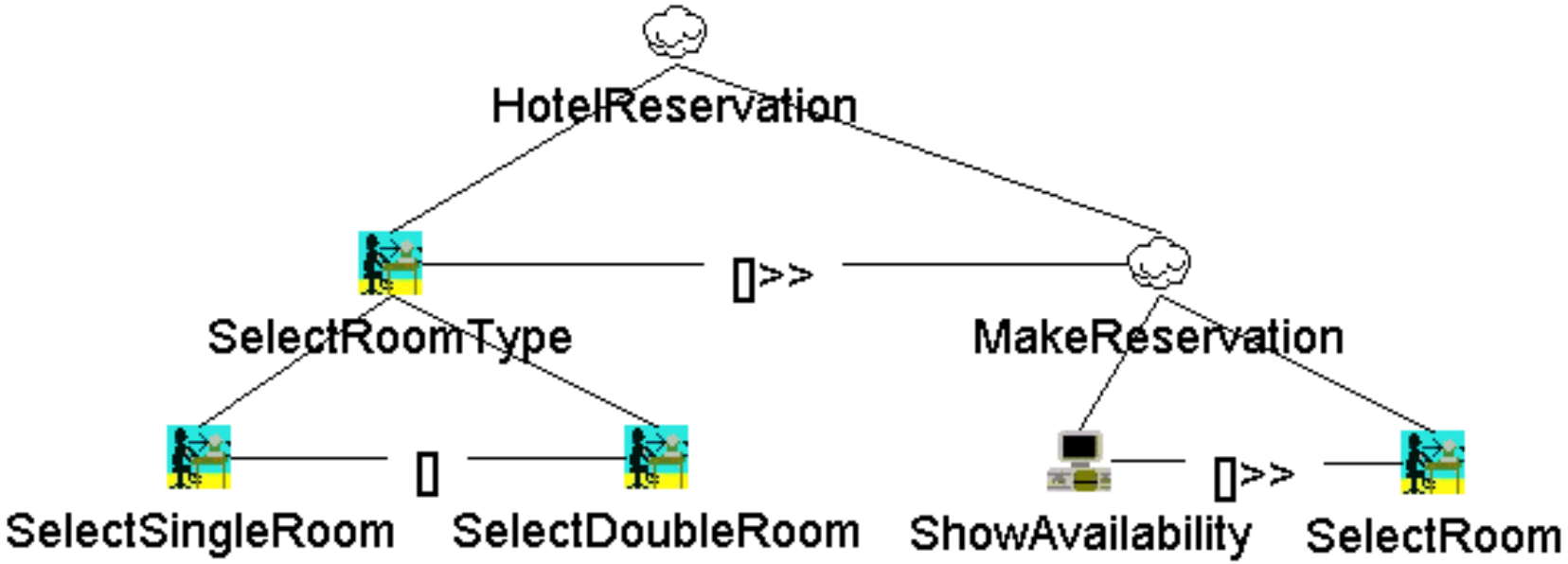
Feedback

Generating alerts

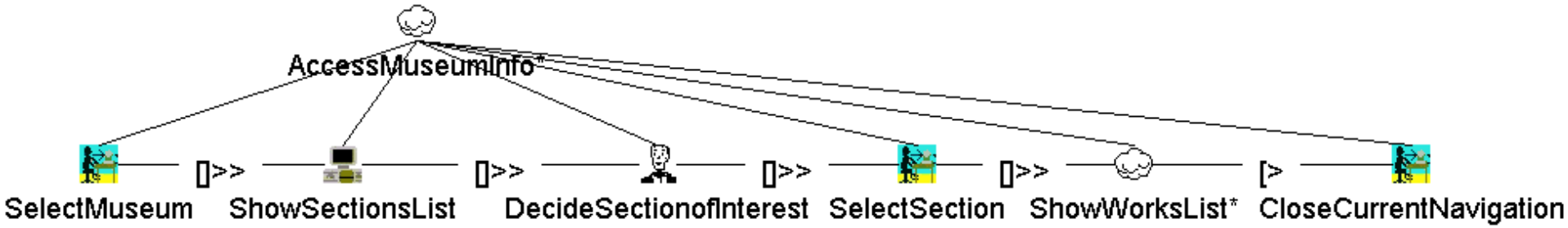
Grouping

…

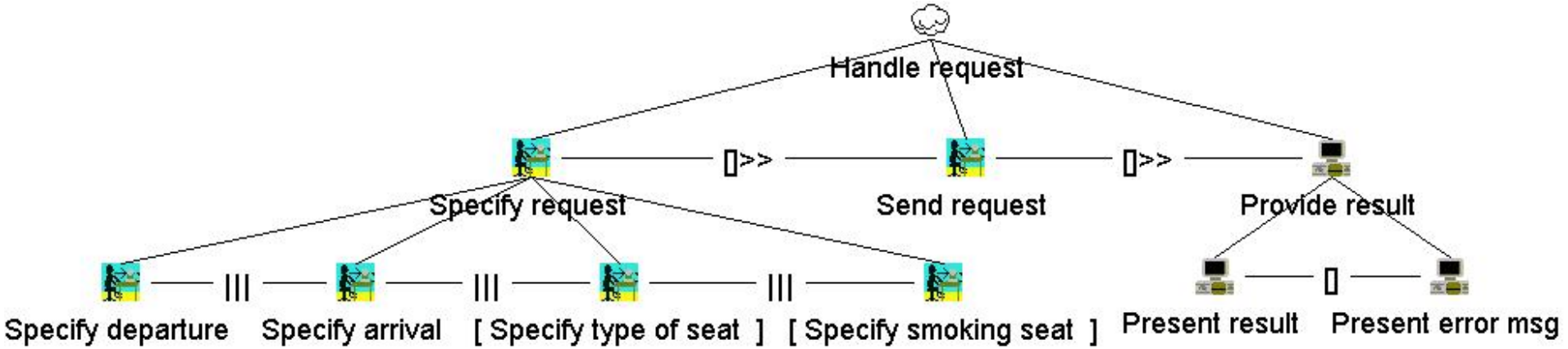
Inheritance of relationships



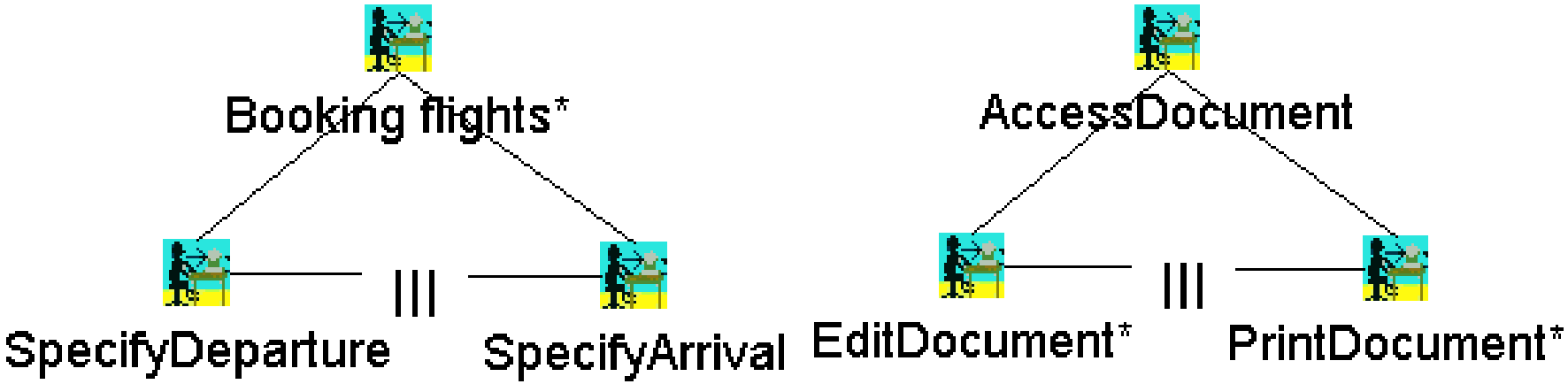
Relationships task/subtasks



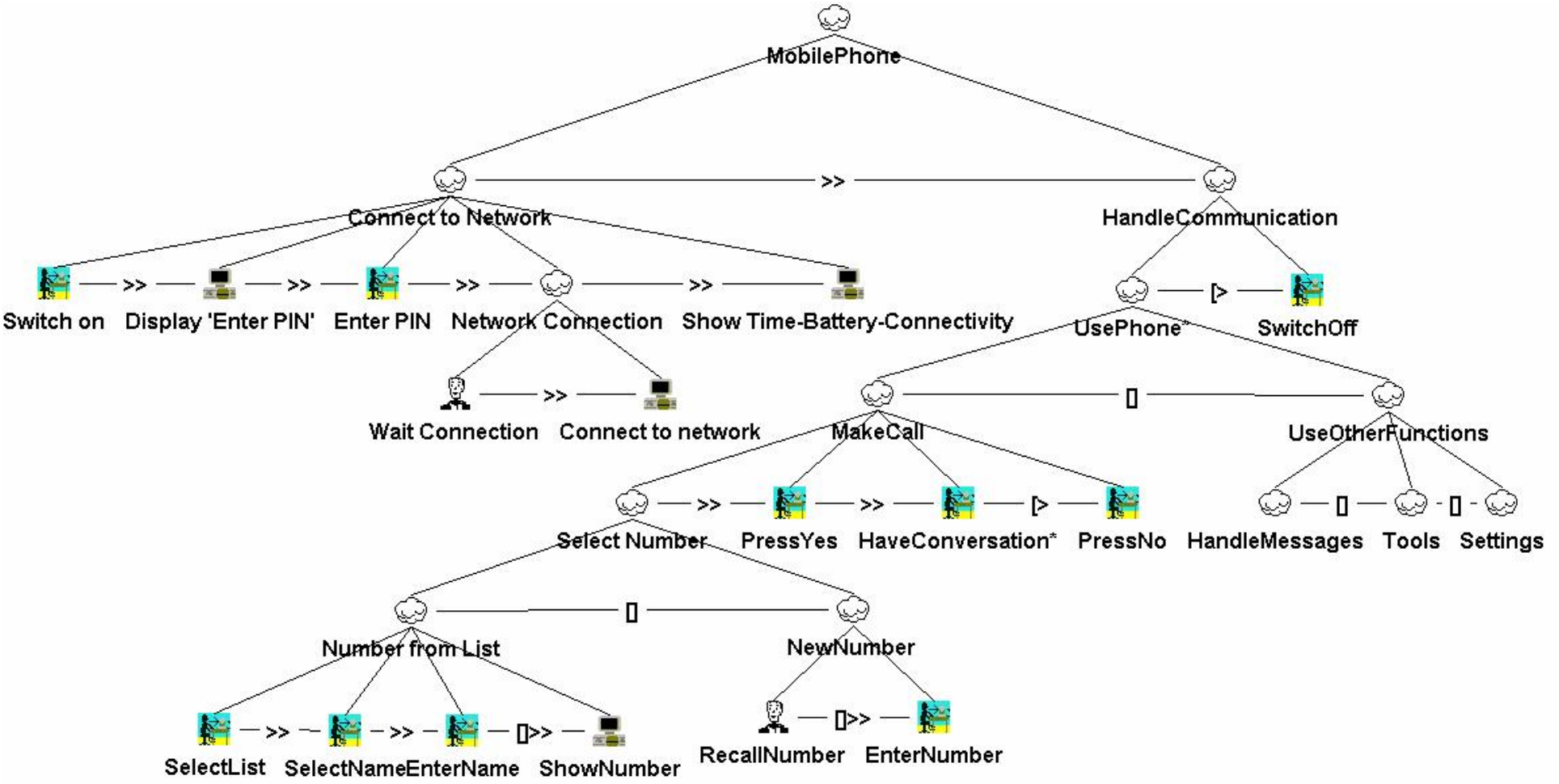
Optional tasks



Multiple performance / continuous interleaving



An Example


MobilePhone
>>
Connect to Network
HandleCommunication
Switch on
>>
Display 'Enter PIN'
>>
Enter PIN
>>
Network Connection
>>
Show Time-Battery-Connectivity
|>
UsePhone*
SwitchOff
Wait Connection
>>
Connect to network
[]
MakeCall
UseOtherFunctions
>>
Select Number
>>
PressYes
|>
HaveConversation*
PressNo
[]
HandleMessages
[]
Tools
[]
Settings
[]
Number from List
NewNumber
>>
SelectList
>>
SelectNameEnterName
[]>>
ShowNumber
[]>>
RecallNumber
EnterNumber

Representations of Task Models

	GOMS	**UAN**	**CTT**	**MAD**	**GTA**
Sequence	X	X	X	X	X
Order independence		X	X		X
Interruption		X	X	X	
Concurrency	**Only CPM-GOMS**	X	X	X	X
Optionality			X	X	
Iteration		X	X	X	X
Allocation			X		X
Objects			X		X
Performance	X		X		X
Pre-post conditions	X	X	X	X	X

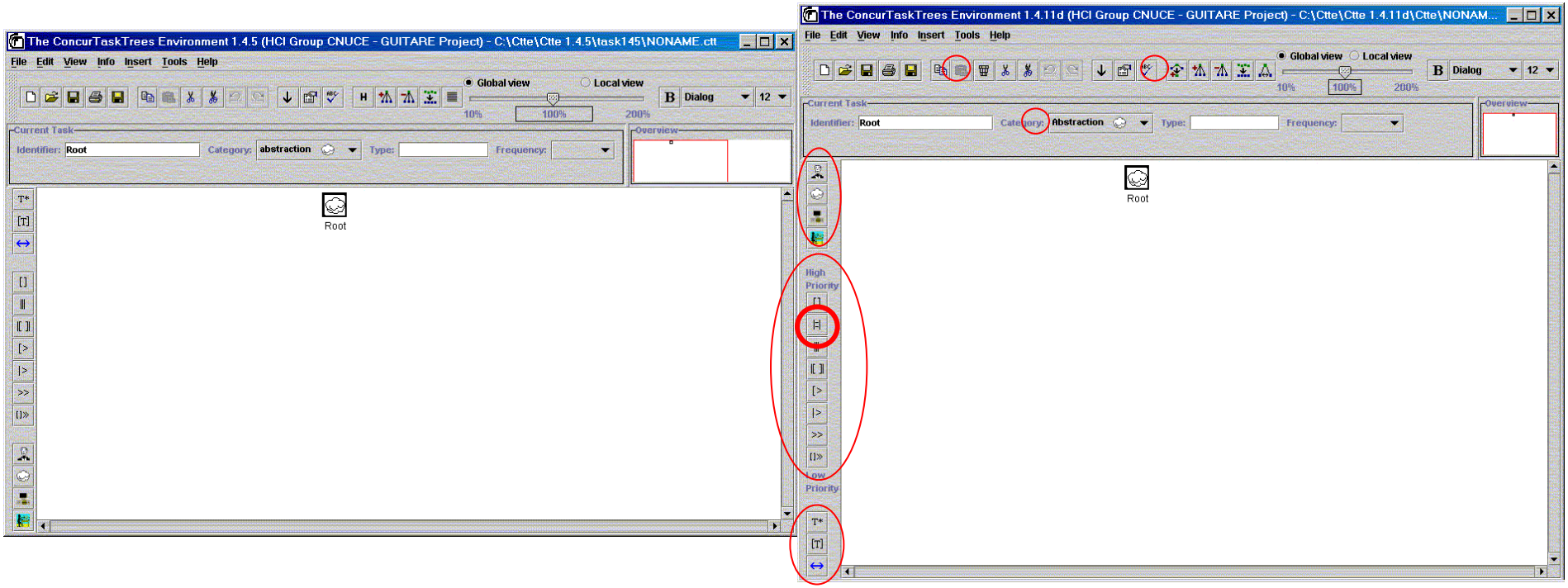
Tool support

- GTA – Euterpe
- VTMB -TAMOSA
- ICO – PetShop
- Teallach
- QGOMS

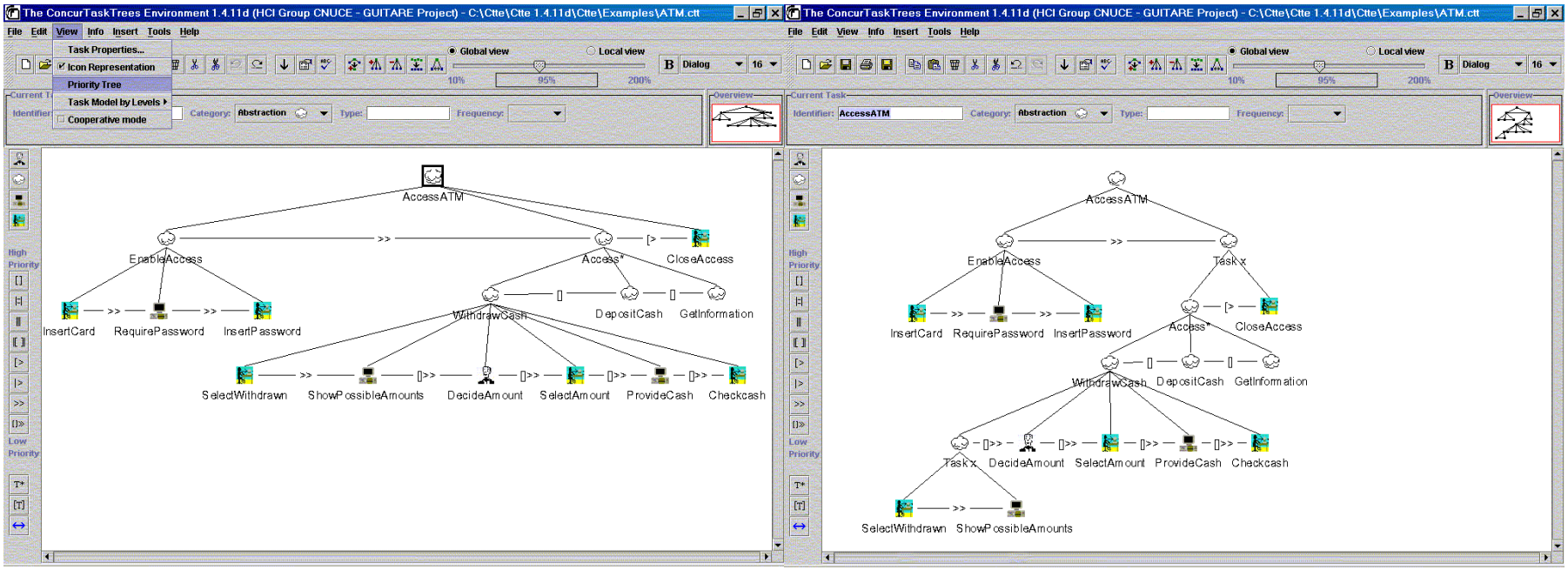
Tool Support in CTTE

- Flexible editing of the task model
- Using informal descriptions in modelling
- Checking completeness of the specification
- Saving the specification in various formats
- Simulating the task model
- Comparing task models
- Running scenarios
- http://giove.cnuce.cnr.it/ctte.html

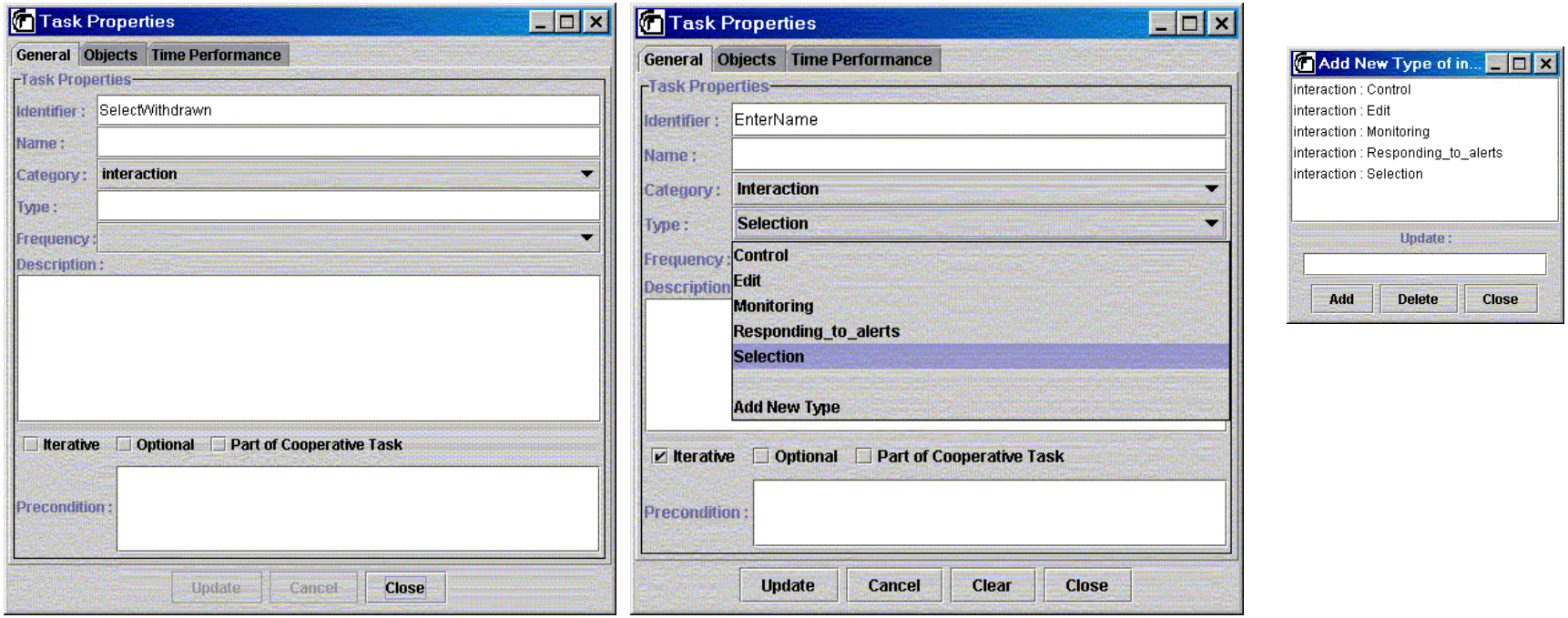
CTTE Changes



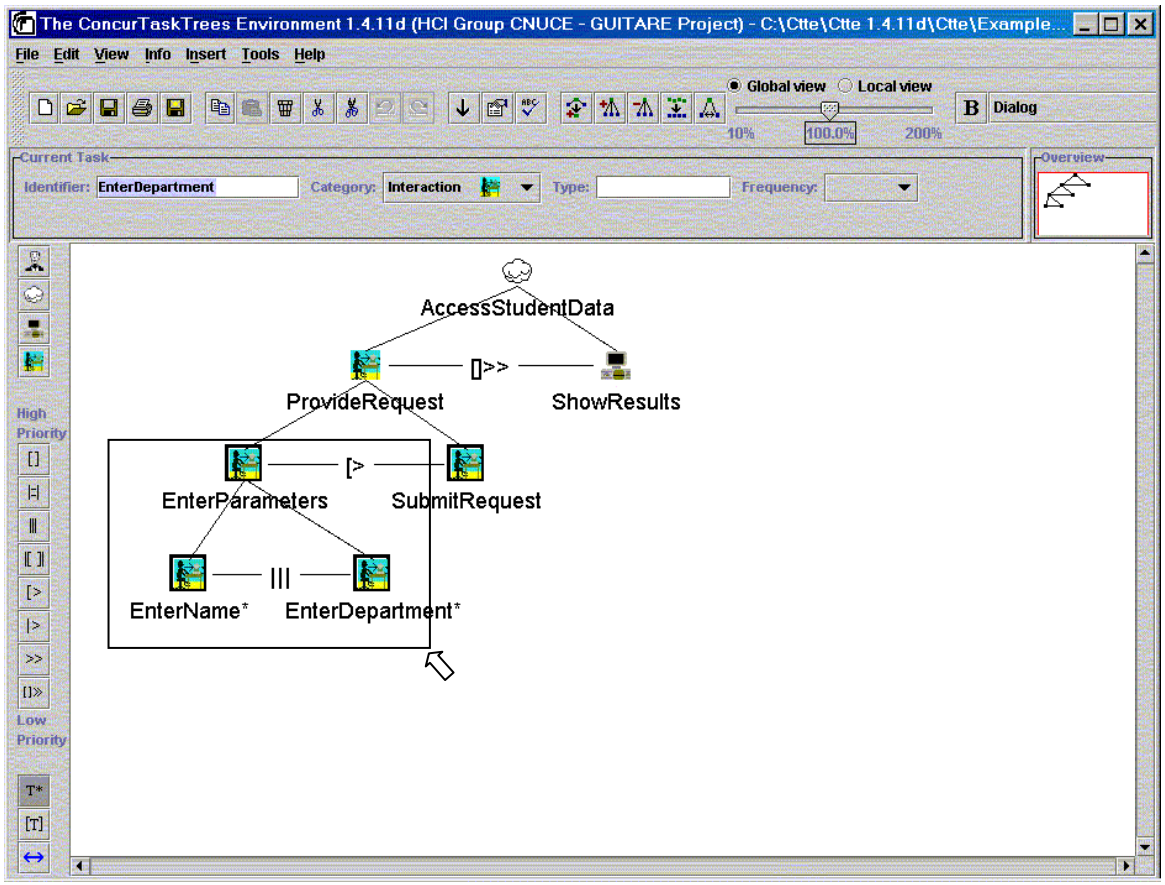
View Priority Tree



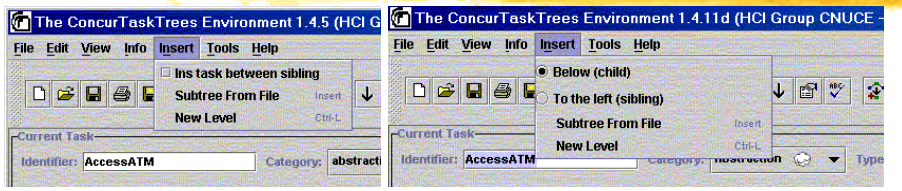
Editing task types

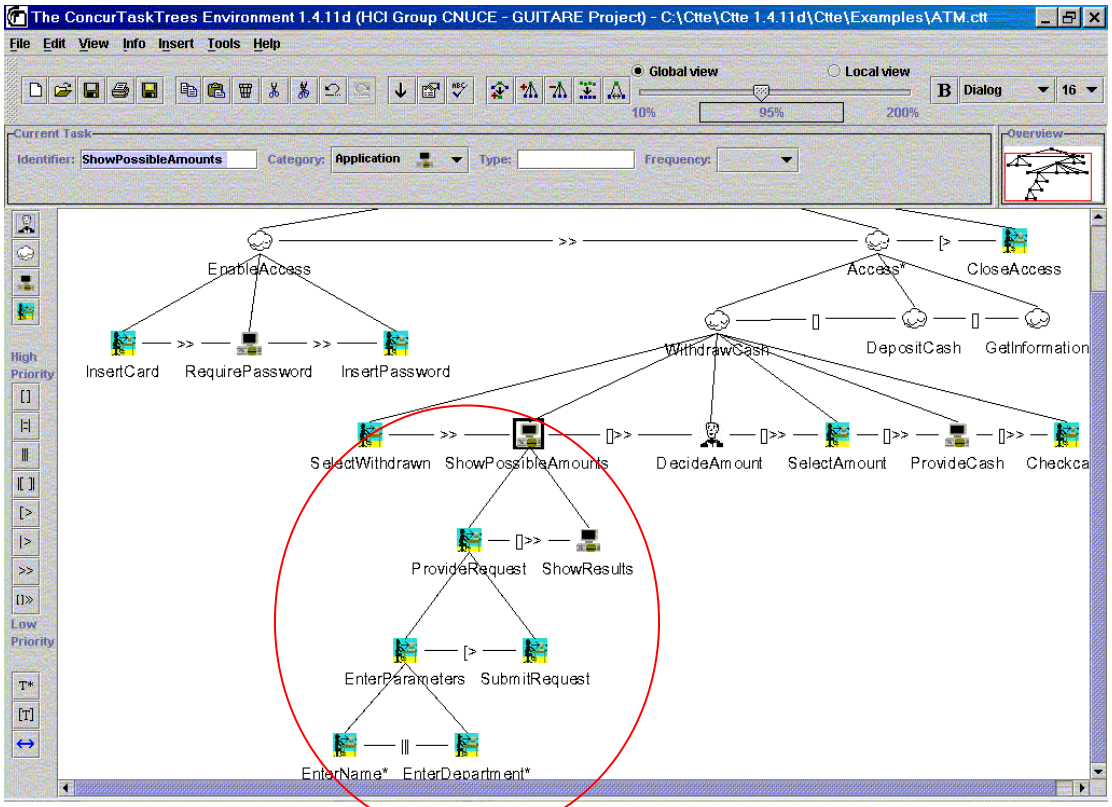


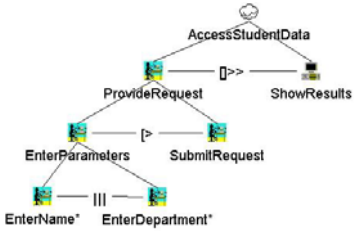
Graphical selection



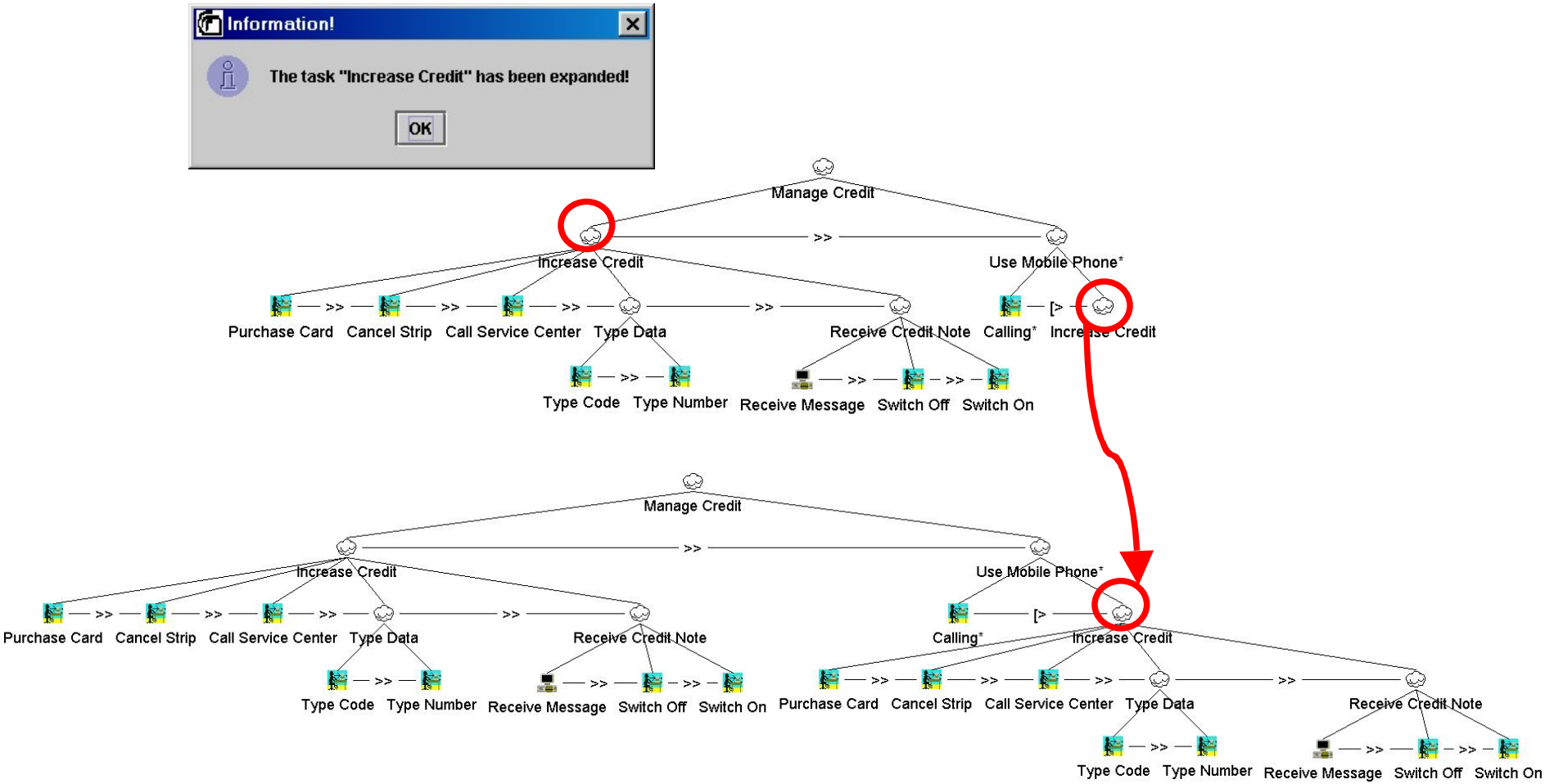
Subtree insertion



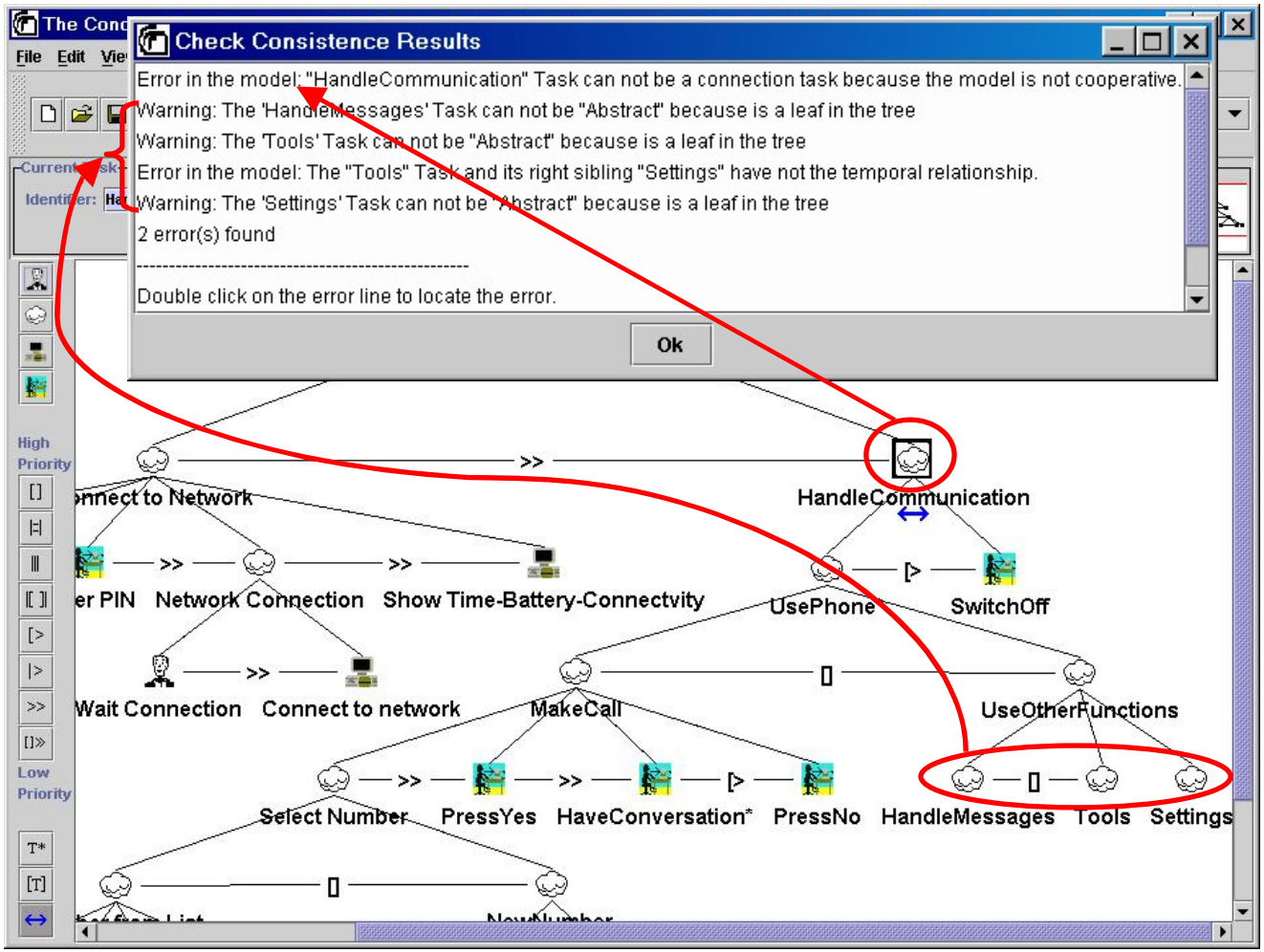




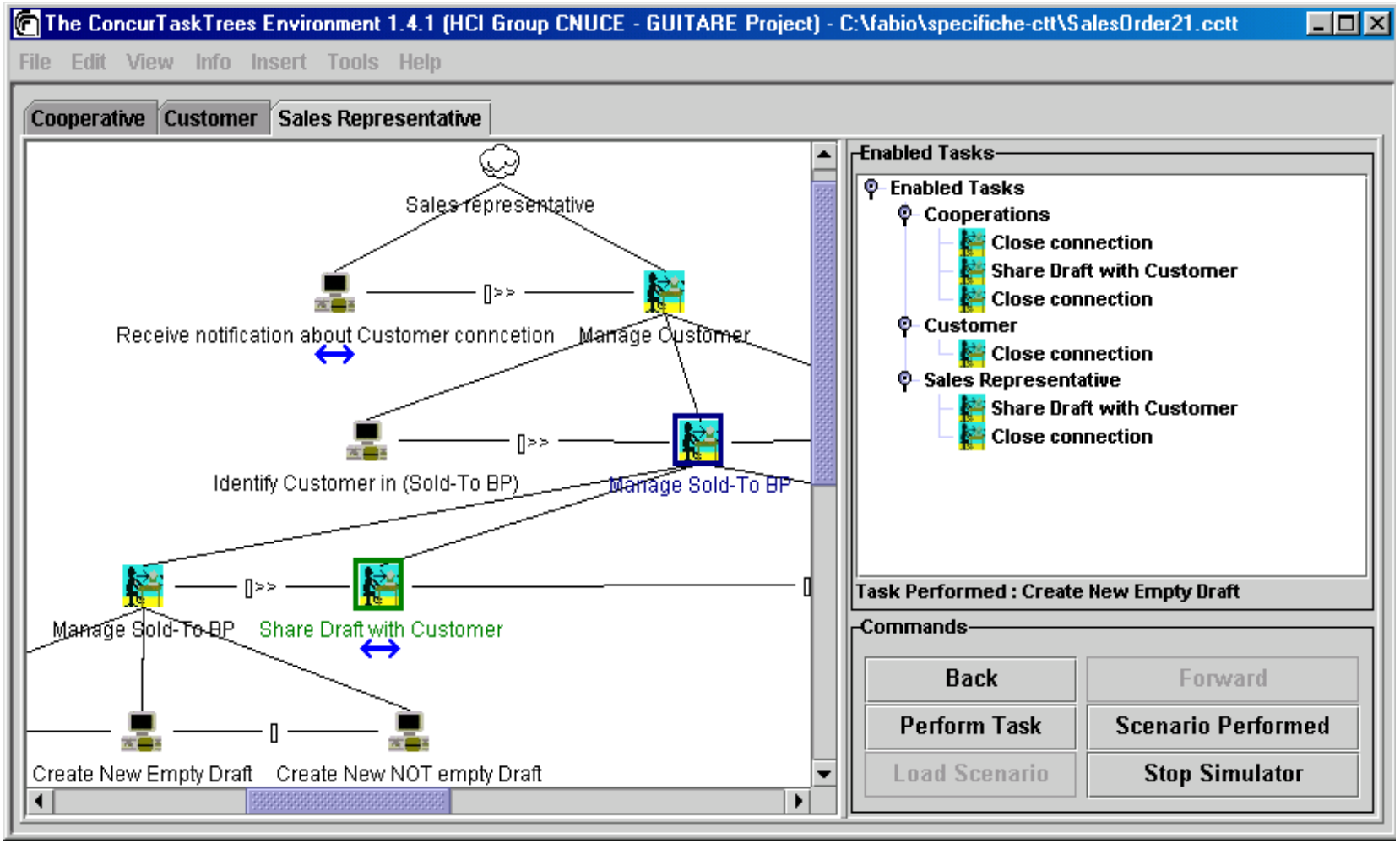
Automatic expansion of task patterns



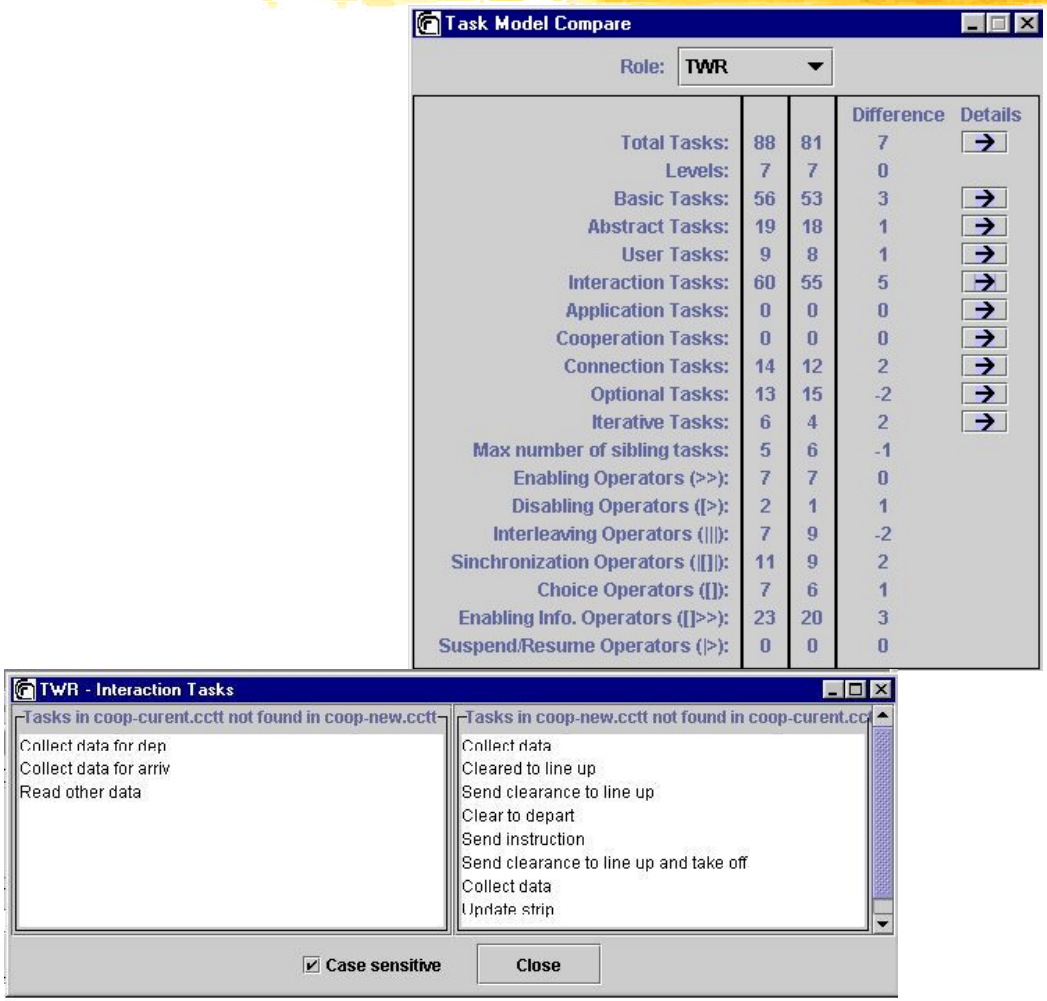
Checking completeness of the specification



Simulating dynamic behaviour

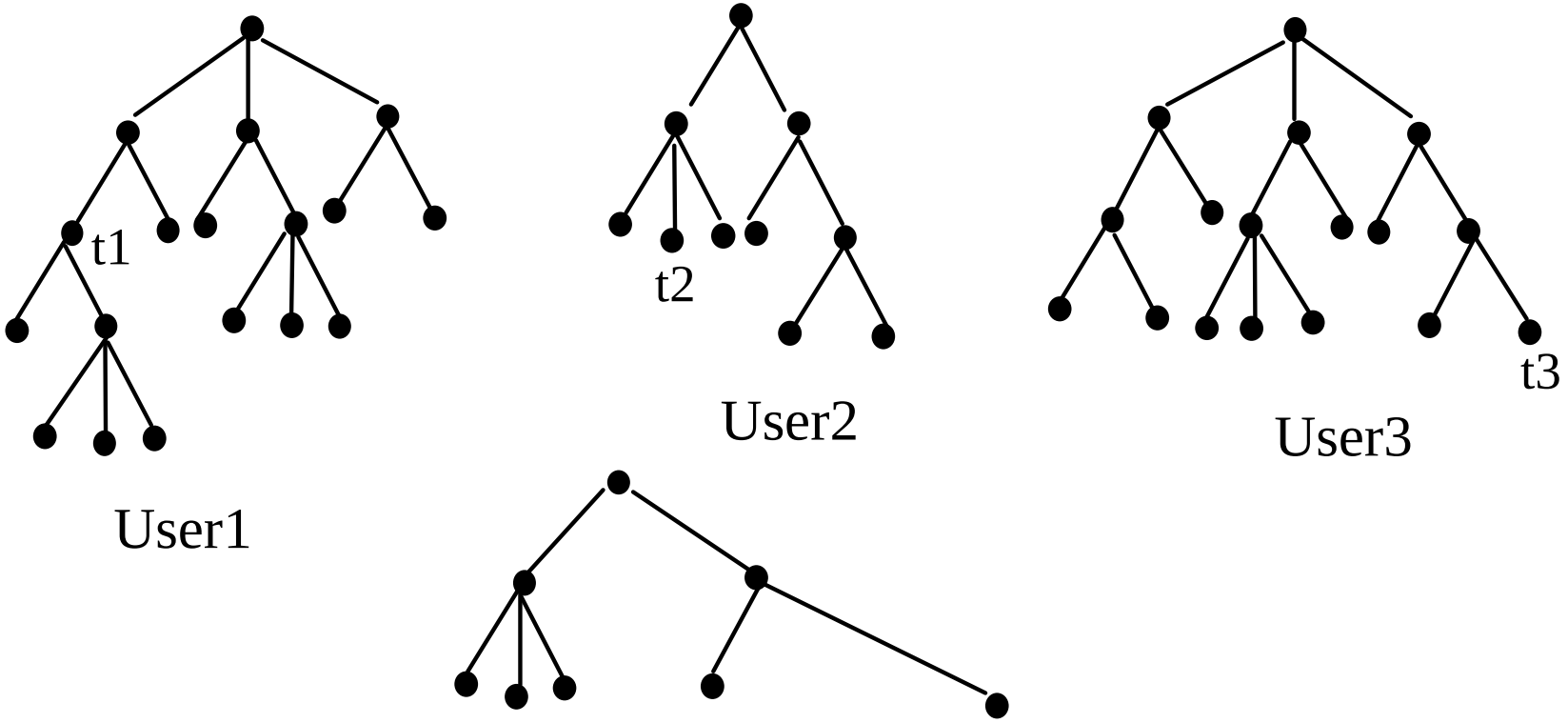


Comparison of Task Models



Tools and Methods based on Task Models

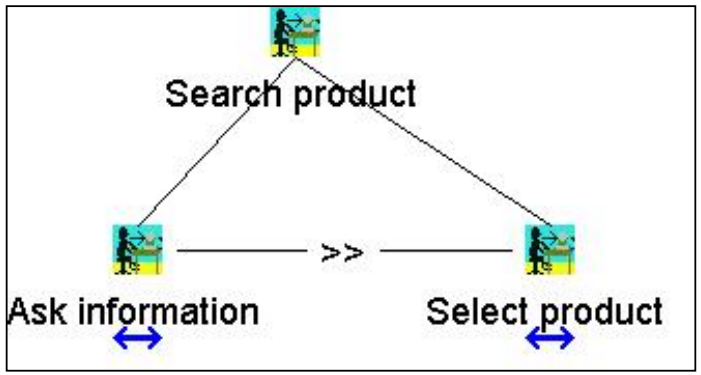
Modelling Multi-User Applications

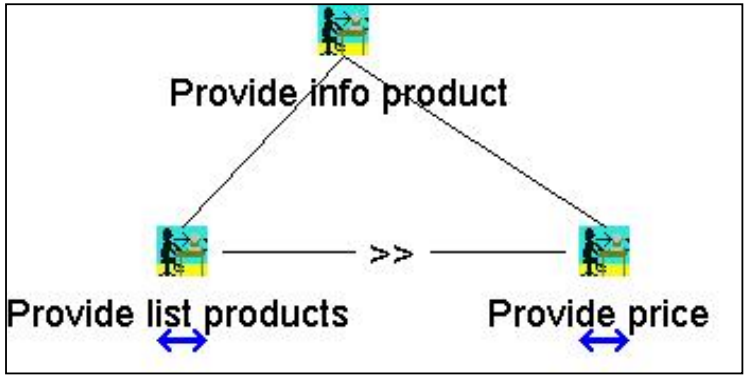


Cooperative aspects

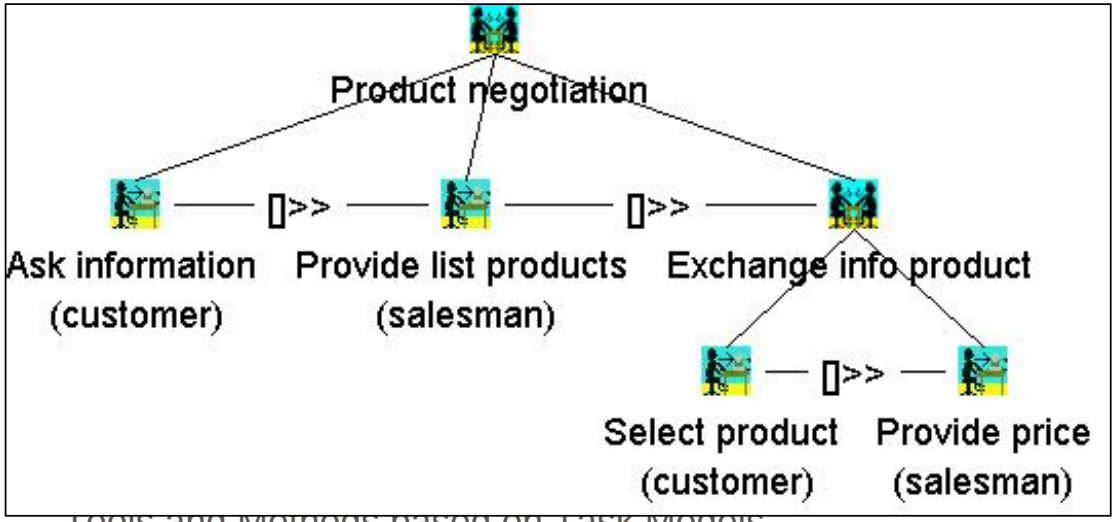
Customer role

Salesman role





Cooperative part



CTTE Demo