Department of Computer Science
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

Presentation of the Content Structure for E-learning

Written Part of Dissertation Exam
Project of Dissertation Thesis

Mgr. Jana Katreniaková

Supervisor: Prof. RNDr. Branislav Rovan, PhD.

Bratislava, 2006

Thanks to my supervisor Prof. Branislav Rovan, PhD. for his support and interest in this work. Finally I would like to thank my husband for his support and encouragement in my research.

Contents

1 Introduction **1**

2 Motivation **3**
2.1 E-learning 3
2.2 Organizational Memory 4

3 Content Structure **6**
3.1 Objects of Content Structure 6
3.2 Process of Content Development 7
3.3 Model of Content Structure 8
3.4 Requirements for Storing and Visualizing 9

4 Storing and Visualizing **10**
4.1 Storing of Content 10
4.1.1 Related XML Standards 10
4.2 Visualizing Content Structure 12

5 Drawing of Content Graph **14**
5.1 Approaches in Graph Drawing 15
5.1.1 Topology-Shape-Metrics Approach 15
5.1.2 Hierarchical Approach 16
5.1.3 Visibility Approach 18
5.1.4 Augmentation Approach 19
5.1.5 Force-Directed Approach 19
5.1.6 Divide and Conquer Approach 19
5.1.7 Incremental Drawing of Graphs 20
5.2 Attributes of Content Structure Graph 20
5.3 Hierarchically Structured Graphs 23
5.3.1 Drawings of Compound Graphs 24
5.3.2 Drawings of Clustered Graphs 24

5.3.3 Visualizing of Hierarchically Structured Graphs 26

6 Existing Solutions **28**
6.1 Learning Management Systems 28
6.2 Personal Information Management Systems 29
6.3 Web-based Solutions 29

7 Conclusion **31**

8 Project of Dissertation Thesis **33**

Bibliogaphy **35**

Chapter 1

Introduction

In many fields is nowadays problem with lot of information, which need to be stored. There are areas, where another problem appears. The stored information has to be presented to other people. One of such fields, where these problems arise is e-learning. Nowadays the importance of e-learning is increasing, since it has many advantages in comparison to traditional learning. The term e-learning denotes a type of learning, where knowledge is mediated to students through electronic media. Existing e-learning systems [bla, ili, moo], besides other functions, provide information (learning content) to students. In the beginning, e-learning materials were created by simple rewriting of texts into an electronic form. Although e-learning materials are more interactive as before, the information is structured very weakly.

We may add structure to materials. The structuring of the learning content is necessary in order to use it efficiently. The content structure results from the process, how learning content may be developed. It consists of learning objects [McG04, QH00] with metadata [Dow02] and relations among them. In general, the content structure may be represented as a graph. For visualizing of the graph is usually used graph drawing, where different possibilities of drawing different type of graphs are explored. In [BETT99], the survey of the techniques used in graph drawing is presented. However, the content graph has an inherent hierarchy, thus the general graph is not appropriate for this purpose. We use compound graph [SM91] instead. The algorithms drawing this types of graphs are mainly heuristics [SM91, Rai, FEC95b], which try to minimize some aesthetic criteria. For the purpose of e-learning, there is also important to provide some basic operations on the graph, as introduced in [EH00]. However, this research is

usefull for the subclass of compound graphs – clustered graphs, which are insufficient for modeling content structure.

The rest of the survey is organized as follows: In Chapter 2 we present some fields, where the research is applicable. Then we present basic definitions of used objects, the formal model and requirements on content structure in Chapter 3. The requirements are handling two main areas – storing and presenting of the content structure. Possible solutions are outlined in Chapter 4. In this chapter we introduce the comparison of methods of storing content structure introduced in [DKR05b], as well as some possibilities of visualizing some information. Most of them were based on graph drawing research, thus we intend to examine this research in more detail in Chapter 5. We describe existing approaches of drawing of graphs in general, then we describe the properties of the content graph and outline some research handling graphs with similar properties. At the end, in Chapter 6, we present some existing solutions in fields mentioned in the first chapter. We conclude, that although storing of the content has been studied, the visualization of the structure is not treated sufficiently. However visualization of content structure is precisely the tool that can help us to avoid getting lost in the amount of stored information.

Chapter 2

Motivation

There is a lot of information, people have to know and use. Therefore it is very important to store the information in a meaningfull way. Nowadays the information is stored unordered. It is a problem when someone needs to find an information, or even present the information set to another person. Although in this work we are interested mainly in e-learning systems, there are also other fields, which require storing a large amount of information and presenting it to others. In this chapter we outline some particular examples:

- E-learning – presenting learning content to student
- Organizational memory – storing and using information about running an organization
- Personal memory – the problem seems to be the same as by organizational memory (but there is a smaller amount of information)

2.1 E-learning

There is no single definition of the term e-learning. E-learning most frequently means an approach to facilitate and enhance learning by means of personal computers, CD ROMs, Digital Television, Mobile Devices and the Internet. Some people and organizations under this term understand also learning via e-mails and discussion forums.

Besides many definitions of e-learning, there are also many usually ad hoc approaches to the development of e-learning systems. Still, the area would benefit from a more unified and systematic approach. In our approach we concentrate mainly on the content of e-learning systems and especially on

its structure and possibilities of storing and representing learning content utilizing this structure.

E-learning is dynamic: E-learner obtains up-to-date content in realtime.
E-learning is accessible: E-learner can access learning content at any time.
E-learning is interactive: E-learner takes an active role in the process and influences the learning path.
E-learning is collaborative: People learn one from another, e-learning connects learners and experts.
E-learning is personalized: Every e-learner selects activities from a personal menu of learning opportunities which are the most relevant to his or her background, job, and career.
E-learning is comprehensive: Experts provide carefully selected, complete, and abundant content and learning instructions from many sources.

The content used in e-learning consists of information, which has to be provided to the e-learner. As it was mentioned, the information can be various. It may consist of text, tests, pictures, videos, etc. The most important relation between these objects is prerequisite, which means, that the first information is necessary to understand the second one. Of course, there are more types of information and relations between them.

There exist several standards used in e-learning. The main purpose for introducing them was to enable sharing of learning objects among different e-learning courses. The most important is IEEE 1484.12.1 - Learning Object Metadata Standard [IEE02] and the metadate schema SCORM - Shareable Content Object Reference Model [SCO02] based on this standard. The important aspects of standards for learning objects and also other standards and schemata are presented in [HC00].

2.2 Organizational Memory

Organizational memory is the organization, creation, sharing and flow of information within organizations. The main task is to make the memory manageable, controllable and measurable.

Many sources consider the organizational memory to be a management strategy to maximize the Human Capital in an organization. It gives an insight into the memory distribution within the organization.

In perhaps every organization, it is very important to store the information about the knowledge, roles, processes and actions in the organization. Information technology has enabled organizations to generate and retain huge amount of information. Unfortunately, many organizations suffer from "infoglut". They have the information they need, but they do not know they have it. Or, knowing they have it, they cannot find it. There could be another situation – a former employee had the information or the knowledge how to do something. Loosing such a long-time employee means the organization could lose the information that only that staff member had.

The best possibility to memorize and keep the information (and find it, when needed) is a special information system responsible for managing and providing the information for the employees of the organization and the organization itself.

We would like to find out, in what ways the information technology can support business processes, but to do this we need to understand how and where information might be of use within organizations.

A standard information contained in organizational memory may include information repositories such as corporate manuals, databases, filing systems, and even stories (which were originally stored in written form). Additionally, the individuals are a prime location for retention of the organization's knowledge. However, organizational memory can be retained in many other places, including organizational culture, processes, and structures.

Chapter 3

Content Structure

If we want to provide some information to a consumer, we first need to arrange it to a form suitable for him or her. In this chapter we describe the content structure, its objects and model. To compare existing approaches and to create useful representation of content later, we have to understand the process of content development and resultant requirements on this representation.

3.1 Objects of Content Structure

In every kind of the motivation of this research, different terms for the same object are used. We decided to use the terms from e-learning, as introduced for example in [McG04, QH00].

In general, *an information object* (or a concept object) is defined as any single digital resource. *A learning object* consist of one or more information objects with metadata (*learning object metadata - LOM*). There is no standard for the size (or granularity) of learning objects. There are more definitions for learning object, however within the framework of this paper we will consider it as an object holding some properties. Learning objects together with relations defined between them create *the learning content*. We will use also a shorter term *content*. The digital resource is not so important from our point of view. More relevant part is its structure and LOMs. The formal representation of this structure is called *content structure*.

3.2 Process of Content Development

At the beginning we have information that we want to use for building the content. This information consists of information objects as they were described in the Section 3.1. These may be documents, database tables, multimedia, web pages, etc[1]. First of all we have to join these information objects into larger groupings, to describe them – add metadata [Dow02]. Then we define relations between these groupings. Each grouping represents a learning object and it can be either a set of several information objects or a particular information object itself. After adding relations to learning objects we obtain the content ready for presentation.

Creating relations between the learning objects can be made manually or automatically. The automatic solutions are based on:

- data mining [HMS01, HK01] or text mining (The practice of an automated search for patterns in large stores of data. Data mining uses statistical computational techniques, pattern recognition, etc.)
- building ontologies [Fen01, GPCFL04]

Various criteria for connecting learning objects using both manual and automatic methods may be used (e.g., similar words or keywords in connected learning objects). In some cases the relations are obvious from the structure of the document (e.g. links connecting webpages).

The overall schema of content structure development also described in [MBA03] is shown in Fig 3.1. The resulting structured information (learning objects and the relations between them) must be stored in some structure. We do not consider the way the information objects (digital resources) are stored and referenced. However, we assume that there exist methods for accessing and presenting them.

The stored content has to be visualized in some way. The visualization should enable different views on learning content as well as different levels of detail. Furthermore, it must be possible to change, to add and to delete relations and learning objects. Thus we are interested mainly in structuring (i.e. adding metadata and relations), visualization and presentation of the information.

[1] We consider information objects as the smallest indivisible units. If it was necessary to break some digital resource into smaller parts, we suppose it was done by pre-processing.

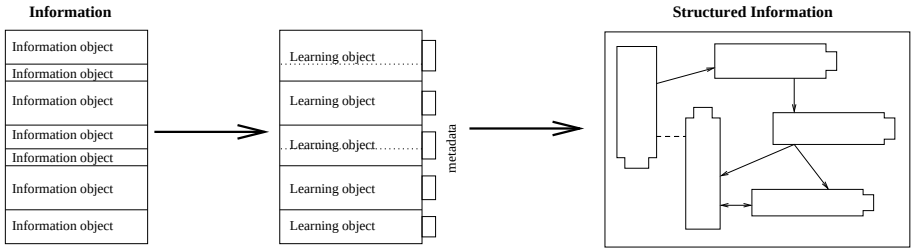


Figure 3.1: Structuring the information

3.3 Model of Content Structure

Formally a content structure can be represented by a graph. We present a *content structure*, where nodes represent learning objects (or rather LOMs) and edges represent relations between them. We started from the definition of the *Content graph* introduced in [WM03] where authors consider two sets of learning objects - abstract learning objects and displayable learning objects. Then relations are differentiated according to the type of connected nodes. This approach is useful, but it appears to be too complex for our needs. Therefore we introduced an easier model in [DKR05b], which was later extended by adding the possibility of clusterization in [DKR05a]. The content structure is defined as follows:

A *content structure* is a pair $CS = (NodeSet, EdgeSet)$, where

NodeSet consists of *nodes* containing identification number, node type (simple or cluster), pointer to resource, resource type (from some defined set $\mathcal{S}_{RT}$) and some other properties.

EdgeSet is the set of *edges* containing identification number, source and target node, edge type (type of the edge can be either from finite set $\mathcal{S}_{ET}$ or of type *contains*) and some application dependent properties.

Each node and edge is identified by a unique *id*. There are two basic groups of edges. Context edges represent semantic relationships between nodes and they have a type from the set $\mathcal{S}_{ET}$ [2] assigned. Edges of the type *contains* belong to the second group. They represent a relationship between an abstract node (cluster) and more specific nodes contained in that node. The model allows any number of cluster levels. However, we shall assume that the clusters are disjoint. This leads to a graph, which has inherent tree hierarchy beside adjacency edges. This graphs, so called hierarchically structured graphs, are analyzed later in the Section 5.3.

[2] The choice of the sets $\mathcal{S}_{RT}$ and $\mathcal{S}_{ET}$ depends on a particular application of the model.

3.4 Requirements for Storing and Visualizing

As mentioned previously, a content structure consists of learning objects and relations between them. Further we deal only with learning object metadata as the important part of learning objects. There are some aspects that we have to take into account when we are deciding how to store and visualize the content structure.

Storing

Storing LOMs: We have to find and use a suitable format. It should allow to describe different properties of a learning object.

Storing relations: In general relations can be represented by a graph. It is important to consider especially the following issues while choosing a format for storing relations:

- relation properties – support for differentiating among relations
- topology – allowed topology of relations
- structure location – are relations stored together with the corresponding information objects or separately?

Support for visualization: Formats for storing LOMs and relations must support visualization of structured information. LOMs' format must enable easy and fast access to the value of a particular property and the format for storing relations should support fast retrieval of relationships between particular learning objects. Thus, the most important criteria is the support for querying.

Visualizing

Easy readable view on the structure of the content is important. Other aspects of information visualization are the interactivity and dynamics of the visual representation. From out point of view, following parts of visualization are important:

Visualization of the content is dependent on the type of the resource and the associated file viewer is used.

Visualization of the content structure and browsing in it (see Chapter 5)

Changing views should allow the user to see only the important part in detail. During the browsing in the content, the view may change.

Chapter 4

Storing and Visualizing

In the previous chapter we defined the content structure and the main requirements on it. Now we shall discuss several well known approaches of storing content structure of the form defined above. We offer a brief summary, which was presented in [DKR05b].

Other important part is information visualization. The information visualization is the use of interactive, sensory representations (typically visual) of abstract data to reinforce cognition.

4.1 Storing of Content

In [DKR05b] we described four possibilities for storing content structure, which are from our point of view important (relational databases, directory structure, web space and structured documents). In Table 4.1 we list all requirements as defined in Section 3.4 and we summarize how do these representations fulfill them.

In general, databases offer good support for storing the structure and querying information. The directory structure and web space effect significant restrictions for the node/edge properties. Moreover the directory structure restricts also the topology of the structure. Using XML, any structure and properties can be stored, but the access to data is slow. The same holds also in the case of directory structure and web space.

4.1.1 Related XML Standards

The family of XML related technologies is extensive. In this section we describe some standards for storing learning content. There exist several

	databases	**directory structure**	**web space**	**XML**
storing LOMs				
- property types	any	restricted (FS dependent)	restricted	any
storing relations				
- supported topology	any	tree or directed graph (FS dependent)	any	any
- relation types	any	restricted (FS dependent)	restricted	any
- location	included in LOM or separate	implicit or included in LOM	included in LOM	implicit or separate
visualization				
- queries	fast	slow	slow	slow

Table 4.1: Comparison (FS = file system)

metadata schemata specifying learning objects' metadata. The main purpose for introducing them was to enable sharing of learning objects among different e-learning courses. We describe a standard and schema for representing LOMs in e-learning. Since the content structure may be seen as a graph, we introduce also an XML standard for representation of graphs.

IEEE 1484.12.1 - Learning Object Metadata Standard [IEE02] describes the LOM data model, also known as the conceptual data schema. This specifies what characteristics of a learning object may be described and how these characteristics should be recorded. It also defines how this data model can be customized by adding extensions (e.g. new vocabularies) or constraints (e.g. restricting the number of elements that may be used). Due to the standard, the metadata are distributed in nine categories (General, Life-Cycle, Meta-Metadata, Technical, Educational, Rights, Relation, Annotation, Classification). These categories group together data elements.

One of the most often used metadata schema based on IEEE Standard for Learning Object Metadata is **SCORM [SCO02]** (Shareable Content Object Reference Model). It is the XML standard for web-based e-learning. It defines how the individual instruction elements are combined on a technical level and sets conditions for the software needed for using the content. The metadata are distributed among subelements defined in IEEE Learning Object Metadata Standard. From the most important we mention following:

General: identifier, title, language, description, keywords
Technical: format, size, location, requirement
Educational: interactivity level, learning resource type, context, difficulty, typical learning time, context, description
Relation: kind, resource

GraphML [graa] is a graph format based on GML[1]. It is capable to store simple directed, undirected and mixed graphs. Advanced features enable to describe clustered graphs, hyperedges and node ports. There are several extensions – attributes holding parsing meta-data, possibility to add specific XML attributes and complex types to the language. An interesting extension is the LEDA (C++ class library dealing with graphs) extension, but it currently supports only the most common types of graphs.
GraphML and other XML formats (GXL, XGMML, GraphXML) are still not as well supported as widely used non-XML formats Dot and GML.

4.2 Visualizing Content Structure

In this section we describe possibilities and tools for visualizing the content structure as it was described in Section 3.3. Formally the content structure is a directed graph. The possibilities of visualization of graphs (not the graph drawing in general) are discussed in [HMM00]. In this section we offer an overview of some tools, that can be used for visualizing the content structure.

Most of the tools are built on principles of graph drawing. Therefore we consider graph drawing to be also a good solution for the visualization of the content structure. It is a general way, how to visualize some structured information. In Chapter 5 we introduce this area of research in context of drawing the content graphs.

The Thinkmap [Inc05] is a commercial system for visualizing different data and information. Since the system helps viewers to understand the architecture of the information and thereby find the information they need, it can be also used in organizational memory management.

It provides four types of visualization of the information, thus it can offer multiple views on the same data. The visualization is especially designed for these types of information:

[1] Graph Modeling Language is a frequently used non-XML language for describing graphs.

- Spider display: for browsing relational information without hierarchy.
- Hierarchy: for visualizing the hierarchy relation.
- Clustering: for groups of related entities, the thickness of the edge represents the strength of cohesiveness of relationships in the cluster.
- Chronology: for visualizing data, where time is an important facet.

The system can visualize data sources that can be both structured (SQL, XML) and unstructured (files, web-pages, e-mails). Thinkmap provides data to unified interfaces, that allows users to search and browse in it.

The Thinkmap also enables users to create their own customized Thinkmap applications with their data, and to integrate this information with their enterprise applications. The examples of such usage of Thinkmap are Visual Thesarus, Sony Music Licensing or Ecosystem Explorer.

Although the Thinkmap seems to be the best of the listed solution, the tool is not sufficient for drawing the content structure. Even if it supports visualizing of clustering, the problem with visualizing graph obtained by combining the clustering with general graph remains unsolved.

Prefuse [HCL05] is an open source user interface toolkit for building highly interactive visualizations of structured and unstructured data. This includes any form of data that can be represented as a set of entities (or nodes) possibly connected by any number of relations (or edges).

This toolkit is based on graph drawing and uses different methods (see Chapter 5) and algorithms to visualize different types of information.

Process of visualizing in Prefuse starts with abstract data in some canonical form. Most of the applications use a graph data set in XML files. It supports any number of attributes in every type of entity (Node, Edge, Aggregate).

Prefuse is used, for example, for visualizing social networks (Orkut etc.).

Prefuse, as the previous tool, do not support visualizing graphs containing inherent hierarchy (e.g. clustered graphs, or compound graphs as defined later in Section 5.2).

InfoVis [Fek04], GraphViz [grab] and others are software packages aimed at simplifying the development of Information Visualization Systems. There are many of such toolkits for visualization of graphs, tables, trees, structured documents and other structured information. All of them are drawing the simple structure, but extensions as clustering or other views are not possible.

Chapter 5

Drawing of Content Graph

Graph drawing is motivated by applications that require visualization, navigation, fabrication, or beauty of its physical or conceptual artifacts that have been embedded into a graph's structure. It is a key ingredient in technologies as varied as VLSI circuit design, social networks, user interface design, software engineering, computer networks, e-commerce, cartography, and bioinformatics.

Basic rules for graph drawing algorithms, the drawing must satisfy, are called *drawing conventions*. The most often used are constraints on the design of edges (*polyline drawing*, *straight-line drawing*, *orthogonal drawing*), placement of vertices (*grid drawing*) and general constraints on the drawing (*planar drawing*, *upward drawing*).

To these basic assumptions, we can add other *aesthetic* criteria, that we would like to apply as much as possible. Among others, there are criteria concerning the total area of the drawing, its aspect ratio, on the edge length (total, maximum, uniform), number of bends on edges (total, maximum, uniform), number of crossings of the edges and a criteria of symmetry.

Drawing conventions and aesthetic criteria are requirements, which the whole drawing has to satisfy (or would be nice if the graph would satisfy them). On the other hand, there are also *constraints*, which can refer to a subgraph or subdrawing. For example, we want, that a given vertex to be in the middle or in exterior of the drawing. Or we request, that some vertices (contained in a cluster) are drawn together. Of course, there are many possibilities on constructing some constraints (they are application dependent).

As supposed in [HE] the criteria were not originally based on experimental data. There might be better criteria which can play an important

role in graph understanding. In this article, authors studied people reading graphs and concluded, that some existing criteria (minimization of bends, minimization of crosses per edge etc.) are really essential.

In this chapter we discuss possibilities of drawing a content graph defined in 3.3. First of all we describe different approaches used in graph drawing in general. Then we describe our graph and find out the requirements (aesthetic criteria, constraints, other specific requirements) the "good" drawing has to fulfill. At the end, we outline some relevant researches, dealing with graphs similar to the content graph.

5.1 Approaches in Graph Drawing

In this section we introduce some of the approaches most used in graph drawing. For more information about basic approaches in drawing of graphs see for example [BETT99].

5.1.1 Topology-Shape-Metrics Approach

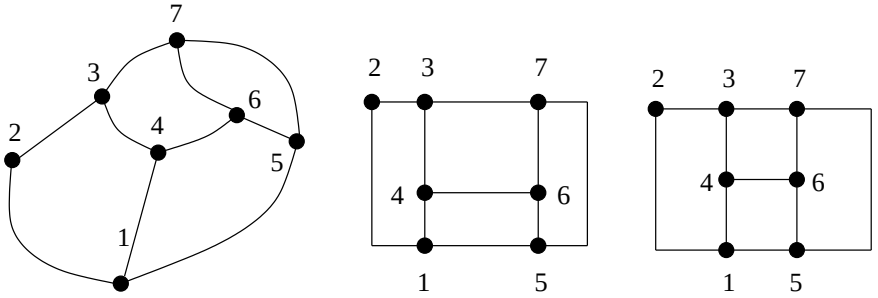
The Topology-Shape-Metrics Approach is often used in real-life applications (entity-relationships, data flow diagrams). It produces a drawing, where given a grid, the vertices are on grid points and edges are sequences of vertical and horizontal segments between grid points. Such drawing is called orthogonal drawing.

The orthogonal drawing (more generally polyline drawing) is characterized by three fundamental properties (equivalence relations): topology, shape (topology stays the same only lengths of edges are modified) and metrics (congruent, up to a translation and/or rotation).

Generating the final drawing consists of three steps (see Figure 5.1):

Planarization step is often used as a part of graph drawing algorithms, which produces a planar graph by adding dummy vertices instead of potential crossings of edges to given general graph. Testing of planarity can be done in linear time [HT74], but to find the maximum planar subgraph of given graph is NP-hard, hence existing planarization techniques use heuristics. The best available algorithm for the maximum planar subgraph problem is described in [JM96].

Orthogonalization step is the main part of the approach. It determines the shape of the drawing. There are different possibilities for producing an orthogonal grid drawing for planar graph where the maximal degree of vertex is at most four (see [BETT99]).



(a) Planar graph (b) Orthogonal drawing (c) Compacted orthogonal drawing

Figure 5.1: Topology-Shape-Metrics Approach

Compaction step minimizes the area of the drawing by shortening the edge lengths.

Due to the sequence of the three steps of the algorithm, the importance of aesthetic criteria is ordered. Minimization of crossings (number of dummy vertices added to the graph) has higher priority as the minimization of bends (orthogonalization step). The minimization of the area of drawing, the minimization of the sum of lengths of the edges and the minimization of the longest edge is done at the end, in the compaction step, if it is still possible.

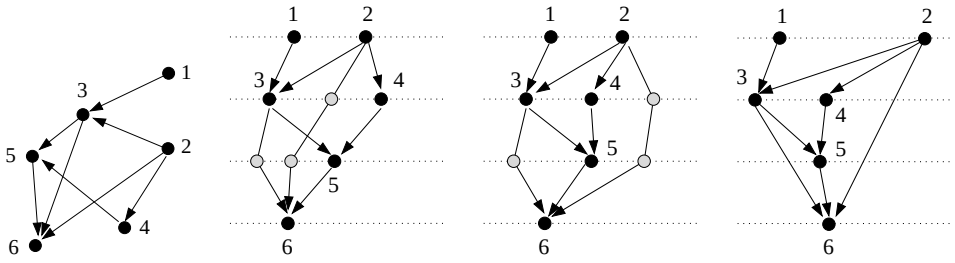
The constraints, as well as the aesthetic criteria, are formed according to the order of the steps of the algorithm. The topological constraints are the most important. The shape constraints depend on the second, orthogonalization phase and are therefore more significant as the metrics.

5.1.2 Hierarchical Approach

The hierarchical approach is used for drawing acyclic digraphs. This type of graphs are often used for modelling dependency relationships, thus the approach is often used in existing systems. The algorithms can be extended to draw a general digraph by adding a step, that forces the graph to be acyclic by reversing a subset of its edges.

The hierarchical approach (originally presented in [STT81]) consists of three steps (see Figure 5.2):

Layer assignment step: The vertices are assigned to horizontal layers. The concept of layering of a digraph is similar as topological number-



(a) Directed Graph (b) Proper Layered Digraph (c) After Crossing Reduction (d) Final Drawing

Figure 5.2: Hierarchical Approach

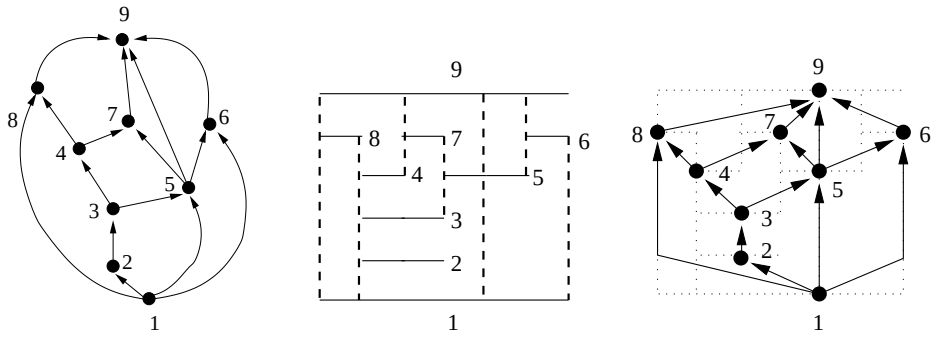
ing. The layering should be as compact as possible, but it should be proper (i.e. every edge connects vertices in neighbouring layers).

We introduce some layering algorithms:

- The Longest Path Layering – the vertex is placed on the layer L_{p+1}, where the longest path from the vertex to a sink has length p. This layering minimizes the height of the drawing.
- Layering to Minimize Width – the problem of finding the layering with minimum width on condition that its height is minimal is NP-complete (easy to derive from [GJ79]). Thus the algorithms for this type of layering are heuristics. An example of heuristic algorithm based on the multiprocessor scheduling theory is *Coffman-Graham-Layering* [CG72].
- Minimizing the Number of Dummy Vertices – the layering can be computed in polynomial time [GKNV93].

Crossing reduction: Having a proper layered digraph we reduce number of crossings by ordering vertices on each layer. In fact, the problem of minimizing edge crossings in layered digraph is NP-complete [GJ83, MNKF90], hence various heuristic algorithms are used for this purpose. The general format of most techniques is the *layer-by-layer sweep*, where the vertex ordering of layer L_{i+1} is derived from the vertex ordering of the layer L_i by minimizing crossings between edges between vertices in these layers.

Horizontal coordinate assignment step: By replacing dummy vertices introduced in first step, bends in edges may occur. This step should



(a) Directed Graph (b) Visibility Representation (c) Polyline drawing

Figure 5.3: Visibility Approach

minimize the angle of bends by choosing the x-coordinate for each vertex.

Some aesthetic criteria can be achieved. By replacing dummy vertices we affect the number and angle of bends, by ordering vertices on layers the symmetry of the graph is influenced.

5.1.3 Visibility Approach

This approach (first presented in [BT88, BTT92]) uses the visibility representation of the graph. A visibility representation of a graph (see Figure 5.3(b)) draws each vertex as horizontal (vertex) segment and each edge as a vertical (edge) segment. The only intersection of the edge segments and vertex segments are the top and bottom points of the edge segments, which are common with segments of vertices to which the edge is incident. Having the visibility representation, we get the positions of vertices by replacing each vertex segment with some point from it (Figure 5.3(c)). The edges are connecting incident vertices containing some part on their edge segment. We obtain the drawing with polyline convention.

Because the visibility representation requires planar graph, the first step in visibility approach algorithms must be the planarization step.

The planarization step brings the same aesthetics criteria and constraints as written before. By producing the visibility representation we can try to minimize the area of the drawing or fullfill some constraints (vertical alignment of selected paths, relative horizontal and vertical positions of pairs of vertices etc.). In the replacement step, several strategies are possible. De-

pending on the strategy, the following can be reached: minimizing the bends, balancing the distribution of edges and vertices, maximizing the symmetry of the drawing, etc..

5.1.4 Augmentation Approach

The augmentation approach is a method for drawing graphs in the polyline drawing convention (see for example [Mut95]). The basic idea is to add edges and vertices to the graph to obtain a new graph with a stronger structure (in this case, the faces are triangles). The method consists of three steps:

Planarization step: the same as in topological-shape-metrics approach

Augmentation step: adds a suitable set of edges to achieve the maximal planar graph (with triangle faces) or a planar graph with certain level of connectivity.

Triangulation drawing step: there are several algorithms for straight-line drawing of graphs with triangle faces. We remove dummy vertices and edges to achieve polyline drawing.

Since the planarization step is done as in topological-shape-metrics approach, the crossing reduction is the most important aesthetic criteria. During other steps different strategies to minimize area, maximize the angular resolution and distribute vertices can be used.

5.1.5 Force-Directed Approach

The Force-Directed Approach is an intuitive method for creating straight-line drawings of undirected graphs. The algorithms using this approach are heuristics. Some of them are empirically analyzed in [BHR96]. The resulting drawing is highly symmetric and the vertices are very well distributed.

The algorithm can be divided into two logical parts. At the beginning we find a suitable force model. Given the force model, we find a local minimum energy configuration.

The variety of constraints is possible according to chosen force model (for example vertices can be placed within a given region or on a given curve). As written above, the drawing fulfills a lot of aesthetic criteria.

5.1.6 Divide and Conquer Approach

Divide and Conquer Algorithms are used for handling structures, that can be easily divided into smaller substructures with the same properties. Es-

pecially trees and series-parallel digraphs are the classes of graphs that are often drawn using this approach. For drawing trees an algorithm producing layered drawing is presented in [RT81].

The common algorithm using divide and conquer approach has the following structure:

```
procedure D&C(graph G);
 if (G is trivial) then Draw(G)
 else begin
    divide G into smaller subgraphs G1.. Gn;
    for i:=1 to n do D&C(Gi);           {draw the subgraphs}
    place the drawings and draw remaining vertices;
 end;
```

5.1.7 Incremental Drawing of Graphs

Incremental techniques of graph drawing address the problem of maintaining a drawing of a graph while the user is interactively modifying the graph. If some operation is performed on the graph, then the new graph should be redrawn.

Most approaches [MHT93, PST97] try to address two main problems: efficiency of the algorithms and preserving the mental map. In [MELS95, Rai] the techniques, where the redrawing of the graph is as minimal as possible, are described. From user's point of view it is important, since the user's mental map is thereby preserved.

There are different scenaria for interactive graph drawing:

full-control scenario: the user has full control over the position of a new vertex in the current drawing.

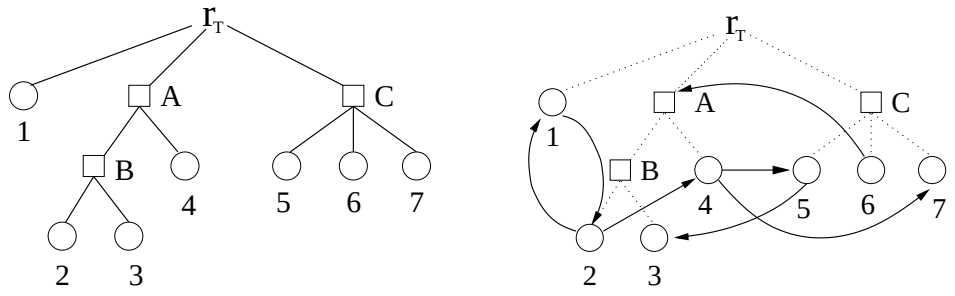
draw-from scratch scenario: every time, user request is posted, the new graph is drawn.

relative coordinates scenario: general shape of the drawing remains the same, coordinates of vertices and edges may change by a small constant.

no-change scenario: the already placed vertices and edges do not changes at all.

5.2 Attributes of Content Structure Graph

The content structure (CS) was defined in the Section 3.3. Formally CG can be viewed as a directed graph. To recall CS is a pair $(NodeSet, EdgeSet)$.



(a) Forest T (b) Directed graph G

Figure 5.4: Clustered content graph

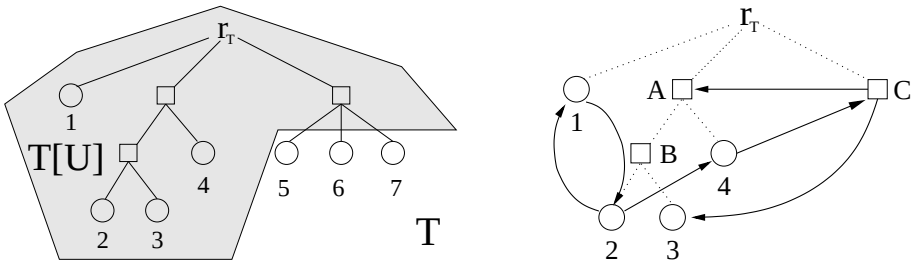
The $NodeSet = V$ consists of two types of nodes – simple nodes and clusters. The $EdgeSet = F$ is a set of directed edges of different types. On type of edge (*contains*) is explicitly defined. We can extract these edges and define a clustered and compound graph as [EFN99, SM91].

Clustered graph is a pair $CG = (G, T)$, which consists of a directed graph $G = (V', F)$ and a tree $T = (V, E)$, where V' are exactly the leaves of the tree T. For our purpose this definition is not sufficient, therefore we define the compound graph.

Compound graph $D = (V, E, F)$ consists of nodes V, *inclusion edges* E (edges of type "contains") and *adjacency edges* F. It is required, that the *inclusion digraph* $T' = (V, E)$ is a rooted tree with the root r_T and no adjacency edge connects a node to one of its descendants or ancestors. In some cases the graph T' may be also a forest, where the roots of the trees contained in the forest are connected to imaginal vertex r_T. In that case, we denote the tree created from the forest with trees $T_1 \dots T_t$ by adding a vertex r_T as T. Otherwise we denote $T := T'$. The graph $G = (V, F)$ is directed graph where no adjacency edge connects a node to one of its descendants or ancestors in T.[1]. A small example of a compound graph is pictured in Figure 5.4.

Let $U \subset V$ be such set of vertices that for each vertex $v \in U$ all siblings of v belong into U as well. The view on the compound graph D containing vertices U is denoted $D[U] = (U, E(U), F(U))$ (see Figure 5.5). The view $D[U]$ fulfills these properties:

[1]Note, that the graph G contains also the clusters as vertices, unlike the clustered graph.



(a) The subtree of T

(b) The view induced by a subgraph U

Figure 5.5: The view on compound graph G

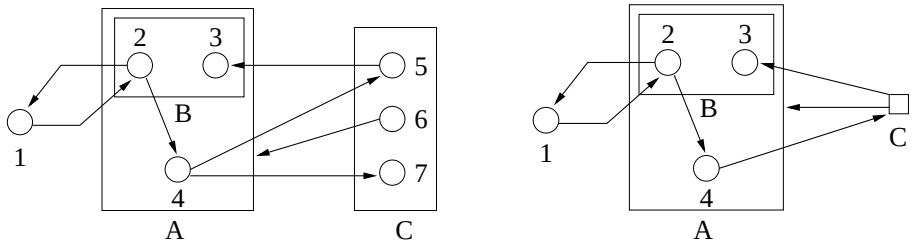
- $E[U] = \{(u_1, u_2) \in E \mid u_1, u_2 \in U\}$
- $T[U] = (U, E[U])$ is a connected subtree of T, which contains the root
- $G[U] = (U, F[U])$, where the set of edges $F[U]$ contains
  - all originall edges $F_1[U]$
    $F_1[U] = \{(u_1, u_2) \mid u_1, u_2 \in U \ \wedge (u_1, u_2) \in F\}$
  - new induced edges $F_2[U]$ (edges going to contracted clusters, which were not contained in F)
    $F_2[U] = \{(u_1, u_2) \mid u_i \in U \wedge \exists v_i \in T(u_i) \cap\cap (V \backslash U) : \ (v_1, v_2) \in F\}$

On the compound graph, we can execute some basic operations manipulating with some object of the graph:

Nodes: adding, removing, changing properties
Edges: adding, removing, changing properties
Clusters: adding, removing, *expanding*, *contracting*

The cluster v is called *contracted* in the view U, if the vertex v is a leaf of $T[U]$. Otherwise the cluster is called *expanded*. The interpretation of expanded (resp. contracted) clusters is that the content (nodes, that are contained in the cluster) of the expanded cluster can be seen unlike the content of the contracted one.

After every operation, we want the graph to be similar to the previous one as much as possible and after executing inverse operation the graph has to be also similar to the initial one.



(a) All clusters expanded (b) Cluster C is contracted

Figure 5.6: Drawing of content graph

In the drawing of a view $D[U]$, the graph $G[U] = (U, F[U])$ is drawn as points and curves as usual (the clusters and simple nodes can be distinguished). For each node $v \in T[U]$ the cluster is drawn as simple closed region (e.g. rectangle), such that all drawings of nodes in subtree $T[U](v)$ rooted in v and edges connecting two of such vertices are drawn in the interior. All other nodes are drawn in the exterior of the region.

The drawing of the compound graph from Figure 5.4 with all clusters expanded and after contracting of cluster C is depicted in Figure 5.6.

The requirements on the drawing of the content structure could be summarized as follows:

– leave the important vertices where they are (if possible)
– after executing an operation, the resulting drawing should be similar to the initial one
– vertices contained in cluster are drawn together – in rectangle
– minimize the crossings and no region crossings

Notice that each node of the graph could be a part of just one cluster. We can extend the definitions above as follows: *Extended compound graph* is $D' = (V, E, F)$, which consists of a **directed acyclic graph** $T = (V, E)$ and a directed graph $G = (V, F)$. Other definitions can be extended in a similar way. The problem "How to draw the clusters with common node?" arises.

5.3 Hierarchically Structured Graphs

The research on dealing with graphs similar to content graph covers problems that arise in conjunction with hierarchically structured graphs, i.e.,

with graphs that have – besides the ordinary adjacency relation – an additional hierarchical structure. Popular examples for this types of graphs are compound graphs [SM91] or clustered graphs [FEC95b].

5.3.1 Drawings of Compound Graphs

As the amount of information increased, the classical graph models were not sufficient for representing it and more powerfull models (hypergraphs, compound graphs etc.) appeared. The classical graph drawing can not be used for drawing of these models, the drawing is too difficult.

One of the general definitions of graphs with hierarchical structure are compound graphs, as defined above.

Until the year 1995, only heuristic algorithms for hierarchical layout of compound digraphs have been presented. An example of such algorithm is introduced in [SM91]. The drawing is produced in four steps and the hierarchical approach is used:

Hierarchization: Produces layered graph

Normalization: Adds dummy vertices to produce the proper layered graph. This step together with hierarchization is the layer assignment step in hierarchical approach.

Vertex Ordering: Orders the nodes on the layer. The vertex ordering algorithm works depth-first.

Metric Layout: Assigns coordinates and dimensions to the nodes of the ordered compound graph. The local coordinates are optimized with the priority method on the metric local hierarchy, which is basically the local hierarchy from previous step.

In [Rai] the algorithm is improved, the update scheme for local changes is presented. Since in vertex ordering step the relative order of nondummy vertices is kept, the users' mental map is preserved.

5.3.2 Drawings of Clustered Graphs

In 1995 were clustered graphs, as a simpler subclass of compound graphs, introduced in [FEC95b]. For clustered graphs several algorithms and approaches were presented.

Let us mention some relevant definitions and algorithms:

The *clustered graph* $C = (G, T)$ consists of an undirected graph G and a rooted tree T in that way, that leaves of T are exactly the vertices of G.

Let $C_1 = (G_1, T_1)$ and $C_2 = (G_2, T_2)$ be two clustered graphs. We say, that C_1 is a sub-clustered graph of C_2 iff T_1 is a subtree of T_2 and for each node $u \in T_1$ the graph $G_1(u)$ [2] is a subgraph of $G_2(u)$. The clustered graph is *connected* if each cluster induces a connected subgraph of G. The clustered graph is *planar* if each cluster induces a planar subgraph of G. The drawing of a clustered graph is *C-planar* if there are no edge-regions crossings and no edge crossings. If a clustered graph has C-planar drawing, it is said to be *C-planar*.

The following results from [FEC95a] characterize C-planarity.

Theorem 5.3.1 *A connected clustered graph $C = (G, T)$ is C-planar if and only if G is planar and there exists a planar drawing $\mathcal{D}$ of G such that for each node u of T all the vertices and edges of $G \setminus G(u)$ are in the outer face of the drawing of $G(u)$.*

Theorem 5.3.2 *A connected clustered graph $C = (G, T)$ is C-planar if and only if it is a sub-clustered graph of a connected and C-planar clustered graph.*

Using these theorems and the theorem of Tutte, that every 3-connected planar graph admits a planar straight-line convex drawing, in [FEC95b] was presented an algorithm for straight-line convex C-planar drawing of a subclass of C-planar clustered graph. It only applies to graphs with a certain strong connectivity property. They also demonstrated the area lower bound ($\Theta(2^n)$) and angle resolution upper bound ($O(1/n)$) for straight-line convex C-planar drawing of C-planar clustered graphs. The question about the existence of an algorithm for drawing straight-line convex drawing for every C-planar clustered graph remains open.

Later in [EF96b] and [EF97] the straight-line constraint was relaxed and orthogonal-grid rectangular clustered drawing (i.e. drawing with edges drawn as sequences of horizontal and vertical segments, vertices drawn on grid points and regions for clusters drawn as rectangles) was produced. The visibility approach was used for this purpose. The performance is as good as existing results for classical graph drawings.

The straight-line constraint was recalled again. In [EFLN97] the question, which remained open in [FEC95b] is answered positively by transforming clustered graphs into hierarchical graphs.

[2] the graph induced by all descendants of u in T_1

5.3.3 Visualizing of Hierarchically Structured Graphs

The approach oriented on drawing of hierarchically structured graphs is also supported by researches, which study possibilities of the visualization of the graph and the operations.

In [EF96a] the traditional 2D representation was extended to 3D multilevel representation. A natural method of such representation is to draw different levels of abstraction on planes with different z-axis. This type of representation is useful in preserving the mental map among abstractions levels.

Problems with large amount of information could be overcome in [EH00] by combining clustering of the graph and the navigation through the graph. Authors introduced the Clustered Graph Architecture (CGA), which was designed for systems, where user manipulates data in four layers.

Graph layer manipulates nodes, edges and their attributes

Clustering layer has two basic operations *create cluster* and *delete cluster*

Abridgment layer shows and hides some information (operations: *open cluster, close cluster, hide node*)

Picture layer makes changes in displayed picture (moving, scaling etc.)

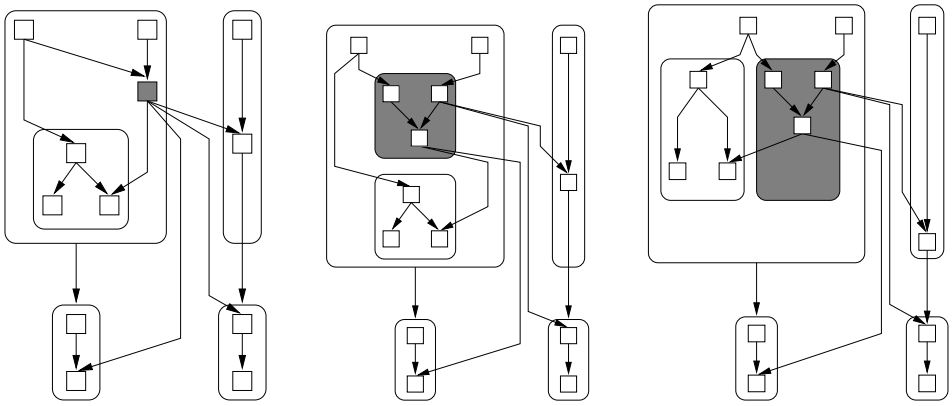
The visual navigation of compound (clustered) graphs by expanding and contracting nodes has been introduced in [SM91]. They seem to implement these operations by reusing the algorithm for drawing compound graph. In [HE01] authors briefly describe a system for handling huge clustered graphs based on the force-directed layout model.

Other possible manipulation with compound graphs is handled in [Rai], where the local update scheme for the algorithm from [SM91] for drawing views of compound graphs is presented. In Figure 5.7 is depicted the difference, between results given by redrawing of the graph by the algorithm from [SM91] and the local updating of the graph as described in [Rai].

This result is the most important from our point of view, because the resulting drawing better preserves the user's mental map of the graph view and has most of the properties, which we need for drawing of the content graph. Moreover, the update schema is more efficient than the redrawing of the graph, because almost all steps of the algorithm are applied only locally.

Two basic operations *expand cluster* and *contract cluster* are defined and performed on the graph.

Expanding: In hierarchization, the nodes from previous view stay on their levels, only new vertices are placed on new levels (within expanded



(a) Before expanding the shaded node

(b) Result of update scheme

(c) Result of redrawing

Figure 5.7: Redrawing vs. Update scheme

node's level). The new edges obtain the orientation from existing edges (all new edges are incident to new vertices), and in second step, they are made proper as before. All old nodes stayed on their level, thus in the third step – vertex ordering, only new nodes and new dummy nodes have to be ordered. The relative order of other nodes has to be preserved. Only the relative coordinates of the expanded node and its ancestors are computed in the metric layout step.

Contracting: Contracting is the visual inverse operation to expanding. It is possible only for nodes expanded with the update scheme. The new nodes are a subset of the old nodes. Results of first, second and third step are restricted to new nodes. The metric layout step is applied.

At the University of Passau started this year the project *Ein Editor für hierarchisch geschachtelte Graphen* [Pas], based on two diploma theses of their students. The main purpose of the project is to produce an editor for hierarchically structured graphs.

Chapter 6

Existing Solutions

In this chapter we present some existing solutions, which help to store some structured information. The existing systems can be divided into groups, according to way they are used. The largest is the group of learning management systems and course management systems, which are used for e-learning. These systems are not designed to store the course information and learning content only. They have also a lot of additional properties (e.g. forums, quizzes, chats, etc.), which are not important for our work.

6.1 Learning Management Systems

A Learning Management System (or LMS) is a software package, usually on a large scale (that scale is decreasing rapidly), that enables the management and delivery of learning content and resources to students. Most LMS systems are web-based to facilitate the concept of "anytime, anywhere" access to learning content and administration. There are many Learning Management Systems available. Comparison and evaluation of some Learning Management Systems was done in 2002 in Germany and Austria [Kri02]. Most of them are very similar to each other. The differences between learning management systems are the availability, localization (languages), price etc. In this section we introduce some systems, which seem to be the most important nowadays.

Moodle [moo] is a course management system – a free, Open Source software package designed by using pedagogical principles, to help educators create effective online learning communities. The Moodle architecture is based on PHP, HTML and one database (which can be shared for several courses).

There is a lot of course activities (Forums, Quizzes, Resources, Choices, Surveys, Assignments, Chats, Workshops) and management modules (site management, user management, course management etc.) in the system.

The most important module, from our point of view, is the *resource module*, which manages the learning resources. This module supports display of any electronic content, external content (via hyperlinks), external applications. The content structure is represented as hierarchical structure (the end nodes could be also web pages outside the system) and presented as web pages.

Ilias [ili] is a powerful web-based learning management system that allows users to create, edit and publish learning and teaching material in an integrated system using their normal web browsers. It came as a winner for German-spoken countries in [Kri02].

Blackboard [bla]) is a commercial software powering teaching and learning in many languages.

6.2 Personal Information Management Systems

Personal information management system is an application software that keeps track of personal information and data. There are several software packages that do this.

Treepad [tre]) is a tool for managing information in a tree structure. The content is presented similarly to the visualization of a directory structure in a file browser. However in this case the underlying format for storing content structure is a mark-up language similar to XML.

6.3 Web-based Solutions

Web-based solutions are very popular, because information is mostly stored in the web-space. Web space can be locally visualized through web browser (no further processing for presentation has to be done) and it provides support for local browsing as well. There are also new approaches, that concentrate on an automatic search of information in web-space. With such systems, the information in web-space would be better arranged and hence easier to find.

Wikipedia [wik] is a multilingual, Web-based, free-content encyclopedia. It is written collaboratively by volunteers with wiki software, which allows articles to be added or changed by nearly anyone.

Wikipedia has all the advantages and disadvantages of web-based representation (described for example in [DKR05b]). The content can be presented directly on the webpage and some additional information is available via hyperlinks. The largest problem of webspace is that one can get lost in the amount of hyperlinks and web pages. We can avoid this by restricting the content graph into tree or DAG. However, this topology may not be sufficient.

mSpace [msp] is a semantic web solution. It is an interaction model that helps to store and explore relationships in information. The application, which won the Semantic Web Challenge 2003, is described in IEEE Intelligent Systems.

The Semantic Web is a project that intends to create a universal medium for information exchange by giving meaning (semantics), in a manner understandable by machines, to the content of documents on the Web. The Semantic Web extends the World Wide Web through the use of standards, mark-up languages and related processing tools. The Semantic Web is at the moment the vision of further solution. This application is one of the first solutions.

The logical description of an mSpace is available in the draft report [GHS04], which provides a tentative mapping to OWL-based Semantic Web concepts.

Chapter 7

Conclusion

In this survey we described the most important aspects of presenting the content structure. As the main area, where presenting the content is applicable we have chosen the e-learning systems, since the learning content has to be stored and then presented to students in a comprehensible way.

E-learning is a frequently used term in recent years. It denotes a type of learning, where knowledge is mediated to students through electronic media. The learning content is mostly published in internet.

Thus we first described the possibilities of storing the content, which result from the way, learning content is produced and from the requirements on it (storing metadata, relations, support for visualization). Very important part connected to storage of the learning content is its visualization. From the psychological point of view, the visual perception can reinforce cognition. We viewed existing tools for visualization of information, but they do not seem to be sufficient for our purpose.

We introduced the graph drawing research, since the learning content can be seen as a graph, where nodes are learning objects and edges are the relations among them. However, the content graph has some special attributes. There are special nodes, so called clusters, which may contain other nodes. In general, this type of graphs are called hierarchically structured graphs. As a good possibility of visualizing the learning content we consider drawing of the content graph.

We gave a brief summary of graph drawing approaches, which may be useful for drawing the content graph. After that, we focused on research on hierarchically structured graphs, e.g., clustered graphs and compound graphs. We described algorithms for drawing such graphs. Another important aspect of visualizing the content is the navigation in the graph. We

summarized also the results in this area.

At last we presented some existing systems for storing and presenting information. We concluded, that even though the existing systems are robust and have a lot of additional properties, the visualization of the content is not included. Thus the presentation of the content is not comprehensible enough and the user may get lost in the amount of given information.

Chapter 8

Project of Dissertation Thesis

In e-learning systems the learning content is presented for students. Most existing e-learning systems are web-based, i.e., the learning content is presented on web pages. Surfing or browsing through the content suggest moving among discrete web pages which potentially have only very loose associations between them. The example of such loose association is relation *next in the book*, which is often used in learning content obtained from existing book. Using such relations one can go far away from the topic, he wants to read about.

The aim of the thesis is to utilize graph drawing algorithms for presenting structured information, in particular, in e-learning systems.

We plan to build upon the graph model for learning content we developed in [DKR05b, DKR05a]. First we need to analyze existing content in order to specify a particular type of graphs, that occur in e-learning environment. In general the content graph is compound graph, as introduced in [EFN99, SM91, Rai], where several algorithms for drawing compound graphs are also introduced. Once this class is specified we shall study the usability of existing graph drawing algorithms for this particular class. We plan to analyze some algorithms on compound graphs as well as other algorithms (e.g. [FEC95b]), which may be suitable for drawing the specified class of graphs.

We expect, that we can find more efficient modifications of those algorithms due to the fact, that we expect the conent graphs to have specific properties. On the other hand, we envision a need to modify the existing algorithms due to the specific needs of the e-learning content presentation

(e.g., nonzero size of nodes, positioning of captions, clustering of nodes, several node and edge types, specific requierments on positioning nodes, incremental changes preserving the general structure, etc.). The main requirement is that the drawing should preserve user's mental map, i.e., after each operation, the drawing of the new graph should be as similar as possible to the previous one. Eventual modification of the requirement above, should be preserving not the whole user's mental map, but only the important parts of it i.e. some important vertices stay on their positions while coordinates of the less important ones may change.

We shall design and study these modifications and compare their efficiency with the existing algorithms.

Finally, we shall need to design and implement a visualization tool that will enable to work interactively with the content graph. The tool should allow browsing through the graph as well as performin basic operations on the graph (adding and removing vertices and edges, expanding and contracting clusters).

We are convinced that utilization of the content structure in existing systems may be good for students. The well structured content and a possibility to exploit this structure can enhance the usability of e-learning systems. It may help students to orient themself in the amount of information in learning content. In addition, some new (semantic) associations can be added to the learning content, which may extent the web pages used in e-learning systems, as it is aimed in the Semantic Web.

Bibliography

[BETT99] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Alan Apt, 1999.

[BHR96] F. J. Brandenburg, M. Himsolt, and C. Rohrer. An experimental comparison of force-directed and randomized graph drawing algorithms. In *Proceedings Graph Drawing '95*, 1996.

[bla] Blackboard. http://www.blackboard.com.

[BT88] G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs. *Theoretical Computer Science*, 61:175–198, 1988.

[BTT92] G. Di Batista, R. Tamassia, and I. G. Tollis. Constrained visibility representations of graphs. *Inform, Process. Letter*, 41:1–7, 1992.

[CG72] E. G. Coffman and R. L. Graham. Optimal scheduling for two processor systems. *Acta Informatica*, 1:200–213, 1972.

[DKR05a] J. Dvořáková, J. Katreniaková, and B. Rovan. Capturing content structure in xml: Formal languages e-course example. E-Learning Conference, Berlin, 2005.

[DKR05b] J. Dvořáková, J. Katreniaková, and B. Rovan. Representing content structure for e-learning systems. In *Information Technologies – Applications and Theory*, pages 361–370, 2005.

[Dow02] S. Downes. Topic representation and learning object metadata, 2002. http://www.downes.ca.

[EF96a] P. Eades and Q. Feng. Multilevel visualization of clustered graphs. In *Graph Drawing, vol. 1190 of LNCS*, pages 101–112, 1996.

[EF96b] P. Eades and Q. Feng. Orthogonal grid drawing of clustered graphs. Technical Report 96-04, The University of Newcastle, Australia, 1996.

[EF97] P. Eades and Q. Feng. Drawing clustered graphs on an orthogonal grid. In *Graph Drawing*, pages 146–157, 1997.

[EFLN97] P. Eades, Q. Feng, X. Lin, and H. Nagamochi. Straight-line drawing algorithms for hierarchical graphs and clustered graphs. Technical Report 98-03, The University of Newcastle, Australia, 28 1997.

[EFN99] P. Eades, Q. Feng, and H. Nagamochi. Drawing clustered graphs on an orthogonal grid. *Journal of Graph Algorithms and Applications*, 3(4):3–29, 1999.

[EH00] P. Eades and M. L. Huang. Navigating clustered graphs using force-directed methods. *J. Graph Algorithms and Applications: Special Issue on Selected Papers from 1998 Symp. Graph Drawing*, 4(3):157–181, 2000.

[FEC95a] Q. Feng, P. Eades, and R. F. Cohen. Clustered graphs and c-planarity. Technical report, The University of Newcastle, Australia, 1995.

[FEC95b] Q. Feng, P. Eades, and R. F. Cohen. Planar drawing of clustered graphs. Technical report, The University of Newcastle, Australia, 1995.

[Fek04] J. D. Fekete. The infovis toolkit, 2004. INRIA Futurs/LRI.

[Fen01] D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001.

[GHS04] N. Gibbins, S. Harris, and M. Schraefel. Applying mspace interfaces to the semantic web. Technical report, University of Southampton, UK, 2004.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, New York, 1979.

[GJ83] M. R. Garey and D. S. Johnson. Crossing number is NP-Complete. *SIAM J. Algebraic Discrete Methods*, 4(3):312–316, 1983.

[GKNV93] E. R. Ganser, E. Koutsofios, S. C. North, and K. P. Vo. Technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19:214–230, 1993.

[GPCFL04] A. Gomez-Perez, O. Corcho, and M. Fernandez-Lopez. *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer, 2004.

[graa] GraphML. http://graphml.graphdrawing.org/.

[grab] GraphViz. http://www.graphviz.org.

[HC00] W. Hodgins and M. Conner. Everything you wanted to know about learning objects but were afraid to ask. *LineZine*, 2000. http://www.linezine.com/2.1/features/wheyewtkls.htm.

[HCL05] J. Heer, S. K. Card, and J. A. Landay. Prefuse: a toolkit for interactive information visualization, 2005.

[HE] W. Huang and P. Eades. How people read graphs.

[HE01] M. L. Huang and P. Eades. A fully animated interactive system for clustering and navigating huge graphs. In *Proc. 6th GD*, pages 232–246, 2001.

[HK01] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Academic Press, 2001.

[HMM00] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[HMS01] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, Cambridge, 2001.

[HT74] J. Hopcroft and R. E. Tarjan. Efficient planarity testing. *Journal ACM*, 21(4):549–568, 1974.

[IEE02] IEEE. IEEE standard for learning object metadata - IEEE 1484.12.1-2002, 2002. http://standards.ieee.org.

[ili] Ilias. http://www.ilias.de.

[Inc05] Thinkmap Inc. Thinkmap sdk v.2.6, technical whitepaper, 2005.

[JM96] M. Jünger and P. Mutzel. Maximum planar subgraphs and nice embeddings: Practical layout tools. *Algorithmica*, 1(1):1–25, 1996. special issue on Graph drawing, edited by G. Di Batista.

[Kri02] R. Kristoöfl. Evaluation von learning management systemen. Technical report, Bundesministeriums für Bildung, Wissenschaft und Kultur, Österreich, 2002.

[MBA03] T. Murray, S. Blessing, and S. Ainsworth, editors. *Authoring Tools for Advanced Technology Learning Environments : Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software*. Springer, 2003.

[McG04] R. McGreal. Learning objects: A practical definition. *International J. of Instrutional Technology and Distance Learning*, 1(9), 2004. http://itdl.org/Journal/Sep_04/index.htm.

[MELS95] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *J. Visual Lang. Comput.*, 6(2):183–210, 1995.

[MHT93] K. Miriyala, S. W. Horick, and R. Tamasia. An incremental approach to aesthetic graph layout. In *Internat. Workshop on Computer-Aided Software Engineering*, 1993.

[MNKF90] S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa. Crossing minimization in linear embeddings of graphs. *IEEE Transactions on Computers*, 39(1):124–127, 1990.

[moo] Moodle. http://moodle.org.

[msp] mSpace. http://mspace.fm.

[Mut95] P. Mutzel. A polyhedral approach to planar augmentation and related problems. In *ESA*, pages 494–507, 1995.

[Pas] University Passau. Ein editor für hierarchisch geschachtelte graphen. http://www.infosun.fmi.uni-passau.de/VisnaCom/.

[PST97] A. Papakostas, J. M. Six, and I. G. Tollis. Experimental and theoretical results in interactive graph drawing. In *Graph Drawing 96*, pages 371–386, 1997.

[QH00] C. Quinn and S. Hobbs. Learning objects and instructional components. *Educational Technology and Society*, 3(2), 2000. http://ifets.ieee.org/periodical/vol_2_2000/.

[Rai] M. Raitner. Visual navigation of compound graphs. http://citeseer.ist.psu.edu/708344.html.

[RT81] E. Reingold and J. Tilford. Tider drawing of trees. *IEEE Transactions on Software Engineering*, 7:223–228, 1981.

[SCO02] SCORM. Sharable content object reference model 2004, 2nd Edition, 2002. http://www.adlnet.org.

[SM91] K. Sugiyama and K. Misue. Visualization of structural information: Automatic drawing of compound digraphs. *IEEE Transactions on Systems, Man and Cybernetics*, 21(4):876–892, 1991.

[STT81] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. *IEEE Transitions on Systems, Man and Cybernetics*, pages 109–125, 1981.

[tre] Treepad. http://treepad.com.

[wik] Wikipedia. http://www.wikipedia.org.

[WM03] J. Wittmann and D. P. F. Moller. The content-graph as a basic data structure to manage authoring and learning processes. In *International Conference on Advanced Learning Technologies ICALT'03*, 2003.