A Specification Language and System for the Three-Dimensional Visualisation of Knowledge Bases

El Atifi, El Mustapha; Falquet, Gilles

How to cite

EL ATIFI, El Mustapha, FALQUET, Gilles. A Specification Language and System for the Three-Dimensional Visualisation of Knowledge Bases. In: 12th International Workshop, DSVIS 2005 - Interactive Systems: Design, Specification, and Verification. Gilroy S.-W. & Harrison, M.-D. (Ed.). Newcastle upon Tyne (United Kingdom). Berlin : Springer, 2006. p. 126–136. (Lecture Notes in Computer Science (LNCS)) doi: 10.1007/11752707_11

This publication URL: https://archive-ouverte.unige.ch/unige:46269
Publication DOI: 10.1007/11752707_11

A Specification Language and System for the Three-Dimensional Visualisation of Knowledge Bases

El Mustapha El Atifi and Gilles Falquet

CUI University of Geneva, 24 Rue Général - Dufour, 1211. Geneva - Switzerland
elmustapha.elatifi;gilles.falquet@cui.unige.ch

Abstract. In this paper we present models and languages to specify 3D interfaces for accessing knowledge bases. In this approach, a specification has an abstract and a concrete level. The abstract specification language describes the contents of nodes, obtained by selecting knowledge base objects, and different categories of links on these nodes. It serves to generate an abstract interface which is a 3D spatial hypertext. The concrete specification language associates styles and layout managers to the abstract interface components, so as to produce a concrete interface in which the nodes have a presentation and a position. This concrete interface is then easily translated in a 3D scene representation language such as VRML or X3D to be displayed.

Keywords. knowledge base, 3D interface, interface specification, spatial hypertext.

1 Introduction

The use of knowledge organisation systems is spreading rapidly to support or enhance new computerized applications. For instance, the implementation of the semantic web idea will require the development of ontologies; multi-agent systems must refer to common ontologies to communicate ; many e-learning environments include knowledge representation components for modelling the domain, the user profiles, etc. In addition, traditional databases or document repositories are often complemented with a knowledge representation layer to form machine treatable knowledge bases.

Several authors have shown that 3D visualisation techniques can enhance the usability of user interfaces, at least for certain tasks. In addition, 3D spaces offer a wide spectrum of opportunities to develop visual metaphors and interaction objects. However, the 3rd dimension has rarely been used to represent knowledge bases. This comes probably from the lack of tools to specify and implement such interfaces.

The aim of our work is to develop a specification language and system to produce three-dimensional interfaces to view and search the content of a knowledge bases. Moreover, we require that the generated interfaces have all the navigation

and representation features of 3D spatial hypertexts. That is, it must be possible to navigate the scene by activating hypertext links, and the geometric positions of the objects must represent some semantic relationship that hold between the knowledge base elements they represent.

In the rest of this section we give some background on knowledge representation and visualisation and on interface specification. In section 2, we introduce the abstract interface model and the abstract specification language. In section 3 we present the concrete model and language. In section 4 we briefly show the current implementation strategy. Section 5 gives our conclusion.

1.1 Knowledge Representation and Knowledge Bases

"Knowledge base" is a general term for a place that contains organized information about a chosen topic. There are many different kinds of knowledge bases. In artificial intelligence, they typically contain formal information on objects, facts or rules; it is thus possible to apply automatic processing to them. Knowledge bases that are used in companies for knowledge management purposes tend to be less formal. They are often made of an indexed document base describing "lessons learned", "best practices", FAQs, etc. [8], [20].

The knowledge elements present in knowledge bases may have different level of formalisation and abstraction. For instance, domain ontologies contain abstract formalized knowledge while text documents may contain factual (concrete) non-formalized knowledge, and databases contain factual formalized knowledge. To deal with this diversity, we selected the RDF-RDFS family of data / knowledge representation models [17].

RDF is a very simple semi-structured data model based on (subject, predicate, object) triples that form a semantic graph. The subject and object are resources (identified by their URI (Universal Resource Identifier)). The predicate is a name that indicates the relation holding between the subject and the object. RDF is mainly intended to create a semantic layer to describe web resources (e.g. HTML pages). It can be considered as the data layer of the semantic web.

RDFS adds a schema layer on top of RDF. RDFS mainly adds the notions of class (of resource), class instance, subclass, and property. RDFS schemas are similar to class specifications in object-oriented systems and languages. The RDFS layer is expressed in terms of RDF triples. For instance, a triple (R, rdf:type, C) indicates that the resource R is an instance of the class C.

Several query languages have been proposed for RDF, the most recent one being SPARQL [22]. They are mostly based on triple patterns. A triple pattern is a triple in which zero ore more components are replaced by variables. Such a pattern matches all the RDF triples that have the same constant values as the pattern. For instance, the pattern `(?X author Bob)` matches all the triples with predicate `author` and object `Bob`. And thus a query of the form `SELECT ?X FROM (?X author Bob)` returns all the resources X such that the author of X is `Bob`.

In the rest of this paper we will consider that a knowledge base consists of document fragments (resources) described by an RDF/RDFS layer. The documents are intended to store the non-formalized knowledge, while the RDF/RDFS layer stores the formalized facts (data) and general knowledge (concept definitions).

1.2 Data and Knowledge Visualization

During the last decade, human-computer interaction researchers have invented various visualization techniques to efficiently present and interact with different data types (linear structures, two-dimensional maps, three-dimensional worlds, temporal structures, multi-dimensional data, trees, and networks). Here we are particularly interested in techniques for visualizing the network structure of a formal ontology.The techniques used until now remain simple and traditional: hypertext interfaces, tabular views, graphs, etc.. As mentioned by Schneiderman [26], there is still much to do in this area. Apart from basic graph drawing, one can mention general techniques like fisheye views [25], or lenses to visualize large networks on a single screen. When a tree structure exists (or can be extracted from the network), techniques like hyperbolic trees [18] or 3-dimensional embedded objects can be used. These techniques have been evaluated with users to assess their effectiveness, see for instance [14] and [7].

The SemNet system [11] is one of the rare attempts at proposing a 3-dimensional view of a knowledge base. It represents concepts and their semantic links as a 3D graph. Another system called MUT [30] proposes a virtual museum metaphor with nested boxes for representing nodes and links from network-structured knowledge bases. Some systems let the user rearrange the visual elements according to their own cognitive model, for instance Workscape [1] or Web Forager [6].

Another knowledge visualization technique is the spatial hypertext. Spatial hypertexts are hypertext systems in which the usual hypertext navigation links are complemented with implicit spatial links [27]. The idea is that the spatial proximity of two nodes implicitly indicates a semantic relationship between these nodes. In spatial hypertexts the nodes are generally (rectangular) objects that lie on a 2D surface. However, some 3D spatial hypertext have been recently developed [13].

1.3 Interface specification techniques and languages

Although formal specification techniques have been extensively studied in the software engineering field, there are only few works on formal specification languages for user interfaces. Among these works we can cite [15] in which Jacob defines a language for specifying direct manipulation interfaces. Other authors have incorporated formal specification techniques (transition diagrams, Petri nets, grammars, etc.) in the interface specification or in dialogue specification, see for instance [5], [2], [21], [3] or [16]. In fact most of the interface specification languages can be found in an interface development environment. In particular,

model based environments such as MECANO [23], MOBI-D [24], MASTERMIND [29], TRIDENT [4] or Teallch [12] have a declarative specification language. They generally include a domain model, a task model, a dialogue model, a presentation model, and a user model. These models have often an abstract level and a concrete level that associates abstract elements to concrete interface objects (widgets, Java components, ...). More recently, interface specification language have been defined with XML, see for instance [19]. Nevertheless, these languages generally have no formal semantics and describe interfaces in terms of "standard" widgets like menus, input field, buttons, etc.

In the database interface field, several tools have been proposed for the declarative specification of Web interfaces, for instance Strudel [10] or Lazy [9].

1.4 A Two-Level Approach to the Specification of 3D Interfaces

Our aim is to specify and generate a 3D spatial hypertext that represents the content of the knowledge base.

The distance between the knowledge representation in the knowledge base model and its representation in the 3D interface model can be large. In particular if we consider 3D representations that are not just graphical views of the objects and relations of the knowledge base (KB). The 3D representation can, for instance, utilize metaphors or geometric relations to convey the meaning of the knowledge base. Thus it is difficult to directly specify the concrete representation of each KB element in the interface. In addition, we want to isolate the general structure of the interface (the semantic content of the interface objects and their links) from the presentation itself (object properties like color, shape, position, etc.). For this purpose, an interface specification consists of two levels: the specification of an abstract interface and the specification of a concrete interface.

The abstract interface specification defines a mapping from a knowledge base content (state) to abstract interface objects, which are abstract spatial hypertext nodes and links. It is expressed in terms of the knowledge base model and the abstract interface model. The concrete specification maps abstract interface components (nodes and links) to concrete components (3D objects and navigation actions).

2 Specification of the abstract interface

At the abstract level, the specification of a 3D interface for a knowledge base consists in specifying how to build a spatial hypertext that represents the content of the knowledge base.

2.1 Abstract interface model

The abstract interface model is based on a three-dimensional version of the spatial hypertext paradigm. In a 3D hypertext each hypertext node is presented

as a three-dimensional object in a 3D scene. Each node can have active elements to trigger link following actions.

An abstract hypertext node can be simple or compound, in this last case the node includes other nodes. The inclusion link between a compound node an a component can have attributes that will play a role in the positioning of the node.

Formally, thus an abstract spatial hypertext is a quadruple (N, I, V, S) where N is a set of nodes, and I, V, and S are sets of inclusion, navigation, and semantic links respectively.

A node is a pair (i, c) where i is a node identifier c is a node content, which is a hierarchy of typed elements. The element contents will be interpreted in the concrete interface where they can give rise to geometric or appearance properties, or texts, or references to other resources, etc. .

The different kinds are defined as follows:

- A navigation link is a pair (s, d) where s is the identifier of the source node, and d is the identifier of the destination node.
- An inclusion link is a quadruple (s, a, u, v) where s is the compound node, a is the element in which the node u is to be included, and v is a set of attribute-value pairs. The attribute values are indications that can be used at the concrete level to position the included node or to determine some presentation attribute.
- A semantic link is a 4-tuple(s, d, t, v) where s is the source node, d is the destination node, t is the type of the link, and v is set of attribute-value pairs. Here again, the attribute values will be interpreted at the concrete level for positioning or presenting the linked nodes.

2.2 Abstract interface specification

An abstract specification is a set of node type specification. A node type specification is a 4-tuple (*name*, *parameters*, *selection*, *content*). An abstract specification can be viewed as a parameterized query to the knowledge base, the results of which are then used to build the node content and links to other nodes. The selection expression is a selection expression of the knowledge base query language. Its free variables represent objects (resources) of the knowledge base.

The content specification is an ordered tree of element specifications. An element specification is a triple (t, l, c) where t is the element's type, l is a, possibly empty, link specification, and c is a content specification. The content specification can be

- a literal specification
- a formal parameter name
- a knowledge base variable
- a list of element specifications

In addition, the content specification c may contain a subtree c_{iter} called the iterated content. All the elements under the root of the iterated content belong

to the iterated content. The purpose of the iterated content is to define a part of the content that will be instantiated for each result yielded by the selection expression. Knowledge base variables may appear only in this subtree.

A link specification is made of a link category (*inclusion*, *navigation*, or *semantic*), a target node designation, a link type name, and a set of link attribute specifications.

Semantics The semantic function I maps a knowledge base state K, a node specification $N = (n, p, s, c)$, and a list of parameter values $A = (p_1 = a_1, \ldots, p_k = a_k)$ to an abstract node and a set of abstract links of the abstract interface model.

This function is formally defined as follows. Let $S = \langle s_1, \ldots, s_l \rangle$ be the result of the selection expression (with the parameter names replaced by their actual values) evaluated on the current state of the database. If the selection has n free variables $x_1, \ldots, x_n$, its evaluation's result is a list of n-tuples of the form $(x_1 : v_1, \ldots, x_n : v_n)$.

The following table defines the interpretation $I^r(e)$ of a content element for a given result tuple $r = (x_1 : v_1, \ldots, x_n : v_n)$ (and for the given parameter assignment A)

element	$I^r(e)$
constant c	c
parameter p_i	a_i
KB variable x_i	v_i
function $f(e_1, \ldots, e_m)$	$I(f)(I^r(e_1), \ldots, I^r(e_m))$
$<type>(e_1, \ldots, e_p)$	$<type>(I^r(e_1), \ldots, I^r(e_p))$

If the element contains a link specification $(cat, target, type, attr)$, its interpretation will yield a link of the appropriate category to the target node, with the specified type and attribute values. The target node designation is composed of a node type name n and a list of parameter specifications $(e_1, \ldots, e_k)$, where each e_i is an atomic element specification (constant, node parameter name, KB variable name, or function call). The target node is thus $(n, (I^r(e_1), \ldots, I^r(e_k))$. The attribute list is interpreted similarly (each attribute value is an atomic element).

2.3 An example

In this example we start with a simple knowledge base that contains concepts, relations between these concepts, and typed links from these concepts to URI of documents (a document can be an *example* or a *description* of a concept). The goal is to create a scene that is like an exhibition hall. In the exhibition, each concept is a stand with a large sign on top and posters with examples and descriptions on the walls. The examples must be on the left wall and the descriptions on the right one.

The abstract specification goes as follows:

```
abstract-node: Exhibition
selection: (?c rdf:type rdf:Class)
content: { inclusion-link: to: ConceptPresentation[c] }
```

The iterated content of the node is placed between { *and* }.

```
abstract-node: LabelAndSuperClassesOf [c]
selection: (c rdfs:label ?l).(c rdfs:subClasseOf ?c2)
content:
   <label>(l),
   {navigation link: to: ConceptPresentation[c2]
       type: "subsumption"
       attributes: (position: "top")
   }

abstract-node: ConceptPresentation [c]
selection: (c ?r ?c2)(?c2 rdf:type rdf:Class)
content:
   inclusion-link: to: LabelAndSuperClassesOf[c]
       attributes: (position: "center")
   inclusion-link: to: ExamplesOf[c]
       attributes: (position: "left")
   inclusion-link: to: DescriptionsOf[c]
       attributes: (position: "right")

abstract-node: ExamplesOf [c]
selection: (c ex:example ?d)
content: { inclusion-link: to: TextPanel[d] }

abstract-node: TextPanel [d]
content: <theText>(d)
```

3 Specification of the concrete interface

The concrete interface is a 3D spatial hypertext this means that the information is conveyed by node (3D) objects that have a shape, a color, a position, etc. and by their linking structure. In spatial hypertext there are implicit links, which are represented by the geometric proximity of nodes, and explicit navigation links, which are represented by anchor objects (buttons) that can lead the user to other nodes when activated. Explicit links can also have a graphical representation (for instance solid lines, or tubes, or roads).

The aim of the concrete specification is to define the interface objects that will represent the knowledge base. A concrete specification determines a mapping from an abstract interface to the concrete 3D objects and actions that form a spatial hypertext. A concrete specification is similar to a style sheet for document

presentation. But it must be more sophisticated about the positionning of sub-objects. For this reason, we have chosen to separate this aspect of the concrete interface from the rest, we take a layout manager approach, i.e. we consider that there exist layout managers that will take care of all the necessary computation to determine the position of each element of the scene. More precisely, each concrete node has its own layout manager take places all its sub-nodes.

A layout manager is essentially an algorithm that takes as input a set of (sub-)nodes and computes their location according to their content and to constraints represented by semantic links. A layout manager can have parameter of two kinds

- global parameters, for instance to set the minimum distance between two objects, or to set the number of lines and columns in a grid style layout.
- constaints (will be associated to implicit links)

3.1 Concrete interface model

What distinguishes the concrete model from the abstract one is essentially the addition of a geometry, an appearance, and a position for each node. The main difficulty in going from the abstract to the concrete interface is to compute the position of each node so as to represent the inclusion (of subnodes into nodes) and the semantic relationships of the abstract model. The positioning of nodes is represented by associating a layout manager to each node. The association between a node and its layout manager is in fact a binding of the layout manager parameters with values which can be constants (numeric or texte) or semantic relations.

Formally, thus a concrete spatial hypertext is a 6-tuple (N, I, V, S, M, B) which denotes (nodes, inclusion links, navigation links, semantic links, layout managers, a node to layout managers bindings)

N, I, V, and S are as in the abstract model, except that nodes are triples (i, c, p) where i is the node identity, c is its content tree, and p is a set of attribute-value pairs used for presentation, such as `shape`, `color`, `visibility`, etc.

A binding b from a node to a layout manager is a set of pairs (p, v) where p is a parameter of the layout manager and v is a value. The value of a parameter is either a simple value (number, string, etc.) or the name of a semantic link type.

Although we call it "concrete", this model is still more abstract than models like VRML, X3D, or Java3D because the positioning of the objects is not given by 3D coordinates but left to layout managers.

There are two strategies to translate a concrete interface into a 3D scene in one of these implemented models. In the static approach, the concrete interface is given as input to the different layout managers that compute the node positions and create a static scene. In the dynamic approach, the 3D scene is created with "active" components that dynamically recompute the object positions each time an event occurs. For instance, a hyperbolic tree layout manager must recompute the positions each time the user selects a new object to become the center of the view.

3.2 Concrete interface specification

A concrete specification is a triple (n, a, l) where n is the abstract node type to which this specification applies; a is a list of *attribute*: *value* pairs; and l is a layout specification.

The attribute values are either constants (as in `shape: Box`, `position: (5, 3, 2.5)`) or expressions computed from some values found in the abstract node (element contents or element attribute values), as in `size: ./size`.

The layout specification is comprised of a layout name and a list of bindings that determine either parameter values or which semantic relation to use for which type of geometric constraint. The binding may include some value translation. For instance, if the `color` attribute takes its value from a content element that has a text value, each possible text value must be associated to a color.

The semantics of a concrete specification is quite straightforward. The value of expressions is obtained by evaluating literal values, path expressions in the abstract node contents, and functions on these values.

3.3 Example (cont. from previous section)

```
concrete-node: Exhibition
shape: Box;
layout-manager: 2DSpringDistances(spring => "semanticRel")

concrete-node: ConceptPresentation[c]
shape:Box
layout-manager:BoxBorder(location => position("left"-> north,
   "rigth"-> south, "center"-> center))

concrete-node: LabelAndSuperClassesOf
shape: Panel
layout-manager:layout-clrtb-LinksRes(linkObject: "cone")

concrete-node: ExamplesOf
shape: Wall
layout-manager: Sequence

concrete-node: TextPanel
shape: Panel
layout-manager:HTMLViewer(content: ./theText)
```

Figure 1 shows a 3D interface generated from a knowledge base and the specifications shown here.

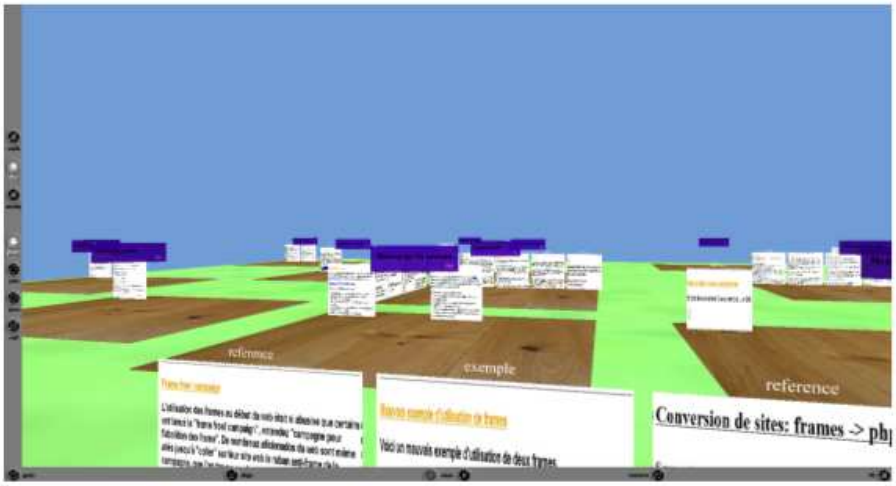

Fig. 1. A 3D view of a hyperbook knowledge base as an exhibition hall

4 Implementation

We have developed specification interpreters for abstract and concrete specifications. The abstract specification interpreter takes as input an abstract specification and a knowledge base and produces an abstract interface. The interpretation starts with a parameterless root node and then recursively interprets the nodes that are referred to (through any kind of links).

In order to re-use existing generation tools, the knowledge base is stored in a relational database and the abstract node specifications are translated to Lazy[1] node specifications. Then the Lazy system is invoked to generate the abstract interface as an XML file.

The interpretation of the concrete specification consists essentially in invoking the selected layout managers to compute the concrete node positions. This is a static approach (as mentioned in the concrete model description), the positions are defined once and for all. The concrete specification interpreter is a Java program that loads the abstract interface through an XML parser, transforms the abstract nodes to (partial) X3D nodes and then invokes the layout manager. The layout manager are Java classes that implement the `Layout` interface.

5 Conclusion

We have shown that a relatively simple specification language is sufficient to generate arbitrarily complex 3D scenes to represent the content of knowledge bases.

[1] Lazy is a declarative hypertext view specification language. It produces XML or HTML contents by querying the database and assembling the query results

The specification process has two phases: the abstract specification produces an abstract hypertextual interface and then the concrete specification generates the factual 3D objects and actions. These languages are declarative, and therefore more abstract than other procedural (even object-oriented) frameworks.

References

1. Ballay J.M. Designing Workscape: An Interdisciplinary Experience, In ACM Conference on Human Computer Interaction (CHI'94), pp. 10-15, 1994
2. Bastide, R., Palanque, P., A Petri Net Based Environment for the Design of Event-Driven Interfaces, 16th International Conference on Application and Theory of Petri Nets (ATPN'95), Torino-Italy, 20-22 June 1995.
3. Berstel Jean, Crespi Reghizzi Stefano, Roussel Gilles, San Pietro Pierluigi, A scalable formal method for design and automatic checking of user interfaces, Proceedings of the 23rd international conference on Software engineering, p.453-462, Toronto, Ontario, Canada, May 12-19, 2001.
4. Bodart F., Hennebert A.-M., Leheureux, J.-M., Vanderdonckt, J. Computer-Aided Window Identification in TRIDENT, in Proc. of 5th IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT'95 (Lillehammer, 27-29 juin 1995), K. Nordbyn, P.H. Helmersen, D.J. Gilmore and S.A. Arnesen (ds.), Chapman & Hall, Londres, 1995, pp. 331-336.
5. Bumbulis, P., Alencar, P.S.C., Cowan, D.D., Lucena, C.J.P. Combining Formal Techniques and Prototyping in User Interface Construction and Verification, in 2nd Eurographics Workshop on Design, Specification, Verification of Interactive Systems (DSV-IS'95). 1995, Springer-Verlag Lecture Notes in Computer Science.
6. Card, S., Robertson, G., York, W, The WebBook and the Web Forager: An Information Workspace for the World-Wide Web, In Proceedings of CHI 96, ACM Conference on Human Factors in Software, 1996.
7. Cockburn, A., McKenzie, B.3D or not 3D?: evaluating the effect of the third dimension in a document management system. In Proc. of the ACM CHI conference on Computer-Human Interaction, pp 434 - 441, 2001.
8. Conklin, J. Designing Organizational Memory: Preserving Assets in a Knowledge Economy, Group Decision Support Systems, 1996.
9. Falquet G., J. Guyot J., Nerima L., In *The World Wide Web and Databases*, International Workshop webDB'98, Valencia, Spain, March 1998, selected papers, LNCS 1590, 1998.
10. Fernandez, M., Florescu, D., Kang, J., Levy A., Suciu, D. Catching the boat with Strudel: experience with a web-site management system, In Proceedings of SIGMOD'98 Conference, 1998.
11. Fairchild, K. M., Poltrock, S. E., Furnas, G. W. SemNet: Three-Dimensional Graphic Representation of Large Knowledge Bases, in Guidon, R. (Ed.), Cognitive Science and its Application for Human-Computer Interaction, Lawrence Erlbaum, Hillsdale, NJ, USA, 1988.
12. Griffiths, T. et al. Teallch: a model-based user interface development environment for object databases, In User Interfaces to Data Intensive Systems,pages"86-96", 1999.
13. Grønbæk, K. Mogensen, P. Hypermedia in the virtual project room - toward open 3D spatial hypermedia. Proceedings of the eleventh ACM on Hypertext and hypermedia, San Antonio, Texas, United States, 2000.

14. Hornbk, K., Frkjr, E. Reading of electronic documents: the usability of linear, fisheye, and overview+detail interfaces. In Proc. of the ACM CHI conference on Computer-Human Interaction, pp 293 - 300, 2001.
15. Jacob Robert J.K, A specification language for direct-manipulation user interfaces, in ACM Transactions on Graphics (TOG), Volume 5, Issue 4, P.283-317, 1986.
16. Jan Van Den Bos, Abstract interaction tools: a language for user interface management systems, in ACM Transactions on Programming Languages and Systems(TOPLAS), Volume 10, Issue 2, P.215-247, 1988.
17. Klyne, G., Caroll, J. (Eds.) *Resource Description Framework (RDF): Concepts and Abstract Syntax.* W3C Recommendation. Retrieved from http://www.w3c.org/TR/rdf-concepts/ on February 5, 2005.
18. Lamping, J., Rao, R. , Pirolli, P. The Hyperbolic Browser: A Focus+Context Technique for Visualizing Large Hierarchies, in Proc. ACM CHI'95 Conf., New York, 1995.
19. XML Markup Languages for User Interface Definition, In The OASIS Cover Pages, Retrieved from http://xml.coverpages.org/userInterfaceXML.html on January 02, 2004
20. O'Leary, D. E. Enterprise Knowledge Management, in IEEE Computer, volume 31, 1998.
21. Palanque, P. Towards an integrated proposal for Interactive Systems design based on TLIM and ICO. In F. Bodart and J. Vanderdonckt, editors, Eurographics Workshop on Design, Specification and Verification of Interactive Systems: Informal Proceedings, pages 69–85, Belgium, 1996.
22. Prud'hommeaux, E., Seaborne, A. SPARQL Query Language for RDF. W3C Working Draft , Retrieved from http://www.w3c.org/TR/rdf-sparql-query/, on February 22, 2005.
23. Puerta, A. R. The MECANO Project: Comprehensive and Integrated Support for Model-Based Interface Development, In Proc. of the 2 Int. W. on Computer-Aided Design of User Interfaces CADUI'96 Namur, 5-7 June 1996.
24. Puerta, A.R., and Maulsby, D. Management of Interface Design Knowledge with MOBI-D, in proc of International Conference on Intelligent User Interfaces (IUI97), P. 249-252, Orlando, January 1997.
25. Schaffer, D., Zuo, Z., Greenberg, S., artram, L., Dill, J., Dubs, S., Roseman, M. Navigating hierarchically, clusetered networks through fisheye and full-zoom methods, ACM Transactions on Computer-Human Interaction, 3 (2), pp. 162-188, 1996.
26. Schneiderman, B. Designing the User Interface: Strategies for Effective Human-Computer Interaction. 3rd ed. Addison-Wesley, Reading, Mass., USA. 1998.
27. Shipman, F., Marshall, C. Spatial hypertext: an alternative to navigational and semantic links. ACM Computing Surveys, Vol. 31, No. 4, December 1999.
28. Sowa, J. F. Principles of Semantic Networks: Explorations in the Representation of Knowledge, 1991.
29. Szekely, P. et al. Declarative Interface Models for User Interface Construction Tools: the MASTERMIND Approach. In Engineering for Human-Computer Interaction, L.J. Bass and C. Unger (eds), Chapman & Hall, London, 1995, pp 120-150.
30. Travers,M., A visual representation for knowledge structures, Proceedings of the second annual ACM conference on Hypertext,147-158, November 1989.