

# Human Computer Interaction

## UNIT-5

### Lecture 2:

### Task modeling and analysis

**Mr. Nachiket Sainis/Ms.Reena Saini**



**B K Birla Institute of Engineering & Technology,  
Pilani**

# Lecture 2:

## Engineering Task Models and CTT

# Objective

- In the previous lecture, we learned about the importance of task modeling and analysis in HCI
- We have also learned about the hierarchical task analysis

# Problems with HTA

- HTA is easy to understand, therefore it can serve as a good starting point for modeling tasks
- However, HTA is not very helpful when implementation comes into the picture

# Problems with HTA

- In order to implement something, we need to specify it in an unambiguous way
  - The HTA lacks the rigor and formalism required for such specification
- Engineering task models are more useful as they can be specified formally

# Engineering Task Models - Characteristics

- Engineering task models should have flexible and expressive notations, which are able to describe clearly the possible activities.
- Notations should be sufficiently powerful to describe interactive and dynamic behaviors.
- Notations should be readable so that they can also be interpreted by people with little formal background.

# Engineering Task Models - Characteristics

- An engineering task model should have systematic methods to support the specification, analysis, and use of task models in the design
- Otherwise, it will be difficult to use the knowledge gained from task analysis
- Such methods can also be incorporated in tools aiming to support interactive designers

# Engineering Task Models - Characteristics

- Another characteristics of an engineering task model is that, it should have support for the reuse of good design solutions to problems that occur across many applications.
  - This is especially relevant in industrial context, where developers often have to design applications that address similar problems.



# Engineering Task Models - Characteristics

- Finally, it is also important that engineering task models make automatic tools available to support the various phases of the design cycle.
  - Tools should have intuitive representations and provide information useful for the logical activities of designers


# Concur Task Tree (CTT)

- CTT is an engineering approach to task modeling
- A CTT consists of tasks and operators
  - Operators are used to depict temporal relationships between tasks

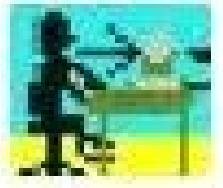
# ConcurTaskTree (CTT)

- The key features of a CTT are
  - Focus on activities that users aim to perform
  - Hierarchical structure
  - Graphical syntax
  - Rich set of temporal operators

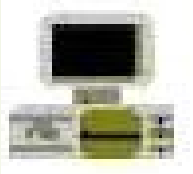
# CTT – Task Categories

- In CTT, four task categories are defined
  - **User task:** these are tasks that represent only internal cognitive activity of a user, such as selecting a strategy to solve a problem
  - It is graphically depicted with the symbol 
  - Can have subtypes such as planning, comparing, problem solving ...

# CTT – Task Categories

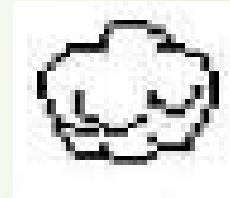
- **Interaction task:** these are user actions with possibility of immediate system feedback, such as editing a diagram
- It is graphically depicted with the symbol 
- Can have subtypes such as selection, edit, control ...

# CTT – Task Categories

- **Application task:** these refer to tasks performed by the system only, such as generating a query result
- It is graphically depicted with the symbol 
- Can have subtypes such as overview, comparison, locate, grouping, processing feedback ...

# CTT – Task Categories

- **Abstract task:** these refer to tasks whose subtasks are of different types (e.g., one user task and one application task) or the task type is not yet decided
- It is graphically depicted with the symbol



# CTT – Task Categories

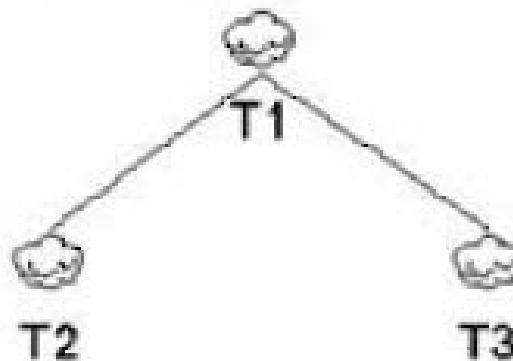
- CTT provides facilities to take care of more task characteristics, namely iterative tasks and optional tasks
  - An iterative task  $T$  is denoted by  $T^*$
  - An optional task  $T$  is denoted by  $[T]$



# CTT – Hierarchy

- CTT supports hierarchical task representation. Task at the same level can represent different options or same abstraction level to be performed

Example: in order to do T1, you have to perform T2 and/or T3



# CTT – Hierarchy

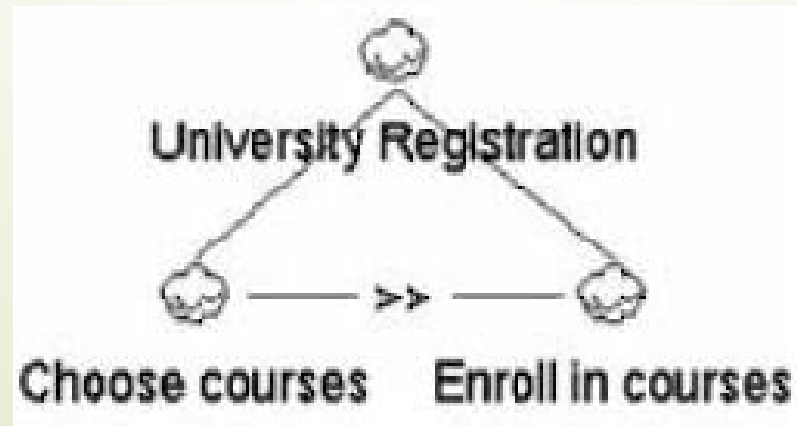
- Note that in CTT, hierarchy does not imply sequence
- In order to represent sequence, the tasks have to be modeled at the same level in a left-to-right manner

# CTT – Temporal Operators

- There are eight temporal operators defined in CTT

**Enabling operator ( $\gg$ ):** if two tasks T1 and T2 are related by the operator as  $T1 \gg T2$ , it means that T2 can not occur before T1

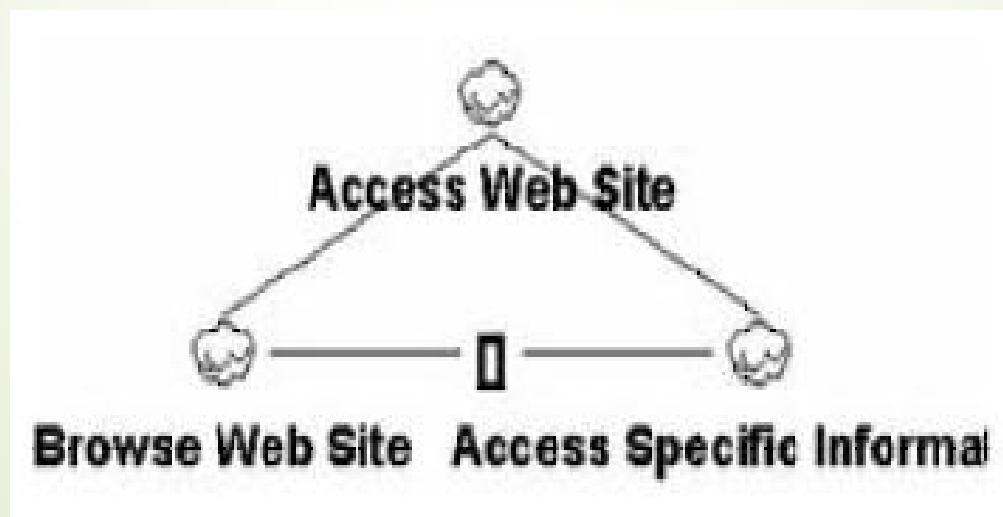
Example: you can not enroll in a course unless you select the course



# CTT – Temporal Operators

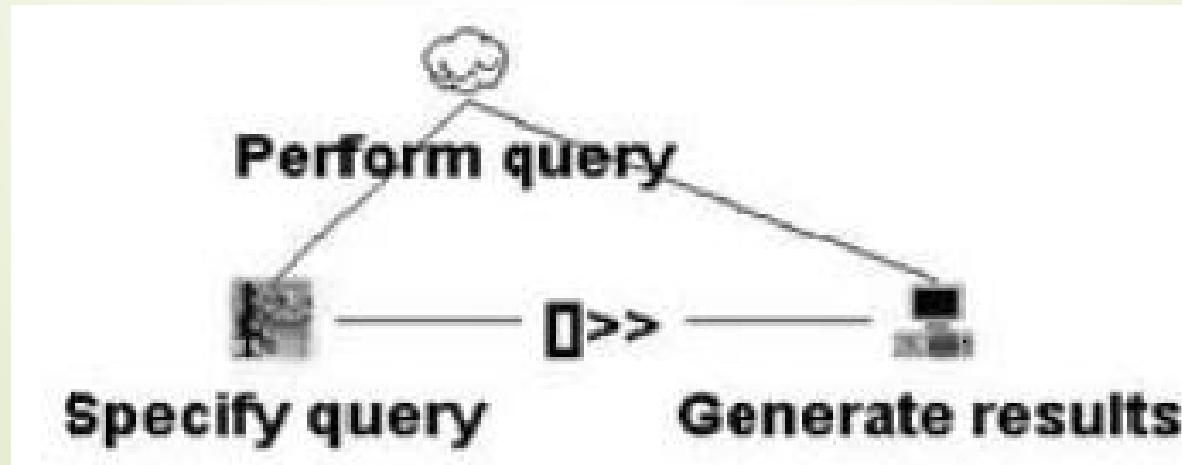
**Choice operator ([ ])**: if two tasks T1 and T2 are related by the operator as  $T1[]T2$ , it means that both T1 and T2 are enabled, however, only one can be performed at a time.

Example: you can either browse a web site or follow a link for details



# CTT – Temporal Operators

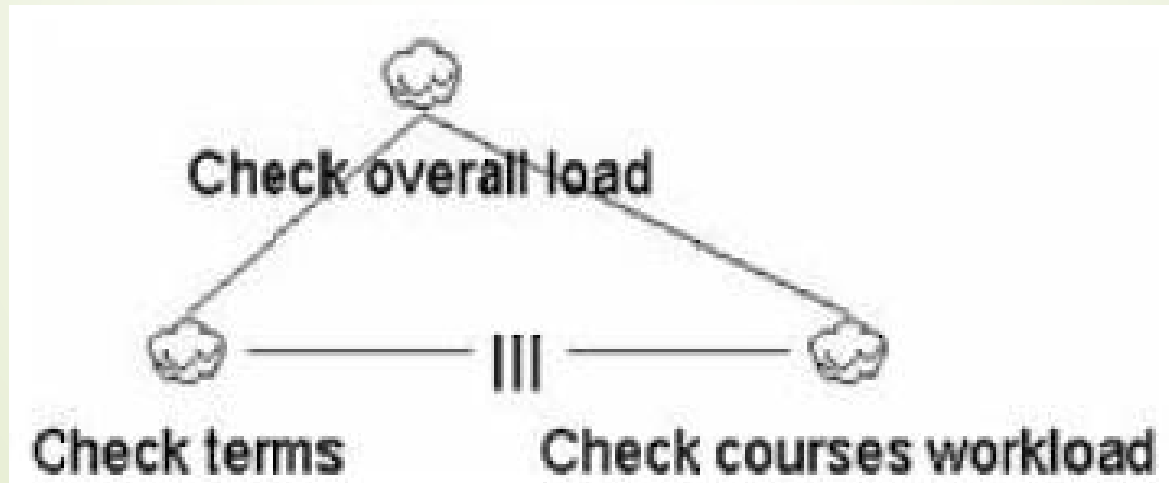
**Enabling with information passing operator** ( $[]>>$ ): if two tasks T1 and T2 are related by the operator as  $T1[]>>T2$ , it means that T2 can't be performed before T1 and depends on the results of T1  
Example: a system can generate result only after user specifies query. The result depends on the query.



# CTT – Temporal Operators

**Concurrent operator (|||):** if two tasks T1 and T2 are related by the operator as  $T1|||T2$ , it means that T1, T2 can be performed at any time, in any order.

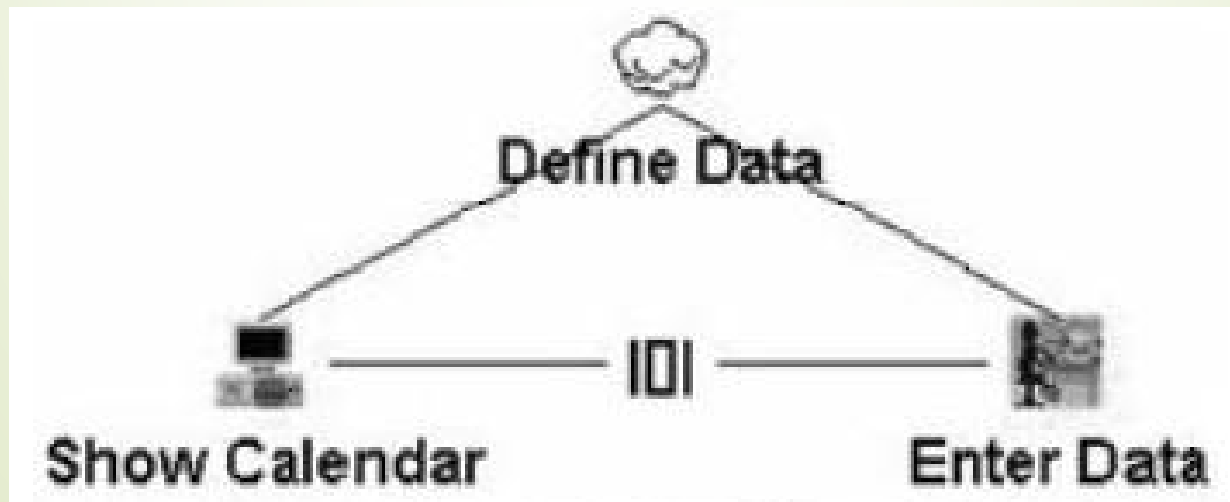
Example: to check the overall course load, we need to check the semester as well as the course workload.



# CTT – Temporal Operators

**Concurrent communicating operator ( $||[]$ ):** if two tasks T1 and T2 are related by the operator as  $T1||[]T2$ , it means that T1, T2 can be performed concurrently and can exchange information.

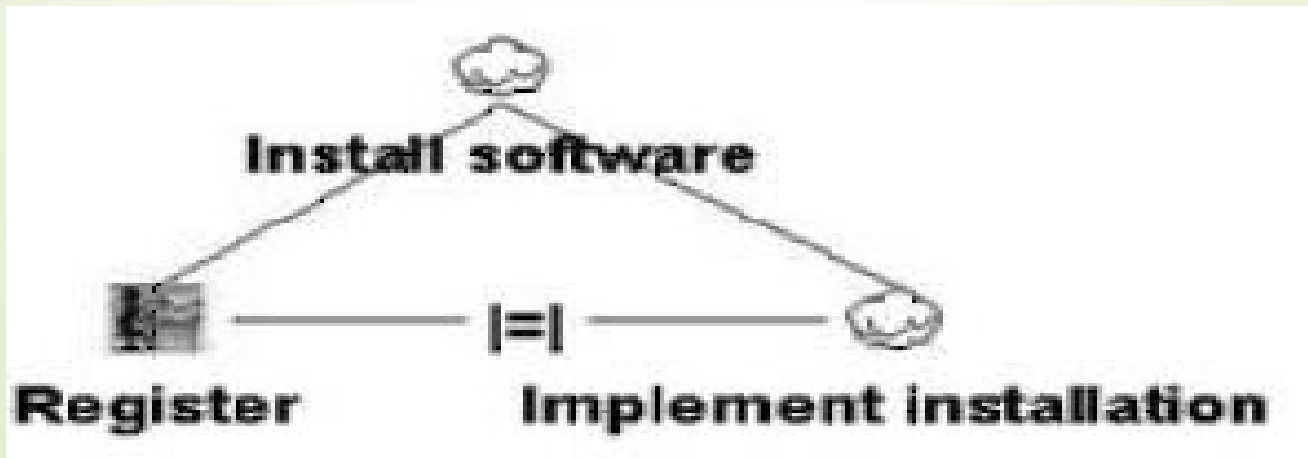
Example: an onscreen calendar highlighting dates being entered by the user.



# CTT – Temporal Operators

**Task independence operator ( $|=|$ ):** if two tasks T1 and T2 are related by the operator as  $T1|=|T2$ , it means that T1, T2 can be performed independent to one another, however, when one starts, it has to finish before the other can start.

Example: when installing a new software, you can either register and then install or vice-versa.

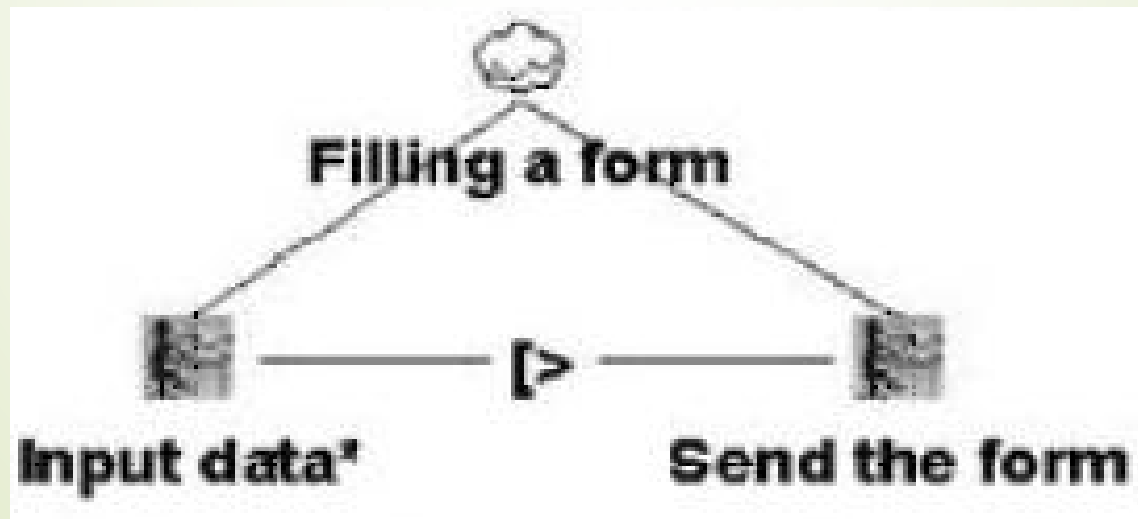




# CTT – Temporal Operators

**Disabling operator ( $[>]$ ):** if two tasks T1 and T2 are related by the operator as  $T1[>T2$ , it means that T1 (usually iterative) is completely interrupted by T2.

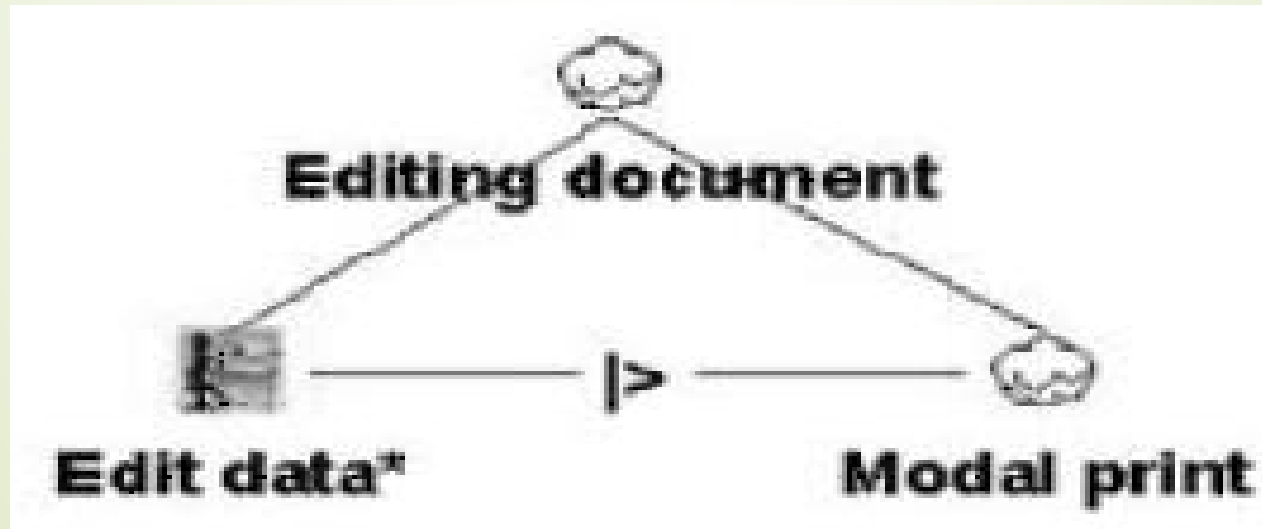
Example: a user can iteratively input data in a form until it is sent.



# CTT – Temporal Operators

**Suspend-resume operator ( $|>$ ):** if two tasks T1 and T2 are related by the operator as  $T1|>T2$ , it means T1 can be interrupted by T2. When T2 ends, T1 can resume from where it stopped.

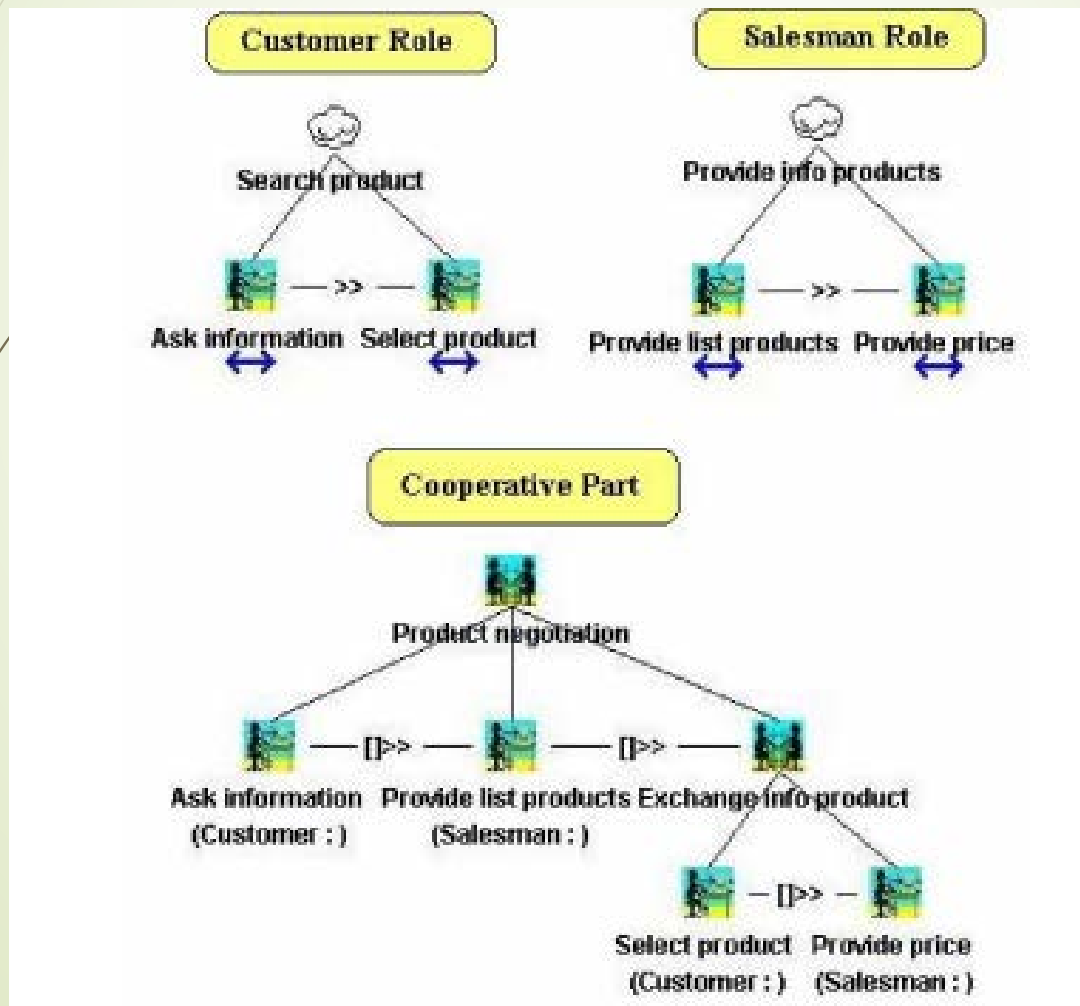
Example: editing some data and then printing it, assuming the two can't be performed together.



# CTT for Cooperative Task



## Cooperative Task



# CTT – Other Features

- Tasks in CTT are used to manipulate *Objects*
- **Two types of objects defined**
  - **Perceivable (user interface) objects** – these refer to output objects for presenting information (e.g., windows, tables, graphs) or items which users can interact with (e.g., menus, icons, windows).
  - **Application (domain) objects:** these are entities that belong to the application domain.

# CTT – Other Features

- Information concerning application objects needs to be mapped onto perceivable objects to be presented to the user.
- Examples of application objects include an order in a business application or a flight in an air traffic control application.

# CTT – Other Features

- Multiple user interface objects can be associated with a domain object.
- For example, temperature (a domain object) can be represented by a bar- chart (an user interface object) or a textual value (another user interface object).
- Each object can be manipulated by one/more tasks
- Tasks can have other information as attribute (e.g., frequency, informal description, estimated performance time etc.)

# CTT - Advantage

- Formal notations help to check for completeness in specification (e.g. each non-basic task has at least two children)
- It allows us to compare two models in terms of number of tasks, number of basic tasks, allocation of tasks, number of instances of temporal operators, the structure of the task models (number of levels, maximum number of sibling tasks etc.)