Task-oriented Information Modelling

A lightweight formal language for the representation of work

Tanguy Wettengel[1], PhD. Computational Linguistics, HDR Language Sciences

[1] Université de Limoges, Centre de Recherches Sémiotiques −39E, rue Camille Guérin 87036 Limoges Cedex France - tanguy@teamtim.com

Abstract. This paper briefly describes a task modelling language (TIM [Task-oriented Information Modelling]) A critical introduction to major task-based work description methods is included.

Keywords: task analysis, task model, task hierarchy, task typology, work modelling

1. Introduction: On task analysis methods

Work modelling has strongly evolved over the past 40 years: leaving aside the attemps of Taylor and the Gilbreths in the beginning of the last century, modelling strategies to represent work appear in the late sixties. We think of HTA and GOMS as "classical" approaches to the field, because they focus on individual performance rather than on team-based activities and therefore do not take the social context of work into account.

HTA, Hierarchical Task Analysis, (Annett & al. [1967]) identifies tasks (work units) to goals. Unlike behaviours, goals are cognitive entities that can not be directly observed. Goals in HTA can be split into sub-goals, activated by an external situation, reached by means of actions and, for sub-tasks, ordered and selected following plans.

Possible drawbacks of the HTA architecture might be the following:

• excepted the task diagram, models are worded in natural language,
• the (crucial) distinction between "task", "operation" and "goal" is somewhat difficult to grasp,
• there is no systematic relation between an operation and a material change in the environment (control operations, for instance, leave the latter unchanged),
• plans are enhancements of the task structure instead of being a part of it.

Despite this rather open architecture (that may cause the quality of models to become too strongly dependent on the analyst's skill), HTA has successfully been engaged in industrial projects and appears as the backbone of some tools dealing with task modelling (TaskArchitect® [see http://taskarchitect.com], TaskModeller® [see http://www.alphaworks.ibm.com]).

While HTA has been initially designed to enable the prevention of errors in process controlling situations, GOMS, Goals, Operators, Methods, Selection rules, (Card & al., [1983]) has been designed for productivity analysis of working strategies based on the estimate execution time of the activities to be performed. GOMS does not provide any means of conducting the task analysis in itself and uses mostly HTA models as input.

GOMS identifies Goals as states of the environment a user is aiming at and Methods with some strategy to obtain these goals. Depending on the working context, Selection Rules allow the choice of the best strategy among the available ones.

A Method comprises a certain number of steps, described by means of Operators. Operators may be motor activities (such as using a mouse to move a pointer), perceptual activities (identifying an alarm, for instance) and mental ones, as for example deciding whether some information should be stored or deleted by performing some kind of evaluation. Each operator is given an execution time, the total amount of time for the operators included in a task providing its duration estimate. Modern versions of GOMS take parallel processing into account, thus allowing to discover critical paths and drawing a fine-grained picture of a system's or a strategy's performances.

The predictive power of GOMS models has shown to be quite precise for productivity measurements, which explains why this approach underwent ongoing evolutions and became computer-assisted. Nonetheless, the degree of detail a GOMS model must reach in order to deliver high-quality predictions translate into costs that limit the use of GOMS to projects where decisions entailing great economic impact have to be met.

It should though be stressed that the concept of "task" does not exist as such in GOMS and that the "operator" category covers simple actions but also complex sub-goals. The "Operator" label applies both to processes taking place within the subject (activation of some knowledge, for example) and on the environment he is acting upon. While it is easy to understand why activities of any kind are equated in GOMS (all of them are time consuming), the levelling of actions and complex sub-goals within operators compromises the translation of GOMS models into a structure systematic enough to support elaborate reasoning.

In the late 80', French research adds a significant contribution to task analysis: MAD, Méthode Analytique de Description des tâches, (Scapin & al. [1989] is a management-oriented framework that enhances the description of tasks not only with temporal information (beginning and end of execution), but also with features relating to task interdependency (parallel vs. sequential processing), hierarchy (main vs. secondary), sensitivity to interruption (interruptible vs. non-interruptible), etc.

The constraints on task description are significantly higher than in HTA, the immediate ancestor of MAD: a MAD task is described by means of a form with a fixed number of fields (name, interruptibility, dependence, priority, etc.), each one having a defined domain of values. From this point of view, MAD pioneered frame-based descriptions in the field. But, the ambition of providing a frame structure that covers all aspects of every task's status in a process, places MAD within the group of the heavy and resource consuming methods for task modelling.

MAD links every task with a user enacting a rôle, having an identified level of experience and of skill. This perspective has open an avenue to representation systems where interaction between actors, rôles and tasks become central, such as CTT, ConcurTaskTrees, (Paternò & al. [1999]) or, even before, GTA, Groupware Task Analysis, (Van der Veer & al. [1996]).

GTA takes not only the actual work, but also the situation in which work is carried out, plus the involved agents (whether human or not) into consideration. Agents have rôles with respect to task accomplishment as well as attributes (competence, skills) that make them qualify to intervene in the different tasks at issue. Work is modelled in terms of a task structure with different levels (complex tasks may engage different rôles). The situation comprises the social, conceptual and physical context bound to work.

Every aspect of the environment relevant to the accomplishment of a task is taken into consideration when the task is described from the situation's perspective. GTA thus provides a holistic view of situated activity, which qualifies it as a rich tool for the understanding of the overall dynamics of a working system but also as a very costly method for conducting instructional aimed task analysis.

2. Task-oriented Information Modelling (TIM)

TIM builds on a techno-centered approach of work: modelling categories such as states, properties, objects, operations, etc., originate in an analytical view of technical environments and do not refer to the subject's knowledge or social context.

2.1 The Structure of TIM Task Hierarchies: Task Types

As task models may be used to represent work on systems where the goals to be achieved depend on the situation and on the user's aims, TIM task structures usually exhibit a series of first-level tasks, each one of them corresponding to a primary goal.

First-level tasks stand for primary user goals and can be split into sub-tasks, if necessary (sub-tasks are often system-dependent complex steps bound to the achievement of some primary goal). Sub-tasks can, in turn, be broken into lower level ones. Any task containing no sub-tasks is a *terminal task*. Only terminal tasks contain the actions (one or more) needed to cause changes in the system. The action(s) contained in a terminal task are known as the task's "*procedure*" in TIM.

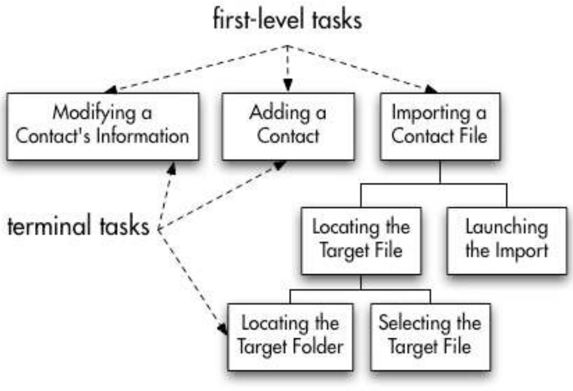
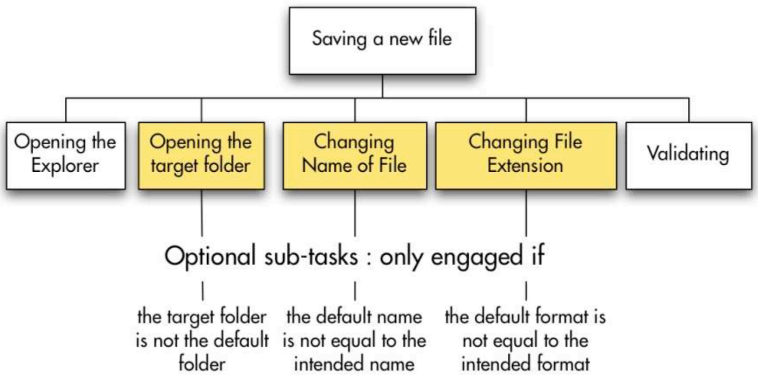


Figure 1: A TIM task structure

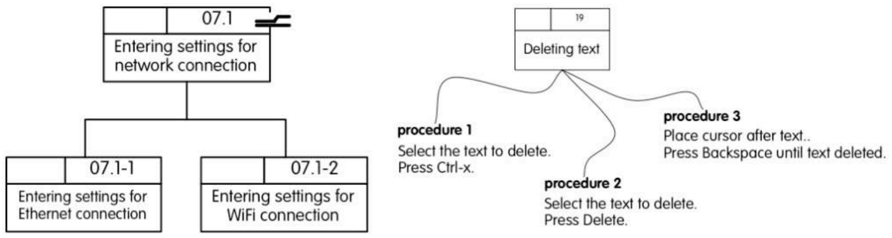
Depending on the situation, some sub-tasks may not be mandatory: when saving a new file, for instance, the "Opening the Target Folder", "Changing Name of File" and the "Changing File Extension" sub-tasks can be skipped if the user decides to accept the default values suggested by the system (see Figure 2). These sub-tasks are termed "*optional*" in TIM.



Saving a new file
Opening the Explorer
Opening the target folder
Changing Name of File
Changing File Extension
Validating
Optional sub-tasks : only engaged if
the target folder is not the default folder
the default name is not equal to the intended name
the default format is not equal to the intended format

Figure 2: Optional and non-optional sub-tasks

Depending on the system, some goals may be achieved in different ways. If these methods involve more than just a series of actions (if they need to be modelled as tasks rather than as procedures), they are bundled together in TIM as sub-tasks of an "*alternative*" task. The goal of an alternative task is inherited by each one of its sub-tasks, which only describe different ways to reach it.



07.1
Entering settings for network connection
07.1-1
Entering settings for Ethernet connection
07.1-2
Entering settings for WiFi connection
19
Deleting text
procedure 1
Select the text to delete.
Press Ctrl-x.
procedure 2
Select the text to delete.
Press Delete.
procedure 3
Place cursor after text..
Press Backspace until text deleted.

Figure 3: Alternative tasks and alternative procedures

Summarising, a TIM task tree structure depicts a workflow by enabling the following distinctions:
- First-level tasks [relating to primary goal] / non-first-level tasks
- Terminal tasks [containing a procedure] / non-terminal tasks
- Optional sub-tasks [not mandatory to reach a goal] / non-optional sub-tasks
- Alternative tasks [including sub-tasks that specify multiple methods to reach a goal] / non-alternative tasks

2.2 Specifying Task Engagement Features (Conditions, Cycle-stop, Triggers): Task Subtypes

If any constraints define the possibility or the obligation of task engagement (or re-engagement), a subtype can be added to the task types listed above. Conditional tasks are those that **cannot be engaged** *unless* some condition specified in the task's content is met. Cyclic tasks (see Figure 4) are those that, if engaged, **must be repeated** *until* some condition specified in the task's content is met. Critical tasks are those that **must be engaged** *if* some triggering condition specified in the task's content is met.

A task may be affected by more than one subtype ("Conditional" and "Cyclic", for instance). Restrictions to sub-typing are the following:
- Optional sub-tasks may not be given the "Critical" subtype
- The "Conditional" and the "Critical" subtypes are mutually exclusive.

Tasks baring no engagement-related subtype are considered as "Standard" tasks.

2.3 The Information Contained in Tasks

Tasks are "working units" producing some change in the system. This general definition points to the fact that the system properties (or the state of the system) change whenever a task is performed. The way TIM captures this change is by the traditional categories of Input and Output, which are the core of the information contained in a task. In fact, the task's name merely summarises the polarity between Input and Output, so that "Closing a File" stands for a transformation having "The file is open" as its Input and "The file is closed" as its Output.

As pointed out in Section 2.2, the engagement of tasks can be subject to conditions, or triggered by some system configuration. The information concerning the state or the properties of the system that cause or allow some task to be engaged, appears in TIM within a component of the task's structure called "Access", similar to the traditional "Pre-condition" category. This component also hosts the configuration to be reached before a task that repeats can be brought to halt.

Besides causing the system's properties or its state to shift from Input to Output, the engagement of a task might carry some secondary effect, as is the loss of data when performing a disk formatting task. This is brought into consideration in TIM by a component of the task's structure named "Side-effect" (similar to the "Post-conditions" traditional category), which states, not the primary goal of the task, but the consequences that are linked to its achievement.

Input, Output, Access and Side-effect are the decisional components of task embedded information. They allow to evaluate the legitimacy of choosing a task in a given situation. This decisional component is termed "Perimeter" in TIM and is bound to every task in the model. Nonetheless, whereas the Input and the Output are systematically populated, the Access and the Side-effect may be void, if there are no conditions nor triggers for the engagement of a task, or no relevant secondary consequences to it.

Section 2.1 underlined that only terminal tasks are enhanced by a Procedure (an ordered list of actions, if more than one). This is the operational component of the task embedded information. It only appears in terminal tasks, because in non-terminal ones, action is distributed among the sub-tasks they contain. Actions can be subject to some condition (Conditional actions). If this condition is not met, the action that bares it is skipped and execution proceeds to the next one in the list.

2.4 The Information Contained in Task Components: Descriptions and Actions

The perimeter of tasks is solely populated by descriptions, that is, pictures of the system that may be seen as persistent (in which case they are Properties) or not (in which case they are States). Descriptions combine objects with an attribute-value structure termed "facet" in TIM ("status: open" is an example of this type of structure). When combining "status: open" (a facet) with "file" (an object), a state is described (namely, the one in which a file is open). Descriptions also provide means to express conditions affecting actions in the procedure, as for example, to convey the instruction of opening some valve if the pressure in the system is higher than 30 *bar*.

The reason why facets exhibit an attribute-value structure is the manifold interpretation that could be attached to bare values: "3m", for example, can mean *length* in some case and *reach* in some other. This explains the disambiguation capacity of facets shaped as attribute-value pairs, such as "length: 3m" and "reach: 3m". Considering the descriptions as a whole (facet + object[s]), naturally suggests a predicate / argument structure, where the facet acts as a predicate and the object(s) affected by it, as the argument(s). The semantics of the predicate, combined with its cardinality, define the rôle of the arguments in the expression. Whereas "length: 3m" has cardinality 1 (even if there is more of one token of the same argument, e.g. different objects having the same length), "location: on" has cardinality 2 (the object finding itself on something and the object acting as support to it), as in "location; on" applied to the arguments "plate" and "table" to mean "The plate is on the table".

The second type of predicate / argument expressions in TIM are actions. Instead of combining objects with facets, actions combine them with operations (such as "turn", "close", "extract from", etc.). As facets, operations can have different cardinality values: while "open" has cardinality 1, "extract from" has cardinality 2.

2.5 An overview of TIM's information structure

Section 2.4 identifies the three elementary building blocks of a TIM task model: facets, operations and objects. Facets coupled with Objects specify descriptions (states or properties, depending on their persistency); Operations coupled with Objects specify actions. Descriptions provide the information to build a task's perimeter, actions are grouped into a task's procedure. Within the procedure, if there are conditional actions, the conditions are expressed by descriptions. Tasks are either split into sub-tasks or contain a procedure. Figure 5 represents a generic TIM model.

2.6 Notation

Expressions in TIM correspond to descriptions and to actions. The order in which they appear define what they stand for. Within a task, four blocks of expressions define (in the following order), Access, Input, Output, Side-effect and, for terminal tasks, a fifth bloc contains the Procedure. Every block may contain one or more expressions. Descriptions are worded as follows:

P01 [length: 3m; O01 [beam]].
meaning that the beam identified as object 01 is of length 3m,
S01 [status : open; O02 [file]].
meaning that the file identified as object 02 is open,
S02 [status : displayed; O03 [document]].*
meaning that any token of type document (identified as O03*) is displayed.

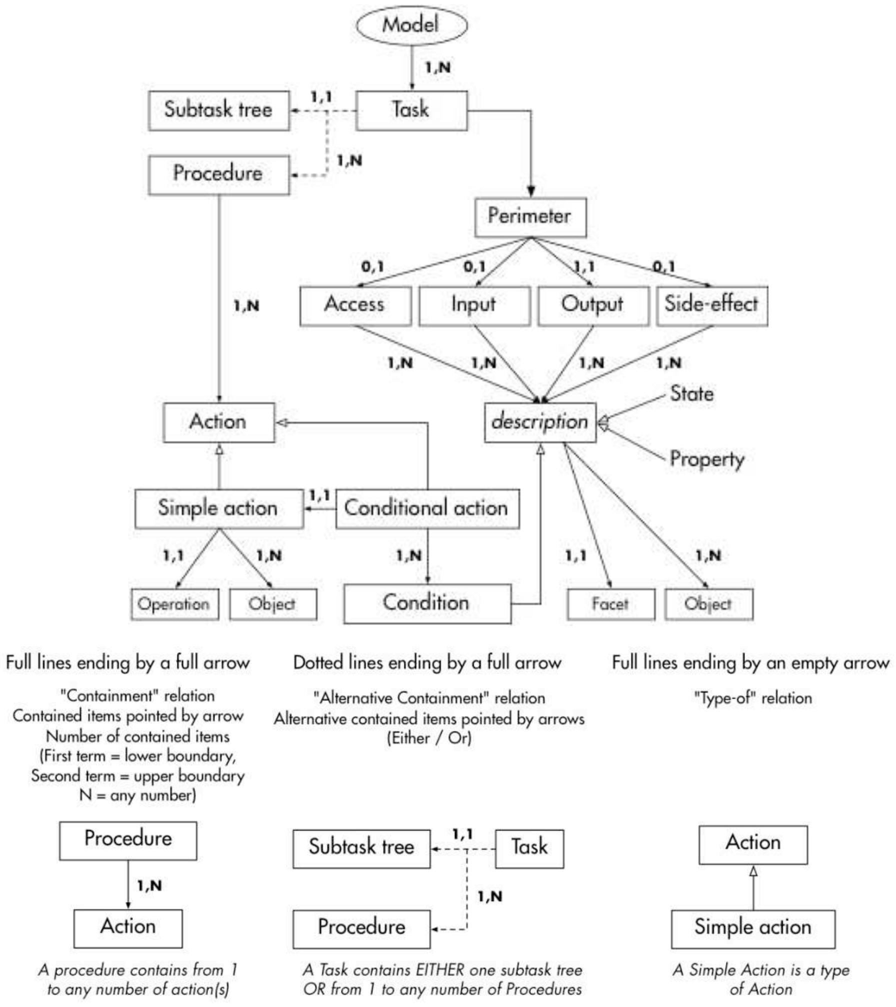

Model
1,N
Subtask tree
1,1
Task
1,N
Procedure
Perimeter
0,1
0,1
1,1
0,1
Access
Input
Output
Side-effect
1,N
1,N
1,N
1,N
1,N
State
Action
description
Property
Simple action
1,1
Conditional action
1,1
1,N
1,N
1,1
1,N
Operation
Object
Condition
Facet
Object
Full lines ending by a full arrow
"Containment" relation
Contained items pointed by arrow
Number of contained items
(First term = lower boundary,
Second term = upper boundary
N = any number)
Dotted lines ending by a full arrow
"Alternative Containment" relation
Alternative contained items pointed by arrows
(Either / Or)
Full lines ending by an empty arrow
"Type-of" relation
Procedure
1,N
Action
A procedure contains from 1 to any number of action(s)
Subtask tree
1,1
Task
1,N
Procedure
A Task contains EITHER one subtask tree OR from 1 to any number of Procedures
Action
Simple action
A Simple Action is a type of Action

Figure 5: The general structure of TIM models

Note that for object 03, the identification number is followed by an asterisk. This means in TIM that the document displayed can be any document (the asterisk meaning "variable" or token of the type named in brackets).

Actions are worded in a similar way:

A01 [open; O02].
meaning that the file identified as object 02 has to be opened.
A02 [extract from; O04 [zip file], O05 [specs file]].
meaning that the file identified as object 05 has to be extracted from the zip file identified as object 04.

As may be noticed, expressions and objects have a unique number so as to allow easy reuse throughout a model, without any rewriting of the full content being necessary. The following is an example of a simple task as written in TIM (italics word the meaning of the expressions in natural language):

T01 [Deleting a file]
Task 1: Deleting a file
S01 [status: displayed; O01* [target file's icon]].
The target file's icon must be displayed.
S02 [status: active; O02* [target file]].
The target file is active.
¬S02.
The target file is inactive.
S03 [location: in; O03 [recycle bin], O02*].
The target file is in the recycle bin.
A01 [right-click; O01*].
Right-click on the target file's icon.
A02 [select; O04 ["Delete"]].
Select "Delete".

References

Annett, J., & Duncan, K. D. (1967). Task analysis and training design. Journal of Occupational Psychology, 41, 211-221.
Card, Stuart; Thomas P. Moran and Allen Newell (1983). The Psychology of Human Computer Interaction. Lawrence Erlbaum Associates. ISBN 0-89859-859-1.
Paternò, F., Mancini , C., Meniconi, S. (1997) : ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models, Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction, ISBN:0-412-80950-8
Scapin D.L., Pierret-Goldbreich C. (1989) : MAD : une méthode analytique de description des tâches, colloque sur l'ingénierie des interfaces homme-machine, Sophia-Antipolis.
Van der Veer, G.C., Lenting B.F., Bergevoet, B. A. G. (1996): GTA: Groupware Task Analysis - Modeling Complexity, Acta Psychologica, 91, ISSN 0001-6918, pp. 297-322.
Wettengel, T. (1999). TIM, un outil pour décrire des environnements techniques. HDR Dissertation. University of Limoges (France).