

## 2.2 Modèle de tâche

La connaissance de l'utilisateur permet d'analyser son **activité** (ce qu'il fait indépendamment du système).

Faire rédiger des **scénarios** par des utilisateurs représentatifs (ou à défaut soi-même) permet d'identifier ces activités.

Il faut ensuite les organiser en un **modèle de tâches**.

## 2.2 Modèle de tâche

Une **tâche** consiste en :

- un **but** (état souhaité) ; et
- une **procédure** pour atteindre ce but.

Une **procédure** est un ensemble de sous-tâches liées par :

- des relations de **composition** ; et
- des relations **temporelles**.

Une **tâche élémentaire** est une tâche décomposable en **actions physiques**.

Une **action physique** est une opération sur un dispositif d'entrée/sortie qui provoque un changement d'état du dispositif (clic, mouvement, affichage, etc.)

## 2.2 Modèle de tâche

Les modèles de tâche sont des **structures arborescentes** dont **les nœuds sont les buts** et **les sous-arbres sont les procédures** pour atteindre ces buts.

Les nœuds peuvent être **décorés** par :

- les **concepts du domaines** (objets référencés, i.e. paramètres, variables globales ...) ;
- les **préconditions** et les **postconditions** ;
- la **fréquence** ;
- la **complexité** ;
- la **criticité** (niveau de danger, caractère irrévocable) ;
- les **contraintes temporelles** (durée maximale) ;
- l'**acteur** responsable de l'exécution de la tâche (utilisateur et/ou système) ;
- toute autre information pertinente (selon le domaine).

## 2.2 Modèle de tâche

Il existe de nombreuses notations pour les modèles de tâche, par exemple UAN, **CTT**, **HTA**, MAD ...  
Les *use cases* et les diagrammes de séquences d'UML peuvent aussi être utilisés.

## 2.2 Modèle de tâche

**UAN**, (User Action Notation) utilise des opérateurs pour dénoter les actions élémentaires :

|              |                                                                      |
|--------------|----------------------------------------------------------------------|
| ~            | move the cursor                                                      |
| [X]          | the context of object X, the "handle" by which X is manipulated      |
| ~[X]         | move cursor into context of object X                                 |
| ~[x, y]      | move the cursor to (arbitrary) point x, y outside any object         |
| ~[x, y in A] | move the cursor to (arbitrary) point within object A                 |
| ~[X in Y]    | move to object X within object Y (e.g., [OK__icon in dialogue__box]) |
| [X]~         | move cursor out of context of object X                               |
| ▼            | depress                                                              |
| ^            | release                                                              |
| X▼           | depress button, key, or switch called X                              |
| X^           | release button, key, or switch X                                     |
| X^▼          | idiom for clicking button, key, or switch X                          |
| X"abc"       | enter literal string, abc, via device X                              |
| X (xyz)      | enter value for variable xyz via device X                            |

## 2.2 Modèle de tâche

**UAN**, (User Action Notation) utilise des opérateurs pour dénoter les relations temporelles :

|     |                                                                                                                                                                                                                                              |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ( ) | grouping mechanism                                                                                                                                                                                                                           |
| *   | iterative closure, task is performed zero or more times                                                                                                                                                                                      |
| +   | task is performed one or more times                                                                                                                                                                                                          |
| { } | enclosed task is optional (performed zero or one time)                                                                                                                                                                                       |
| A B | sequence; perform A, then B (same if A and B are on separate, but adjacent, lines)                                                                                                                                                           |
| OR  | disjunction, choice of tasks (used to show alternative ways to perform a task)                                                                                                                                                               |
| &   | order independence; connected tasks must all be performed, but relative order is immaterial                                                                                                                                                  |
| ↔   | interleavability; performance of connected tasks can be interleaved in time                                                                                                                                                                  |
|     | concurrency; connected tasks can be performed simultaneously                                                                                                                                                                                 |
| ;   | task interrupt symbol; used to indicate that user may interrupt the current task at this point (the effect of this interrupt is specified as well, otherwise it is undefined, i.e., as though the user never performed the previous actions) |

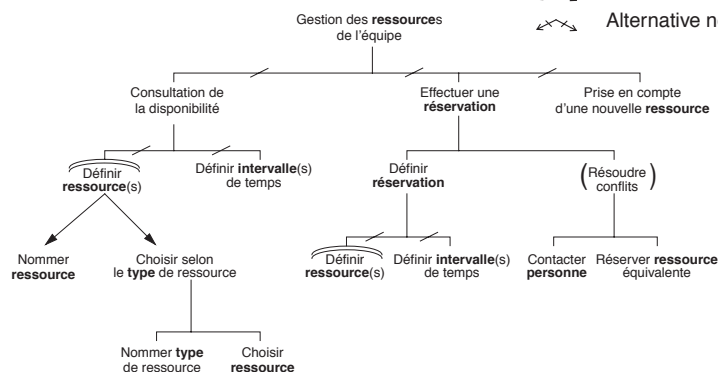
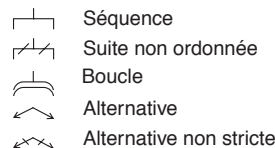
## 2.2 Modèle de tâche

**UAN**, (User Action Notation) utilise des opérateurs pour dénoter le *feedback* :

|                      |                                                        |
|----------------------|--------------------------------------------------------|
| !                    | highlight object                                       |
| !-                   | dehighlight object                                     |
| !!                   | same as !, but use an alternative highlight            |
| !-!                  | blink highlight                                        |
| (!-!) <sup>n</sup>   | blink highlight <i>n</i> times                         |
| @ <i>x, y</i>        | at point <i>x, y</i>                                   |
| @ <i>X</i>           | at object <i>X</i>                                     |
| display ( <i>X</i> ) | display object <i>X</i>                                |
| erase ( <i>X</i> )   | erase object <i>X</i>                                  |
| <i>X</i> > ~         | object <i>X</i> follows (is dragged by) cursor         |
| <i>X</i> >> ~        | object <i>X</i> is rubber-banded as its follows cursor |
| outline ( <i>X</i> ) | outline of object <i>X</i>                             |

## 2.2 Modèle de tâche

Un exemple d'arbre de tâches noté avec **HTA** (*Hierarchical Task Analysis*).



## 2.2 Modèle de tâche

Il n'y a pas de notation standard,

**CTT** (*Concurrent Task Tree*) propose les opérateurs suivants :

- $T1 \gg T2$  : séquence ;
- $T1 [data] \gg T2$  : séquence & transmission d'information ;
- $T1 \sqcup T2$  : disjonction ;
- $T^*$  : fermeture (répétition de zéro à autant de fois qu'on veut) ;
- $T^n$  : répétition ;
- $[T]$  : option ;
- $T1 [> T2$  : interruption définitive ;
- $T1 |> T2$  : interruption avec reprise ;
- $T1 ||| T2$  : parallélisme.

Il existe un éditeur qui permet l'édition de modèles de tâche : **CTTE**.

<http://giove.cnuce.cnr.it/ctte.html>

## 2.2 Modèle de tâche

Les *use case models* d'UML spécifient les requis fonctionnels du point de vue des acteurs.

| Use Case Name |                                      |
|---------------|--------------------------------------|
| 1             | Brief Description                    |
| 2             | Actors*                              |
| 3             | Flows of Events                      |
| 3.1           | Main (basic) Flow                    |
| 3.1.1         | Step 1                               |
| 3.1.2         | Step 2                               |
| 3.1.3         | Step ...                             |
| 3.2           | Alternative Flows                    |
| 3.2.1         | Alternative flow 1                   |
| 3.2.1.1       | Step 1                               |
| 3.2.1.2       | Step 2                               |
| 3.2.1.3       | Step ...                             |
| 3.2.2         | Alternative flow 2                   |
| 3.2.3         | Alternative flow ...                 |
| 4             | Special Requirements                 |
| 4.1           | Usability requirements               |
| 4.2           | Business rules                       |
| 4.3           | Other non-functional requirements... |
| 5             | Pre-conditions                       |
| 6             | Post-conditions                      |
| 7             | Extension Points                     |

## 2.2 Modèle de tâche

### 3. Flow of Events

#### 3.1 Basic Flow

##### 3.1.1 BID

This use case starts when the Buyer bids on a displayed item currently available for auction.

##### 3.1.2 ENTER AMOUNT

The Buyer enters the bid amount. The system validates the bid amount. The entered bid must be greater than the current (i.e., greatest) bid by an amount greater than the minimum bid increment specified for the auction.

##### 3.1.3 BUYER CONFIRMS BID

The system provides information that tells the legal obligations of placing a bid. The Buyer confirms that the bid should be placed.

##### 3.1.4 POST BID

The system posts the bid for the auction. The entered bid becomes the current (i.e., greatest) bid.

##### 3.1.5 SEND EMAIL

The system sends an email confirmation to the Buyer, including the bid amount for the auction item, as well as when the auction will close.

##### 3.1.6 SYSTEM CONFIRM BID

The system notifies the Buyer that the bid has been accepted and displays the Buyer's name and email address, as well as the item name and the bid amount. The use case ends.

#### 3.2 Alternative Flows

##### 3.2.1 AUCTION IS CLOSED

At BF BID, the auction for the item has been closed, a message is displayed to the Buyer stating that the auction has been closed, and the bid is not accepted. The use case ends.

##### 3.2.2 BUYER HAS PENDING PAYMENTS

At BF BID, the Buyer has pending payments, the system displays a message to the Buyer, reminding him/her that he/she has pending payments that are due and that until those payments are made, the Buyer cannot participate in an auction, either as a Buyer or a Seller. The use case ends.

##### 3.2.3 ENTERED BID IS INVALID

At BF ENTER AMOUNT, the User enters an invalid bid, the system indicates to the User the reason the bid is invalid (e.g., the entered bid was not greater than the greatest bid by an amount greater than the minimum bid increment specified for the auction). Resume at BF ENTER AMOUNT.

##### 3.2.4 BID NOT CONFIRMED

At BF CONFIRM BID, the Buyer does not confirm that the bid should be placed. The use case resumes at BF ENTER AMOUNT.