ELSEVIER Artificial Intelligence in Medicine 18 (2000) 83–103

Artificial Intelligence in Medicine

www.elsevier.com/locate/artmed

An object-oriented design for automated navigation of semantic networks inside a medical data dictionary

W. Ruan *, T. Bürkle, J. Dudeck

Institute of Medical Informatics, University of Gießen, Heinrich-Buff-Ring 44, 35392 Gießen, Germany

Received 14 January 1999; received in revised form 3 May 1999; accepted 31 May 1999

Abstract

In this paper we present a data dictionary server for the automated navigation of information sources. The underlying knowledge is represented within a medical data dictionary. The mapping between medical terms and information sources is based on a semantic network. The key aspect of implementing the dictionary server is how to represent the semantic network in a way that is easier to navigate and to operate, i.e. how to abstract the semantic network and to represent it in memory for various operations. This paper describes an object-oriented design based on Java that represents the semantic network in terms of a group of objects. A node and its relationships to its neighbors are encapsulated in one object. Based on such a representation model, several operations have been implemented. They comprise the extraction of parts of the semantic network which can be reached from a given node as well as finding all paths between a start node and a predefined destination node. This solution is independent of any given layout of the semantic structure. Therefore the module, called Gießen Data Dictionary Server can act independent of a specific clinical information system. The dictionary server will be used to present clinical information, e.g. treatment guidelines or drug information sources to the clinician in an appropriate working context. The server is invoked from clinical documentation applications which contain an infobutton. Automated navigation will guide the user to all the information relevant to her/his topic, which is currently available inside our closed clinical network.

* Corresponding author. Tel.: +49-641-9941370; fax: +49-641-9941359.

E-mail addresses: wen.ruan@informatik.med.uni-giessen.de (W. Ruan), thomas.buerkle@informatik.med.uni-giessen.de (T. Bürkle), joanchim.w.dudeck@informatik.med.uni-giessen.de (J. Dudeck)

0933-3657/00/$ - see front matter
PII: S0933-3657(99)00030-5

Keywords: Medical data dictionary; Semantic network; Context sensitive help; Dictionary server; Infobutton; Navigation method

1. Introduction

1.1. Problem and solution

Modern hospital information systems (HIS) strive to comprise a complete electronic patient record. They can however no longer be considered as singular applications. Van de Velde [38] defines a modern hospital information system with the following words:

> An HIS serves essentially as a medium for communication between the diverse collaborating functional subsystems or units in a hospital. It acts therefore as skeleton to facilitate the integration of various systems operating at subsystem level.

In addition to patient data such information systems will offer access to an increasing variety of medical information sources. Such sources can comprise medical textbooks and guidelines as well as literature search programs such as MEDLINE. They can provide full library access, drug data sources, teaching programs and research databases. In some institutions the integration of medical contents has been formalized, e.g. in the American IAIMS (integrated advanced information management systems) program [17]. However, with more and more medical information sources available, information searching becomes increasingly tedious and cannot be performed quickly during daily routine any more. Approaches have been proposed in the shape of so called infobuttons [15,11]. The generic idea behind this approach is that data from an application the user is working with, are fed into a search mechanism which narrows the search results to the most appropriate information sources in the given circumstance.

Considering van de Velde's definition of an HIS as an integration skeleton it becomes clear that such infobutton mechanisms must be independent from any specific HIS application. Only then they can be integrated in many different existing or future HIS applications dealing with patient data. Furthermore, infobutton mechanisms should also be independent of any specific information source or guideline. They should rather be able to deal with as many different kinds of information sources as possible. Within this paper, we present navigation mechanisms and an implemented solution based on a medical data dictionary as means to achieve those goals. We coined the expression 'context sensitive information presentation' for this solution which allows us to follow links to different appropriate information sources based on terms derived from the HIS application [5].

In our case the clinical application must be merely complemented with an infobutton. Clinical staff, which record e.g. clinical findings or intensive care plans on a computer, can press this infobutton to retrieve information for a clinical term they have marked in their application. Such clinical terms may comprise, e.g. medical findings, medical events or procedures. The infobutton activates an Internet browser and sends the term of interest to the Gießen data dictionary server (GDDS). Receiving the search term the dictionary server starts searching the semantic network of the data dictionary and returns the semantically related website information to the Internet browser.

1.2. Hospital background

Gießen University hospital is a 1300 bed facility caring for about 38 000 admissions and 300 000 outpatient treatments per year. The Gießen University Hospital Information System is a comprehensive and continuously evolving HIS with its roots dating back to 1989. A central clinical database, maintained on a Tandem mainframe, still forms the backbone and contains most of the clinical patient data [35]. The information system itself has been converted to a client server architecture integrating a variety of commercial and non-commercial departmental applications as described by van de Velde [27]. A total of 2000 clinical workstations and printers spread throughout the hospital in wards, doctors' offices, outpatient clinics, theatres and partially at bedsides have access to the HIS. Starting in 1993, a variety of commercial information sources such as MEDLINE, clinical textbooks, etc. as well as in-house compiled information sources called the electronic books have been made available on clinical workstations as separate applications [1,8,32,33]. The concept has always been that all required clinical information sources should be made available on each single workstation, rendering possible what we termed one stop information shopping [32]. Many information sources have been converted to HTML (hypertext markup language), the presentation format used also in the Internet and an intranet has been established with Web browsers available on most clinical workstations.

1.3. Medical data dictionary

To provide an independent mechanism for the linking of on-line information sources to clinical applications a medical data dictionary has been used. Medical data dictionaries can be defined as

> A central thesaurus for the controlled definition of the medical vocabulary to be applied in an HIS, which is also capable to represent the semantic relationships existing between all HIS objects and to link the local vocabulary to standardized international nomenclatures and knowledge sources [34].

There are two common structures to arrange terms in medical data dictionaries (MDDs) [3,13]. The terms in early data dictionaries are grouped in strict hierarchies [22,26,39]. More modern data dictionary concepts allow semantic links besides hierarchy [16,23], overcoming the limitations imposed by purely hierarchic relationships. Another shortcoming of some elder data dictionaries was a limitation in the depth of a hierarchy and the number of items per hierarchy level [29].

Semantic structure is flexible and has no structural limitation. Medical terms can be linked with semantic relationships as needed. It allows for a simple and straightforward representation of real world conditions. Therefore it is comparatively easy to link any given medical term or procedure derived from clinical applications to any information source. On the other hand, navigating semantic networks is somewhat complicated. The method to represent the semantic network for navigation purposes should have no limitation either.

Gaining its first experience with the US derived PTXT (pointer to text) data dictionary [22] during an experimental HELP HIS installation, the staff of Gießen University medical information processing department has developed several of its own data dictionaries [3,28,30,31]. The data dictionary used in our project is based on the architecture of MDD-GIPHARM [31] which was originally designed to support drug charting and the implementation of knowledge based functions to monitor the prescription [7]. The MDD-GIPHARM architecture allows the definition of a vocabulary of medical terms, which can be linked by semantic relationships. In the following sections whenever we reference MDD this architecture is meant.

To allow flexible linking between on-line information sources and different clinical applications an independent web based dictionary server has been implemented on top of the MDD. When passive data dictionaries can be described as a thesaurus which can be used merely for lookup purposes, a dictionary server can perform active duties offering some kind of application programming interface (API) to different applications. Therefore it is independent of any given information system. The concept of the active dictionary service is described in Refs. [4,5]. We will reference our Gießen data dictionary server as GDDS.

2. Method

2.1. Graph theory

Graphs, by providing a means of explicitly presenting relations using arcs and nodes, have proved to be an ideal vehicle for formalizing association theories of knowledge. A semantic network represents knowledge as a graph with the nodes corresponding to facts or concepts and the arcs to relations or associations between concepts. Both nodes and links are generally labeled [25]. Therefore in the following sections, the terminology of graph theory is used to explain our method and algorithms.

A generic semantic network can be viewed as a labeled, directed, unilaterally connected simple graph shown in Fig. 1. Node A is said to have relationship R with node B if an arc labeled R begins at node A and ends at node B. Then node B is a neighbor of node A [24]. Node B and relationship R is called a *relation pair* of node A in this paper. Please note that node A is not a neighbor of node B because of the direction of the arc. In Fig. 1, node A has relationship r_1 with node B, relationship r_2 with node E and relationship r_3 with node C. Therefore, node A has three neighbors, thus three relation pairs B versus r_1, E versus r_2 and C versus r_3. Similarly, node B has three relation pairs H versus r_4, D versus r_5, E versus r_6. Node K has only one neighbor node A and thus one relation pair A versus r_{11}.

There are two traditional ways of maintaining a graph in the memory of a computer [18,24,37]. One way, called sequential representation of a graph, is by means of its adjacency matrix. The other way, called linked representation of a graph, is by means of linked lists of neighbors. The sequential representation of a graph by its adjacency matrix has a number of drawbacks. First of all, the size of the matrix may need to be changed when nodes are inserted or deleted. Furthermore, if the number of arcs grows linearly according to the nodes, the matrix will be sparse, thus lot of space will be wasted. Linked representation of a graph, based on pointers, overcomes the above shortcomings of matrix representation. However, pointers are prone to errors and are not part of Java programming language [10].

In this paper we present an object-oriented representation of a graph based on Java in which a graph is viewed as a group of objects that encapsulate a node and its relations with neighbors.

2.2. The semantic network structure inside the MDD

Basically the MDD architecture manages terms and their semantic relationships within a set of relational tables. In principle any RDBMS (relational database management system) can be used as a technical platform [31]. By definition we group terms in concept classes. For example a drug substance called Furosemide would be considered a concept. It would belong to the concept class, drug substance. The resulting semantic network can be presented as a labeled, directed,

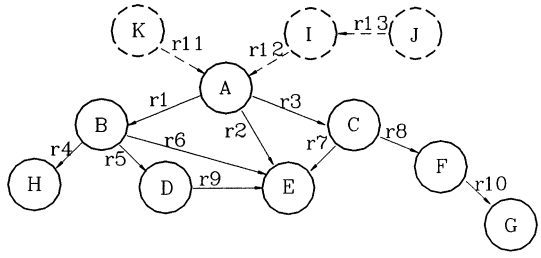

Fig. 1. A generic semantic network — a directed, unilaterally connected simple graph. Circles and arcs in solid lines represent the reachable part of the semantic network from node A. Those in broken line are not reachable from node A.

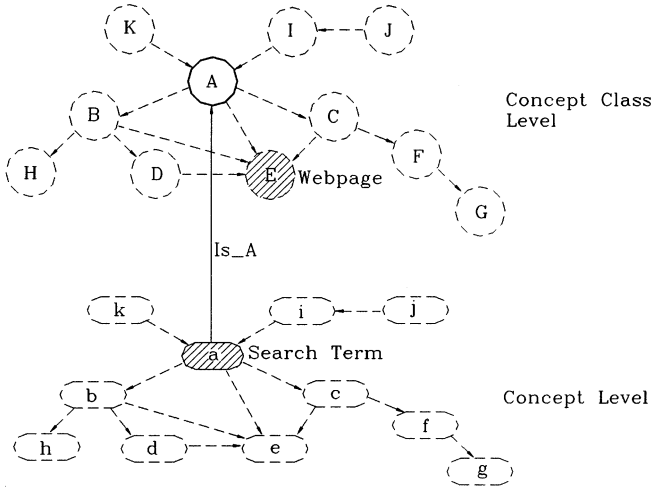

Fig. 2. The generic semantic network inside medical data dictionary. Abstract semantic network between concept classes is on top level, network between concepts on lower level. For clarification, most IS_A relation arcs as well as some of the arc labels have been omitted.

unilaterally connected simple graph in two levels (see Fig. 2). The semantic relationship between a concept and its concept class is the IS_A relationship. Each concept (identified with lower letter) maintains an IS_A relationship to the concept class with the corresponding capital letter. For clarification, most of the IS_A relation arcs as well as some of the arc labels have been omitted in Fig. 2 (The omitted arc labels between the concept classes should be the same as those between the corresponding nodes in Fig. 1). The concept classes represent a complete abstract semantic network (top level of Fig. 2). The concepts (lower level of Fig. 2) can but must not maintain all the relationships that are defined in the abstract semantic network. An automated mechanism allows the unambiguous and nonredundant integration of any new concept or concept class complete with its proper relationships into the MDD.

For this project we defined the concept class, *application term*, for any term which is sent by an existing or future HIS application. It has already been mentioned that application terms comprise e.g. medical findings, medical events or procedures which are used by the clinician for his documentation task and for which he requires context sensitive information. Furthermore, we defined the concept class *Webpage* for any medical information source to which a possible link should be established [36]. Such information sources could comprise, e.g. medical textbooks or their pages, treatment guidelines, literature source pages or drug information sources. Concepts belonging to the concept class Webpage will provide a uniform resource locator (URL) and can be considered the navigation goal of the context sensitive help function.

2.3. The dictionary service API

The goal of the dictionary service is an automated navigation mechanism through a semantic network. This navigation mechanism, starting from a given concept, has the task to find all the possible concepts which can be reached following the semantic net and which belong to a defined concept class. More specifically, in our case the task is to find any concept belonging to the concept class Webpage. To complicate the situation even more, the navigation mechanism has to memorize all successful navigation pathways through the semantic net. Those pathways will allow the clinician to assess quickly if the related information is helpful for his task. He may choose for example only to look at information sources which are in a specific way related to his term of interest.

In such a task, a generic design of the dictionary server is required, which is able to accommodate any possible semantic network layout. The concept class of the destination node of the automated navigation is fixed but the start node is variable depending on the search term which is submitted by the application. This implies variable pathways through the semantic network. Furthermore, the semantic network may be updated at any time with new concepts, concept classes and relationships which have to be considered in the search mechanism.

Therefore the GDDS is designed as an automated navigator of the semantic network that finds its way to web pages independent of the start point and the layout of the semantic network. It finds first the concept class of the search term, from which it retrieves all the reachable parts of the semantic network on concept class level and stores them in memory. Then it deletes nodes that do not lead to its navigation goal, the Webpage concept class. Thus a semantic network on concept class level beginning at the search term's concept class and ending at Webpage concept class is stored in memory. The navigating mechanism then sorts out all linked web pages on concept level according to the potential pathways on concept class level. Finally it returns the page information as well as the path information, i.e. the relations between the terms in the path, to the Web browser.

2.4. Object-oriented representation of the semantic network

As the name object-oriented implies, objects are key to understanding object-oriented technology. Objects store data (variables) and provide methods for accessing and modifying them. Every object is an instance of a *class*, which defines the variables and methods common to all objects of a certain kind [10]. The object's variables make up the nucleus of the object. Methods surround and hide the object's nucleus from other objects in the program. Packaging an object's variables within the protective custody of its methods is called encapsulation.

In a semantic network, the basic object is a node, a concept in our data dictionary. From the navigator's point of view, the neighbors and corresponding relations represent the state of a node object. An important method of a node object is to find its neighbors. We define the *node object* class for node objects in a semantic network (see Table 1).

Table 1
The definition of variables and methods of node object class

Variables	Methods
Node name	SetName( )
Neighbor list (hashtable)	GetName( )
	FindNeighbors( )

Every instance of the node object class contains a node name and a neighbor list as its variables. *Node name* is a string object holding the name of a concept or concept class in our data dictionary. *Neighbor list* is a hashtable that stores the relation pairs of a node. *Hashtable* is a Java data type that maps associated data values by keys. It is a fast way to access data using the associated key value [21]. Any non-null object can be used as a key or as a value. For example, both node name and relationship are string objects. Therefore, any relation pair of a node can be stored in a hashtable with the name of the neighbor as a key and relationship as a value. From this neighbor list hashtable, one can quickly retrieve neighbors (keys) and the corresponding relationships (values).

Every instance of the node object class contains methods that can access the variables. For example, *SetName*(), *GetName*() can be referenced from outside to set or get the value of node name. The method *FindNeighbors*() can find neighbors of the node and set the key value pairs in the neighbor list of the node object.

An instance of node object class is node object A in Fig. 1. It represents node A. The state of node object A is shown in Table 2. The node name is A. The neighbor list contains its neighbors node B, E and C as keys and relationships r_1, r_2 and r_3 as values, respectively. According to the definition of the node object class, all the information stored in the semantic net (Fig. 1) is memorized within the node Objects A–K. All the node objects possess the methods defined in the node object class.

To facilitate the navigation of the semantic network, a *node repository* has been implemented to store the objects from the node object class. The node repository is a vector, an extensible array of objects, the size of which can grow or shrink as needed to accommodate adding and removing items after the vector has been created.

Table 2
The state of node object A, an instance of node object class

A	
Node name (key)	Relationship (value)
Neighbor list	
B	r_1
E	r_2
C	r_3

In the following part we will discuss how to traverse the semantic network to fulfil the navigation objective. The semantic network shown in Fig. 2 is used as an example with node A as the start node and node E as the desired destination node Webpage.

2.5. Algorithms of active dictionary service

In this part we will discuss the steps to build the active dictionary service and the algorithms for each step. The whole task is divided into three major sections.

2.5.1. Retrieving and storage of the reachable semantic network on concept class level

Using the concept class of the search term as the start node, a complete search of all the reachable nodes on concept class level is performed. All the nodes met during the search are put into the node repository excluding however duplicates. The program flow diagram is shown in Fig. 3. This algorithm executes a breadth-first search on a directed simple graph beginning at node A.

The node repository holding all reachable nodes of the semantic network looks like a deformed semantic network shown in Fig. 4. First, node object A is inserted

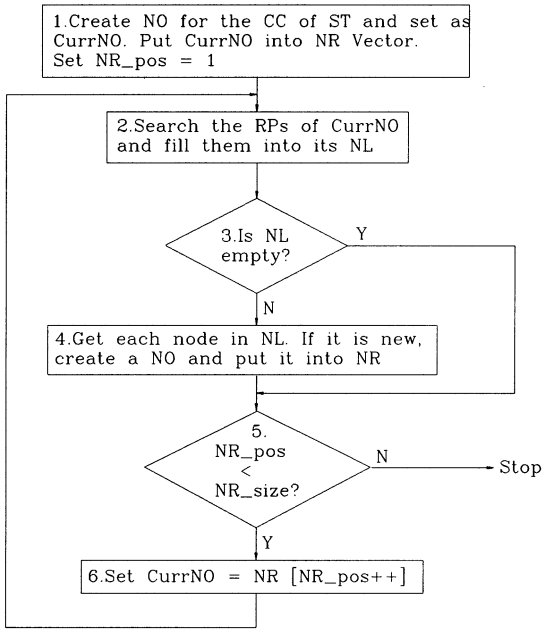

Fig. 3. Algorithm to construct the node repository. NR_size is an integer that counts the number of objects in node repository. NR_pos is an integer that points the position of the node in the vector that is being processed. CurrNO is the node object being processed in the repository. The abbreviations used in the flow diagram: ST, search term; CC, concept class; NO, node object; NR, node repository; RP, relation pair; NL, neighbor list; Curr, current.

into the node repository vector at the first position. Its neighbors node B, E and C are found and inserted into the neighbor list of node object A. Node objects B, E and C are constructed and put into the node repository since node B, E and C are new to the node repository. Then the neighbors of node B are searched and filled in node object B's neighbor list. New node objects are constructed and put into node repository. Step by step, node H, D, F and G are found and the corresponding node objects are constructed and put into the node repository.

2.5.2. Deleting sink nodes that are not goal of the navigation process

Obviously several pathways exist on concept class level between A and the predefined navigation goal E. But the navigator found also pathways that never lead to E, for example the pathway A — C — F — G in Fig. 2. The reason for such phenomena is the existence of sink nodes, nodes without outgoing arcs, in the semantic network. Our navigation goal concept class E is a sink node, as well as concept class H and G. All the nodes that only lead to sink nodes other than the navigation goal should be deleted so that only useful pathways are kept in memory. The program flow diagram is shown in Fig. 5. This algorithm executes a repeated process to delete nodes that are sink nodes other than the search aim since some nodes may become sink nodes when the existing sink nodes are deleted.

During the first iteration, node object H and G, which have empty neighbor list, are deleted. The arcs pointing to node object H and G are deleted as well. In the second iteration, node object F and the corresponding arc are deleted. In the third iteration, no node object is deleted. Therefore the iteration stops. The result is shown in Fig. 6.

Up to now the node repository stores sufficient information about the abstract semantic network beginning at node A and ending at node E Webpage. The information in the node repository covers all the possible ways to navigate from concept class A to concept class E on concept class level:

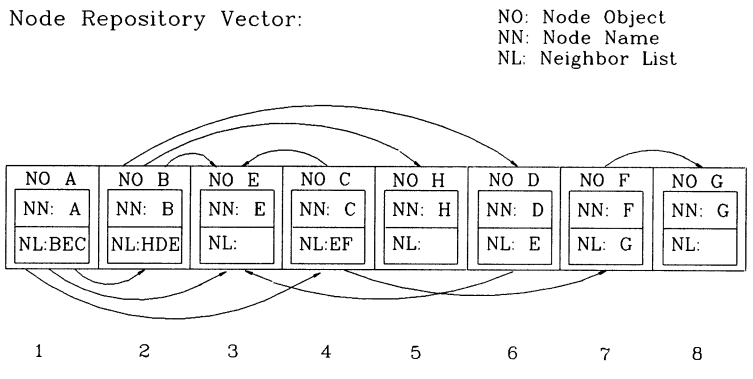

Fig. 4. Node repository of node object A built with the algorithm in Fig. 3. The node names in the neighbor list of each object and the arcs show the relations between the nodes and their neighbors. Node object E, H and G have no neighbors.

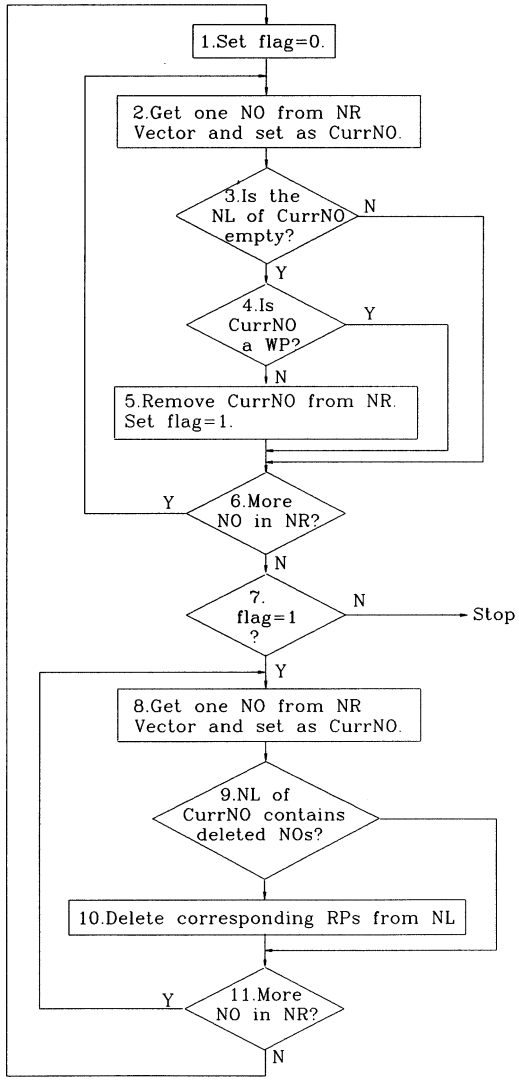

Fig. 5. Algorithm to delete sink nodes from node repository. The abbreviations used in the flow diagram: NO, node object; NR, node repository; WP, web page; RP, relation pair; NL, neighbor list; Curr, current.

(1) A — $<r2>$ — E;

(2) A — $<r1>$ — B — $<r6>$ — E;

(3) A — $<r1>$ — B — $<r5>$ — D — $<r9>$ — E;

(4) A — $<r3>$ — C — $<r7>$ — E.

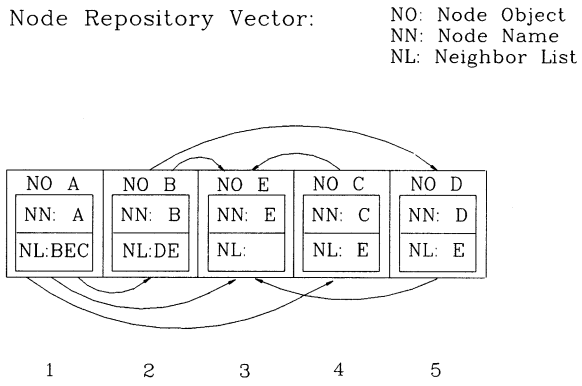

Fig. 6. Nodes in node repository after deleting sink nodes other than the navigation goal (node object E).

2.5.3. Sorting out paths on concept level according to paths on concept class level

The next step is navigating on concept level in parallel to the pathways defined in the abstract semantic network. The navigation paths on concept level can be represented in a general tree structure (Fig. 7). For each path on concept class level there might be many possible results on concept level. The number of results varies according to disparate search terms. The task of this part is to sort out the existing paths for a specific search term, i.e. to traverse the tree structure.

The conventional tree traversal algorithm includes preorder, inorder and post-order traversal algorithms that differ in the order of the node being processed [24]. Inorder and postorder algorithms are not suitable since we need a traversal performed from root to leaves. Preorder algorithm is used, however it needs a set of modifications since its function is to visit every node in the tree without remembering the routes — the branch information. However in our case the route

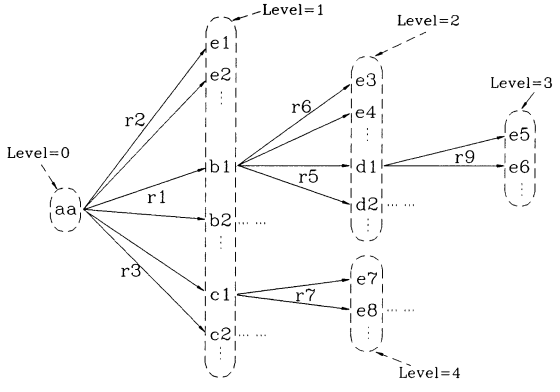

Fig. 7. A generic structure of general tree on concept level based on the paths on concept class level. The search level of each node is identified.

to the destination contains relevant information for the clinician and must be remembered. Therefore, we introduced a notation of search level as a complement to the preorder traversal algorithm.

Let us assume the following terminology: if N is a node with successors S_1, S_2, …, S_m, then N is called the *parent* of the S_i's, the S_i's are called *children* of N, and the S_i's are called *siblings* of each other [24]. Then the nodes are said to be in the same *search level* of each other if they are siblings.

Therefore in Fig. 7, search term aa is the root, the concepts from concept class E, e.g. e1, e2, …, e8, etc., are leaves. Node e1, e2, b1, b2, c1 and c2 are children of root aa, siblings of each other and therefore in the same search level. Similarly node e3, e4, d1, d2 are siblings of each other and in the same search level. The same is true for node e7 and e8, node e5 and e6, respectively.

To facilitate our search objective, another object class called concept object class is defined for the modified preorder tree traversal on concept level (see Table 3). It contains concept name, class name, in-relationship, URL and level as its variables and a set of corresponding methods.

Concept name represents the name of the concept. *Class name* is the name of the concept class to which the concept belongs. *In-relationship* contains the relationship between the last concept in the path with the current concept, i.e. the arc begins at the last concept in the path and ends at the current concept. *URL* is the address of a resource on the Web. If this concept has website information, it should have the correct address. *Level* is an integer to remember the search level of the concept during the traversal. The methods *SetParameters*() and *GetParameters*() can set or get the value of the above defined variables.

Fig. 8 demonstrates the flow diagram of the modified preorder tree traversal algorithm. Starting from the search term submitted by applications, the algorithm sorts out all the existing paths to the destination nodes according to the path information stored in the node repository. A path vector is used to store all the concept objects in a path. A level variable initiated as 0 increases by 1 after a node is processed. A stack, which is used to store a last-in-first-out (LIFO) queue of objects, is used to store the concept objects waiting for processing.

The level of a concept object decides its position in the path vector, in which objects are ordered with an increasing level. When a path is finished, i.e. a destination node with Web address is found, an unprocessed node (concept object)

Table 3
The definition of variables and methods of concept object class

Variables	Methods
Concept name	SetParameters( )
Class name	GetParameters( )
In-relationship	
URL	
Level	

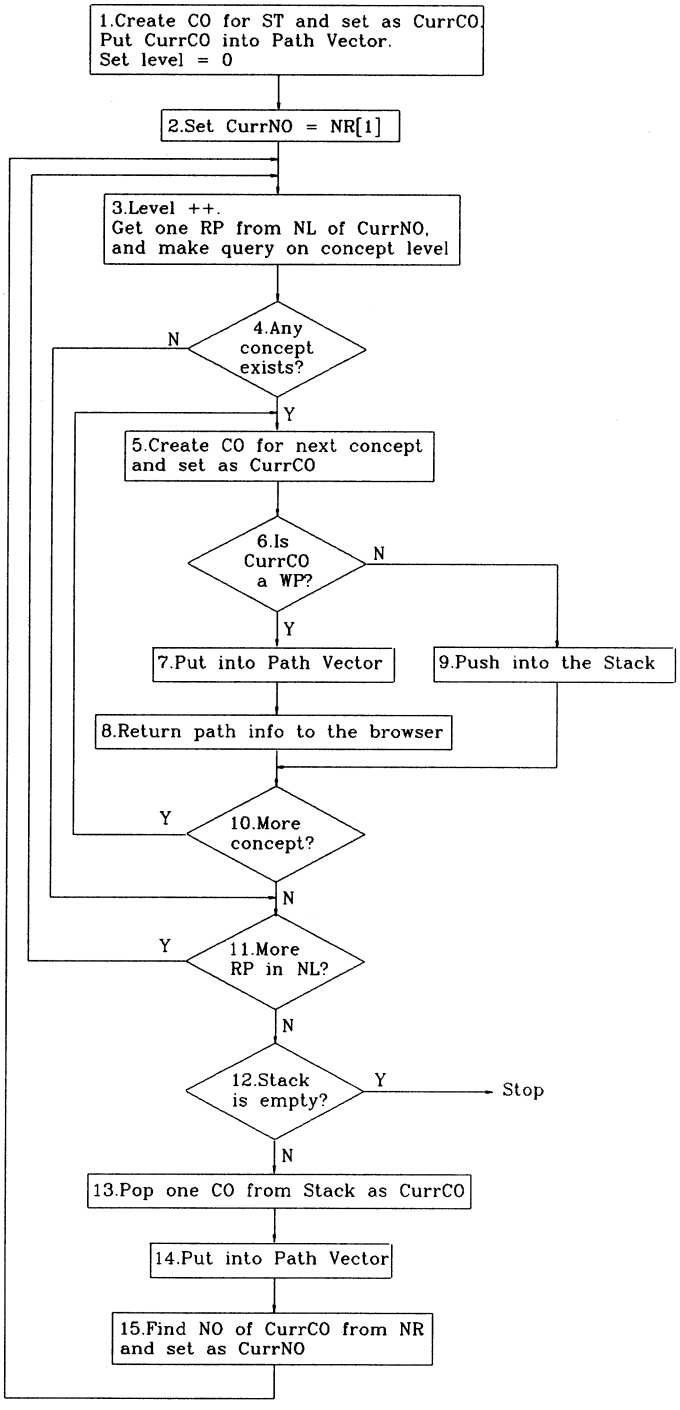

Fig. 8. Tree traversal algorithm to retrieve paths (Fig. 7) from the node repository. The abbreviations used in the flow diagram: ST, search term; CO, concept object; NO, node object; NR, node repository; WP, web page; RP, relation pair; NL, neighbor list; Curr, current.

is popped from the stack and put into its sibling node's position in the path. That means the new path will inherit the nodes with lower level and then continue with the current one.

When all three sections of the dictionary server algorithm have been completed, an HTML page is returned to the client browser which displays all the related on-line information sources plus various possible semantic relation paths.

3. Applications

For the first prototype an HIS application for nurses was chosen to supply active help functions. This application is used on intensive care units to document nursing acuity and medical procedures which have been performed for a patient [2,6]. Nursing acuity is documented using the TISS scheme (therapeutic intervention scoring system). TISS comprises a set of 75 medical and nursing procedures [20], e.g. dressing changes or urinary catheterization. We provide context sensitive information from our existing electronic guidelines for those procedures. As the first step we use the hygienic guidelines. The pages of the hygienic guidelines as well as the index terms of those guidelines have been mapped into the MDD.

We shall illustrate an example. Let us assume that the clinical user has submitted the expression 'inserting of wound drain'. (This is the procedure he is just recording for his patient). Receiving the search term our dictionary server starts its automated information retrieval process. First the server finds the concept class of the search term which is 'application term'. Then the server extracts semantic pathways on the abstract concept class level:

1. 'Application Term' — <is_related_to> — 'Index Term'
   — <info_available_on> — 'Webpage';
2. 'Application Term' — <info_available_on> — 'Webpage'.

The first (indirect) relationship on concept class level relates to the possibility to find information sources (Webpage) based upon the index terms of the electronic guidelines for hygiene. The second relationship relates to the possibility to find information sources directly. Both potential pathways are now searched on concept level using the given term 'inserting of wound drain'. The results can be seen in Fig. 9. The background of Fig. 9 is the adapted user interface of the TISS module with an infobutton. In the upper part of the browser window, 'inserting of wound drain' is the intervention sent from the application. It relates directly to a book chapter called 'Guideline dressing changes' and to an index term of the electronic book which is called 'Guideline Sterile Instruments'. This index term 'Guideline Sterile Instruments' has <info_available_on> book chapter 'Available sterile sets'. Thus the given intervention 'inserting of wound drain' points also to 'available sterile sets' indirectly. The corresponding information of each link can be chosen and displayed in the lower window.

Several other applications are currently under consideration for the same mechanism of context sensitive information delivery to the clinical user. The most promising area is drug therapy and the presentation of related drug knowledge. Several drug therapy charting applications are currently in routine use at Gießen University Hospital, for example in the medical ICU [9] and in the departmental pharmacies in our surgical building [40]. A variety of drug information sources are available for active search in our clinical network, e.g. a hospital formulary [1], as well as the German RedList formulary and DRUGDEX® from Micromedix [32,8]. We consider mapping those drug sources to a common drug vocabulary inside the data dictionary. Having achieved this step, all drug information sources availabe in HTML format could be offered simultaneously with the same infobutton mechanism according to the user's need.

Other potential application areas for context sensitive help functions are, e.g. the combination of lab results and lab test normal ranges, or the combination of microbiology results and guidelines on infectious diseases treatment.

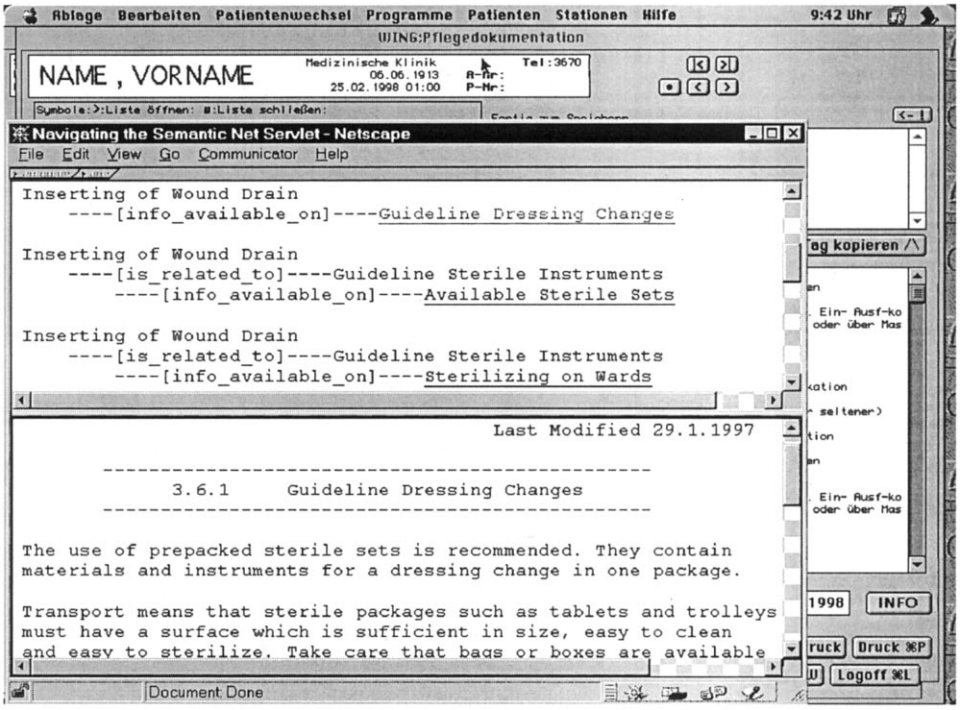

Fig. 9. Example of the search result of the search term 'inserting of wound drain'. The background is the user interface of the TISS module with infobutton.

4. Discussion

4.1. Structural limitation

A generic method of searching all the possible paths in a semantic network from a randomly chosen start node to a fixed destination node is described in this paper. The method uses an object-oriented representation of a semantic network and accommodates any semantic network layout as long as it can be represented as a directed, unilaterally linked, simple graph.

In our method, a semantic network is represented as a group of objects called node object that encapsulates a node, its neighbors and corresponding relations. The neighbors and the relations are recorded as relation pair in a variable length hashtable inside a node object. The node objects are stored in a variable length array called vector. The capacity of both hashtable and vector is identified by a 32-bit integer, i.e. almost no limitation of the number of nodes in a semantic network and no limitation of the number of neighbors at least in the following foreseeable years or decades.

4.2. Other limitations

The object-oriented representation of semantic network depicted in this paper is suitable for a directed, unilaterally connected, simply graph. A directed graph is said unilaterally connected if for any pair u, v of nodes in graph G there is a path from u to v or a path from v to u. A directed graph G is said to be simple if G has no parallel arcs.

In the medical informatics domain there are some exceptions, e.g. in some conditions parallel arcs are necessary in the MDD-GIPHARM. Drug substance and diagnosis/symptom are concept classes. A drug substance can have indication or have contra-indication to a disease. For example, arrhythmia is a kind of heart disease. Some drug substance is effective in treating one type of arrhythmia, but it may be contra-indicated for another type of arrhythmia. Therefore there are two or even more relationships (parallel arcs) between two concept classes. Another example for parallel arcs in a semantic network can be found in the Columbia Medical Entity Dictionary (MED) [14,12]. There the semantic relationships are always bidirectional and semantic attributes are paired with inverse attributes. For example, the two inverse relationships ‘has part’ and ‘part of’ are paired to result in a bilaterally connected graph.

Both types of parallel arcs cannot be accommodated directly in our search mechanism, since we use a hashtable which maps keys to values to store the relation pair. One concept class is mapped to only one relationship. Parallel relationships can not be stored in a hashtable. One compromise is to bundle the parallel arcs as one relation on concept class level. Then on concept level, when such a bundle relation is met it is replaced with a group of real existing relations. Another alternative is to switch to other data structure to store the relation pairs.

For the bilaterally connected graph it is much more complex to make automated navigation and much easier to cause directed cycles. Some rules should be considered to decide which one of the semantic pairs should be used for the navigation.

4.3. Adaptation of a tree traversal algorithm and its implications

In Section 2.5.3, our objective is to record all the nodes in all the possible paths from root to leaves in a tree structure. Although the preorder tree traversal algorithm is already well developed, it does not remember the branch position of each node. In our method a notation of search level is introduced for the preorder tree traversal algorithm, which identifies branch positions by setting a search level for each node met during the retrieval. The children of one parent node have the same level number that increases by 1 after a node is processed. According to the traversing order of the preorder traversal algorithm, the nodes ordered in a path from the root to a leaf should possess increasing level number. Therefore, the complete search paths can be recorded according to the search level of each node in the traversing process.

4.4. Advantages of a predefined concept class level

Retrieving semantic network pathways on a meta level of concept classes is a main principle in this method. In theory the search strategies in Sections 2.5.1 and 2.5.2 might be performed on concept level only. The reasons why we chose to extract the semantic network on concept class level are: (a) the semantic structure on concept class level supplies us with the complete abstract semantic network that forms the backbone of the MDD. On concept level some of those search pathways might not be found for an individual term or concept; (b) the second and third search section are repeated several times if the search is performed on concept level. On concept class level the semantic pathways are extracted only once and can then be reused to follow the relationships of the given concept.

4.5. Preventing directed cycles

Directed cycles that can easily cause problems for an automated navigation mechanism are not allowed in most semantic network based coding systems [19]. MED [16] and UMLS (unified medical language system) [23] both support acyclic graphs only. Our MDD however is not restricted in order to assign concept classes and relationships according to real world conditions. In addition, the content of the MDD may be updated by disparate people. Therefore it might be unavoidable to have directed cycles existing in the semantic network.

Fortunately, directed cycles do not cause endless circulation in our method. In the first step of extracting the semantic network on concept class level, the object-oriented design helps successfully to prevent circulation. From the start node, each new node found during the search is added to the node repository. Each node is put into the node repository only once, and each search is done only within

the direct neighbors of one node object. Therefore, the directed cycles are divided and put into different node objects. The circulation has no chance to start, but the directed cycles are still existing in the node repository — the deformed semantic network. In the second step (deleting sink nodes) the directed cycles cannot be deleted either, since each node in the directed cycles has a non-empty neighbor list. In the last step, the directed cycles could become active and cause the disaster of endless circulation. However this problem will appear only when the program does not remember the intermediate nodes it passes. In our search strategy, the main goal is to supply the intermediate search path to the user. Therefore, the circulation problem is solved easily by checking the current node if it has already existed in the search path before it is put into the path vector. If it has already existed, the search along this path will stop and a new path will start.

5. Conclusions

This paper described an object-oriented representation of a generic semantic network. Based on this representation a method of automated navigation of the network has been implemented. The method does not depend on any given layout of the semantic network. The resulting module called GDDS (Gießen data dictionary server) can act independent of our hospital information system. Any clinical application can be supplied with context sensitive help functions using a simple API. On the other hand, any on-line information sources can be mapped into the medical data dictionary using appropriate semantic relationships. As pointed out in Section 3, a huge set of information sources is available already at our university site for clinical staff. Making those sources accessible at just a click on an infobutton which delivers appropriate information for the current problem will render more efficient help for our clinicians.

Simultaneously our implementation demonstrated the power of active dictionary servers. We consider this development as a first step in designing an active dictionary server that can supply a multitude of applications with the required terminology and mapping services. Active dictionary servers may become a valuable asset for the improved cooperation of disparate clinical information systems in a confederated hospital information system environment. They will allow the smooth integration of commercial and noncommercial applications that do not maintain an own data dictionary.

References

[1] Brumhard M, Prokosch HU. Die elektronische Arzneimittelliste-Neue Wege zur Informationsbereitstellung im Gießener Universitätsklinikum. Krankenhauspharmazie 1996;17(4):129–33.

[2] Bürkle T, Michel A, Horch W, Schleifenbaum L, Dudeck J. Computer based nursing documentation — means to achieve the goal. Int J Med Inform 1998;52(1/3):71–80.

[3] Bürkle T, Prokosch HU, Michel A, Dudeck J. Data dictionaries at Giessen University Hospital: past–present–future, In: Chute CG, editor. Proceedings of the 1998 Annual Fall Symposium of the IMIA. Philadelphia: Hanley and Belfus, 1998. pp. 875–879.

[4] Bürkle T, Ruan W, Dudeck J. Darstellung klinischen Wissens am Arztarbeitsplatz: Der Arbeitskontext sollte berücksichtigt werden, In: Greiser E, et al., editors. Methoden der medizinischen Informatik, Biometrie und Epidemiologie in der modernen Informationsgesellschaft. München: MMV Medien & Medizin Verlag, 1998, pp. 222–225.
[5] Bürkle T, Ruan W, Michel A, Dudeck J. On the way to a web based hospital information system: concepts for the use of a medical data dictionary to present context sensitive information in an intranet environment, In: Cesnik B, et al., editors. Proceedings of the MEDINFO'98. Amsterdam: IOS Press, 1998, pp. 917–921.
[6] Bürkle T, Michel A, Horch W, Dudeck J, Schleifenbaum L. Does TISS pave a way towards the nurses care documentation? In: Pappas C, et al., editors. Proceedings of the Medical Informatics Europe, 1997, Porto Carras/Greece. Amsterdam: IOS Press, 1997, pp. 152–156.
[7] Bürkle T, Prokosch HU, Hussak G, Dudeck J. Knowledge based functions for routine use at a German University Hospital setting: the issue of fine tuning, In: Masys R, editor. 1997 Proceedings of the Annual Fall Symposium of the AMIA (formerly SCAMC). Philadelphia: Hanley and Belfus, 1997, pp. 61–65.
[8] Bürkle T, Prokosch HU, Dudeck J. Knowledge and information based functions in the Gießen hospital information system — present and future, In: Waegemann CP, et al., editors. Toward An Electronic Health Record Europe 1996 Proceedings. London: Omnipress, 1996, pp. 100–107.
[9] Bürkle T, Prokosch HU, Dudeck J. Steps towards integration of knowledge based functions into hospital information system, In: Brender et al., editors. MIE 1996 Proceedings. Amsterdam: IOS-press, 1996, pp. 286–290.
[10] Campione M, Walrath K. The Java Tutorial Second Edition: Object-Oriented Programming for the Internet. Reading, MA: Addison-Wesley Longman, 1998.
[11] Cimino JJ, Elhanan G, Zeng Q. Supporting Infobuttons with terminological knowledge, In: Masys DR, editor. Proceedings of the AMIA Annual Fall Symposium. Philadelphia: Hanley & Belfus, 1997, pp. 528–532.
[12] Cimino JJ, Johnson SB, Hripcsak G, Hill CL, Clayton PD. Managing vocabulary for a centralized clinical system, In: Greenes RA, et al., editors. MEDINFO 95 Proceedings. Edmonton: HC & CC, 1995, pp. 117–120.
[13] Cimino JJ. Coding system in health care. Yearbook of Medical Informatics 1995. Stuttgart: Schattauer Verlagsgesellschaft, 1995, pp. 71–85.
[14] Cimino JJ, Clayton PD, Hripcsak G, Johnson SB. Knowledge-based approaches to the maintenance of a large controlled medical terminology. J Am Med Inform Assoc 1994;1:35–50.
[15] Cimino JJ, Johnson SB, Aquirre A, Roderer N, Clayton PD. The medline button, In: Frisse ME, editor. SCAMC 1992 Proceedings. New York: McGraw Hill, 1992, pp. 81–85.
[16] Cimino JJ, Hripcsak G, Johnson SB, Clayton PD. Designing an introspective, multipurpose, controlled medical vocabulary, In: Kingsland LC, editor. Proceedings of 13th SCAMC. Washington DC: IEEE Computer Society, 1989, pp. 513–518.
[17] Clayton PD, Cimino JJ, Anderson RK, Roderer NK, McCormack M. Integrated advanced information management system (IAIMS): accomplishments from a global and local perspective, In: Prokosch HU, Dudeck J, editors. Hospital Information Systems-Design and Developments Characteristics; Impact and Future Architecture. Amsterdam: Elsevier, 1995, pp. 287–311.
[18] Goodrich MT, Tamassia R. Data Structures and Algorithms in Java. New York: Wiley, 1998.
[19] Hull R, King R. A tutorial on semantic database modeling, In: Cardenas AF, McLeod D, editors. Research Foundation In Object-Oriented and Semantic Database System. Englewood Cliffs: Prentice-Hall, 1990, pp. 1–33.
[20] Keene AR, Cullen DJ. Therapeutic Intervention Scoring System: Update. Crit Care Med 1983;11(1):1–3.
[21] Knuth DE. The art of computer programming, In: Sorting and Searching, vol. 3. Massachusetts: Addison–Wesley, 1973.
[22] Kuperman GJ, Gardner RM, Pryor TA. HELP: A Dynamic Hospital Information System. New York: Springer, 1991.
[23] Lindberg DAB, Humphreys BL, McGray AT. The unified medical language system. Meth Inform Med 1993;32:281–91.

[24] Lipschutz S. Data Structures. New York: McGraw-Hill, 1986.
[25] Luger GF, Stubblefield WA. Artificial Intelligence, Structures and Strategies for Complex Problem Solving. Reading, MA: Addison-Wesley Longman, 1998.
[26] McDonald CJ, Blevins L, Tierney WM, Martin DK. The Regenstrief medical records. MD Computing 1988;5:34–47.
[27] Michel A. Migration steps from a mainframe based HIS approach to an open HIS environment. In: Prokosch HU, Dudeck J, editors. Hospital Information Systems-Design and Developments Characteristics; Impact and Future Architecture. Amsterdam: Elsevier, 1995:267–86.
[28] Michel A, Prokosch HU, Dudeck J. Concepts for a medical data dictionary. In: Barber B, et al., editors. MEDINFO 1989 Proceedings. Amsterdam: North-Holland, 1989:805–8.
[29] Miller PL. Aspects of IAIMS implementation that require further research. JAMIA 1997;4(2):52–61.
[30] Prokosch HU, Amiri F, Krause D, Neek G, Dudeck J. A semantic network model for the medical record of a rheumatology clinic. In: Greenes RA, et al., editors. MEDINFO 95 Proceedings. Edmonton: HC & CC, 1995:240–4.
[31] Prokosch HU, Bürkle T, Storch J, Strunz A, Müller M, Dudeck J, Dirks B, Keller F. MDD-GIPHARM: design and realization of a medical data dictionary for decision support systems in drug therapy. Inform Biom Epidemiol Med Biol 1995;26(3):250–61.
[32] Prokosch HU, Puhle B, Dudeck J. One-stop-information-shopping — do we meet the information needs of the hospital staff? In: Barahona P, et al., editors. Proceedings of the 12th Medical Informatics Europe Congress MIE 94 Lisbon, 1994:362–5.
[33] Prokosch HU, Puhle B, Müller M, Wagner P, Junghans G, Marquardt K, Dudeck J. From HIS to IAIMS: expanding the scope of information processing application in a German University Hospital. In: Ozbolt J, editor. SCAMC 1994 Proceedings of the AMIA. Philadelphia: Hanley and Belfus, 1994:115–9.
[34] Prokosch HU, Dudeck J, Michel A. Standards for data dictionaries. In: Bakker AR, et al., editors. Hospital Information Systems: Scope—Design—Architecture. Amsterdam: North Holland, 1992:189–95.
[35] Prokosch HU, Dudeck J, Junghans G, Marquardt K, Sebald P, Michel A. WING – entering a new phase of electronic data processing at the Gießen University Hospital. Methods Inform Med 1991;30:289–98.
[36] Ruan W, Bürkle T, Dudeck J. Context sensitive information at the clinical workstation — the role of the medical data dictionary. In: Greiser E, et al., editors. Methoden der medizinische Informatik, Biometrie und Epidemiologie in der modernen Informationsgesellshaft. München: Medien and Medizin Verlag, 1998:214–7.
[37] Standish TA. Data Structures in Java. Reading, MA: Addison Wesley Longman, 1998.
[38] Van de Velde R. Hospital Information Systems: The Next Generation. Berlin: Springer, 1992.
[39] Whiting-O'Keefee QE, Whiting A, Henke J. The STOR clinical information system. MD Computing 1988;5:8–21.
[40] Wieczorek D, Prokosch HU, Neidhart B, Kreckel H, Dudeck J. One year of experience with EDP support for drug therapy at Gießen University Hospital. In: Reichert A, et al., editors. MIE 93 Eleventh Congress Proceedings. Israel: Freund, 1993:322–5.