Multimodal Interaction

Lesson 4
Designing Interaction

Maria De Marsico
demarsico@di.uniroma1.it

Dialog Design

- User models (difficult)
  - Cognitive models
  - Task models
- Technology models (easy)
- Sentence models (dialog sentences) to produce

Hierarchies of goals and tasks

- Abstraction of internal user models
- Mental processing as divide-and-conquer
- Example: produce a selling report

produce report
gather data
. find book names
. . do keywords search of names database
. . . … *further sub-goals*
. . sift through names and abstracts by hand
. . . … *further sub-goals*
. search sales database - further sub-goals
layout tables and histograms - further sub-goals
write description - further sub-goals

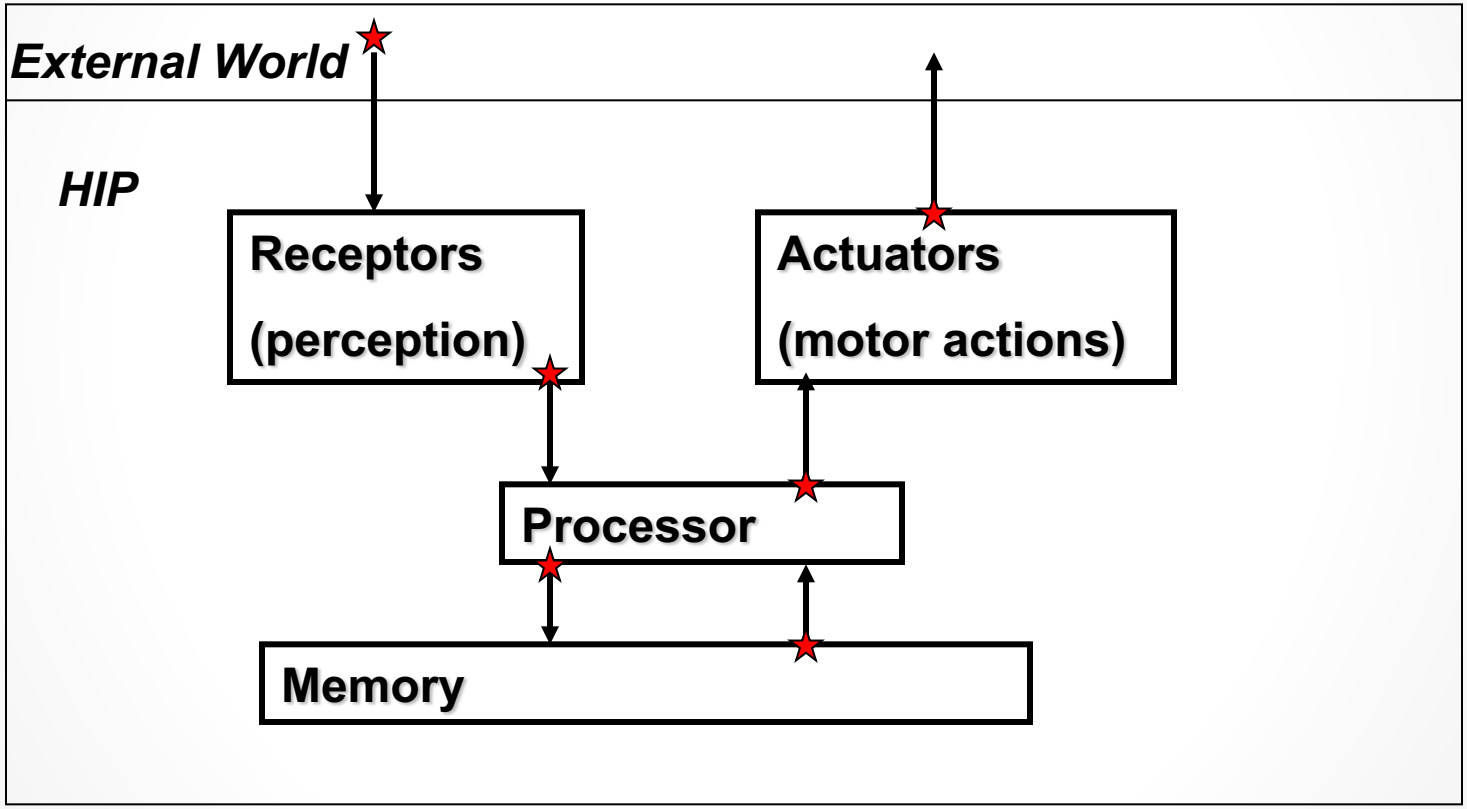
Goals vs. tasks

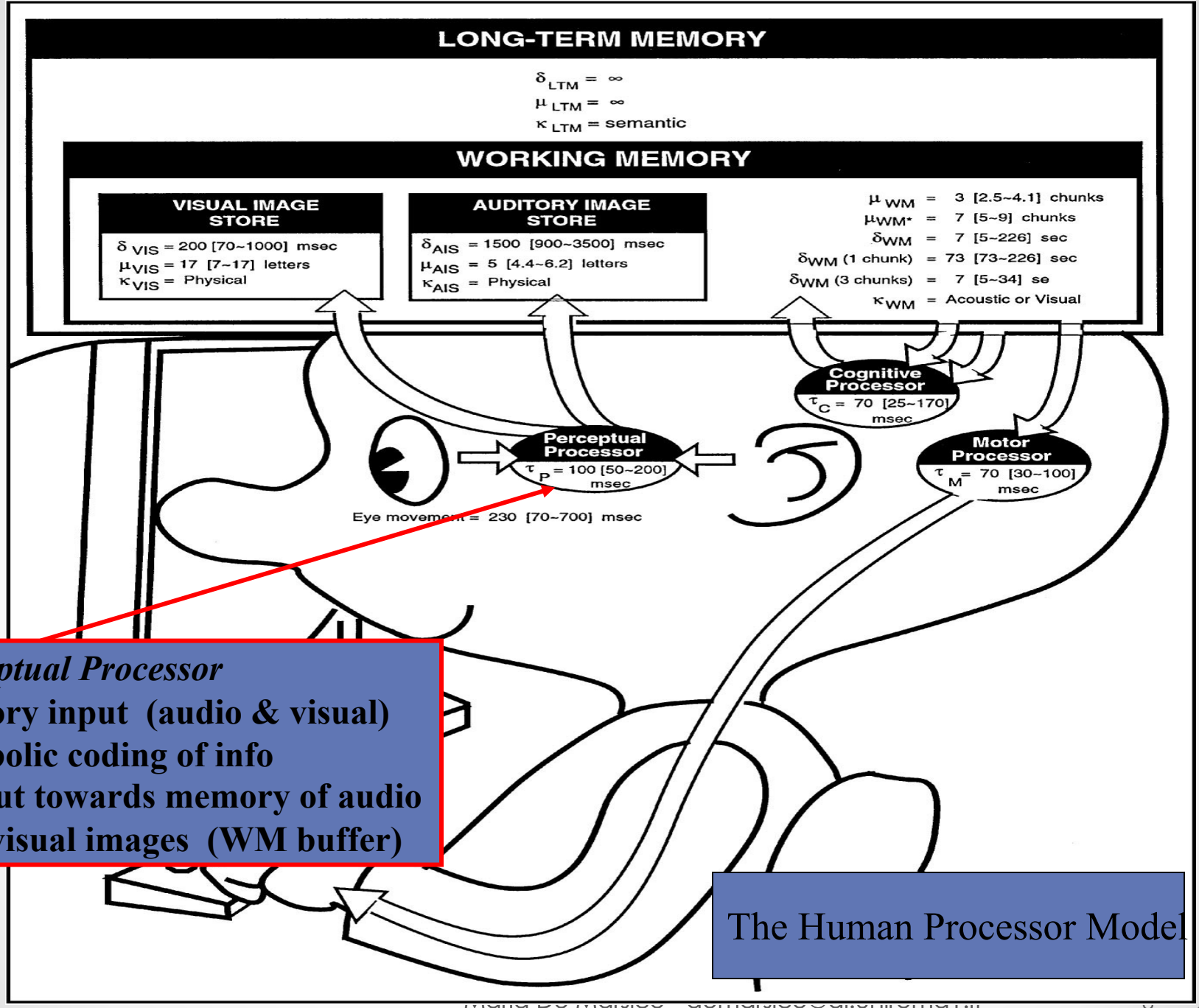
- goals – intentions
  What one would want
- tasks – actions
  How to obtain it

- GOMS – internal goals

- HTA – external actions
  – tasks are abstractions

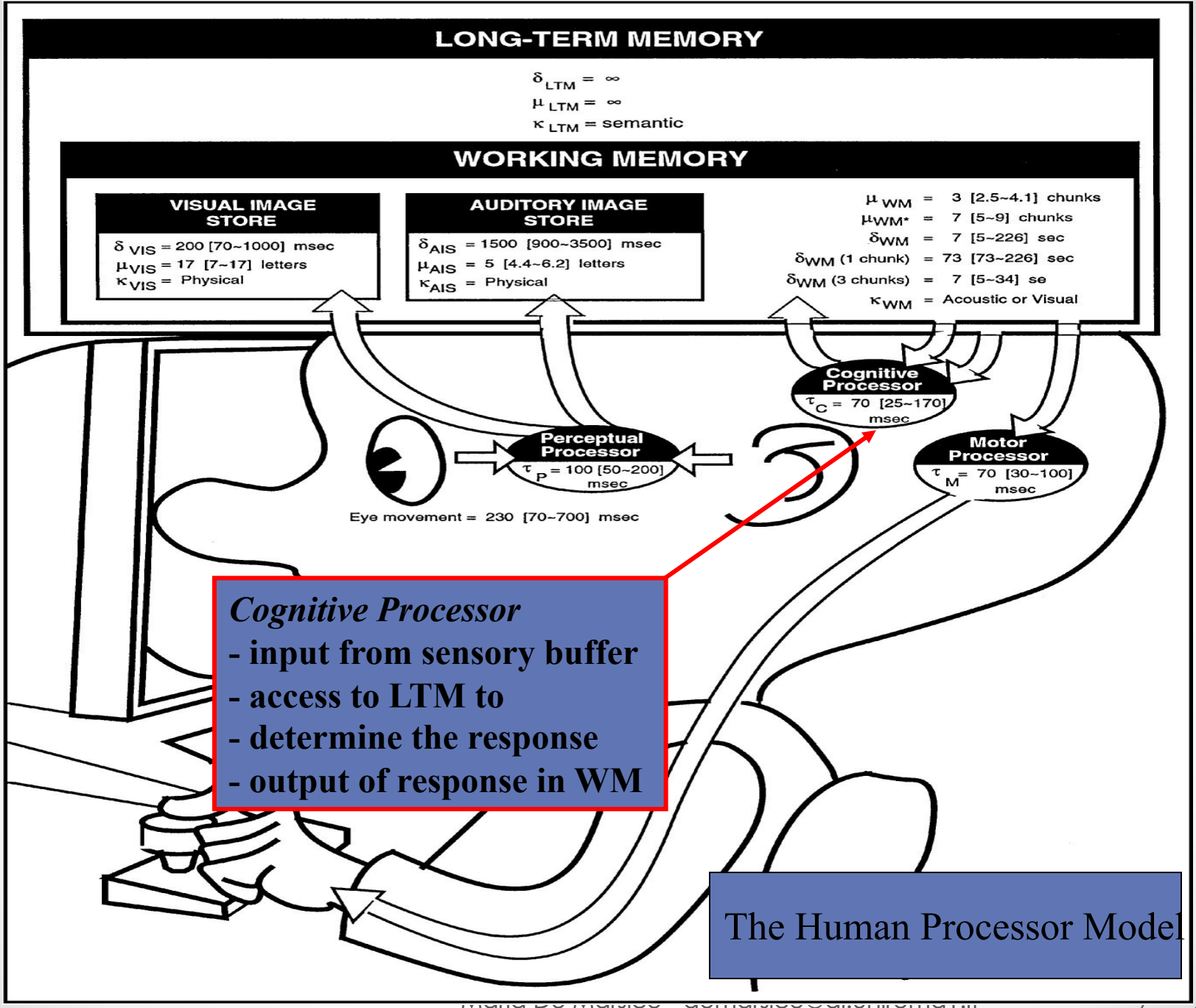
Cognitive modeling: human processor

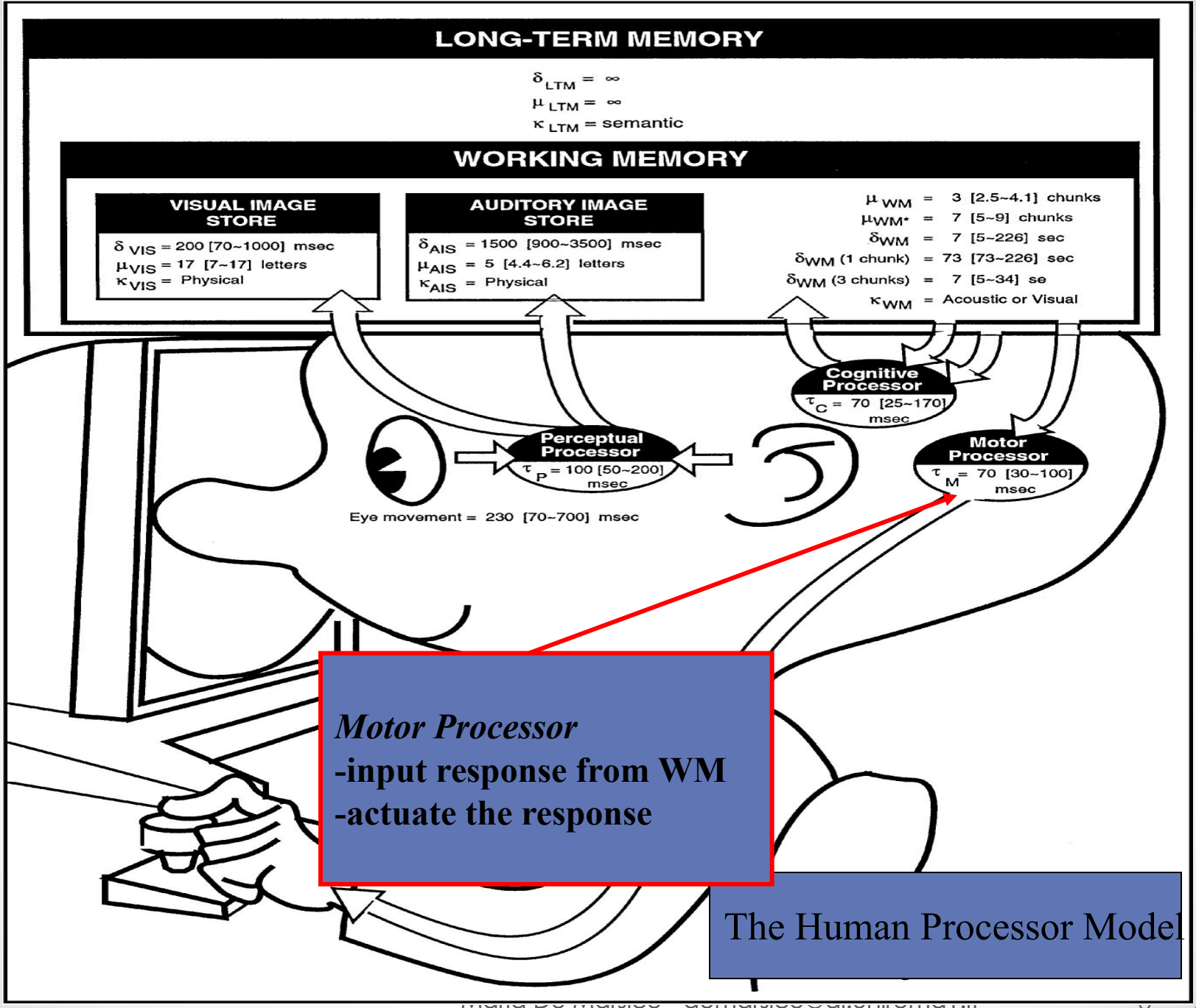




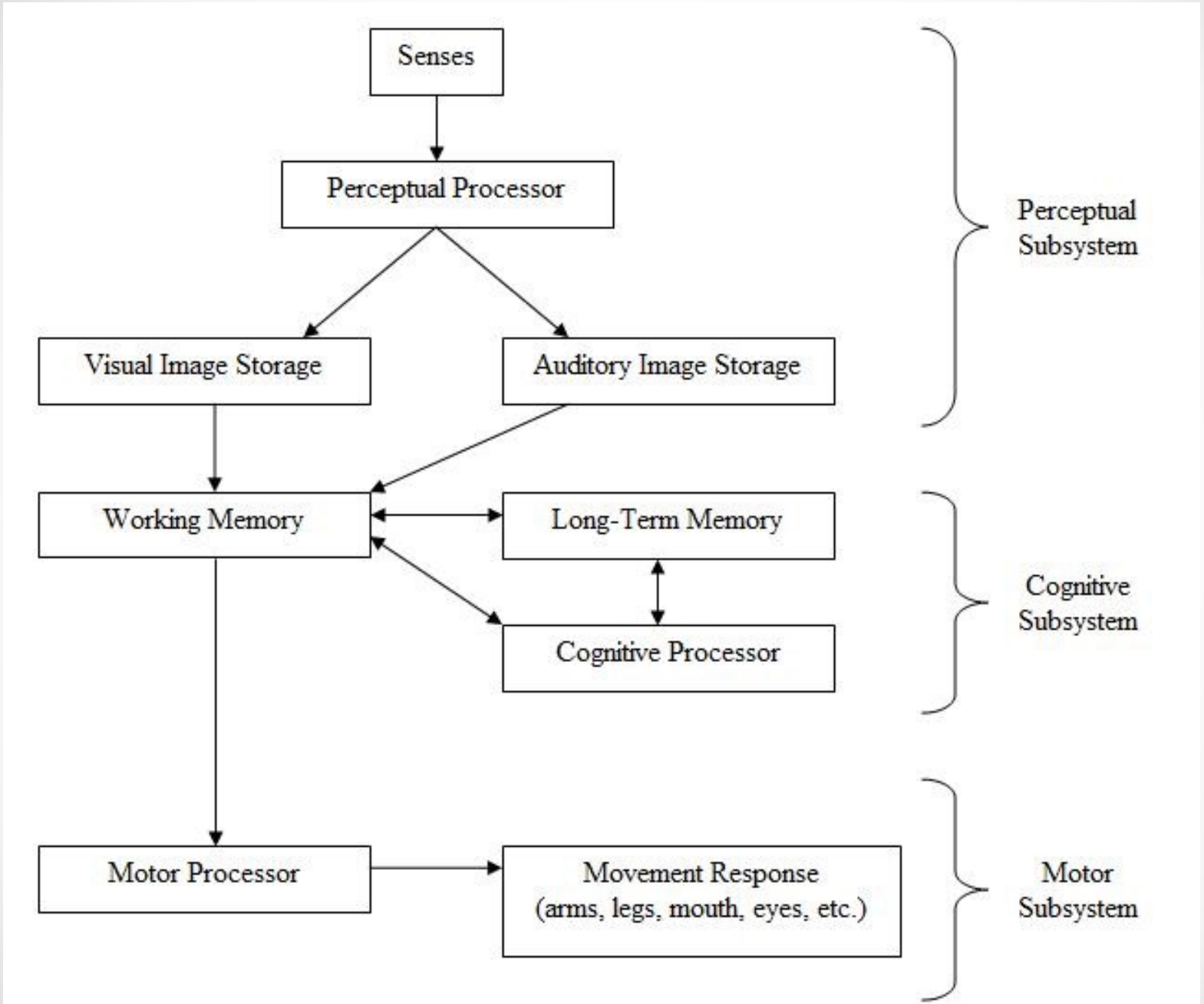
Perceptual Processor
-sensory input (audio & visual)
-symbolic coding of info
-output towards memory of audio and visual images (WM buffer)

The Human Processor Model


LONG-TERM MEMORY
δ LTM = ∞
μ LTM = ∞
κ LTM = semantic
WORKING MEMORY
VISUAL IMAGE STORE
δ VIS = 200 [70~1000] msec
μ VIS = 17 [7~17] letters
κ VIS = Physical
AUDITORY IMAGE STORE
δ AIS = 1500 [900~3500] msec
μ AIS = 5 [4.4~6.2] letters
κ AIS = Physical
μ WM = 3 [2.5~4.1] chunks
μ WM* = 7 [5~9] chunks
δ WM = 7 [5~226] sec
δ WM (1 chunk) = 73 [73~226] sec
δ WM (3 chunks) = 7 [5~34] se
κ WM = Acoustic or Visual
Cognitive Processor
τ C = 70 [25~170] msec
Perceptual Processor
τ P = 100 [50~200] msec
Motor Processor
τ M = 70 [30~100] msec
Eye movement = 230 [70~700] msec
Cognitive Processor
- input from sensory buffer
- access to LTM to
- determine the response
- output of response in WM
The Human Processor Model


LONG-TERM MEMORY
δ LTM = ∞
μ LTM = ∞
κ LTM = semantic
WORKING MEMORY
VISUAL IMAGE STORE
δ VIS = 200 [70~1000] msec
μ VIS = 17 [7~17] letters
κ VIS = Physical
AUDITORY IMAGE STORE
δ AIS = 1500 [900~3500] msec
μ AIS = 5 [4.4~6.2] letters
κ AIS = Physical
μ WM = 3 [2.5~4.1] chunks
μ WM* = 7 [5~9] chunks
δ WM = 7 [5~226] sec
δ WM (1 chunk) = 73 [73~226] sec
δ WM (3 chunks) = 7 [5~34] se
κ WM = Acoustic or Visual
Cognitive Processor
τ C = 70 [25~170] msec
Perceptual Processor
τ P = 100 [50~200] msec
Motor Processor
τ M = 70 [30~100] msec
Eye movement = 230 [70~700] msec
Motor Processor
-input response from WM
-actuate the response
The Human Processor Model

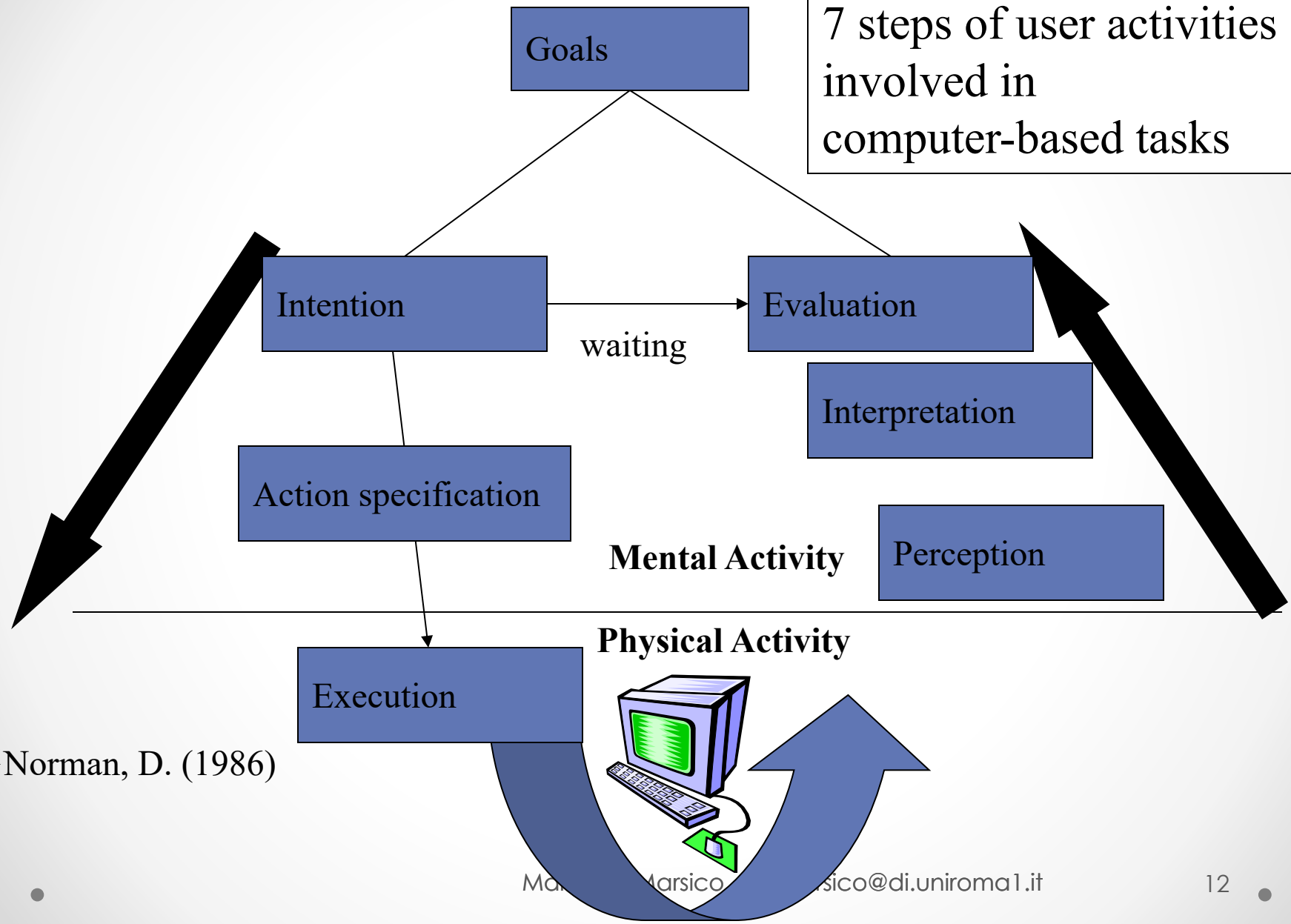
Parameter	Mean	Range
Eye movement time	230 ms	70-700 ms
Decay half-life of visual image storage	200 ms	90-1000 ms
Visual Capacity	17 letters	7-17 letters
Decay half-life of auditory storage	1500 ms	90-3500 ms
Auditory Capacity	5 letters	4.4-6.2 letters
Perceptual processor cycle time	100 ms	50-200 ms
Cognitive processor cycle time	70 ms	25-170 ms
Motor processor cycle time	70 ms	30-100 ms
Effective working memory capacity	7 chunks	5-9 chunks
Pure working memory capacity	3 chunks	2.5-4.2 chunks
Decay half-life of working memory	7 sec	5-226 sec
Decay half-life of 1 chunk working memory	73 sec	73-226 sec
Decay half-life of 3 chunks working memory	7 sec	5-34 sec


Senses
Perceptual Processor
Visual Image Storage
Auditory Image Storage
Working Memory
Long-Term Memory
Cognitive Processor
Motor Processor
Movement Response
(arms, legs, mouth, eyes, etc.)
Perceptual Subsystem
Cognitive Subsystem
Motor Subsystem

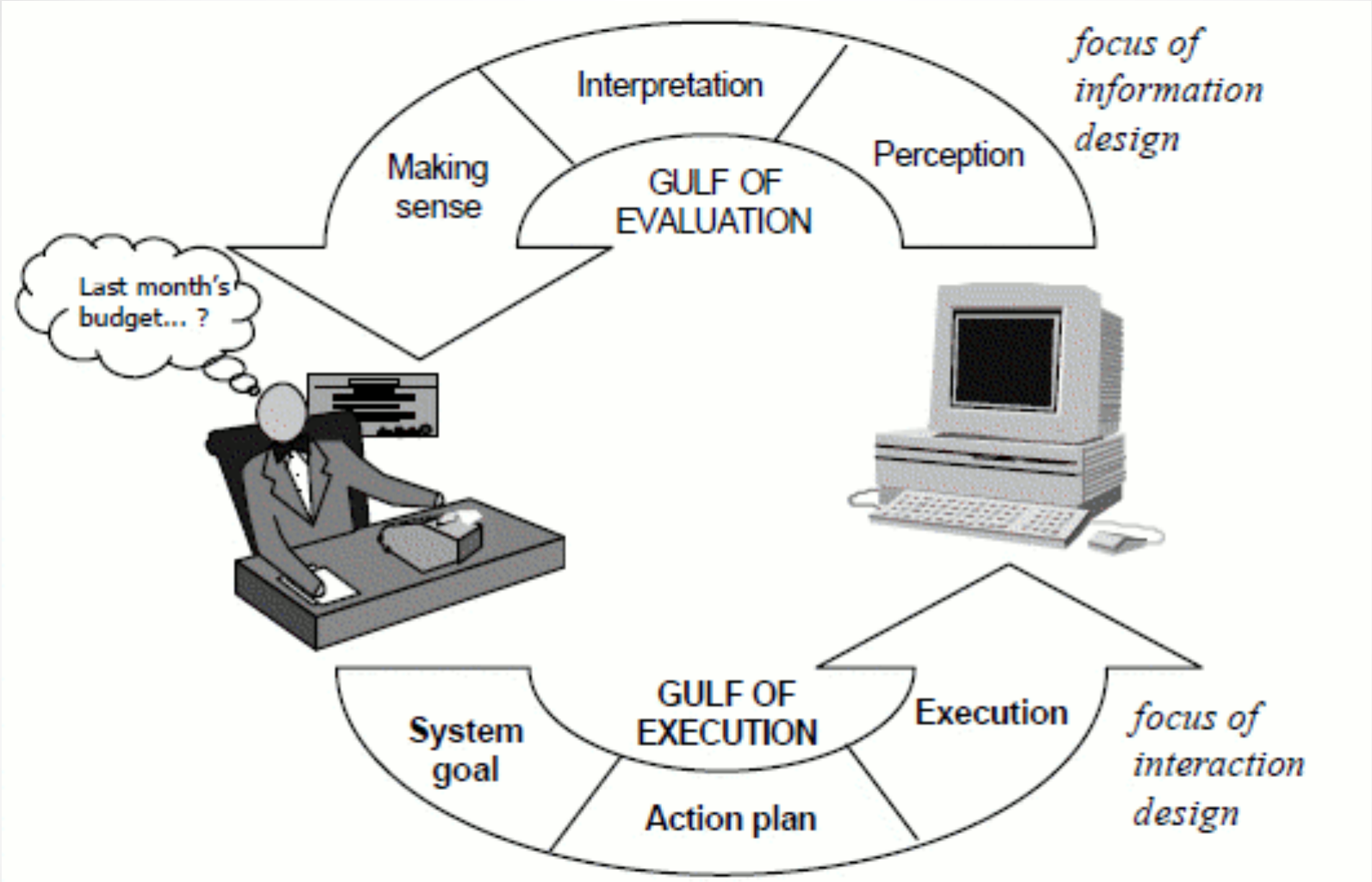
Norman Stages of action

- We start with a **goal**
  - Real tasks are defined in an imprecise way
- We must do something: **execution**
  - **intentions:** low level statements to define what we need to do
  - **sequence of actions**
  - **execution**
- We verify il goal was achieved
  - **perception** of what happened
  - **interpretation** of what happened
  - **evaluation** of correspondence with goals

7 steps of user activities involved in computer-based tasks



[1]Norman, D. (1986)


focus of information design
Interpretation
Perception
Making sense
GULF OF EVALUATION
Last month's budget... ?
GULF OF EXECUTION
Execution
System goal
Action plan
focus of interaction design

WHAT DO THE GULFS MEAN?

Let's look at an interesting blog page on the Nielsen Normal Group page https://www.nngroup.com/

The Two UX Gulfs: Evaluation and Execution

Kathryn Whitenton
March 11, 2018

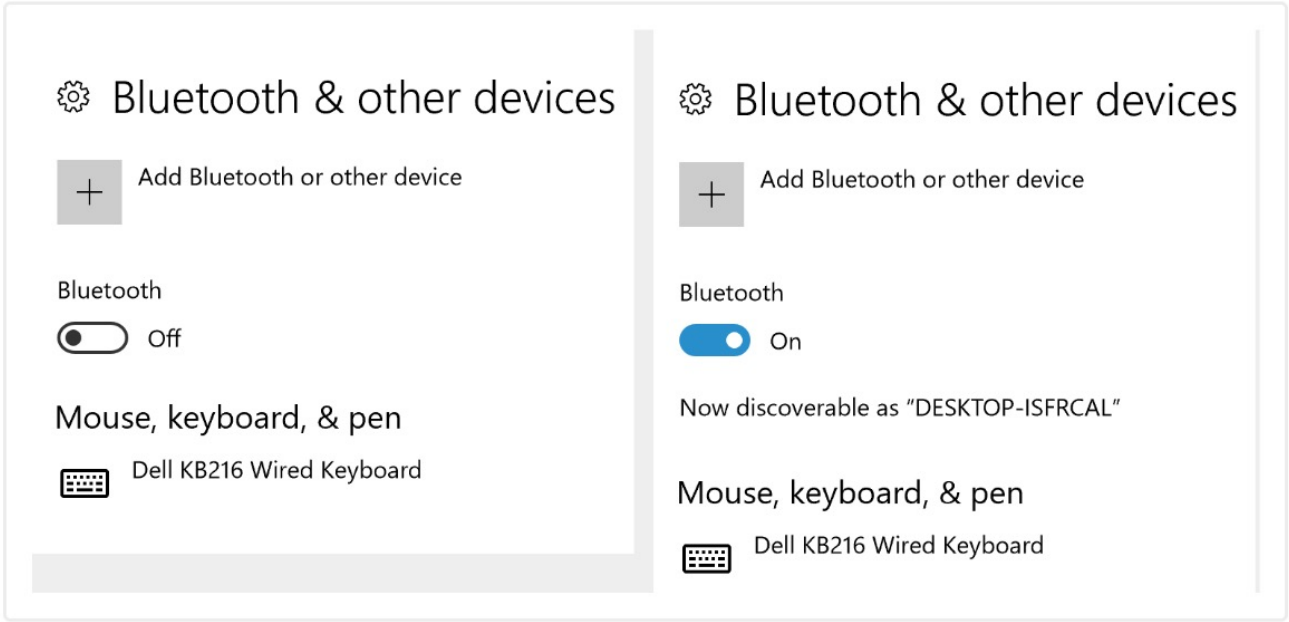
Share

Summary: With every interaction, users must overcome the twin challenges of understanding the current state of a system and figuring out how to change it. Designers can support them by being aware of these gulfs and bridging them with a transparent conceptual model.

Last week, I bought a great new Bluetooth headset — then, sadly, proceeded to spend over an hour on a frustrating quest to connect it to my computer. Despite the promises made by both the headset manual and the computer-support site, the headset did not automatically connect to the computer. After re-reading the instructions, checking to make sure both devices were on, scouring the headset reviews to confirm it was compatible with my computer, and even testing the headset by syncing it to a different computer, I was just about ready to give up and return the darn thing when I found a random help page (from a completely different hardware manufacturer), which changed everything.

https://www.nngroup.com/articles/two-ux-gulfs-evaluation-execution/

This new help page showed an actual screenshot of what the Windows 10 Bluetooth settings switch looks like when it is turned on. Looking at this screenshot, I instantly realized my mistake: the Bluetooth settings screen I'd been staring at for the past hour was actually showing me that Bluetooth was off on my computer.



This Bluetooth-connectivity switch has a circle that can be shifted from side to side to change the state to Off *(left) or* On *(right). The label reflects the current state of the control, not what will happen when the switch will be moved to the right. (Unlike the label for the plus icon, which explains what will happen when clicking it.)*

I felt pretty dumb when I realized my mistake. But my failure to understand the current state of the device is actually an extremely common usability problem — so common that visibility of system status is the very first in Jakob Nielsen's famous 10 Usability Heuristics.

Looking at this design, it's clear that the creators knew it was important to make the status visible: they even included an explicit status label, *Off*, next to the control switch.

So what went wrong? To understand, we need to dig deeper.

In This Article:

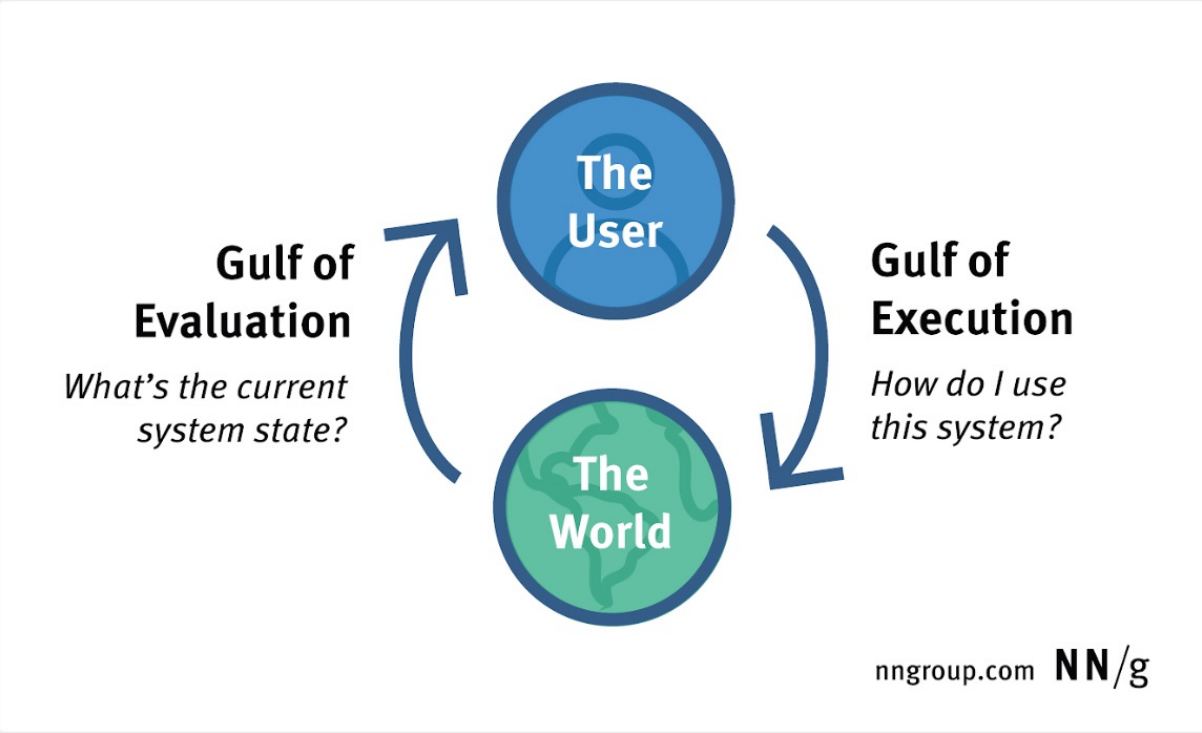
The Gulf of Evaluation and the Gulf of Execution

Two of the many challenges people must overcome to successfully interact with technology are:

- Evaluation: Understanding the state of the system
- Execution: Taking action to accomplish a specific goal

These challenges are described as the "gulf of evaluation" and the "gulf of execution" because, without effective design elements to support users, they can become insurmountable barriers between users and their goals.

The terms *gulf of evaluation* and *gulf of execution* were invented in 1986 by Ed Hutchins, Jim Hollan, and Don Norman, when they wrote about the virtues of direct manipulation in helping users bridge these gulfs. (Their work was published in the book "User Centered System Design," edited by Norman and Draper, which was the first use of the term 'user-centered design', long before Don joined Jakob Nielsen to establish the Nielsen Norman Group). Don's book, "The Design of Everyday Things," tells the story of the gulfs and details their importance in the design process. Thirty years later, the two gulfs are still essential concepts in our field.



The gulf of evaluation and the gulf of execution describe two major challenges that users must overcome to successfully interact with any device. As indicated by the figure, users are stuck in an (almost) endless cycle, alternating these two issues: after you execute, you must evaluate the outcome, plan to execute your next step, evaluate that outcome, and so on, until you reach your desired end-goal and quit.

Norman's cycle revisited

- In the case of multimodal interactive cycles, Norman's model of interaction may be reformulated as follows:

1. Establishing the goal.
2. Forming the intention.
3. Specifying the **multimodal** action sequence in terms of **human output modalities**.
4. Executing the **multimodal** action.
5. Perceiving the system state in terms of **human input modalities**.
6. Interpreting the system state.
7. Evaluating the system state with respect to the goals and the intentions.

From goal to intention

- **Establishing the goal**: is, as usual, the stage when the user determines what needs to be done in the given domain, in terms of a **suitable task language**

- **Forming the intention**: at this stage the goal is translated into more precise user's intention, which will help the user determining the right sequence of actions that should be performed to achieve the goal

Action plan

Specifying the multimodal action sequence: the sequence of actions performed to accomplish the required task should be precisely stated at this stage.

Complexity of multimodal interaction appears for the first time in the cycle. Each multimodal action can be specified in terms of:

1. ***Complementary human output sensory modalities*** (i.e., multiple utterances at once form the action)

and/or

2. ***Alternative human output sensory modalities*** (i.e., alternative, redundant utterances for the same action).

Some **unintentional** utterances: blood pressure, temperature, heartbeat, excretion, etc.

A user may move an object in the interaction scene by speaking and pointing at (gesturing) the new object location (**complementary** modalities).

Then, instead of gesturing (s)he may want to gaze at the new location on the interface where the object should be moved (**alternative** modality).

Execution

Executing each multimodal action: at this stage, each human modality used to specify an action is translated into corresponding interaction modes.

Each action is executed through

1. ***Complementary modes***

or

2. ***Alternative modes***

Text, speech, Braille, mimicking, eye/motion capture, haptics, bio-electrical sensoring are examples of modes used to translate human output modalities into the system input language.

When the execution of the whole sequence of multimodal actions is complete, the system reaches a new state and communicates it to the user again exploiting (possibly multiple) interaction modes, such as speech synthesis, display, haptic/tactile feedback, smell rendering and so on.

Perception

Perceiving the system state: at this stage, the evaluation phase of the cycle begins.

Depending on the combination of system output modes, the user may perceive the new state through multiple input sensory modalities, such as visual, auditory, tactile, and (in some revolutionary interfaces) even smelling and tasting.

Interpretation and Evaluation

- **Interpreting the system state**: here the user is supposed to interpret the output of her/his sequence of actions to evaluate what has happened.

- **Evaluating the system state with respect to the goals and the intentions**: at the final stage, the user compares the new system state with her/his expectations, to evaluate if the initial goal has been effectively reached.

Bridging the gulfs with mental models

- **In multimodal interaction the two gulfs may even worsen, unless the communication over the different channels is accurately designed to avoid any conflict or ambiguity.**

- Form the same blog page:
- «Interpretation requires effort, and most people try to minimize this effort by relying on a mental model to understand a system.
  A **mental model** is a theory of how a system works, what its signals mean, and what the outcomes of different user actions will be. To save time, most people rely on their past experiences to quickly build mental models for new systems.»

- Interesting to continue reading!

Goal hierarchies

- Granularity
  - Where to start from?
  - Where to stop?
- Routine tasks vs. problem solving
  - Unit tasks
- Conflict
  - Different ways to reach a goal
- Errors

Tecniques

- Goals, Operators, Methods and Selection (GOMS) [Card, Moran, Newell 1983]
- Cognitive Complexity Theory (CCT) [Kieras & Polson 1990 (basata su GOMS)]
- Hierarchical Task Analysis (HTA) [Shepherd tardi anni '60]

GOMS

Goals
- what the user wants to obtain

Operators
- base actions executed by the user

Methods
- decompose goals in sub-goals / operators

Selection
- ways to choose among alternative methods

Example GOMS

```
GOAL: CLOSE-WINDOW
.   [select GOAL: USE-MENU-METHOD
            .   MOVE-MOUSE-TO-FILE-MENU
            .   PULL-DOWN-FILE-MENU
            .   CLICK-OVER-CLOSE-OPTION
            GOAL: USE-CTRL-W-METHOD
            .   PRESS-CONTROL-W-KEYS]

For a particular user:

   Rule 1: Select USE-MENU-METHOD unless another
           rule applies
   Rule 2: If the application is GAME,
           select CTRL-W-METHOD
```

Action Language

Reisner (1983) "Psychological BNF"

- The analysis is focused on user's behaviour and communication
- Attention for human knowledge and knowledge sources

Example of Action Language

Get web document date ::= USE MUSCLE MEMORY |
(<retrieve syntax info> + <inspect date>)

<retrieve syntax info> ::= <retrieve from human memory> |
<retrieve from external source>

<retrieve from human memory> ::= RETRIEVE FROM LTM |
RETRIEVE FROM WM

<retrieve from external source> ::= RETRIEVE FROM BOOK |
ASK SOMEONE | EXPERIMENT | USE ON-LINE HELP

<inspect date> ::= <open document source window> +
SCAN WINDOW FOR DATE + <delete window>

<open document source window> ::=PRESS RIGHT MOUSE BUTTON +
POINTER TO DOCUMENT INFO + RELEASE MOUSE BUTTON

<delete window> ::= .....

Cognitive complexity theory (CCT)

Kieras & Polson 1990 (based on GOMS)

- Based on a model of information processing by human user
- Production rules :
  - Learned and stored in Long Term Memory (LTM)
  - Used during interaction with computer, through processes activated from Working Memory (WM)

Cognitive Complexity Theory (CCT)

- Two parallel descriptions:
  - User production Rules
  - Generalised Transition Networks for the device
- Production rules :
  - if condition then action

Cognitive complexity theory (CCT)

Production rules in in LTM:

if (condition) then (action)

- condition: test on the content of WM
- action: transforms the content of WM towards an activity

If more rules can be alpplied: priority rule

Complexity:

- number of production rules in LTM: index of *learnability*
- number of cycles in WM: index of *ease* of use
- number of shared productions for 2 systems: index of *transferrability*

Example: editing with vi

- Production rules in LTM
- WM modeled as attribute-value associations:

  (GOAL perform unit task)

  (TEXT task is insert space)

  (TEXT task is at 5 23)

  (CURSOR 8 7)
- Rules matched with WM

  LOOK-TEXT task is at %LINE %COLUMN
  is true, with LINE = 5 COLUMN = 23.

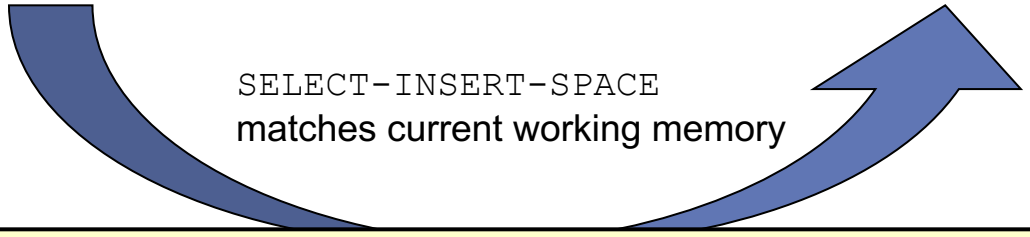
Modeling insertion of a space

Active rules:

```
SELECT-INSERT-SPACE
INSERT-SPACE-MOVE-FIRST
INSERT-SPACE-DOIT
INSERT-SPACE-DONE
```

New working memory

```
(GOAL insert space)
(NOTE executing insert space)
(LINE 5) (COLUMN 23)
```




```
(SELECT-INSERT-SPACE
IF (AND (TEST-GOAL perform unit task)
        (TEST-TEXT task is insert space)
        (NOT (TEST-GOAL insert space))
        (NOT (TEST-NOTE executing insert space)))
THEN (  (ADD-GOAL insert space)
        (ADD-NOTE executing insert space)
        (LOOK-TEXT task is at %LINE %COLUMN)))
```

Notes on CCT

- Parallel Model
- Actions are expressed as procedures
- Different rules for experts and novices
- Possible to represent error
- Measures
  - Depth of goal structure
  - Number of di rules
  - Matching with device description

Problems with goal hyerarchies

- Post hoc technique
- Experts versus novices
- How cognitive they are?

Linguistic Notations

- Try to understand user's behaviour and cognitive difficulties according to the analysis of the language between **user** and **system**
- Backus–Naur Form (BNF)
- Task–Action Grammar (TAG)

Backus-Naur Form (BNF)

- Computer Science Notation
- Dialog is considered from a purely syntactic point of view
- Terminals(symbols)
  - lower level of user behaviour
    e.g. CLICK-MOUSE, MOVE-MOUSE
- Non-terminals
  - terminal ordering
  - higher level of abstraction
    e.g. select-menu, position-mouse

Esempio di BNF

- Basic syntax:
  - nonterminal ::= expression
- Expression
  - Includes both terminals and non-terminals
  - Combined in sequence (+) or as alternatives (|)

draw line ::= select line + choose points + last point
select line ::= pos mouse + CLICK MOUSE
choose points ::= choose one | choose one + choose points
choose one ::= pos mouse + CLICK MOUSE
last point ::= pos mouse + DBL CLICK MOUSE
pos mouse ::= NULL | MOVE MOUSE+ pos mouse

Measures for BNF

- Number of rules
- Number of + e | operators

- Problems
  - same syntax for different semantics
  - no reflections on perception
  - low consistency control

Task Action Grammar (TAG)

- Consistence made explicit
- Code user knowledge
- Grammar rules are **parametrized**
- Non-terminals are modified to include **semantics**

Consistency in TAG

- In BNF, 3 UNIX commands are described as :

  copy ::= cp + filename + filename | cp + filenames + directory

  move ::= mv + filename + filename | mv + filenames + directory

  link ::= ln + filename + filename | ln + filenames + directory

- No measure would distinguish a less consistent grammar where :

  link::= ln + filename + filename | ln + directory + filenames

Consistency in TAG

- Consistency of the argument made explicit using **parameters**
- **Possible** parameter values

  Op = copy; move; link

- Rules

  file-op[Op] ::= command[Op] + filename + filename
  | command[Op] + filenames + directory
  command[Op = copy] ::= cp
  command[Op = move] ::= mv
  command[Op = link] ::= ln

Modified non-terminals

Task Action Grammar

List of commands **possible values**

- move cursor one character forward ctrl-C
- move cursor one character backward meta-C
- move cursor one word forward ctrl-W
- move cursor one word backward meta-W

List of features

- direction forward, backward
- unit character, word

dictionary of simple tasks

- move cursor one character forward {direction=forward, unit=char}
- move cursor one character backward {direction=backward, unit=char}
- move cursor one word forward {direction=forward, unit=word}
- move cursor word backward {direction=backward, unit=word}

rule schemas

task [direction, unit] → symbol [direction] + letter [unit]

symbol [direction = forward] → "ctrl"

symbol [direction = backward] → "meta"

letter [unit = word] → "W"

letter [unit =character] → "C"

Task Analysis

Methods per analyze user **tasks** :

- **What** they do
- **What** they are working **with**
- **What** they have to **know**

Example

- To clean house
  - get the vacuum cleaner out
  - fix the appropriate attachments
  - clean the rooms
  - when the dust bag gets full, empty it
  - put the vacuum cleaner and tools away
- Need to know about:
  - vacuum cleaners, their attachments, dust bags, cupboards, rooms etc.

Approaches to task analysis

- Task decomposition
  - Sub-tasks ordered both temporally and casually
- Knowledge-based techniques
  - User knowledge about task and its organization e
- Analysis based on entitis/objects
  - Relations among objects, actions, and users performing them

General method

- To observe
- To gather lists of keywords and actions
- To organize them using notations and/or diagrams

Decomposition into tasks

Aim:

- To describe the actions performed by users that have to interact with the system
- Structure them in a hierarchy of tasks and sub-tasks
- To describe the order of sub-tasks

Variations:

- Hierarchical Task Analysis (HTA)
  - more widespread
- CTT (CNUCE, Pisa)
  - uses temporal operators from LOTOS

HTA
Textual description

Description of the hierarchy...

0. to clean house
    1. get the vacuum cleaner out
    2. fix the appropriate attachments
    3. clean the rooms
        3.1. clean the living room
        3.2. clean the bedrooms
        3.3. clean the bathrooms
    4. empty the dust bag
    5. put the vacuum cleaner and tools away

... and of plans

Plan 0: execute 1 - 2 - 3 - 5 in order, when the dust bag gets full execute 4

Plan 3: execute 3.1, 3.2 or 3.3 in any order, according to the rooms to clean

N.B. only plans denote order

How to generate the hierarchy

1. Get the list of tasks
2. Group tasks in higher level tasks
3. Further decompose lower level tasks

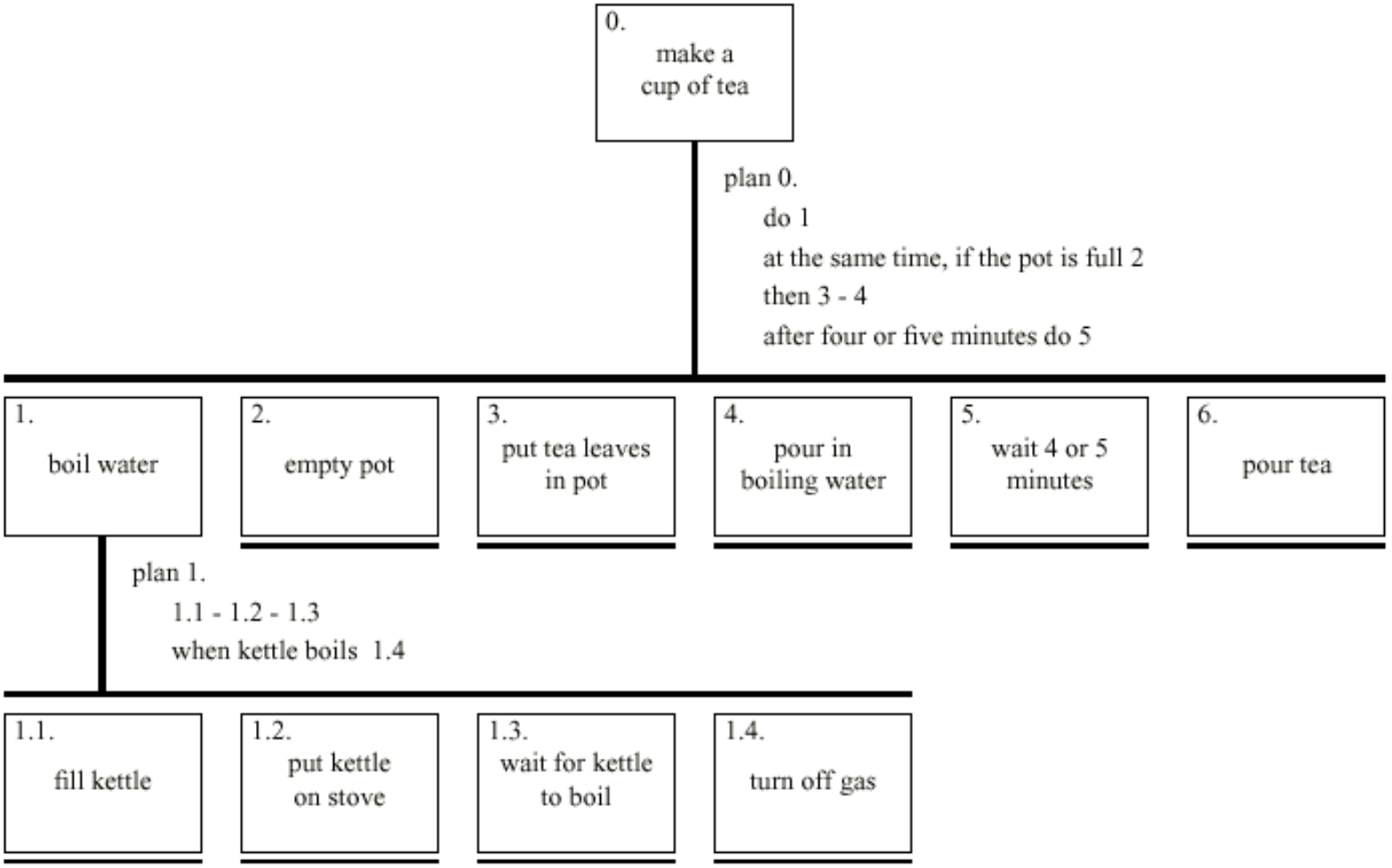
Stop rules

- How to know when to stop?
- A task like "empty the dust bag" is easy enough?
- Aim: to expand only relevant tasks
- Motor actions : lower reasonable level

Note

- Asking:
  what are you doing?
- One could get answers:
  I'm typing ctrl-B
  I'm making a word become bold
  I'm emphasizing a word
  I'm editing a document
  I'm writing a letter
  I'm working on a legal practice

HTA
Diagrammatic description



Description refinement

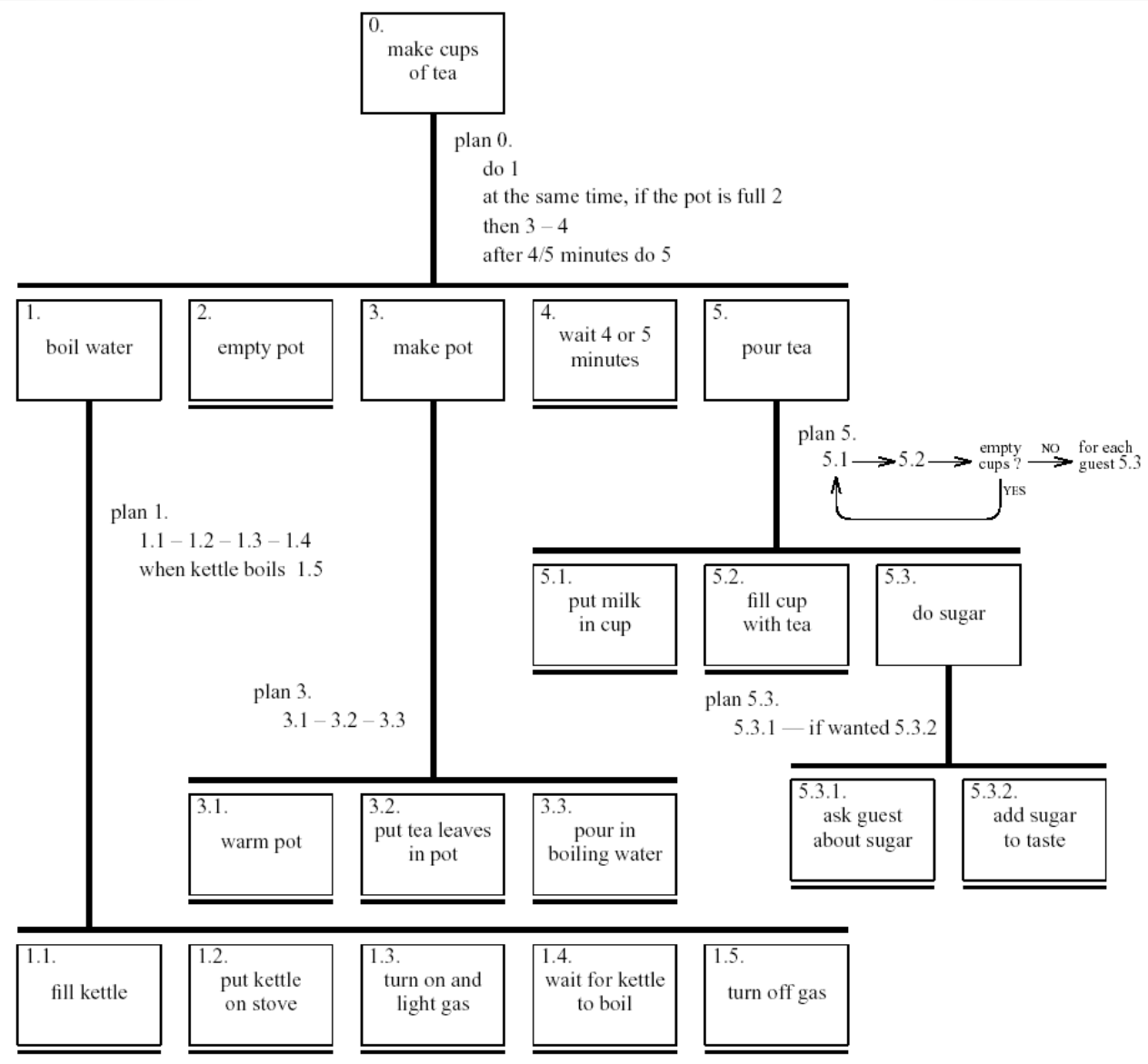
Given a HTA (textual or diagrammatic)

How to verify/improve it?

Euristiche:

Paired actions	e.g., `turn on gas‘ is missing
Restructure	e.g., generate task`make pot'
Equilibrate	e.g.`pour tea' simpler than “make pot”?
Generalize	e.g., “make a cup ….. or more”

Diagrammatic HTA refined

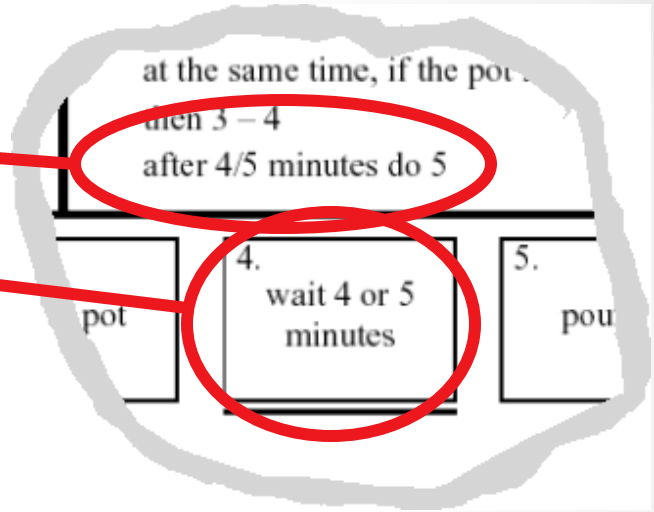


Kinds of plan

Fixed sequence	-	1.1 then 1.2 then 1.3
Optional tasks	-	if the pot is full 2
Event wait	-	when kettle boils 1.4
Cycles	-	do 5.1 5.2 while there are still empty cups
Concurrent	-	do 1; at the same time ...
Discretionary	-	do any of 3.1, 3.2 or 3.3 in any order
Mix	-	most plans involve several of the above

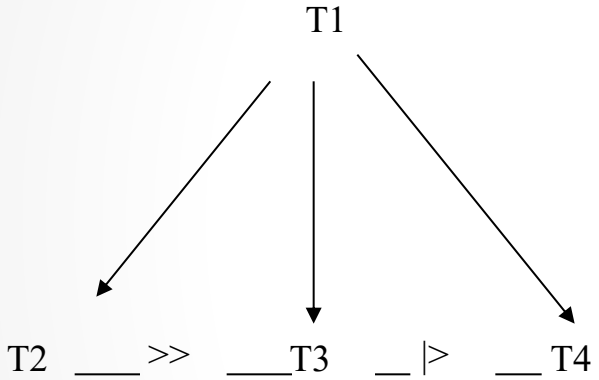
Waiting forms…

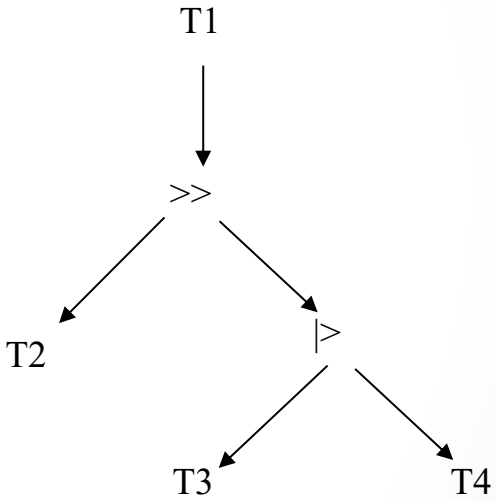
- Part of a plan?
  … or of a task?
- In general
  - Of task – if 'busy' wait
    - One actively waits
  - Of plan – if the end of the delay is an event
    - e.g. "when alarm rings", "when reply arrives"



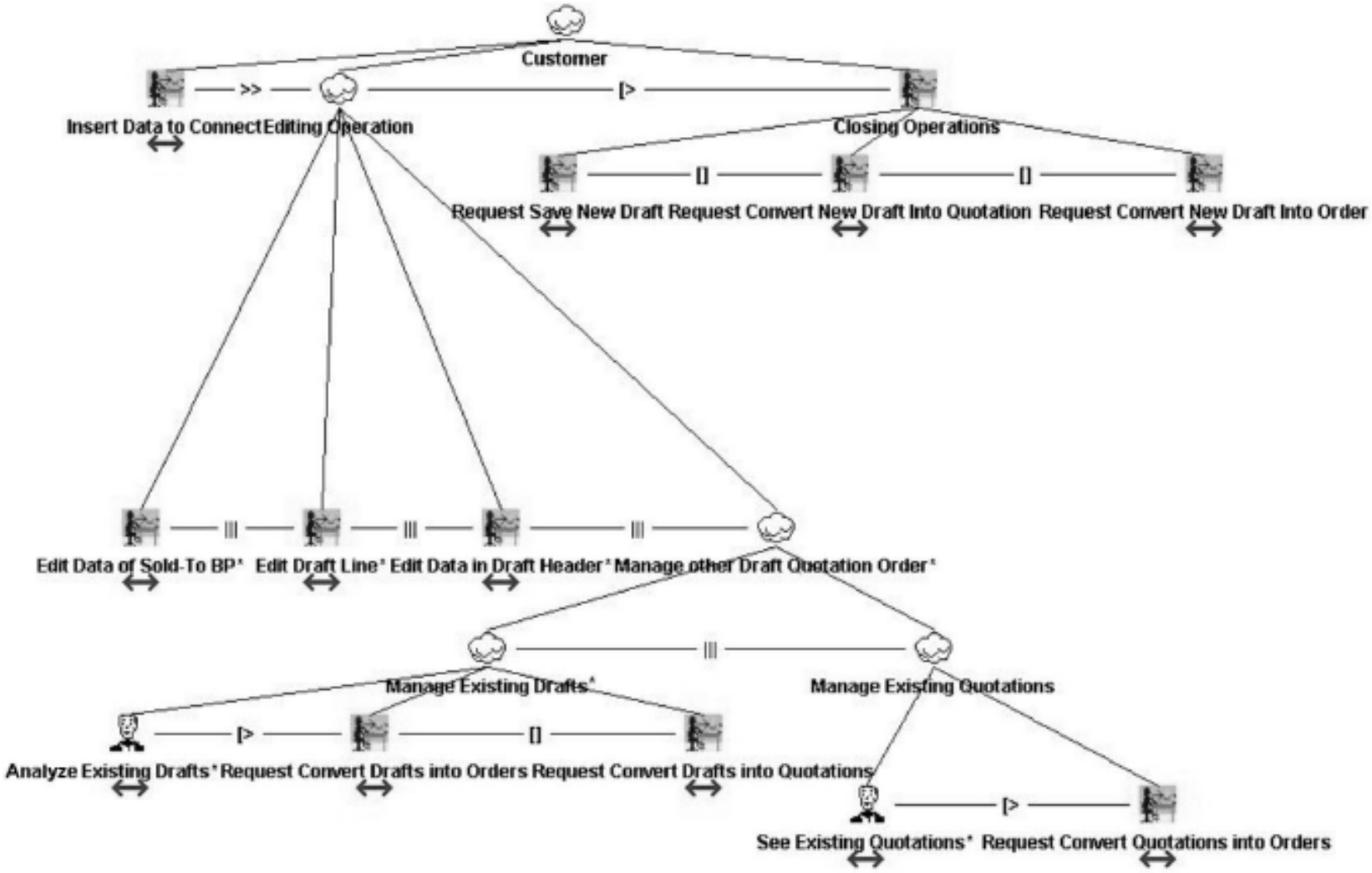
CTT

- Logical-temporal operators

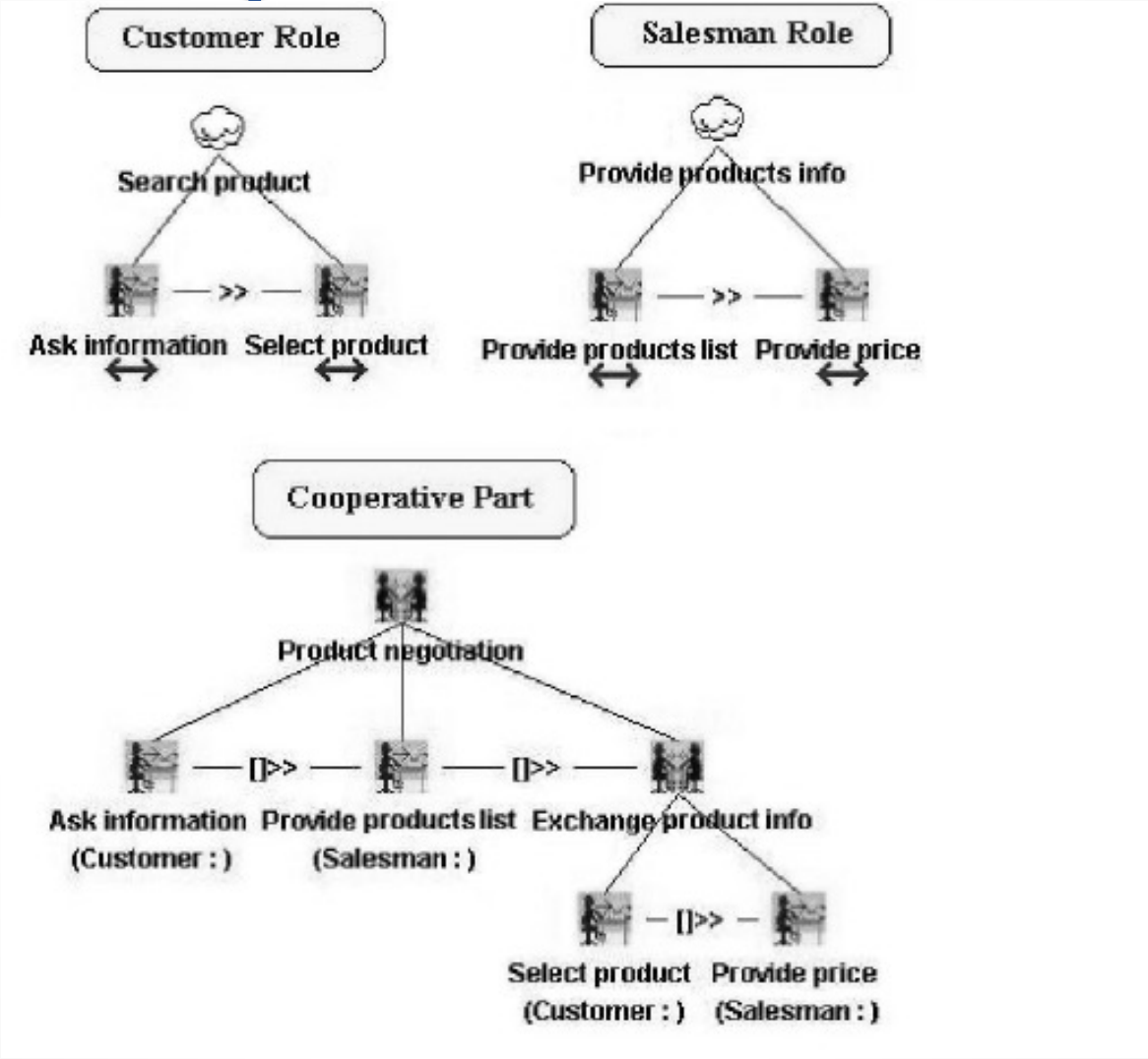




A complex task



Cooperative tasks



Operators I

Independent Concurrency (T1 ||| T2)	Actions belonging to two tasks can be performed in any order without any specific constraints, for example monitoring a screen and speaking in a microphone;
Choice (T1 [] T2)	It is possible to choose from a set of tasks and, once the choice has been made the task chosen can be performed and other tasks are not available at least until it has been terminated. This operator is useful in the design of user interfaces because it is often important to enable the user to choose from various tasks. An example is, at the beginning of a word processor session when it is possible to choose whether to open an existing file or a new one. Also the system can choose to perform one task from a set of application tasks depending on its current state.
Concurrency with information exchange (T1 |[]| T2)	Two tasks can be executed concurrently but they have to synchronise in order to exchange information. For example, a word processor application where editing a file and scrolling its contents can be performed in any order and they exchange information when they are performed because it is possible to edit only the information that the scrolling has made visible.

Deactivation (T1 [> T2)	The first task is definitively deactivated once the first action of the second task has been performed. This concept is often used in many user interface implementations when the user can deactivate the option of performing a set of tasks and enable a new set of possible task accomplishments by a specific action (for example by selecting a button).
Enabling (T1 >> T2)	In this case one task enables a second one when it terminates, for example, a database where users have first to register and then they can interact with the data.
Enabling with information passing (T1 []>> T2)	In this case task T1 provides some information to task T2 other than enabling it. For example, T1 allows the user to specify a query and T2 provides the query result that obviously depends on the information generated by T1.
Suspend-resume (T1 |> T2)	This operator gives T2 the possibility of interrupting T1 and then when T2 is terminated, T1 can be reactivated from the state reached before the interruption. For example, the editing text task which, in some applications can be suspended by a modal printing task, and once the printing task is accomplished then editing can be carried on from the state reached beforehand. For example, this operator can be used to model a type of interruption.

Operatori II

Iteration T*	In the tasks specification we can have some tasks with the * symbol next to their name. This means that the tasks are performed repetitively: when they terminate, the performance of their actions automatically starts to be executed again from the beginning. This continues until the task is deactivated by another task.
Finite Iteration (T1(n))	It is used when designers know in advance how many times a task will be performed.
Optional tasks ([T])	They give the possibility of indicating that the performance of a task is optional. Optional tasks are indicated in square brackets. For example, we have optional tasks when we fill a form in and there are some fields that are mandatory and others optional.
Recursion	This means that in the subtree originated by the task considered there is another occurrence of it. This possibility is used, for example, with tasks that, for each recursion, allows performance of the recursive tasks with the additional possibility of performing some new tasks, until a task interrupting the recursion is performed.

Readings

Norman D. The design of everyday things. Doubleday, New York, 1988.

http://www.interaction-design.org/

http://www.demystifyingusability.com/2007/06/multimodal_desi.html

Sharon Oviatt. Ten myths of multimodal interaction. http://www.kevinli.net/courses/mobilehci_w2012/papers/p74-oviatt.pdf