EDITION OF STRUCTURED DOCUMENTS IN A HYPER-TEXT ENVIRONMENT

Christine Buors
Jean-Louis Vignaud

Cap Gemini Sogeti Group
Cap Sesa Research Center
Chemin du vieux chêne
ZIRST 38240 MEYLAN
FRANCE

Abstract

This paper presents the TWIST (Technical Writer's Integrated Support Tool) project. The system developed within TWIST can be viewed as a high level hypertext system interface, as well as an environment for the edition of structured documents. This paper explains in which way the two points of view are addressed. In particular, it describes TWIST objectives and the means carried out to reach them. This presentation will led us to introduce the hypertext and structured edition concepts

Key words

Documentation / document / role / structured edition / hypertext

1. Introduction

Documentation is one of the key components in any project, and particularly in the software production domain. Indeed documents are often the most suitable and the most reliable communication media. Even if some people agree with the idea that we generally write too much, we cannot avoid the production of a minimum set of documents. The current systems carried out in most companies, leave the user feeling uncomfortable: everybody has experienced, at least one time in his life, the agony of the poor lonesome writer. We can easily imagine that a good documentation support, or environment, is really a need in any activity domain (and therefore in the computer based activities).
In the context of ESF (Eureka Software Factory) this problem has been addressed. ESF is an Eureka project whose major interest is the management of large projects (that means, projects involving a lot of people). In this context, problems such as activity management and control, integration, and man-machine interface design are addressed. The aim of the project is to provide companies with a way to design their own software environment, according to their specific needs and resources.
ESF is a very large project divided into sub-projects according to different themes. TWIST - Technical Writer Integrated Support Tool - is one of these sub-projects. As indicated by its name, TWIST deals with documentation management problems. Its two main objectives are:
- to improve the quality and the productivity of documents within a software project;
- to provide some help in writing and using the documentation during the whole project life cycle.
To meet these objectives an environment is under specification. This paper presents the main functionalities which will be supported and the underlying concepts. The choice of the functionalities to be provided has been based on an investigation of users' needs and on our own unfortunate experiences.

2. The documentation problem

Management of documents is a well-known problem in any domain. There are generally many kinds of worries:

- the problem of writing: a lot of people still dream of a fancy editor allowing the use of bold characters without using barbarous signs such as \\f-, $(, and so on, not to mention the luxurious versions allowing the "What You See Is What You Get" !

- the problem of homogeneity: how is it possible to avoid, in the same project, the use of anarchical document formats, other than doing a visual control ?

- the problem of consistency: when documents share information (for example a schema), how is it possible to control the evolutions of this information in all the documents ?

- the problem of organization: when documents are finally written (phew!), the last problem is to localize each document among others. This work is really important: particularly in that it introduces a reading order.

- and the last problem (but not the least !), how to manage a document all along its life cycle (versions, archiving, etc.) ?

In the domain of computer science the problem is particularly important since any activity is sanctioned by a document. The set of the documents produced within the whole life cycle of a project therefore constitutes a mountain! In the context of large projects, all the problems previously mentioned are increased.

Before going further, let us introduce a few vocabulary that will be used in this paper.

A **role** corresponds to the representation of people functions in the organization of the production process. For example, in the software production, we may identify the following roles: secretary, support manager, project manager, programmer, analyst, sales manager, system engineer, etc. With roles are related activities, responsibilities and ability to perform actions to access (more or less "confidential") information. The different roles in a company are identified by the method used for the production. Obviously, roles are played by physical persons named **actors.**

A **document** is an organized set of text, graphics, formulae, etc. For TWIST, documents are structured and constituted of three types of information: the structure, the presentation and the content of the document. The content is updated by the user while editing the document. The structure and the presentation correspond to a standardization of the document.

The **documentation** of a project is the set of all the documents produced during the project life cycle. To be easily managed these documents must be classified. This classification generally depends on the organization of the company, and on the method used to develop the project. The method, also indicates which role is responsible for the creation and the update of a set of documents (or a specific document) and indicates which roles may consult or modify this set.

3.. TWIST functionalities

The aim of TWIST is to prove that the current technology in computer science allows the design of an efficient solution to all the problems described above.
We think that the improvement of documents quality and productivity goes with methodological support for organization of documentation, standardization and reuse of documents, automatic generation of documents, archiving capacities and maintenance help.

For doing so, TWIST will provide some functionalities that could be classified in three large classes:
- those dealing with the organization of the documentation
- those dealing with the management of the documentation of a specific project
- those dealing with the manipulation of content of documents

Each of these functionalities corresponds to the need of a particular role within a company or a project. Each of the specific roles allows the definition of an environment in which concerned functionalities will be implemented by tools.

3.1 Management of documentation organization

Design of the organization of a documentation consists in combining documents in sets, describing all the relationships between these sets, defining standards of presentation and structure for the documents contained in these sets, and associating a list of roles with their rights on each set.

TWIST provides the possibility to describe several structures of documentation. These structures will be used as models for documentation of projects: each time a project starts, the best adapted structure of documentation is chosen, and then duplicated. A *project documentation* can therefore be considered as an instance of a *model documentation*

Documentation models manipulations and creation of instances for projects are attributed to the same role (for example, the method engineer) and grouped in an environment named **administrator environment.**

3.2 Management of a specific project documentation

A project documentation is therefore an instance of a model of documentation. This instance may be particularized to fit special constraints bound to the project . For example, a set of documents defined in the model may be obsolete for a specific project, or a set of documents may need a particular presentation for a typical project. TWIST allows this kind of adaptation. In this case, the modifications performed are local to the concerned project documentation.

Unlike documentation models, project documentations have a content: the documents realized during the project life cycle. Documents belonging to a library may be included in a project documentation. On the opposite, documents belonging to a project documentation may be stored in a library.

When created (see section 3.1.3), documents may be archived or moved from a set of documents to another one. These manipulations are also available for sets of documents.

Otherwise, TWIST allows automatic generation of a document providing that its content has been described.

All the manipulations described in this section are generally performed by the responsible of documentation within a project, and grouped in an environment named **document manager environment.**

3.3 Management of documents' content and versions

As soon as a project documentation has been defined, one can create documents. Creation, destruction, and consultation of documents are obviously realized using an editor. Nevertheless, TWIST provides new functionalities concerning co-authoring, information sharing and inter-document references. These capacities are also supported by the editor. Additional functionalities concern versions of documents (listing versions, purging versions), and impression of documents (printing facilities).

These manipulations are grouped in an environment named **writer/reader environment** or **editing environment**.

4. How these functionalities meet TWIST objectives

Since we mentioned TWIST objectives in the introduction of chapter 3, we must now examine in which way the functionalities proposed will ensure them.

4.1 Methodological support

The organization of the documentation and the association of role/rights upon each set of documents depend on the method used to develop the project.
Providing a way to describe documentation models, TWIST therefore ensure a methodological support for the development of any project.
If the scheduling of document production is not a goal for TWIST, any component able to perform this capacity can be integrated to TWIST (in the context of software factory).

4.2 Standardization of documents

We have seen that structure, presentation and content of documents can be controlled independently. Using TWIST environment, the work of a the author of a document is reduced to writing its content. That is good way to increase the productivity. Moreover the use of the same structure and the same presentation for all documents of a same set warrant at least an equal quality of presentation.

4.3 Reuse of parts of documents

TWIST allows information sharing among documents of a same documentation. Another way to increase the productivity is in reusing documents existing in other documentations. Generally in a company some reference documents could always be used to make new documents (for example, contract, quality plan, etc.). This corresponds to the culture (or experience) of the company. These kinds of documents should be in free access libraries: one can then import any document desired in his own documentation, eventually attributing some specific properties to it (for example, new roles and access rights).

4.4 Automatic generation of documents

Generally in a company, many software environments are used assisting people in their work. Each of these environments manages its own data. For example, an environment supporting project management has the data needed to produce monthly reports. Generally this environment is able to produce these kinds of documents. The problem is that the documents produced are not managed by the documentation support environment, and therefore they are not consulted, printed or archived like the other documents of the project.
In the context of software factories, where environments or tools are able to communicate easily, all the documents may reside in the documentation environment. In particular, TWIST is able to generate documents providing that their structures and presentations are described, and principally their content is specified. Describing the content of a document consists in providing the paths to find some data in another environment. A special mechanism is then able to interpret them, generating the specified docu-

ment.

4.5 Editing support

TWIST provides capabilities of:
- managing the co-authoring, in indicating the part of documents reserved by authors
- sharing parts of documents, in maintaining the consistency between the different documents sharing an information
- inter-document referencing, in managing links between parts of documents.

These functionalities are not managed by the editor: this allows the possibility to integrate to TWIST any editor (providing that it manages structure and presentation standards).

4.6 Archiving capacities

Documents have to be archived in the life cycle of the project. Archiving a document means that this document is considered finished and no more modifiable.
TWIST allows the definition of an archiving property which will be attached to documents. The management of relations between archived documents and non-archived ones is ensured.

4.7 Maintenance help

Documentation maintenance help requires at least two capacities:
- brownsing of the documentation
- maintaining consistency within the documentation.

Using the documentation structure, a user may be able to find (more or less easily) a specific information. This action is more difficult to realize since the expression of the request is not very precise but corresponds to a semantic information (for example, request as "I would like to find a quality plan concerning real-time applications in the nuclear domain"). These kinds of request may be assumed by TWIST providing that documents are annotated with comments or specific information.

TWIST manages information sharing and references between documents. It allows archiving and versions control.

5. Concepts used to implement TWIST

The two main underlying concepts of the TWIST implementation are structured edition and hypertext structures and manipulations

5.1 Structured document edition

There are two kinds of documents manipulation system based on two document models. A possible model consists in describing documents as characters flows containing special control characters (line jump, page jump, spaces,..). Another kind of model consists in taking into account the logical structures of documents: instead of only describing their physical presentations, it describes their organization in term of chapters, sections, subsections, titles, etc. The physical representation is described separately.

We have chosen to use this second approach. A document is defined by two generic descriptions: its logical structure, its default presentation. These descriptions allow the classification of documents in classes.
The logical structures are essentially

tree structures: nodes are components of the logical structure (chapter, section, etc.), and links represent inclusion relations among nodes (e.g., section I -> sub-section I.1). The structure is described using rules. The brows of the document is equivalent to a path in the tree.
The physical presentation is attached to the logical structure. The presentation is defined by a set of rules. It allows the definition of multiple views of a same document (e.g., the document is viewed as a whole or as its table of contents).
The content of the document is dispatched among its nodes.

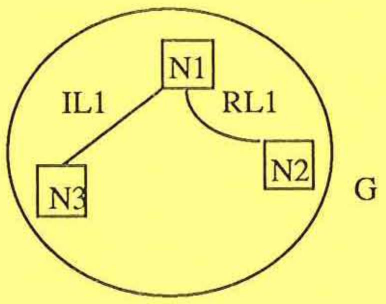
5.2. Hypertext concepts

An Hypertext abstract machine (or HAM) allows the manipulation of five types of objects: the type-graph, the type-context, the type-node, the type-including-link and the type-reference-link. Objects of these types are, respectively, graph, context, node, including link, and reference link.
An object is identified by a name. Each time an object is modified a new version of it is created.
Each class of object may own attributes/values pairs. The attributes and the values are either imposed or specified by the user (i.e., the user can create all the attributes/values pairs he needs).
The manipulations provided by the HAM are creation, destruction, modification of all these objects.

We can now described more precisely the different objects introduced below, and explained in which way they can be used within TWIST.

5.2.1 Graph

A graph is a set of nodes, including links and reference links. For example, G, set of nodes N1, N2, N3, reference link RL1 and including link IL1, is a graph



○ : graph   \ : including link

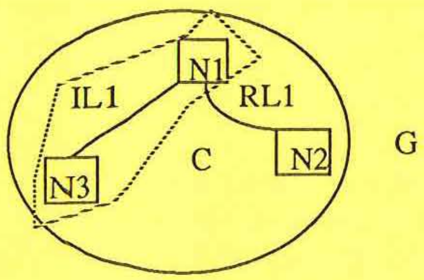
□ : node   ╰ : reference link

Property:
All the graphs are disjoint (i.e., two different graph - which have different names - cannot include the same node, or the same link).

Use in TWIST:
A project documentation in TWIST is a graph.

5.2.2 Context

A context is a part of a graph. It is a set of nodes, including links and reference links. For example C = {N1, N3, IL1} means that C is a context of the graph G



○ : graph \ : including link

□ : node ⟍ : reference link

◇:context

Property:
Contexts do not have intersection. Therefore, they can be used to create a partition of the graph.

Use in TWIST:
A context may have an intersection with another one, particularly a context may include other contexts.
A document (or a set of documents) in TWIST is a context.

5.2.3. Including link, Reference link

A link is associated to a single orderly pair of nodes. It is the only a way to connect nodes in a graph. For example the including link IL1, in the graph G1, is associated to the (N1, N3) pair.
An including link defines an including relation between two objects. A reference link defines a simple relation between the two nodes.

Properties:
An including link and its two associated nodes belong to the contexts including the first node in the associated pair of nodes. (I.e., if N1 belongs to C and IL1 relies N1 to N2 then IL1 and N2 belong to C)
If a link (reference or including) is associated to two nodes belonging to the same context, then the link belongs also to the context (i.e., if N1 and N2 belong to C and L connects N1 to N2 then L belongs to C).
If a link (reference or including) belongs to a context then its two associated nodes belong also to the context. (I.e., if L belongs to C and L connects N1 to N2 then N1 and N2 belong C)
If a link (reference or including) is associated to two nodes, and one of the two nodes is destroyed, then the link is also destroyed.

Use in TWIST:
Including links in TWIST are used to realized documents structures, and reference links are used to realize references between documents or parts of documents.

5.2.4 Node

From the HAM point of view, a node is an atomic object characterized by its content.

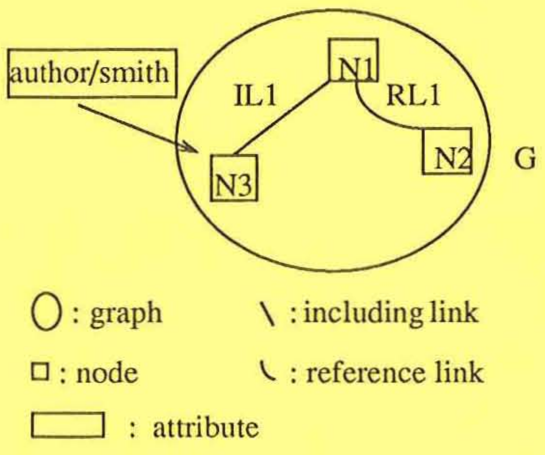
Use in TWIST:
Within TWIST, the content of a document is stored in nodes.

5.2.5 Attribute

An attribute is a property bound to an object (graph, context, node, including link, reference link). This object is character-

ized by the value of its property. An object may own several (attribute/value) pairs.
Attributes are not pre-defined : one may create any property desired.

Use in TWIST:
Within TWIST, attributes are used to specify special properties. For example, the pair (author/J.Smith) bound to a specific node N3, means that J.Smith is the author of the content of the node N3.



5.3 Summary

The two concepts previously described concern respectively the two types of objects manipulated by TWIST: the documents, and the set of documents (documentation).

The structured edition clearly allows an easy manipulation of parts of a document, the definition of any kind of link inside the document. The structured edition concern essentially the reader/writer user. The edition in TWIST is supported by GRIF [Quint 86], [Quint 87].

The hypertext concepts [Con 87] association is a "plus": it allows a large range of manipulations among documents and set of documents. Using HAM concepts [Del 86], [Del 87],, the documentation can be viewed as an *hyper-document.* The sharing and reference (among documents) functionalities can be easily carried out. The functionalities provided for the administrator and responsible of a project documentation users are essentially supported using hypertext mechanisms..

6. Conclusion

The starting point of our work has been a functional description of TWIST. This first approach assured us that providing the user with an integrated documentation support is not an utopia. Moreover the concepts carried out are well-known. The current step consists in specifying and designing a storage component based on the HAM concepts, aiming at providing, for September 1989, a prototype which will support the three environments described in this paper.
We expect that before the end of 1989, at least one project in our company will use TWIST.
We hope that this prototype will be the starting point of further investigations both in the improvement of the edition functionalities and in the extension of the use of hypertext concepts to other storage components within ESF.

References:

[Con 87]: J. Conklin. Hypertext: An Introduction and Survey, *IEEE Computer, September 1987*

[Del 86]: Delisle N. and Schwartz M. - Neptune : A hypertext system for CAD applications, *Proceedings ACM SIGMOD '86* (Washington D.C., May 28-30, 1986), 132-142.

[Del 87]: Delisle N. and Schwartz M. - Contexts : A partitioning concept for Hypertext, *ACM Transactions on Office Information systems 5, 2* (April 1987), 168-186.

[Quint 86]: V.Quint, I.Vatton,. GRIF: An Interactive System for Structured Document Manipulation, *Proceedings of the International Conference, J.C van Vliet, ed., Cambridge University Press 1986*

[Quint 87]: V.Quint. Une approche de l'édition structurée des documents, *Thèse d'état, Université scientifique Technologique et Médicale de Grenoble, Mai 1987*