TIM and DITA

[Contributed by Tanguy Wettengel]

Before comparing TIM to DITA, it be should stressed that both of them allow to describe activities by means of frame-based task structures. The following analysis relates to the capacity of TIM and DITA to model information that can translate into instructional literature (manuals, on-line helps, etc.).

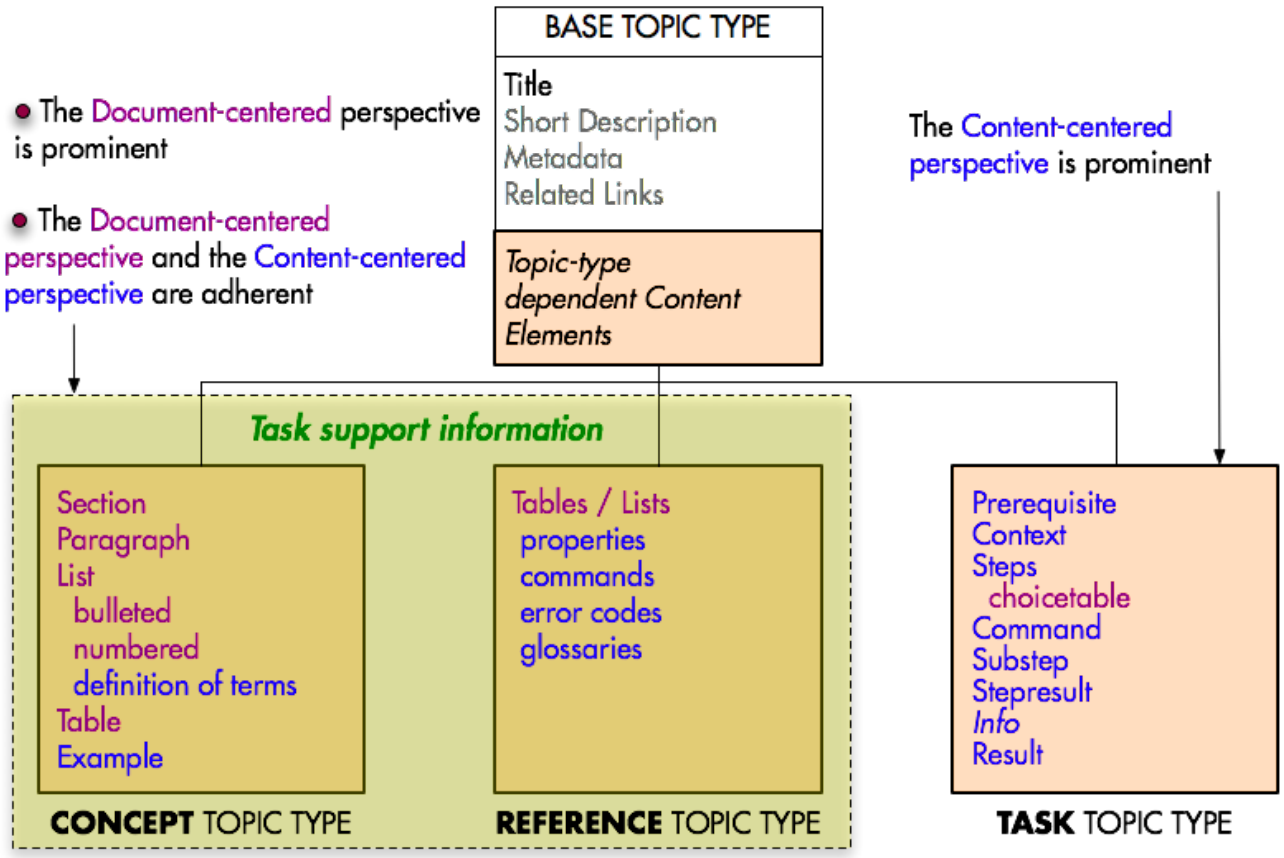
There are software tools that assist modelling and allow not only for direct publishing of manuals and on-line help systems but also for modular storage and publishing of module collections as documents. These tools are generally XML based and include export facilities for rendering of the content in different formats (HTML, PDF, CHM...): some major XML editors for DITA, TimBox® for TIM. The structure of the model's content is defined either by DTD's (one or more), or by Schemas.

As for TIM, the corresponding DTD is not the only rule-based expression of task descriptions: a formal language (Task-oriented Information Modelling Language), engineered between 1992 and 1999, has been used since to describe software, machines and industrial processes from the point of view of their users. Expressions in this language may be broken up into basic constituents (the lowest level ones being "object", "facet" and "operation"), which enables incremental modelling via libraries (in order to avoid the well known uncertainties bound to natural language).

DITA

DITA (Darwin Information Typing Architecture) allows XML file validation. The DITA DTDs encompass an information meta-model to structure technical contents, the core of this meta-model being the task concept. Whereas TIM and DITA share the view that tasks are central when describing working environments, they differ dramatically in the way the overall content template has been designed. TIM does not allow any specification of document partitions (sections, paragraphs) or information layout (lists, tables ...) into its models; DITA includes these alongside with semantic-oriented structuring facilities in its models.

This choice places the DITA user in a double perspective: as an expert in his own domain (electronics, software, medicine, etc.) and at the same time as a technical writer. Holding this double view simultaneously is generally considered as challenging, but, if unavoidable, a clear-cut separation between the domain knowledge layer and the presentation layer would seem beneficial for the consistency of both the information as such and the rendering of it in documents. The graphics below underlines a relatively strong adherence of these layers ("definition of terms" is a subtype of "list", properties and commands are bound to table or list formats).



There is equally a substantial difference between the category types ("elements") embedded in the "Concept Topic" type and in the "Task Topic" type: whereas the latter are generally equipped with semantic values related to the information they convey ("Prerequisite", "Steps", "Substeps", "Stepresult" appear as the content of a task), the former (except "Definition of terms" and "Exemple") are semantically empty.

Because of this lack of balance, "element" (the sub-unit of a Topic Type) merely becomes a structuring factor without constant properties (it sometimes defines the way the content is presented or a document is divided into parts, sometimes defines the sub-concepts bundled in an abstract view of work units [tasks]) . Moreover, the document-oriented containers (Concept Topic type and the Reference Topic type) are levelled to the domain-centered one (the Task Topic type) due to the fact they are all instances of the same base category ("topic"). In other words, when producing content, the structure and layout of a document (sections, paragraphs, lists, tables) and the semantic structure of activities (steps, sub-steps results, etc.) must be tackled simultaneously by means of containers that are formally equivalent (split up into "elements") but semantically uneven.

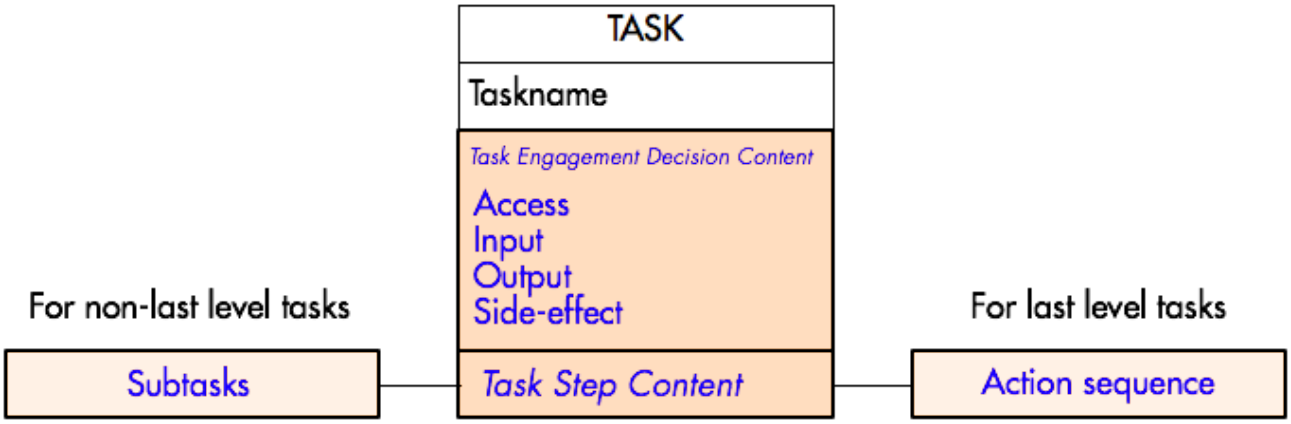
As a closing comment, we would like to stress that the criterion applied to separate the three topic types (Concept, Reference and Task) is not easy to sustain from a systematic point of view: whereas "Task" information clearly differs from information needed to perform a task (general knowledge, set of commands, parameters and such, included in the "Concept" and "Reference" topics), the contrast between the "Concept" and Reference" topics is based on the ancillary character of the "Reference" contents with respect to those of the other two types.

Even if this content structure may be adapted to particular needs through the "Specialisation" facility, the generic DITA content model that we have discussed strongly interweaves content structure and layout management. Unlike TIM, DITA does not draw a strict borderline between domain-centered model engineering and publishing.

TIM

TIM is an environment allowing to model work in different contexts. TIM models can be used to provide the content of instructions (such as manuals and on-line helps) but equally to plan production processes or define functional specifications. TIM is thus a method for describing work and not a tool for describing documents. The choice of sharply separating task modelling and content publishing explains why TIM does not include any means to specify the structure or the layout of documents.

Unsurprisingly, a TIM task's meta-model only bears semantic components related to the analysis of work, such as Input, Output, Side-effect, etc, as shown by the image below.

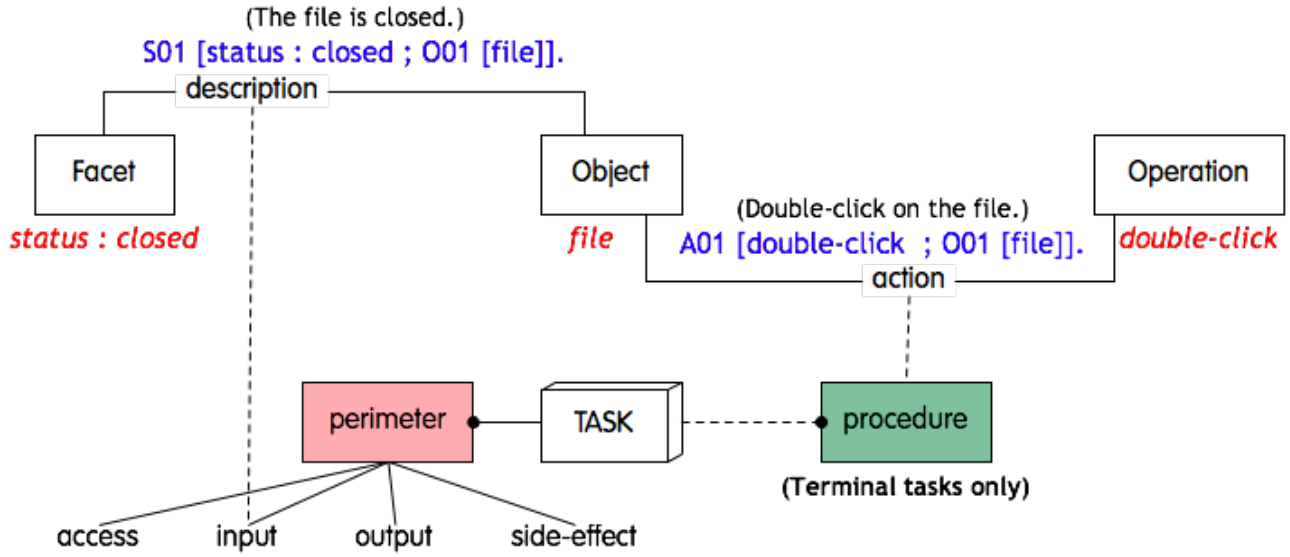


Information relevant to task engagement is distributed between categories named "Access" (pre-requisites), Input, Output and Side-effect. Last level tasks express the activity allowing the transition from Input to Output as a list of actions named "Procedure", while other task distribute this activity among the subtasks included in them. Task engagement related information is limited to descriptions that are typed to Properties or to States, depending on the persistence throughout the model of the situations they describe. States are reversible (a light that is on, in case it might as well be off) whereas Properties are not (the colour of a cable, for example).

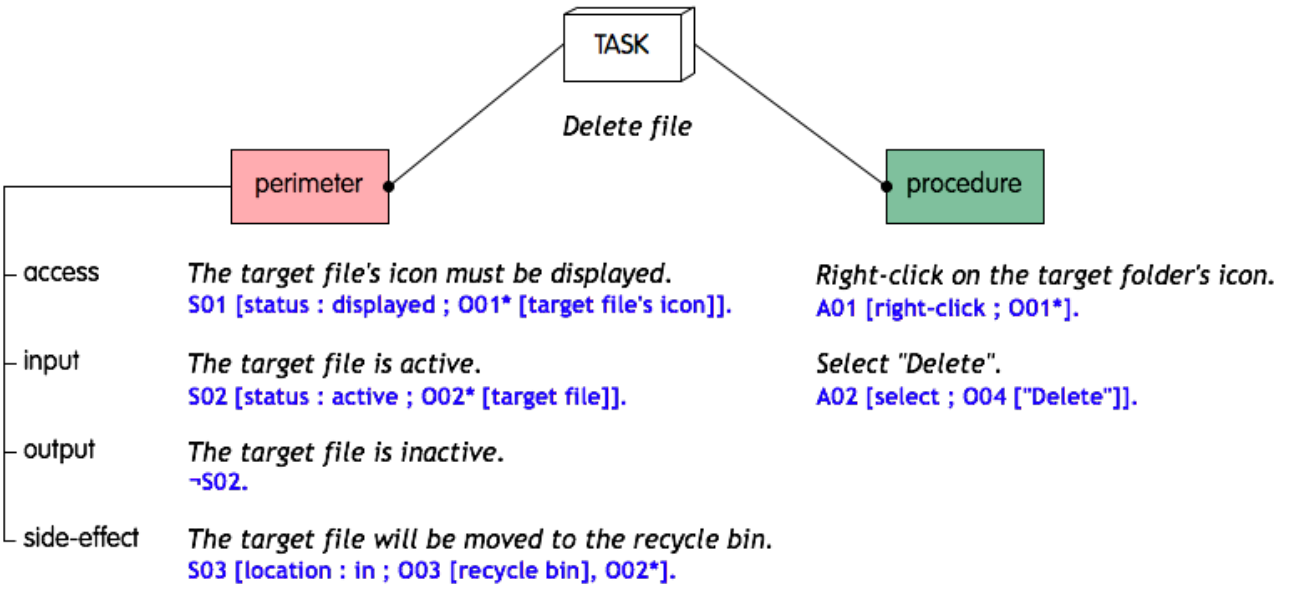
Each task holds a place in a task tree. If a task is split into sub-tasks, the sub-tasks included in it may be mandatory or optional. Any task may bare engagement conditions or not (the ones that do, are called "conditional" tasks in TIM). Any task that may need to be re-engaged in order to allow satisfaction of a condition stated in its Access is viewed as a "Cyclic" task. If the engagement of a single subtask is enough to produce the transition from Input to Output, the siblings of the same level are considered as "Alternatives". Because of this rich structure, TIM task trees conveys complete information about mandatory sequences, options, alternatives, conditions and cycles. They therefore are not

bound to external enhancements, such as the "Plans" that are mapped onto task trees in HTA.

The formal language monitoring this architecture is built up by combining three primitive entities, namely "Objects", "Facets" and "Operations". Objects are entities baring properties or being affected by operations, Facets stand for some feature of an object in a given situation, Operations express concrete or abstract manipulations performed on objects.



States and Properties (descriptions) populate the Perimeter of a task (Access, Input, Output, Side-effect); Actions (at least one) build up the task's procedure. Whereas the goal (the polarity between Input and Output) distinguishes optional tasks from mandatory ones in a given situation, the Access allows to code task features, such as "conditional" or "cyclic" without any enhancement of the basic task tree. The blue text in the-figure below shows a syntactic model of a simple task.



The actual model reduces in fact to the following expressions:

```
T01 [Deleting a file].

S01 [status : displayed ; O01* [target file's icon]].
S02 [status : active ; O02* [target file]].
¬S02.
S03 [location : in ; O03 [recycle bin], O02*].

A01 [right-click ; O01*].
A02 [select ; O04 ["Delete"]].
```

The positions of the elements define the category they belong to (in sequence, task name, access, input, output, side-effect and procedure). The combination of the meaning of the expressions and the category they belong to allow appropriate translation into natural language: the access is here translated into "the target file's icon must be displayed", "must" expressiong the fact that the description is placed within the access. Moreover, the fact that the access is not empty reveals the fact that the task is conditional (in our example task, if the icon is not displayed, the procedure cannot be performed).

CONCLUDING REMARKS

DITA is an environment designed for producing different types of content to be published as technical documents. We have here examined the capacity of this architecture for content modelling and underlined a strong dependency between categories related to the document's structure, the presentation of the information and the semantic value of this information. The prominence of the publishing perspective leads us to consider DITA as a tool for structuring the content of documents rather than a semantic-oriented modelling environment.

TIM does not allow to attach any documentary related information to content. It is an environment exclusively engineered for modelling work by means of clearly identified semantic categories. These categories are represented in a model by a meaningfull sequence of expressions. Expressions combine a fixed set of symbols according to a defined syntax. The only type of content TIM is able to handle is a task structure.