Automatic Generation of Documentation for Information Systems

Lurdes Jesus
Rogério Carapuça

IST/INESC
Rua Alves Redol, 9, 1000 Lisboa, Portugal
Phone: +351.1.3100326 - Fax: +351.1.525843 - Email: mlp@eniac.inesc.pt

Abstract. Documentation plays a central role in Information Systems (IS) development and use. Software tools must be built to allow its automatic generation. In the same way as the need of prototyping for IS construction is generally accepted, so must be the need for "Documentation Prototyping". In this paper a paradigm for the generation of documentation is proposed. It can be adapted to any IS development approach. In order to illustrate the proposed ideas, the Infolog approach was chosen in this paper.

On-line documentation is, in most cases, preferable to off-line one (the traditional paper manuals). The paradigm for generation of documentation herein proposed can be used to obtain both on-line and off-line documentation. Moreover, it is obtained in hypertext form for better browsing. The linearization of the hypertext network is also possible in order to produce the often required linear form (for example to obtain paper manuals).

The main contributions of this work are the generation paradigm, which uses the semantics of the modelling primitives in order to obtain natural language text fragments describing the specified models, and the generation of the "User Manual", the more critical of the documents to be produced.

1 Introduction

Documentation plays a central role in Information Systems (IS) development and use. Software tools for IS development must facilitate the production of documentation based on the specification models for requirements and design. That way it is possible to keep up-to-date the documents as changes are made to the objects described [6].

Manual elaboration of documentation is fastidious and time consuming. In the current state of computational tools for IS development, if automatic documentation generation is not supported, there is a big disparity between the time consumed to produce a version of the IS ready to use and the time to build the corresponding documentation.

Hence we want to see fast "documentation prototyping" going at a same pace as the already common notion of IS prototyping.

Automatic generation is possible when the requirements for documentation are described and the necessary information is available. The generation should (must) be possible either using documentation templates supplied with the tool or user defined templates. This work recognizes a set of requirements and proposes a set of templates for IS documentation. A method to automatically produce the defined documents from stored specifications is also defined.

The structure of the documentation could be the traditional linear or in hypertext form [6]. In the work reported in this paper, the hypertext structure was adopted in order to support both kinds of presentation (the linear form can be obtained from the hypertext one). Furthermore, the structured nature of the stored information and the requirements on how to browse through it, suggest naturally the hypertext presentation form.

Currently it can still be said that most of the documentation produced to support applications is essentially linear [6], presented either in paper manuals or on-line context documentation. On-line context documentation is, for many users, preferable to paper manuals. Specially if they are able to use sophisticated browsing capabilities in hypertext form instead of linear text.

IS development can generally be divided into the traditional analysis, design, construction and use/maintenance phases. According with each particular methodology, a sequence of models (Universe of Discourse interpretations) is obtained during the execution of these phases. The set of concepts (or primitives) used to build each interpretation is a Concept Structure. In order to present the ideas of this work any Concept Structure could be adopted. In this paper the one of Infolog [7,8] was chosen. It has the advantage of being rich enough so as to allow the derivation of good descriptions due to its underlying semantics. In section 2.1 the Infolog approach is briefly presented.

Specifications resulting from analysis and design normally include:

- diagrams;
- sentences in some specification language;
- unstructured (natural language) text fragments.

Using the semantics of the underlying Concept Structure it is possible to derive descriptions for the specified models. The richer the Concept Structure, the more precise and adapted to the UoD, the derived descriptions are. Furthermore, from the stored definitions of the Human-Machine Interfaces (HMIs) it is also possible to derive the User Manual. The non-automatic construction of a User Manual is, in fact, a rather complex, repetitive and time consuming operation. So, to automate its production means to considerably simplify the documentation production task. The use of the semantics of the underlying Concept Structure to produce descriptions for the specified models and the generation of the User Manual are the two fundamental contributions of this work.

It should also be noted that according to [1] HMI default definitions can also be produced automatically. This process can be coupled with that of producing the default User Manual.

There are several documents to be considered, according to the objects described and the expected reader. As referred in [9], two kinds of requirements descriptions are usually suggested in the Software Engineering field: the Requirements Definition stating, in natural language, what user services the system is expected to provide, and the Requirements Specification stating, in a formal notation, the system services in a more detailed way. The requirements definition must be understandable by non-specialist staff, like management people and potential system users. As mentioned in [9], the requirements specification "must be couched in such terms that is understandable to technical staff from both procurers and developers, and it is reasonable to assume some understanding of the software engineering process on the part of the procurer" .

The proposed documents to be generated are: the Requirements Definition Document and the Requirements Specification Document, both resulting from the Requirements Analysis phase, the Design Specification Document, that results from Design phase and the User Manual used in the Operation/maintenance phase (note that this manual does not result from the Operation/maintenance phase, instead it is used in that phase).

In the next section a brief overview of the Infolog approach and some hypertext concepts are presented. In section 3 the proposed paradigm for generation of documentation is described. Some rules used to derive the natural language description of an Infolog schema are included. An example of its application is given in section 4 where fragments of the Requirements Definition Document and of the User Manual automatically generated are shown. In section 5 the computational tool that is being developed applying the proposed ideas and its integration in an workbench for IS prototyping are referred. Some conclusions are made in section 6.

2 Basic Concepts

The objective of this section is to give an overview of the background concepts. In section 2.1 the Concept Structure of the Infolog static model is described. In section 2.2 some hypertext concepts are focused.

2.1 The underlying approach for IS development

In this section the Concept Structure of the Infolog static model is briefly presented. The basic Infolog abstractions are described along with their graphical definitions. Illustrations are presented using a fragment of the already "famous" IFIP Conference example. The Infolog schema for that UoD is presented in fig. 1.

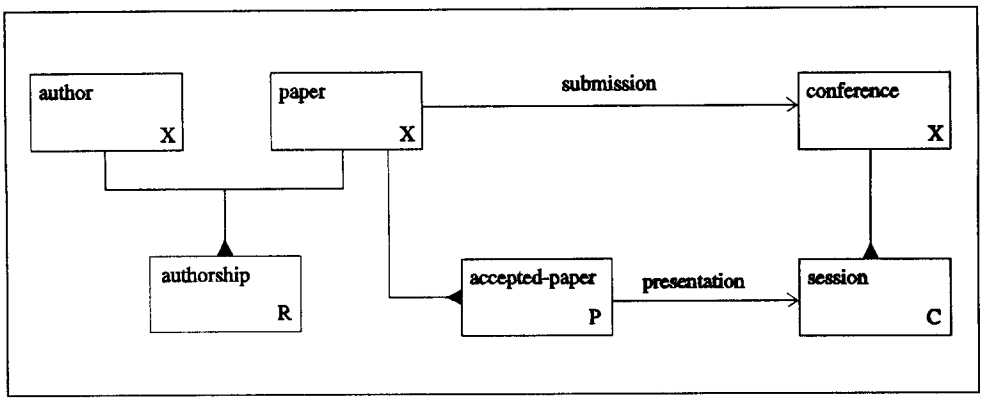


Fig.1. Infolog Conceptual Schema for the Conferences UoD (X - surrogate; R - relation; P - specialization; C - characteristic)

For a better understanding of the Infolog primitives, in the next figure the corresponding ER schema is depicted.

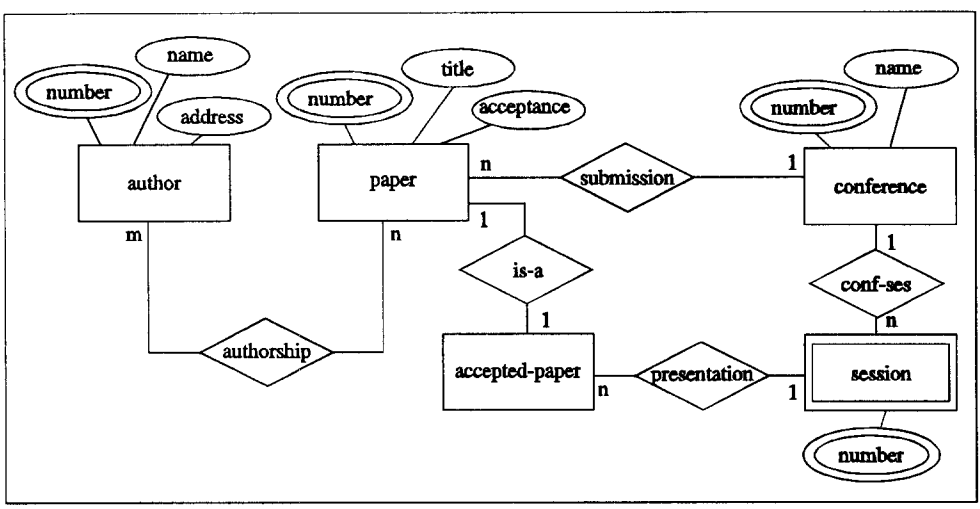


Fig. 2. ER Conceptual Schema for the Conferences UoD

The basic Infolog abstraction at the static level is the archetype. An archetype sort is a class of similar objects which are called archetype instances. An archetype instance aggregates the information about some real-world entity or association between entities. For example, *paper* is an archetype representing an entity and *authorship* is an archetype representing an association between *paper* and *author*. The word archetype is often used in this text to designate both the type and the instances. For each particular case, the reader will distinguish from context if a reference is made to a type or to an instance.

In the archetype, the information is structured in attributes. An attribute is a mathematical function whose domain is an archetype. Attributes can be either properties or designators. The codomain of a property is a data type whereas the codomain of a

designator is an archetype. For example, *title* is a property of *paper* whose codomain is the *string* data type and *submission* is a designator of *paper* whose codomain is the *conference* archetype. By means of the *submission* designator a *conference* (instance) is associated to each *paper* (instance) - the conference where the paper has been submitted. Designators correspond to a one to many relationship in the ER approach [3].

There are several categories of archetype. Each of the archetype categories have a different key mechanism (identification mechanism for the instances) and corresponds to a different real world abstraction. In this paper only the surrogate, relation, characteristic and specialization categories of archetype are described, the ones that appear in the Conference schema. For more details on the Infolog approach see [7] and [8].

A surrogate archetype represents an entity whose existence and identification mechanism does not depend of the rest of the UoD entities. The corresponding ER semantic primitive is the entity type whose instance naming and existence are independent of other entities. In the case of the Conference UoD, *paper* is a surrogate archetype sort. Each paper is uniquely identified by a number that does not depend of any other object of the Conference UoD. *number* is the key property of *paper*.

All other categories represent entities or associations that depend for definition on other - the arguments. A relation archetype represents a many to many association between the argument archetypes. The corresponding ER approach primitive is the M:N relationship (with arity equal to the number of archetype arguments).

In the case of the Conference UoD, *authorship* is a relation archetype sort. In fact, each author may contribute to more than one paper whereas, on the other hand, one paper may be authored by more than one author. So it corresponds to a M:N relationship between entities *author* and *paper*.

A characteristic archetype sort is an archetype whose instances are dependent for existence and naming on other archetype instances. In particular, a unary characteristic archetype corresponds to an ER 1:N relationship between two entities A and B where the B entity (the one with the N multiplicity in the relationship) is a week entity - the relationship between A and B is necessary to identify its instances. For example *session* is a unary characteristic of *conference*. That means that for each conference there are many sessions and that a session is related to only one conference. Moreover, to identify a session the related conference is needed. Note that the identification of a conference is not enough to identify a particular session, since there are several sessions for each conference. So a partial key mechanism must be defined in order to distinguish the sessions related to each conference. The definition of a characteristic sort includes a partial key property. Instances of that sort related to the same argument instances can not have repeated values for the partial key property. In the Conference example, *session* as a partial key property *number*. All sessions of the same conference must have different numbers. Sessions of different conferences may have equal numbers.

A specialization archetype sort denotes a sub-class of some other class (its argument archetype). Whenever a specialization is defined, a discriminant property must be introduced in the corresponding argument. The value associated through this property with each argument instance indicates if a specialization instance must or not be defined as well.

In the case of the Conference UoD, a paper may or not be accepted. In the archetype *paper* a property *acceptance* is defined to discriminate between accepted papers and non-accepted ones. If a paper is accepted then its *acceptance* property has the value "acc" and an instance of *accepted-paper* must be defined.

2.2 Hypertext

Hypertext, in opposition to traditional text, has not implicit a sequential reading order. Traditional text has a single linear sequence defining the order in which the text is to be read (page 1 -> page 2 -> ...). By contrast, in hypertext there is no single order that determines the sequence in which the text is to be read [5]. Hypertext has a network structure associated (fig. 3). Each node of that network has a text fragment[1] with a certain subject. Links are used to establish associations between subjects: each node has links for the nodes with related subjects.

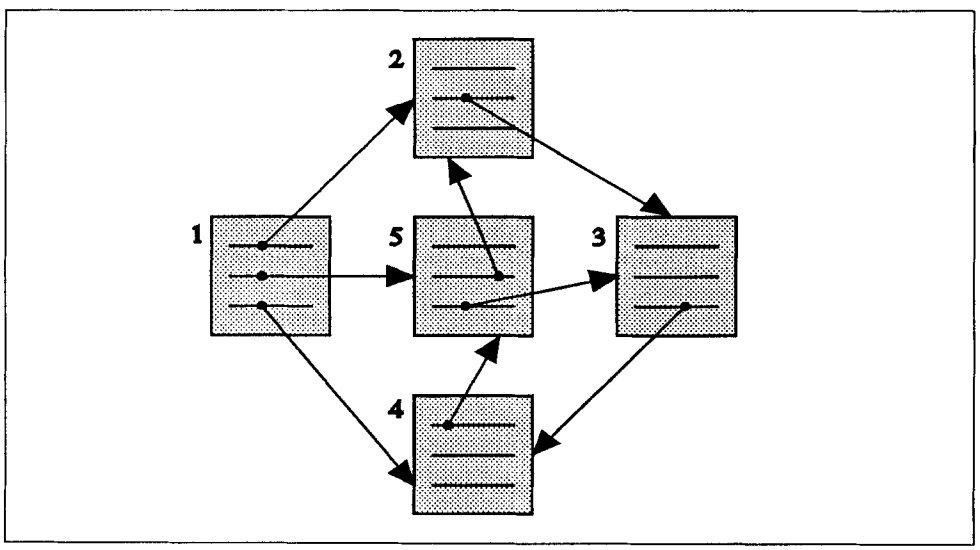


Fig. 3. Hypertext has a network structure

[1] Sometimes when the information in one node is expressed in non-textual form (usually graphics, video images or sound), the term hypermedia is used. Nevertheless, some authors (Jackob Nielsen [Niel90], for instance) use indistinctly the term hypertext defending that there is no need to reserve a special term for text-only systems.

An hypertext system offers an interactive presentation form of documentation highly adaptable to the user needs (interest level, understanding and other individual factors). Hypertext reading is made by navigation through the network (using the defined links), according to the user interest. For instance, in fig. 3 from node 1 the reader can go to node 2 or 5 or 4. As mentioned in [5], hypertext links are frequently associated with specific parts of the departure nodes (as depicted in fig. 3, where links are anchored at specific locations in the departure node). A typical application of this feature, that will be used in this paper, is to have the definition of a link anchored at a certain word in the departure node. That word can be presented, for instance, in reverse video, a different color or simply underlined to denote the link. In this paper underlined words represent the links, as exemplified in fig. 4.

Infolog

Infolog is a data oriented modelling approach.Nevertheless, not only the statical aspects of the UoD may be captured, but also the dynamical aspects. ...

Approach

An approach is formed by a Concept Structure and a methodology. ER, Infolog and Relational are examples of approaches.

UoD

Universe of Discourse - real word fragment that we want to represent.

Fig. 4. An hypertext fragment where links are denoted by underlined words

3 The proposed paradigm

The paradigm herein proposed for documentation generation is based on a set of rules that, applied to the models obtained as result of the analysis and design phases, gives an hypertext network describing those models. Moreover, and most important, there are rules that applied to the interface specifications, generate the user manual.

Several groups of rules are defined, according to the Concept Structure used and the derived types of description. Namely, there are rules to generate from Infolog schemas

not only its formal descriptions, but also descriptions in a fragment of natural language. This two kinds of descriptions are also derived from the specifications of dynamic aspects. From relational schemas only technical descriptions are derived. From HMIs definitions two kinds of descriptions are generated: a technical description and a description of its presentation and expected interactions (user manual). In this paper only the natural language description of Infolog schemas and the user manual will be focused. In [4] the complete work underlying this paper was reported.

Each set of rules describes how the nodes and links of the hypertext network for each schema are defined. In the rules the (schema-)text of the nodes is defined. Underlined words correspond to links. Like in BNF, words between '<' and '>' must be replaced, text between '[' and ']' is optional and '|' means alternative. By convention, generated text appears between '«' and '»'.

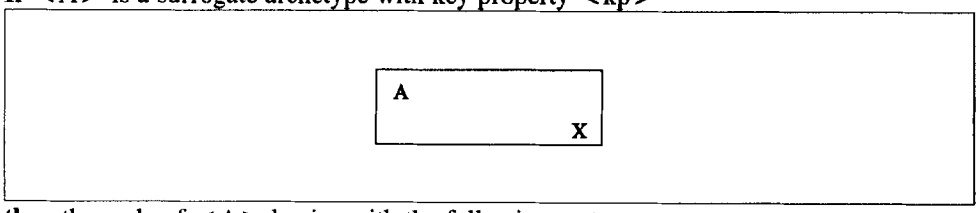
3.1 Natural language description of an Infolog schema

There is a node of the hypertext network for each archetype. That node contains text describing the archetype. The text depends on the archetype category, its attributes and its relationships. Hypertext links result from links between archetypes (either from archetype categories or designators). If in a node N, associated to an archetype A, an archetype B, with associated node M, is mentioned then there is a link from node N to node M.

The text of each node results from the concatenation of several fragments of text obtained from the application of different rules. To obtain the text of the node associated to a given archetype the application of the rules is done by the following order: first the rule that corresponds to the archetype category, next the rule for the attributes and finally the rules that corresponds to the categories of the associations from which the archetype is argument.

As an example, the next rule gives the description for surrogate archetypes. A surrogate archetype has a naming mechanism of its occurrences which is independent of those of other entities. Each instance of a surrogate archetype is uniquely identified by the value of its key property.

Rule 1:
If <A> is a surrogate archetype with key property <kp>



then the node of <A> begins with the following text:
«A <A> is uniquely identified by its <kp>.»
¤

An example of application of this rule:

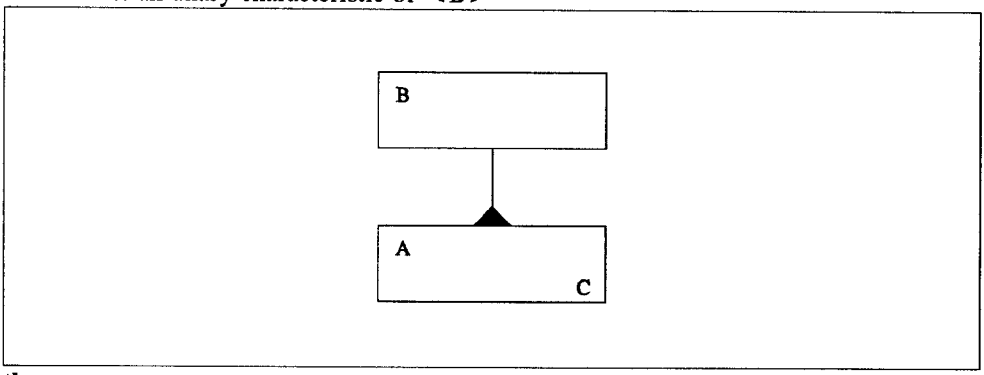
«A author is uniquely identified by its number.».

Note in the example that an 'An' should be used instead of 'A'. This and other "linguistic features" are not introduced in the current version of this work.

A typical relationship is the one-to-many relationship were a weak entity is involved. That corresponds to the unary characteristic in Infolog. In the next rule there is a one-to-many relationship between B and A and to identify an instance of A the related B instance is needed.

Rule 2:

If <A> is an unary characteristic of <B>



then

- In the node of <B>:

«A <B> has several <A> related.»

- In the node of <A>:

«There are several <A> for each <B>. Each <A> is identified by the related <B> in conjunction with the <pkp> of <A>.».

¤

To illustrate the application of this rule lets see how we would begin the description of *session*, since it is a characteristic of *conference* (see all the example of section 2.1):

«There are several session for each conference. Each session is identified by the related conference in conjunction with the number of session.».

This relationship between session and conference generates also a piece of text in the description of conference:

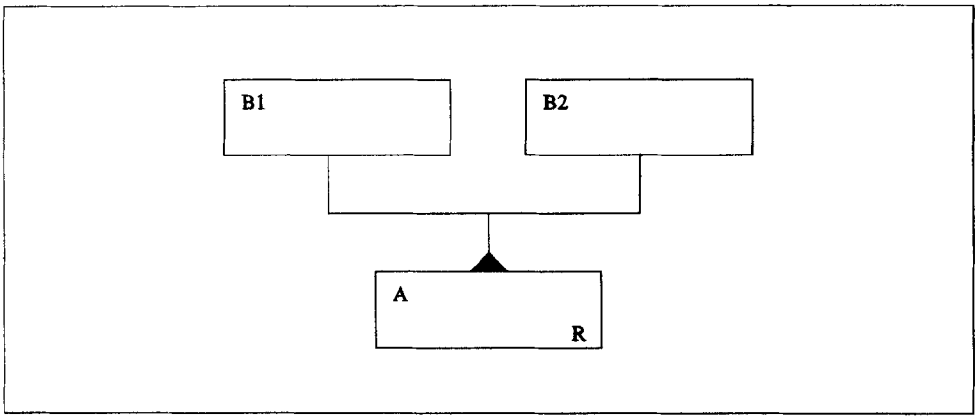
«A conference has several session related.».

That text in the node of conference establishes the link to the node of session.

Note here again the lack of concordance in 'several session' due to the yet limited linguistic features.

Another common relationship is the many-to-many relationship. The next rule establishes the description to be generated in the case that A is a many-to-many relationship between entities B1 and B2.

Rule 3:
If <A> is a binary relation of $<B_1>$ and $<B_2>$



then
- In the node of $<B_{1(2)}>$ appears the following text:

«A $<B_{1(2)}>$ can have several $\underline{<A>}$, one for each $\underline{<B_{2(1)}>}$.»;

- In the node of <A>:

«A <A> is related to a $\underline{<B_1>}$ and to a $\underline{<B_2>}$. There can be several <A> related to each $\underline{<B_1>}$, one for each $\underline{<B_2>}$. In the same way, there can be several <A> related to each $\underline{<B_2>}$, one for each $\underline{<B_1>}$.».

¤

In the case of the Conference example, *authorship* is a binary relation of *author* and *paper*. Applying the previous rule it results:
- in the description of *authorship*:

«A author can have several authorship, one for each paper.»;

- in the description of *paper*:

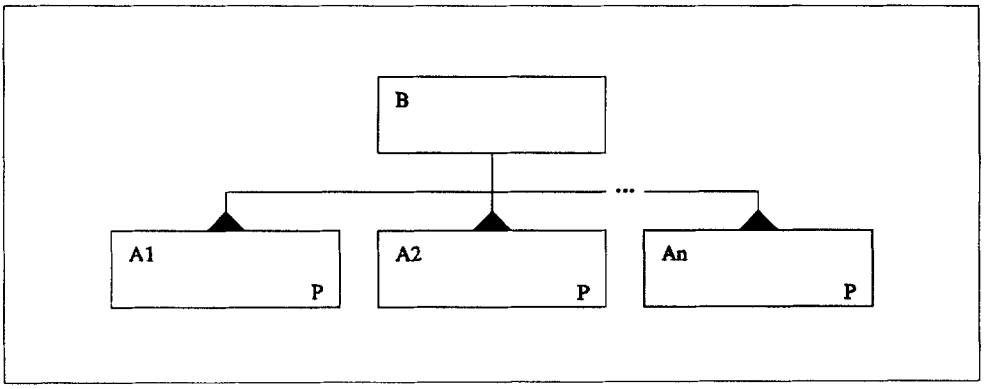
«A paper can have several authorship, one for each author.»;

- and the description of *authorship* begins with the text:

«A authorship is related to a author and a paper. There can be several authorship related to each author, one for each paper. In the same way, there can be several authorship related to each paper, one for each author.».

The is-a relationship is defined in Infolog by means of the specialization category of archetype. In the next rule "each A_i is a B" and a B can only be of one kind (A_1 or A_2 ... or A_n). The discriminant property of B indicates for each instance if it is an A_1 or an A_2 ... or an A_n.

Rule 4:
If $<A_1>$, ..., $<A_n>$ are specializations of <B> with discriminant property <dp>



then
- In the node of <B>:
«A <B> can be a $\underline{<A_1>}$ or a $\underline{<A_2>}$, ... or $\underline{<A_n>}$, depending on its <dp>.»;
- In the node of $<A_i>$:
«A $<A_i>$ is a $\underline{<B>}$.».
¤

Applying this last rule to the specialization of *paper*, *accepted-paper*, and being acceptance the discriminant property, results:
- in the description of *paper*
  «A paper can be a accepted-paper, depending on its acceptance.»;
- and the description of *accepted-paper* begins with the text:
  «A accepted-paper is a paper.».

Note that similar rules could be defined for the ER approach. If roles are attached to the arguments of the defined associations, the generated descriptions can be well adapted to the UoD.

4 Applying the paradigm

In this section a fragment of the generated hypertext network for the Conference example is presented. In section 4.1 the natural language fragment included in the Requirements Definition Document is represented. In section 4.2 the problem of generation of the User Manual is discussed and illustrated.

To represent the hypertext network without drawing the links, the nodes have been labeled with a number. Whenever there is a link from a word in a certain node to a node n, that same word is underlined and indexed by n. For instance "authorship^3" indicates a link to node number 3 associated to "authorship".

4.1 Requirements Definition Document

The Requirements Definition Document contains a natural language fragment description of the specification that results from the analysis phase (see fig. 5). It should be noted that it is easily understood by everyone. No technical knowledge is needed. Moreover it can be used to validate the specified model.

1 **author**
A author is uniquely identified by its number. It is characterized by its name and address. A author can have several authorship[3], one for each paper[2].

2 **paper**
A paper is uniquely identified by its number. It is characterized by its title. It is also characterized by its submission which is a conference[5]. A paper can have several authorship[3], one for each author[1]. A paper can be a accepted-paper[4], depending on its acceptance.

3 **authorship**
A authorship is related to a author[1] and a paper[2]. There can be several authorship related to each author[1], one for each paper[2]. In the same way, there can be several authorship related to each paper[2], one for each author[1].

4 **accepted-paper**
A accepted-paper is a paper[2]. It is also characterized by its presentation which is a session[6].

5 **conference**
A conference is uniquely identified by its number. It is characterized by its name. A conference has several session[6] related.

6 **session**
There are several session for each conference[5]. Each session is identified by the related conference[5] in conjunction with the number of session.

Fig. 5. Natural language fragment of the Requirements Definition Document

4.2 User Manual

The User Manual is produced from the HMI specifications. Those specifications are in a format described in [2]. In order to shorten this presentation neither the HMI specification format nor the rules to obtain the HMI descriptions from its definitions are presented. In [4] both aspects are extensively reported.

The objective of this section is to exemplify the format of the User Manual by presenting a fragment of the automatically generated description. That description consists on a list of the interfaces that appear to the user on the different situations to which he may go (see fig. 7 as an example).

One of the basic building blocks of the HMI specifications is the Card (the "Card" designation follows the one of the HyperCard objects). An example of a Card Widget is shown in fig. 7 where the card associated with archetype *paper* is presented.

Each HMI has the following structure: it has a Main Menu and several cards, where each card is an image of an object of the UoD. The Main Menu has several buttons, each one giving access to a Card (see fig. 6).

In fig. 6 the automatically generated description for the Main Menu of a HMI for the Conferences example is presented.

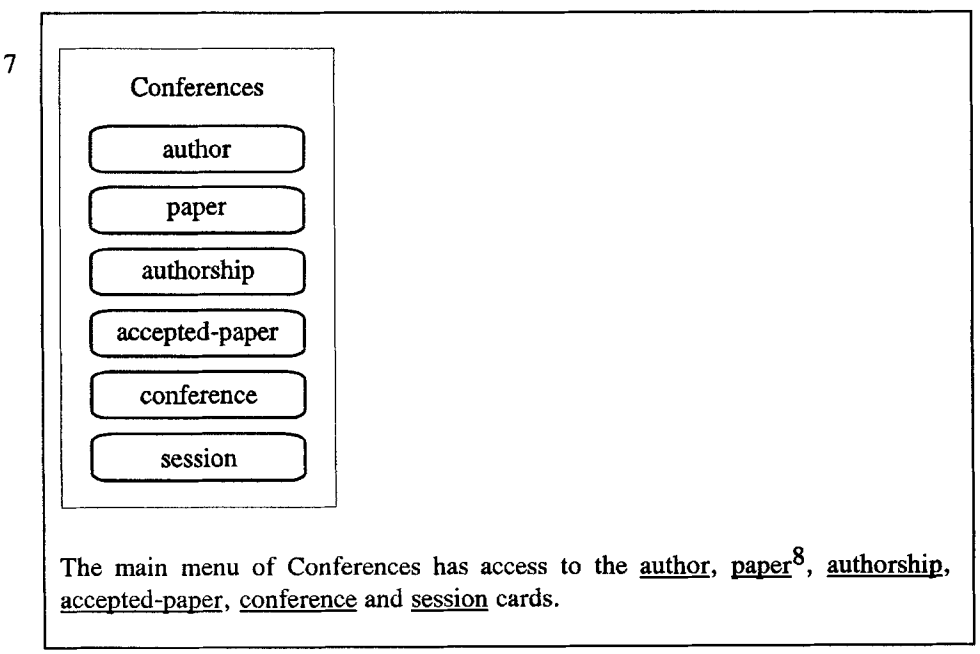
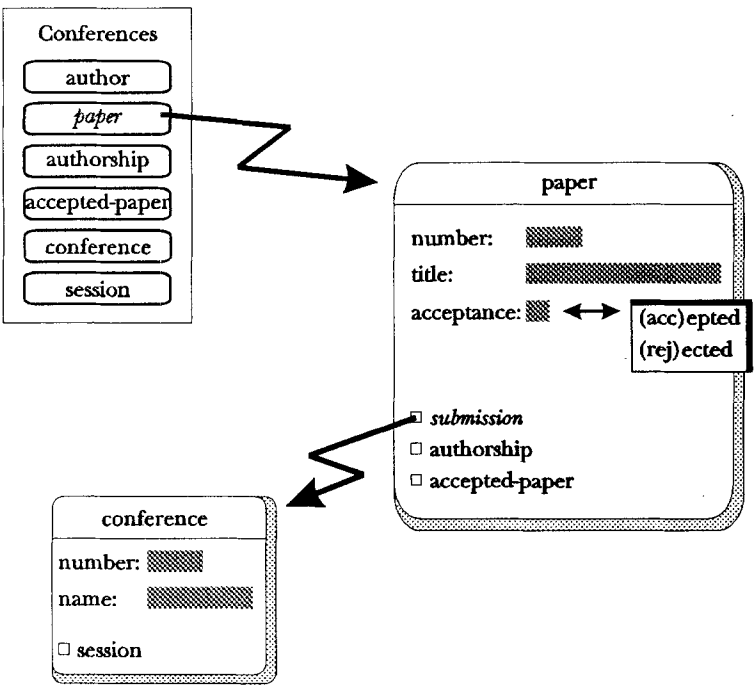


Fig. 6. Description of the main menu of a HMI for the Conferences example

A card has four components: title, properties zone, icons zone and messages device. The title is a text expression that informs the user about the archetype represented by the

card ("paper" in the case of the card of fig. 7). The properties zone contains an entry for each property of the archetype. It includes either key and non-key attributes. The icons zone includes a set of navigation points, each one leading to another card. This is the case of buttons *submission*, *authorship* and *accepted-paper* in the card of *paper* depicted in fig. 7. Activating an icon, the user invokes a new card through which he can access instances related with the data reported in the present card. For example, the icon *submission*, when set, maps an instance of the card corresponding to the archetype *conference*. The messages device is used for visualizing error, warning or information messages.

8



The field acceptance has the following possible values:

(acc)epted

(rej)ected

Clicking on the submission icon the card of conference is invoked.

Clicking on the authorship icon the card of authorship is invoked.

Clicking on the accepted-paper icon the card of accepted-paper is invoked.

Fig. 7. Description of the card of *paper*

5 Documentation Generation Support System

IS prototyping is widely accepted and increasingly used. The Caravela workbench [1] was developed with that purpose. In this paper "Documentation prototyping" is proposed. The computational tool that is being developed applying the proposed ideas will be integrated in the Caravela workbench, as depicted in fig. 8 where the global architecture of the Caravela workbench is presented. The tool proposed in this work is shadowed.

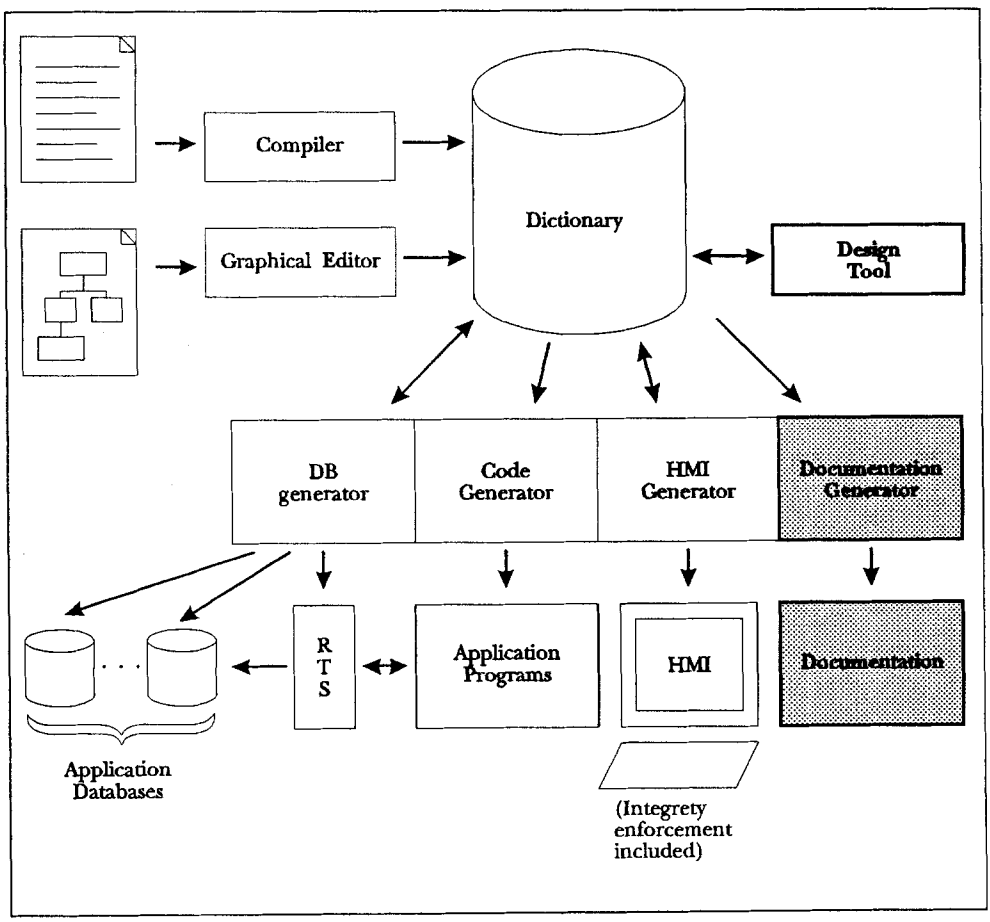


Fig. 8. Global architecture of the Caravela Workbench - adapted from [2]

In an abbreviated form it may be said that the input specifications, either in graphical or textual form, are stored in the dictionary. From this specifications, several 'products' are generated: relational schemas for each local database, default HMI, application code and documentation.

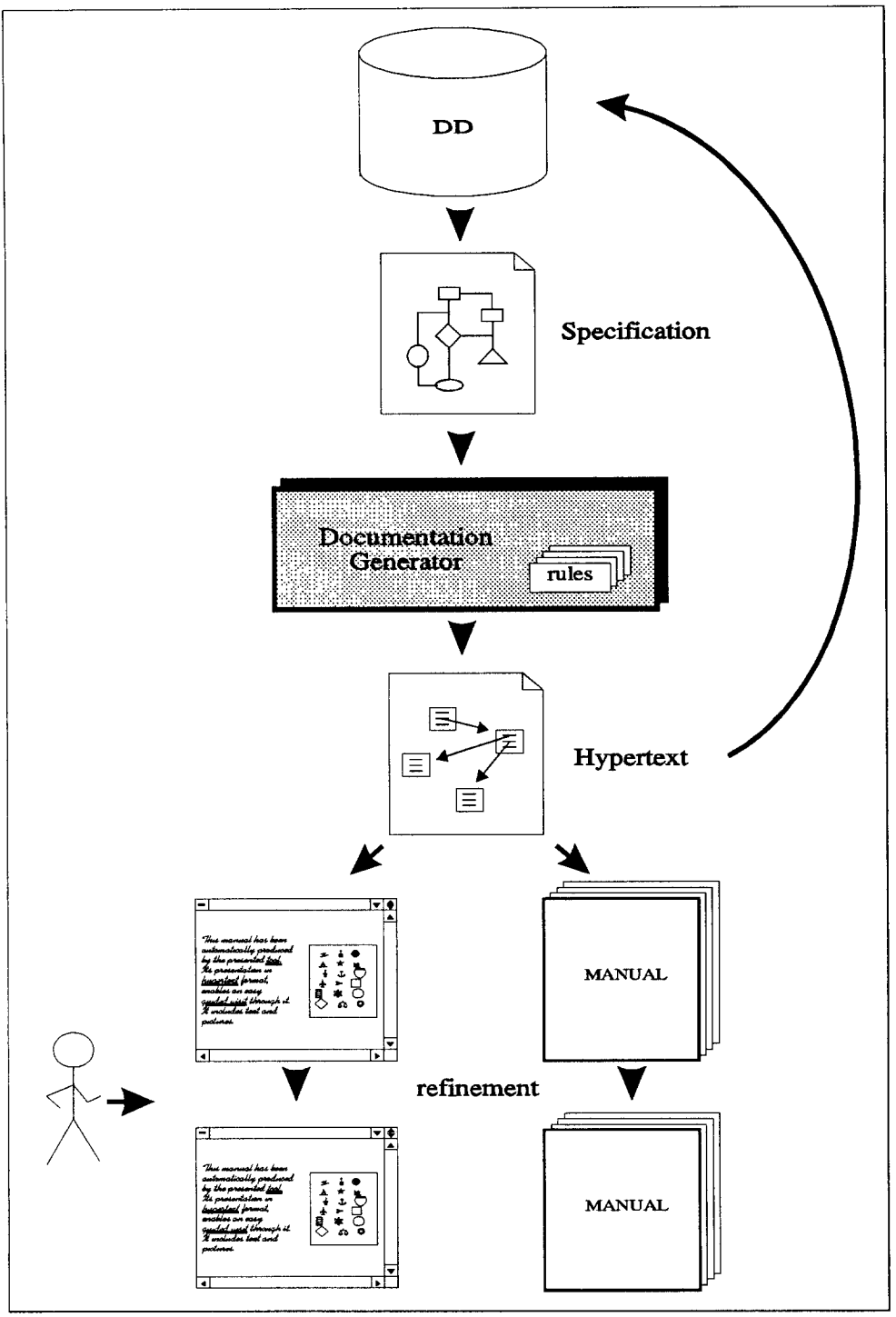


Fig. 9. Documentation Generator

In fig. 9 the Documentation Generator is depicted. From the stored specifications and using a set of rules (like the exemplified in 3.1), the Documentation Generator produces a representation, in hypertext format, of the documents. From this representation, several forms of presenting the documents can be produced. In particular, the hypertext one and paper supported (paper manuals).

6 Conclusions

The simplification of the task of documentation production requires the inclusion of the automatic generation of documentation in CASE tools for IS development. In this paper, a paradigm for the generation of documentation was proposed.

A demonstrator of the proposed ideas is being developed. It will be integrated in the Caravela CASE Workbench [1].

The exhaustive experiment of the generated User Manual is needed. The resulting feedback will serve to improve the present ideas. In fact, only with a great deal of experience it is possible to come up with ideas to improve this manual. Also the introduction of linguistic features in all the generated fragments of text is a foreseen future work.

References

1. R. Carapuça, L. Andrade and A. Sernadas, "A Database Design and Construction Workbench", in Computerized Assistance During the Information Systems Life Cycle, T.W. Olle, A.A. Verrijn-Stuart, L. Bhabuta (Eds), Elsevier Science Publishers B.V. (North-Holland), IFIP 1988
2. R. Carapuça et all., "Computer-Aided Database Development Tools: Extending the Caravela workbench", ESPRIT project report, IACIS, April 1991
3. P. Chen, "The Entity-Relationship Model: Toward an Unified View of Data", ACM TODS, 1:1, pp9-36, 1976
4. L. Jesus, "Geração Automática de Documentação para Sistemas de Informação", MsC thesis, IST, October 1991
5. J. Nielsen, "Hypertext & Hypermedia", Academic Press, Inc., 1990
6. J. Parkinson, "CASE - An Introduction, IBC Financial Books, 1989
7. A. Sernadas and C. Sernadas, "Infolog: An Integrated Model of Data and Processes", IFIP WG8.1 Meeting, York, July 1983
8. A. Sernadas e C. Sernadas, "Infolog 86", (in Portuguese) Revista de Informática, Vol. 5 Núm. 10, 1986
9. I. Sommerville, "Software Engineering", Addison-Wesley Publishing Company, 1989