

# **Task Modeling and Analysis: Engineering Task Models and CTT**

# Learning 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
- In this lecture, we shall discuss the engg. task models

# Learning Objective

- In particular, we shall learn about the following
  - The need for engineering task models
  - ConcurTaskTrees (CTT) – an engineering task model
  - An example to better understand the CTT (Case Study)

# 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


# ConcurTaskTree (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

- In CTT, four task categories are defined
  - **Interaction task:** these are user actions with possibility of immediate system feedback, such as editing a diagram
  - It is graphically depicted with the symbol A small, pixelated icon showing a person sitting at a desk with a computer, representing a user interaction.
  - Can have subtypes such as selection, edit, control ...

# CTT – Task Categories

- In CTT, four task categories are defined
  - **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

- In CTT, four task categories are defined
  - **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

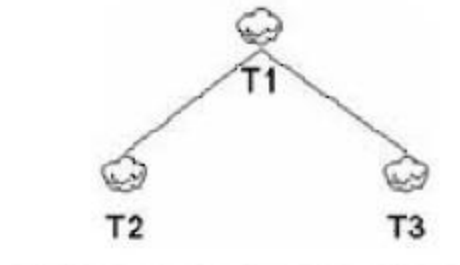
- CTT provides facilities to take care of more task characteristics, namely iterative tasks and optional tasks
  - An iterative task  $T$  is denoted by  $T^*$
  - An optional task  $T$  is denoted by  $[T]$



# 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

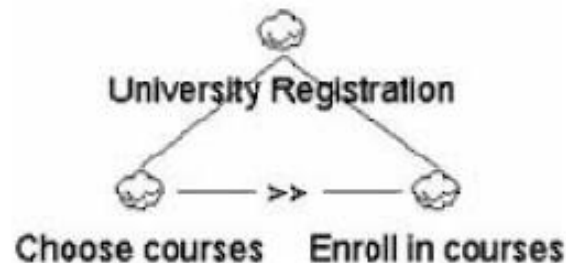
- Note that in CTT, hierarchy does not imply sequence
- In order to represent sequence, the tasks have to be modeled at the same level in a left-to-right manner

# 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  $T \gg T2$ , it means that T2 can not occur before T1

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

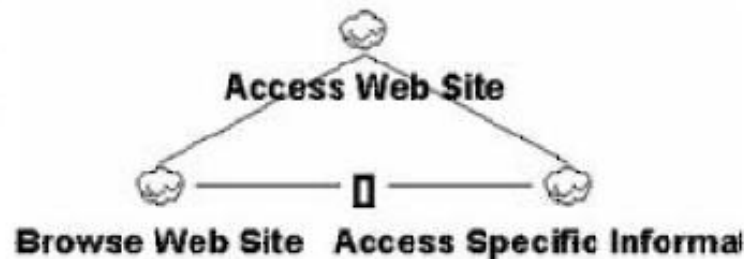


# CTT – Temporal Operators

- There are eight temporal operators defined in CTT

**Choice operator ([ ])**: if two tasks T1 and T2 are related by the operator as  $T[ ]T2$ , 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

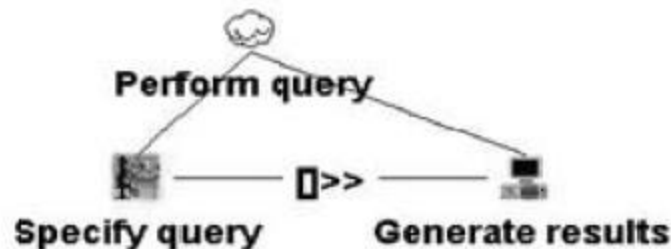


# CTT – Temporal Operators

- There are eight temporal operators defined in CTT

**Enabling with information passing operator ( $[]>>$ ):** if two tasks T1 and T2 are related by the operator as  $T[]>>T2$ , 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

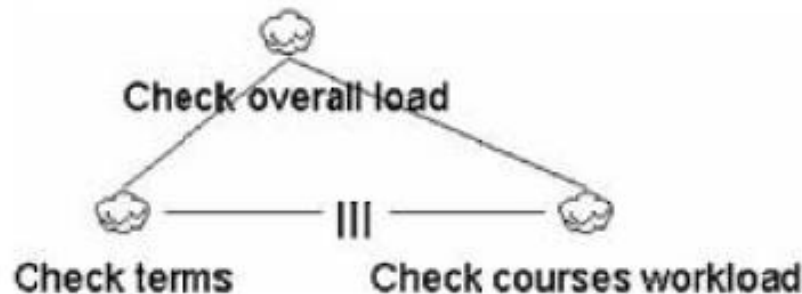


# CTT – Temporal Operators

- There are eight temporal operators defined in CTT

**Concurrent operator (|||):** if two tasks T1 and T2 are related by the operator as  $T ||| T2$ , 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

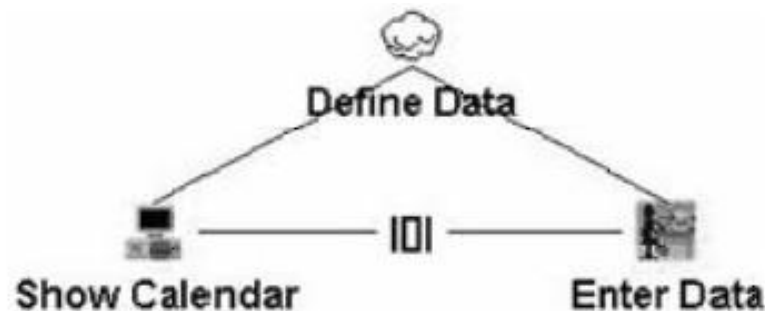


# CTT – Temporal Operators

- There are eight temporal operators defined in CTT

**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

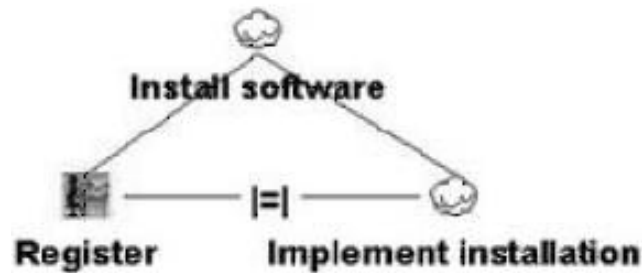


# CTT – Temporal Operators

- There are eight temporal operators defined in CTT

**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



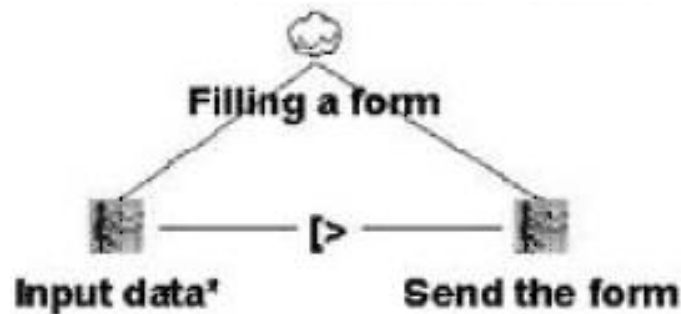


# CTT – Temporal Operators

- There are eight temporal operators defined in CTT

**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

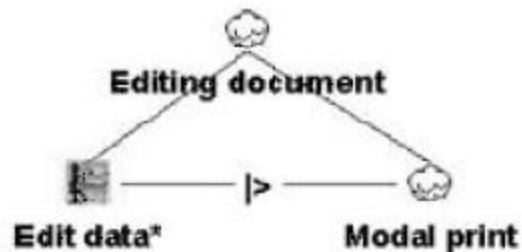


# CTT – Temporal Operators

- There are eight temporal operators defined in CTT

**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 – 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)

# CTT – Other Features

- 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, no. of instances of temporal operators, the structure of the task models (number of levels, maximum number of sibling tasks etc.)

# CTT – A Case Study

Suppose we want to model the user tasks in operating a mobile phone. Further assume that we are interested in modeling the tasks related to making a phone call only (not taking photos or listening to music!)

- The questions of interest are:
  - What are the tasks?
  - What are the task types (user tasks, interaction task etc.)?
  - How the tasks are related (what temporal operators are applicable between tasks)?

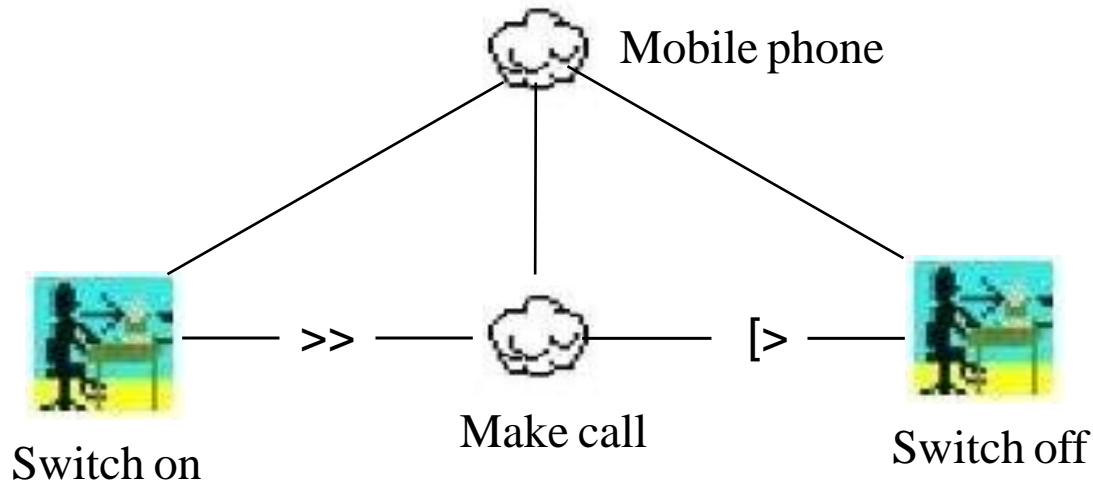


# CTT – A Case Study

- First thing to note that there are three sequential tasks:
  - We first switch-on the phone (assuming the phone is always switched-off after making a call)
  - Then we perform the tasks necessary to make phone call
  - Finally we switch-off the phone
- Switching on/off the phone are interaction tasks
- The type of the subtasks necessary to perform a phone call are not of single types. Hence it's an abstract task

# CTT – A Case Study

- Phone call can't take place before the phone is switched on. Thus, these two are related by the enabling operator ( $\gg$ )
- Phone can be switched off any time during a call. So, these two tasks are related by a disabling operator ( $[>$ )
- The top level structure of the CTT looks like the following:



# CTT – A Case Study

- Now, let us determine the subtasks for making a call
  - We first have to select the number of the person to be called (task: select number)
  - Then, we dial the number using a “make call” button (task: dial number)
  - Once the called person picks up the phone, we start conversation (task: conversation)
  - Finally, at the end, we disconnect (task: disconnect)
- What are the task types?

# CTT – A Case Study

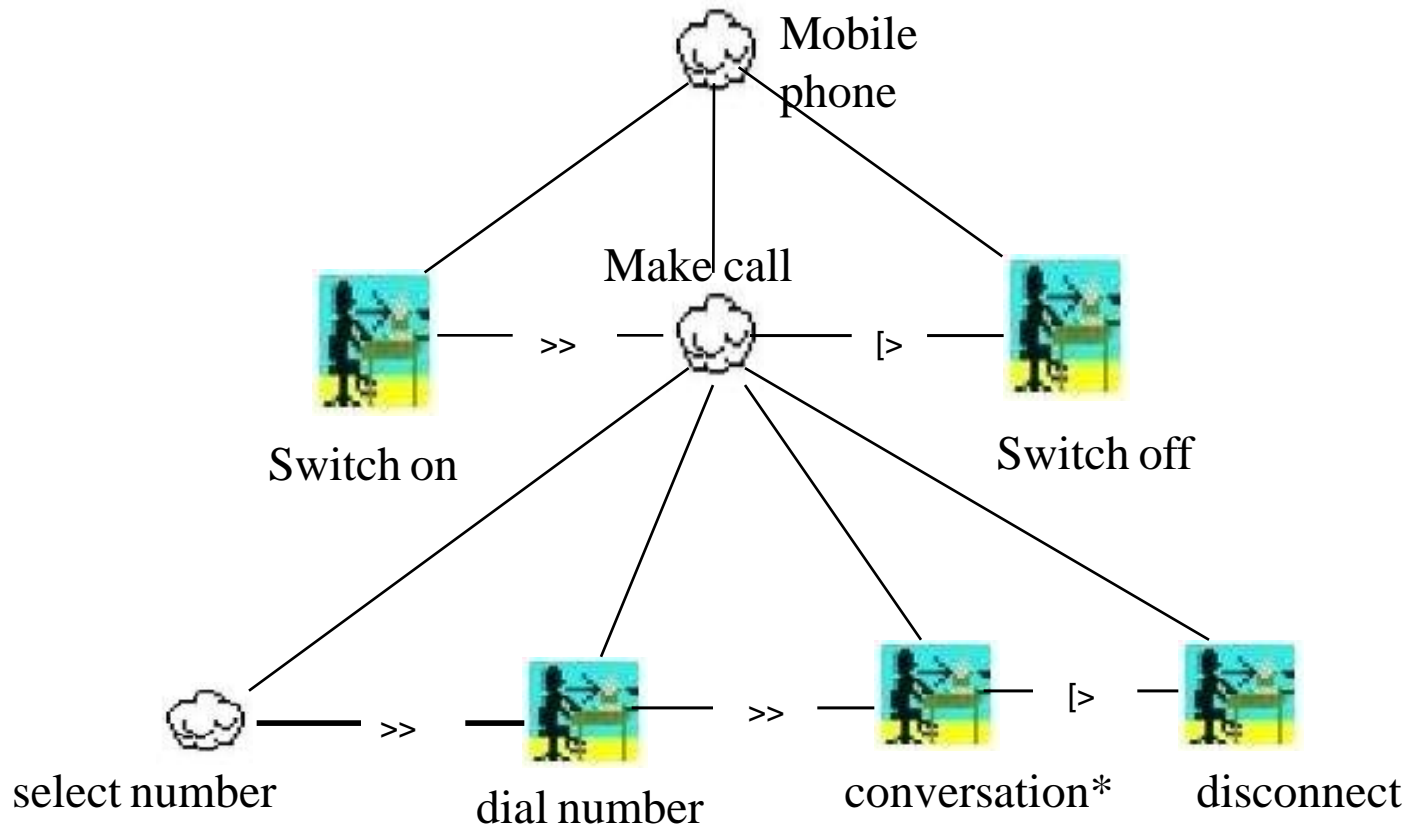
- The “select number” tasks involve subtasks of different types (as we shall see). So, it is an abstract task
- “Dial number” is clearly an interaction task
- “Conversation” is an interaction task. Moreover, it is a repetitive task if we assume that a conversation represents each pair of utterance (called/caller)-response (caller/called)
- The remaining task (“disconnect”) is also interaction task

# CTT – A Case Study

- Next, we need to find the relationship between these tasks
- It is obvious that “select number” precedes “dial number”, which in turn precedes “conversation”
- “disconnect” comes at the end of the sequence of tasks
- Therefore, “select number” and “dial number” should be connected by the enabling operator ( $\gg$ ). The same operator should apply between “dial number” and “conversation”
- Since a call can be disconnected any time during conversation, “conversation” and “disconnect” should be related with a disabling operator ( $[>$ )

# CTT – A Case Study

- Thus, the CTT looks like the following after expansion

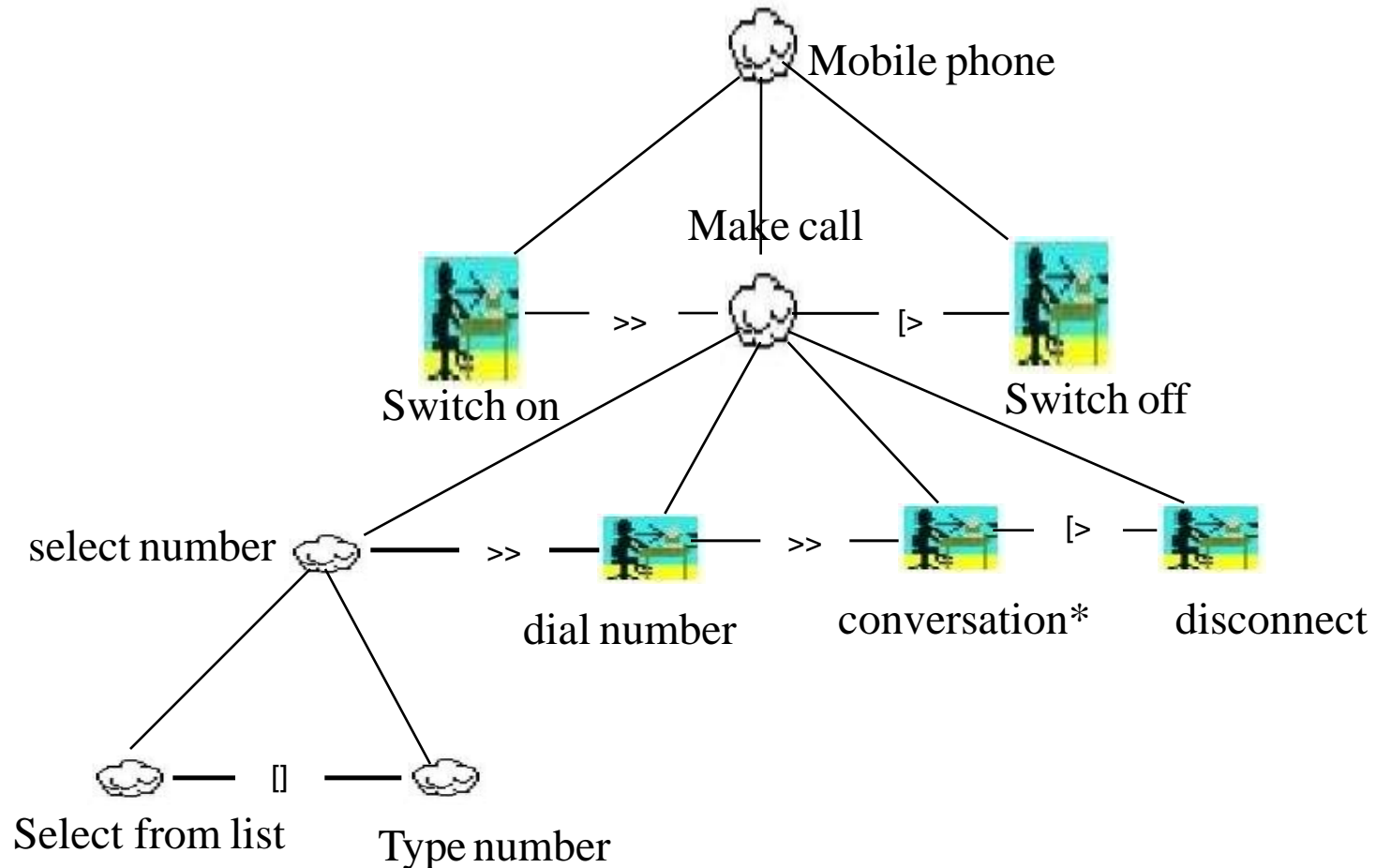


# CTT – A Case Study

- The task “select number” can be performed in either of the two ways
  - We can select the number from a contact list (task: select from list)
  - Or we can type the number (task: type number)
- Both the above tasks are abstract type (as they can be further divided into different types of subtasks) and connected by the choice operator ([ ])

# CTT – A Case Study

- Thus, the CTT looks like the following after expansion





# CTT – A Case Study

- The task “select from list” involves four sub tasks
  - First, select the appropriate button/menu option to display the list (task: open list)
  - Next, browse the list elements till you arrive at the name of the called person (task: find name)
- The task “select from list” involve four sub tasks
  - Then, select the name using appropriate button (task: select name)
  - Finally, once the name is selected, the corresponding number is displayed on the screen (task: show number)

# CTT – A Case Study

- The first three tasks (display list, find name, select name) are all interaction tasks
- Since these have to be performed in sequence, they are connected by the enabling operator (>>)
- The last task (display number) is of type application task. Moreover, it is related with the last task in the previous sequence (select name) through the enabling with information passing operator ([ ]>>)

# CTT – A Case Study

- The task “type number” involves two subtasks
  - We first need to recall the number (task: recall number).  
Clearly, this is a user task
  - Then we need to actually type the numbers (task: enter number).  
This is clearly an interaction task
- The two tasks are related through the enabling with information passing operator ( $[]>>$ )

# CTT – A Case Study

- Thus, the CTT for the entire task looks like the following

