A Model of Versioned Web Sites*

Mária Bieliková and Ivan Noris

Faculty of Informatics and Information Technologies
Slovak University of Technology
Ilkovičova 3, 812 19 Bratislava, Slovakia
bielik@fiit.stuba.sk
http://www.fiit.stuba.sk/~bielik

Abstract. In this paper we present a model of versioned web sites which is aimed at building a web site configuration. The web site configuration is a consistent version of the web site and serves for navigation purposes. We exploit the fact that the versioning of web sites is in many aspects similar to versioning of software systems (and their components). On the other hand, specific characteristics related to the web environment and web sites in particular are considered. The web site is modelled by an AND/OR type graph. The model serves as a useful abstraction simplifying the process of configuration building. Being essentially a graph search, it is inevitable to have a method for selecting a proper version. Presented approach is best suited for web sites where several variants of web pages exist. It is advantageous for example for presentation of multilingual web sites. We briefly discuss developed software tool for versioning and navigation on the multilingual web site which is based on proposed model of versioned web site.

1 Introduction

Web sites evolve by changing their content and structure over time. Change is inevitable. However, the web today as a rule supports only one version of a document – the current one. Requirement to store and access previous versions of the web content, retrieve the history of the content, annotate revisions with comments about the changes, or navigate through a versioned web site is explicitly noted already in [2]. This requirement follows the evolution in the area of hypermedia research, where version control has been identified as a critically important task [8].

The literature lists many reasons for saving the history of an object (be it software component, hypermedia node, or web page), including distributed and collaborative development, keeping old versions for later use, assuring the safety of recent work against various kinds of accidents, preserving cited work in the original state. The purpose of document versions is not only a change. For

* This work has been supported by the Grant Agency of Slovak Republic grant No. VG1/ 0162/03 "Collaborative accessing, analysis and presentation of documents in internet environment using modern software tools".

example, to be accessible to a large audience web sites often contain information written in more than one language – each can be considered as a version.

Evolution of information presented on the web and related problems are discussed in several works [18,14,4,17]. Content management systems often provide versioning in the sense of software systems versioning [6]. A user – content manager – can store snapshots of his work (revisions) and come back to them later, or develop the web site by collaboration in a distributed team. Such scheme can be sufficient from a developer point of view, but not from a user (reader) point of view, who stands in a need of navigation through versioned web site.

In this paper we address the problem of computer support for navigation within a versioned web site. We have proposed a model of versioned web site which is simple, but still sufficiently rich to reflect the principal relations and properties which are decisive in the web site configuration building. The web site configuration is a consistent version of the web site that serves for navigation purposes. We have adopted the AND/OR graph model formalized for software systems in [3]. Semantics of the model is specified according to specific properties of web sites.

Central notions of proposed model include following: web pages are characterized by properties defined at three levels: family level, variant level and revision level; links are established with respect to a family of target page rather than the pages themselves; page to page connections are established at time of a configuration building; several revisions (of different page variants) can be included in the configuration.

2 Modelling a web site

A model is used to express a web site structure, respecting in our case the point of view of navigation of the versioned web site (i.e. building its configuration). The web site configuration can be used for off-line browsing within selected version of the versioned web site.

2.1 Elements of the model

A web site consists of several independent parts (nodes) interconnected by *links*. Each node – a *web page* – primarily comprises the content going to be presented. Obviously, there exists a mechanism for presentation layout definition (either embedded in each node or represented separately in one place for several nodes, or the whole web site). The web page often comprises both the content in the form of a text or other media and chunks of programming source code which provide dynamic content.

We consider two scenarios for a web page version creation [5,6,18]. First, versions are created to represent alternate forms of a page. Such 'parallel' versions, or *variants*, are frequently results of alternative realizations of the same concept (e.g., multilingual variants). The variants can evolve independently. Second, versions are created to represent improvements of previous ones, or as modifications caused by an error correction, content enhancement, and/or adaptation to

changes in the environment. Such 'serial' versions, or *revisions*, are frequently results of modifications of the same variant. A *family* of web pages comprises all web pages which are versions of one another. Note that the concept of parallel versions induces an equivalence relation within the set of web pages. We shall use the term *version* in cases where both variants or revisions can be considered.

Let us formulate the notions more formally now. Let P_S be a set of web pages of a web site S. Then a binary relation $is\_version_S \subseteq P_S \times P_S$ is given as the reflexive and transitive closure of another binary relation which is defined by elementary transformations describing such modifications of web pages that they can still be considered to be expressing essentially the same content. Relation $is\_version_S$ is reflexive, symmetric and transitive.

A set of all equivalence classes induced by the $is\_version_S$ relation is denoted F_S and called a set of families of web pages of the web site S. An element of F_S is called a family of web sites. In other words, a family consists of web pages which are related by relation $is\_version$. Usually such web pages are presented by means of the relation $is\_developed\_from$ as so called version graph [11,10,6]. Nodes denote various versions as they are created; an arrow from version A to version B indicates that B was created from A. All the web pages included in the version graph form a family of web pages.

We define for each versioned web page a set of properties (using attribute-value pairs) and its content (usually a HTML text, scripts, and embedded media). Based on that, we consider *variants* as sets of those web pages which share certain properties. This conceptual design choice does not impose any serious limitations in most cases. On the contrary, it provides a considerable flexibility to the specification of versioned web site navigation. It offers a useful abstraction that should simplify the process of navigation. In order to describe variants, we define a binary relation $is\_variant$ which determines a set of web pages with the same subset of properties within a given family. The binary relation $is\_variant_S \subseteq P_S \times P_S$ is defined by:

$x\ is\_variant_S\ y \Leftrightarrow x\ is\_version_S\ y \wedge x.VariantAttr = y.VariantAttr$

Variants are important to simplify management of web page versions when selecting a revision of some page, or during an automatic navigation within the versioned web site. We can treat a whole group of web pages in a uniform way due to the fact that all of them have the relevant properties defined as equal (e.g., the content is written in English).

One consequence of our design decision of taking variants to be sets of web pages is that from the two kinds of versions of web pages, only revisions are left to represent actual single web pages. Figure 1 depicts the above defined relationships.

The distribution of web pages to variants depends on a decision which properties are considered as variant properties and which are considered as revision properties, i.e. unique properties of the actual web page. Decision about distributing attributes is left open for a developer in our approach because it depends on the project, its size, problem domain, etc. Typical recommendations applicable in many cases are to consider as variant attributes specific characteris-

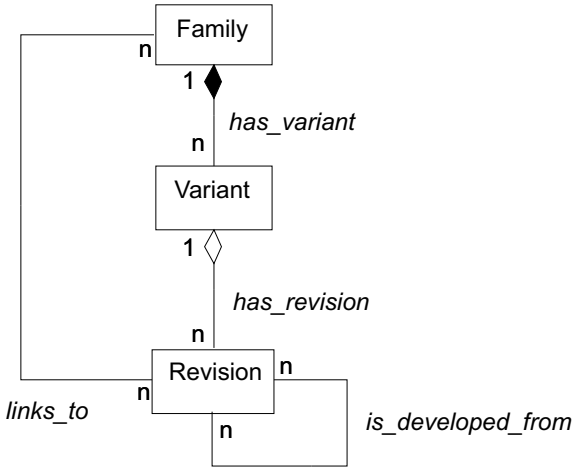

Fig. 1. Family-Variant-Revision relationships.

tics of the user knowledge (e.g., novice, intermediate, advances), characteristics of a developer environment (e.g., HTML version), or characteristics of a user environment (e.g., browser used). This means that a change of these properties leads to a new variant. Properties related to the development process such as state, change description, author, date, time are often considered as revision attributes, i.e. their change leads to a new revision.

2.2 Model formulation

The concepts introduced above allow us to formulate a model of a versioned web site which would support a navigation. Determining versions for navigation resembles a configuration building in software configuration management. From the point of view of a family, a model should involve families and variants included in them. Links from a family to all its variants are defined by the relation $has\_variant_S \subseteq F_S \times V_S : x\ has\_variant_S\ y \Leftrightarrow y \subseteq x$.

From the point of view of a variant, the model should represent links to all those families which are referred to in revisions of that variant. When building a configuration, for each family already included in a configuration there must be selected at least one variant. For each variant already included in a configuration, there must be included precisely one revision. For all selected revisions, all the families related by links to that revision must be included. A configuration comprising more than one variant of the web page is inevitable for example in case of building the configuration of a multilingual web site with specific requirements for English and Slovak language pages.

Our method of modelling a web site S is to describe it by an oriented graph, with nodes representing families and revisions in such a way that these two kinds of nodes alternate on every path. Let F_S be a set of families of web pages, P_S be a set of web page revisions of a web site S. Let FN be a set of names and $f\_name_S : F_S \rightarrow FN$ an injective function which assigns a unique name to each family of a web site S. Let $A \subseteq P_S \times FN$ be a binary relation defined as:

$e_1\ A\ e_2 \Leftrightarrow \exists r(r \in e_1.Link \wedge r.FamilyId = e_2)$

Let $O \subseteq FN \times P_S$ be a binary relation defined as:

$e_1\ O\ e_2 \Leftrightarrow e_2 \in f\_name_S^{-1}(e_1)$.

We define a model of a web site S to be an oriented graph $M_S = (N, E)$, where $N = FN_S \cup P_S$ is a set of nodes with $FN_S = \{\, x \mid x \in FN \wedge f\_name_S^{-1}(x) \in F_S\}$, and $E = A \cup O$ is a set of edges such that every maximal connected subgraph has at least one root.

We remark that the binary relation A stands for hyperlinks (relating revisions to families) and the relation O mirrors *has_revision* relation. Such A/O graph model refers for the previously introduced notions (revision, variant, family). Variants are covered in the model implicitly through sets of A-nodes which represent revisions.

The usual interpretation is that A-nodes are origins of edges leading to nodes, all of which must be considered provided the A-node is under consideration (logical AND). Similarly, O-nodes are origins of edges leading to nodes, from among which exactly one must be considered provided the O-node is under consideration (logical OR).

The example of a web site model is depicted in Figure 2. For the sake of clarity we depict also variants and relation *has_variant*. One possible configuration for the content written in Slovak language is highlighted (bold).

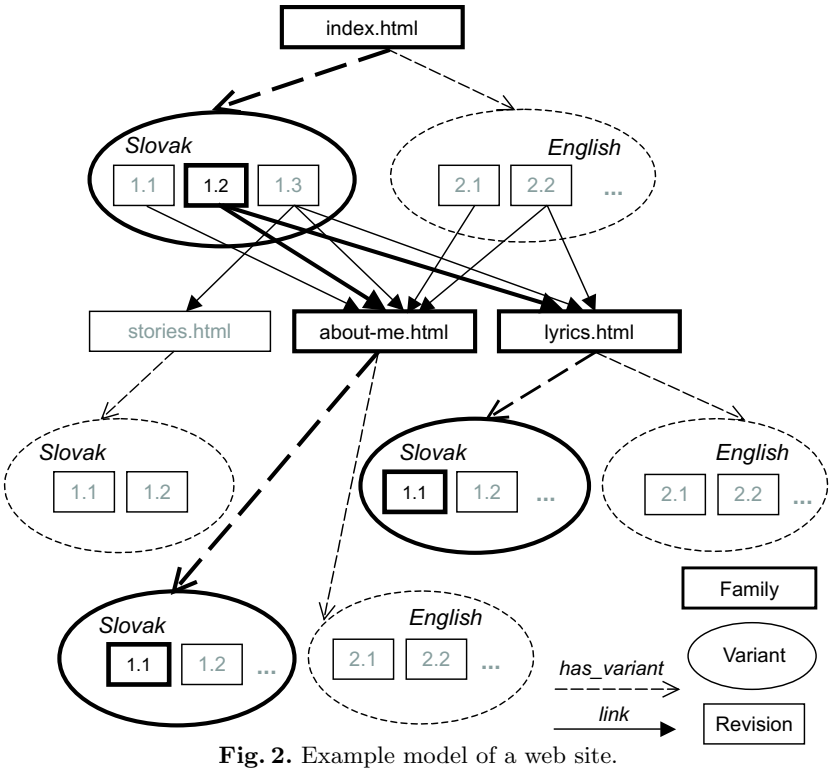

Fig. 2. Example model of a web site.

Proposed model is a version–family kind of model (an analogy of intertwined model known in software engineering [6]). The navigation relations are defined for each revision and the revisions are connected to the web page families only. Therefore, when a new version of target of a relation is created, the relation itself is not affected, so there is no need to create new version of the source component. Definition of links within revisions and families compromises the complexity of the model and subsequent support for navigation, and its flexibility.

If all links were defined on the variant level (all revisions within a variant share the same links), we would be able to exploit the model on several levels of abstraction: as an abstract model containing families and variants; a generic configuration containing selected variants, and bound configuration containing interconnected revisions. However, requirement for links definition on the variant level is in many cases too limiting. It can be successfully used for multilingual sites with only several revisions.

3 Navigation and configuration building

Process of navigation within a versioned web site is based on determining a target of the selected link. The navigation procedure is defined as a graph search, where the graph constitutes model of the versioned web site. We use the version selection filters which are conditions applied to all versions of appropriate families of the web pages [13].

Any version can have its own and independent attribute structure, so it can be modified without affecting other versions. This makes the attribute structure sufficiently flexible. On the other hand, some attributes (e.g., Language, Browser) are supposed to be shared between several revisions of the web page (or web page families). Several types of attributes are defined (e.g., string, number, time, list, set). A set of system attributes further improves the management of meta-data related to the versioning. The system attributes are automatically set by a tool providing the model of versioned web site. We distinguish several types of system attributes:

Read-only system attributes: their values are set only once after creating a new revision (e.g., InsertTime, InsertUser). The values cannot be changed.

Auto-updating system attributes: their values are updated automatically after each version significant change (e.g., ChangeTime, ChangeUser).

Default-value system attributes: they have some predefined values, but can be changed later (e.g., VersionCodename, Owners, Author, Keywords).

In the proposed model we use variant attributes to distinguish between revisions of various variant sets. Each variant wires revisions with the same values of the variant attributes. In the course of version selection process the variant attributes are evaluated first. Any attribute may be flagged as the variant attribute. The revisions are grouped into variants considering their properties.

The version selection is a two-step process: first, the corresponding variant and then the revision is selected. Thus, the variants exist only on a "logical

level". Version selection is described and formalized in [12]. We concentrate in this paper on specific features applicable for version selection of web pages.

3.1 Version selection filters

Version of a web page is selected using a set of selection filters. Selection filter is represented by a logical expression which operates on version properties.

Limitation of the proposed approach is that selection filters need not guarantee that exactly one version is selected. If the sequence of filters is too strict, none of the versions would match. On the other hand, more than one version could match loose filters. To avoid such cases, an implementation of the proposed approach should allow to refine (add, modify or even remove) the filter, at least as a user-initiated action. Also internal restrictions based on e.g., last modification time, can be implemented to filter out all but one matching version.

Version selection filters can be of two kinds: user-defined filters, and default filters. The option to save a sequence of filters as a "named configuration specification" makes the version selection mechanism more flexible and allows its reuse.

User-defined filters. The user-defined filters can be entered by any visitor of the versioned web site. The filters are defined explicitly on the user request. The number of filters is not limited in our proposal, however, some implementation limitations are expected to occur. The user-defined filters can be defined for any attribute and any required values.

Default filters. Default filters are defined by the web page author and are automatically applied when accessing a versioned web page. The need for default filters is based on the fact that a version-unaware users could easily visit the web site. Such users are not expected to be concerned with version control and may feel confused when dealing with the attributes and configuration specifications. On the other hand, it may be useful for the content author to guide the visitors in some way.

We distinguish static and dynamic default filters. A static default filter specifies a version in such a way that its evaluation will lead always to the same version. The dynamic default filters allow the author to set conditions which will automatically apply when the version within the family should be selected. The simplest dynamic filter selects the last inserted revision.

The dynamic default filters are flexible and allow the author to write conditions which ensure a version selection according to the current browsing session. The browser type, preferred language, client's top-level domain etc. can be used to determine user's requirements. The dynamic default filters can also be used to distinguish local and remote users. They are evaluated automatically, if the filter trigger has been activated. The trigger's actual value is compared with the author-supplied value. If the values match, the trigger is activated and the filter based on the trigger would apply. If more filters use the same trigger, they are defined separately. In developed software tool (see next section) we have used environment variables as filter triggers (e.g., `HTTP_USER_AGENT` variable stores

identification of the current client; `HTTP_ACCEPT_LANGUAGE` contains accept language codes sorted by client's preference). These variables are set automatically by the web server.

Specified sequence of default selection filters can be set at the logon time for certain users or groups of users. Presentation of a versioned web site using described mechanisms becomes adaptive.

We have also proposed a mechanism for accession alternatives of currently displayed page. This is accomplished by conditions for alternatives. The alternative versions have different values for combinations of the specified attributes. Conditions for alternatives together with displaying mechanism allow a user to see links to the corresponding versions with the alternative content (e.g. written in different language) on each visited page.

On a user request the value of selected attribute in selected filter changes and the alternative version is provided. In developed software tool, we use only one attribute for updating the conditions (the *ContentLanguage* attribute). This simplification is partially based on the purpose of the developed tool (to provide a support for multilingual sites). However, the greater number of attributes implies the greater number of possible combinations of their values produced, i.e. the number of alternate contents would grow exponentially.

3.2 Configuration building

A configuration building is the process of selection appropriate revisions for all web pages families to be included in the configuration. The configuration can be built for off-line content reading or by actual browsing through the versioned web site. This process consists of four basic steps:

1. Select a starting family f of web pages (the O-node in the model); obviously the starting family is the root of the web site model.
2. Select appropriate revision v of the web page (one of the A-nodes connected to the family f by *has_revision* relation); this step is based on specified version selection filters; at first variant is selected and then revision within this variant [13,12].
3. Provide attached elements for revision v, provide versions of attached elements.
4. IF the stop condition is false THEN select next web page to be included in the configuration (all those families which are referred to in the revision v are considered) and go to the step 2.

If the configuration is created in a browser, the family considered in the next cycle (step 4) is determined by a visitor by clicking on a link. Otherwise, next family for processing is selected according the search strategy adopted (e.g., depth-search or breath-search strategy).

4 An application of the model

We developed a software tool called *DiVer* for support of the navigation on versioned web sites [13]. DiVer implements described model for web pages written in HTML. Its primary purpose was presentation of multilingual web sites. We considered following requirements while developing a prototype of versioned web site navigation tool:

1. URLs pointed to the web page from the external (unversioned) space should remain working;
2. a reader should understand that he is viewing a versioned page; he should be able to access different versions of the displayed page;
3. there must be a mechanism for navigation within versioned space (method for web site configuration presented in the previous section is exploited); this mechanism should mimic standard web navigating (as without versions);
4. web pages with alternative content (written for example in different languages) should be easily accessible.

DiVer tool similarly to the V-Web tool [14] adds to the top of a web page a frame containing a textual depiction of the web page's version information together with possibility to select a version, define selection filters or see alternative versions. Original HTML page is replaced by a new HTML composite page which comprises menu and the content of the selected version from an archive of versions (see Figure 3).

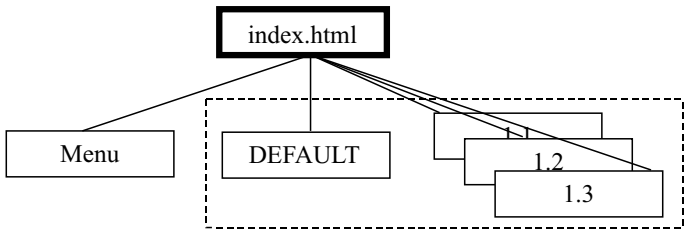

Fig. 3. Versioned web page.

The version control related attributes are defined only for versions (variants, revisions) of the HTML-elements. In our opinion, there is no need to define attributes of other elements, e.g., images, because the images are included in the composite HTML elements and not selected into configurations separately. This simplification significantly influences the data that have to be stored and processed. Of course, the attached components are also versioned and can be identified and accessed by the version number. Therefore, only HTML-elements are represented in the model. When the HTML-element revision is selected, appropriate revisions of all attached elements are selected. The information about attached elements and their versions is stored in the attribute structure of the HTML-element revision.

Our software tool uses RCS [15] as a revision control back end on the server side. The graphical front end on the client side is developed in Perl language. The

archive library is used to save and restore version attributes which are stored separately in XML archives. Version attribute structure is used to select appropriate version while creating the configuration. "Cookie" mechanism is used to transfer the selection filters. RCS uses a check-in/check-out paradigm to create revisions. It organizes the revisions into an ancestral graph and stores them in a file called an archive. We suppose that a tool which implements proposed model will rest the version control responsibility for the files with a revision control system such as RCS, while responsibility for maintaining the relations between the files will reside with the tool.

The DiVer tool conforms WebDAV protocol (Distributed Authoring and Versioning over the Web) [1] and its DeltaV extension (Web Versioning and Configuration Management) [11] (however, it was not developed with assumption of a web server supporting such extended HTTP protocol because of non availability of the WebDAV web server support in time of the tool development). The tool for navigation in versioned web space built on our model should be responsible for substituting the family URL (the same as for unversioned page) by the DeltaV stable URL.

5 Related work

In the hypermedia field, the problems connected to version control and configuration management have been frequently examined and discussed [16]. Also a wide variety of research has attempted to deal with versioning issues on the web. Example of this work include [14,1,2,11,17].

Two basic version models are used in area of hypermedia [7]: state-based versioning which maintains the version of an individual resource, and task-based versioning which focuses on tracking versions of complex systems as a whole. These concepts are similar to those of state-based and change-based versioning as known in software engineering [6]. Proposed model is oriented towards state-based versioning. While it does not support the tracking of a set of changes, it enables effective and efficient realization within web environment.

Several hypermedia systems define links on the level of particular pages (nodes) (see a comparison of hypermedia data models presented in [19]). In our model links are defined on the version—family level. This means that the link points to the family of web pages and is resolved on time of a configuration building (or navigating the web site). Linking on the level of page versions can lead into many broken links when versioned items are deleted from the public web repository (even if this is not consistent with the idea behind versioning – to preserve all states of an entity – deleting some versions from the web public presentation is prevalent).

Important issue while discussing versioning of web documents (or hypermedia in general) is that of links versioning. Several different solutions are described in [16]. There is no consensus on the issue whether links should be modelled (and represented) separately from the content. We can find approaches where links are embedded in the content; or links are represented separately; or some links

are local and modelled within the content (their change causes creation of the new version) and some links are external (represented separately, their change does not cause creation of a new version). Although at present, linking on the web consists in primarily of tags embedded in HTML, XLink proposal provides for storing links between XML documents externally to the documents they reference (see `http://www.w3c.org/XML/Linking`). Realization of our model does not restrict in any way the representation of versioned entities. Defining version—family model covers the structure versioning, i.e. within each revision relationships are maintained.

We do not introduce a new hypermedia model. But we build on the Dexter hypermedia model [9] and propose its specialisation with the aim of allowing efficient navigation within versioned web site.

6 Conclusions

Versioned web site offers significant advantages to the content developers and readers. It provides mechanisms to allow version-dependent navigation through the site. Content developers can concentrate on the content and relationships between versioned families of web pages. They do not have to deal with complexities of versioning and versioned navigation implementation.

We have proposed a model for versioned web site which aims at computer support of the process of the web site configuration building. We have concentrated in this paper on modelling presentation space (on the contrary of many existing approaches which stress on the development space). In fact, both spaces can exploit devised model. Basic distinction would be in granularity of versioning. A limitation of our approach is that the model does not provide substantive web site structure modelling. However, there is no model which suits all purposes.

The advantages of our proposal include:

- the model is simple and effective (version-family links present the main advantage of the model which produces also above mentioned limitation),
- the model can be used in web site design on several levels of abstraction,
- navigation within versioned site is intuitive,
- the model is ready for using with current web technologies (as demonstrated by the DiVer software tool).

Dynamic approach to resolving links improves the maintenance of the web site integrity. If a user has stored a bookmark to the particular version, it points to the page family with version selection data. In the case of missing version (or no version satisfying the selection filter) the user has still an option to select different version.

The model can serve also during the web site design (its structure and navigation). Several models on various level of abstraction can be constructed. Levels of abstraction regard to hierarchy of composite elements (web site versus web page) and to interconnections (explicit versus implicit links).

References

1. IETF WEBDAV working group. `www.ics.uci.edu/pub/ietf/webdav`.
2. T. Berners-Lee. Versioning, 1990. A web page that is part of the original design notes for the WWW, available at `www.w3.org/DesignIssues/Versioning.html`.
3. M. Bieliková and P. Návrat. Modelling software systems in configuration management. *Applied Mathematics and Computer Science*, 5(4):751–764, 1995.
4. M. Bieliková and P. Návrat. Modelling versioned hypertext documents. In B. Magnusson, editor, *System Configuration Management, ECOOP'98 SCM-8 Symposium*, pages 188–197, Brussels, Belgium, 1998. Springer-Verlag, LNCS 1439.
5. M. Bieliková and P. Návrat. An approach to automated building of software system configurations. *International Journal of Software Engineering and Knowledge Engineering*, 9(1):73–95, 1999.
6. R. Conradi and B. Westfechtel. Version models for software configuration management. *ACM Computing Surveys*, 30(2):232–282, June 1998.
7. A. Haake and D. Hicks. VerSE: Towards hypertext versioning styles. In *Proc. of the 7th ACM Conf. on Hypertext*, pages 224–234, Washington DC, USA, March 1996. Available at `www.cs.unc.edu/~barman/HT96/`.
8. F.G. Halasz. Reflections on notecards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM*, 31(7):836–852, 1988.
9. F.G. Halasz and M. Schwartz. The dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.
10. D.L. Hicks, J.J. Leggett, P.J. Nurnberg, and J.L. Svhnase. A hypermedia version control framework. *ACM Transactions on Information Systems*, 16(2):127–160, April 1998.
11. J.J. Hunt and J. Reuter. Using the web for document versioning: An implementation report for DeltaV. In *Proc. of the 23rd Int. Conf. on Software Engineering*, pages 507–513, Toronto, May 2001. IEEE Press.
12. P. Návrat and M. Bieliková. Knowledge controlled version selection in software configuration management. *Software – Concepts and Tools*, 17:40–48, 1996.
13. I. Noris. Building a configuration of hypertext documents. Master's thesis, Department of Computer Science and Engineering, Slovak University of Technology, 2000. (supervised by Mária Bieliková).
14. I. Sommerville, T. Rodden, P. Rayson, A. Kirby, and A. Dix. Supporting information evolution on the WWW. *World Wide Web*, 1(1):45–54, January 1998.
15. W.F. Tichy. RCS - a system for version control. *Software-Practice and Experience*, 15(7):637–654, July 1985.
16. F. Vitali. Versioning hypermedia. *ACM Computing Surveys*, 31(4es), Dec. 1999.
17. F. Vitali and D.G. Durand. Using versioning to support collaboration on the WWW. In *Proc. of 4th World Wide Web Conference*, 1995. Available at `www.w3.org/pub/Conferences/WWW4`.
18. E.J. Whitehead. *An analysis of the hypertext versioning domain.* PhD thesis, University of California, Irvine, 2000.
19. E.J. Whitehead. Uniform comparison of data models using containment modeling. In *Proc. of ACM Conf. on Hypertext - HT'02*, pages 182–191. ACM, June 2002.