Advanced Engineering Informatics 25 (2011) 177–184

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

A semantic file system for integrated product data management

Oliver Eck [a,*], Dirk Schaefer [b]

[a] Department of Computer Science, HTWG Konstanz, Germany
[b] Systems Realization Laboratory, Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, USA

ARTICLE INFO

Article history:
Received 31 October 2009
Received in revised form 10 June 2010
Accepted 17 August 2010
Available online 20 September 2010

Keywords:
Information retrieval
Semantics
Semantic file system
Engineering design
PDM
PLM
CAD
CAE

ABSTRACT

A mechatronic system is a synergistic integration of mechanical, electrical, electronic and software technologies into electromechanical systems. Unfortunately, mechanical, electrical, and software data are often handled in separate Product Data Management (PDM) systems with no automated sharing of data between them or links between their data. Presently, this is a significant drawback with regard to supporting the collaborative and integrated design of mechatronic systems. One underlying research question to be addressed is: "*How can such domain-specific PDM systems be integrated and what are the standards needed for this*?" Our starting point for addressing this question is to look into how engineering data is stored today. We believe that the more intelligent and sophisticated the mechanisms for file management are, the more Computer-aided Engineering systems will benefit from them in terms of making integrated cross-disciplinary product development more efficient. However, while Computer-Aided Design and Engineering (CAD/CAE) systems and related tools such as, for example, Product Data Management (PDM), Engineering Data Management (EDM), etc. have significantly matured over the past ten years, most of their associated database systems are still based on traditional file systems and hierarchical directory structures.

In light of this context, we will initially discuss a number of disadvantages of current file management systems. In the body of the paper our main contribution is presented. That is, a formal mathematical model of a new semantic file system, *SIL* (*Semantics Instead of Location*), that allows engineers to access data based on semantic information rather than storage location is proposed. A major difference between our approach and previous related work is that we do not aim at yet another point solution and, instead, propose an approach that may be employed by next generation engineering data processing systems on a larger scale. In addition, a corresponding programming interface along with a graphical user interface used as a file browser is presented and the benefits of utilizing the proposed semantic file system for product data management in the field of integrated design of mechatronic systems are discussed.

1. Introduction

1.1. Motivation and background

A mechatronic system is a synergistic integration of mechanical, electrical, electronic and software technologies into electromechanical systems. Mechatronic systems are excellent candidates for design process optimization due to the high complexity of mechatronic design, the high degree of integration of electrical, mechanical and information-processing components, the overlapping design disciplines and system behaviors, and the critical nature of optimizing the overall system. Unfortunately, mechanical, electrical, and software data are often handled in separate Product Data Management (PDM) systems with no automated sharing of data between the systems or links between the data. As a result, a significant drawback exists with regard to supporting the collaborative and integrated design of mechatronic systems [3]. In light of this, the key research question we address is: "*How can such domain-specific PDM systems be integrated and what are the standards needed for the integration*?" The work presented in this paper is a contribution towards answering the first part of the above-stated research question.

While PDM and other related Computer-Aided Design and Engineering (CAD/CAE) systems have significantly matured over the past ten years, their underlying database systems are still based on traditional file systems and hierarchical directory structures. However, in order to facilitate integrated cross-disciplinary computer-aided product creation, new sophisticated mechanisms for intelligent information retrieval and file management are required.

Most engineering information systems store and retrieve data in the form of files of different types. Especially in the computer-aided engineering and design field, the number of files associated with individual products, parts, components and their associated

* Corresponding author. Tel.: +49 7531 206 630; fax: +49 7531 206 559.
E-mail address: eck@htwg-konstanz.de (O. Eck).

1474-0346/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.aei.2010.08.005

CAD/CAE/PDM/PLM systems and system users has reached a level of vast complexity and little transparency [4]. This is due to the fact that current file systems are still based on traditional hierarchical directory structures. In such file systems, users are required to organize their data in hierarchical directory structures. In such directory structures, specific path and file names have to be determined and correctly entered in order to store and retrieve data. Unfortunately, these traditional hierarchies often do not scale to large data collections and to the fine-grained classifications required [5]. As the data classification hierarchy grows, files are increasingly likely to be assigned several different classifications that are all linked to only one associated file location. To locate and retrieve a particular file at a later time, system users must remember the one specific classification that was originally chosen.

Another significant issue with traditional file systems is that information, which identifies a document, must be captured in both the pathname and filename. Unfortunately, standards do not exist for describing the exact information to be represented in pathnames and filenames, respectively. Today, many companies define their own proprietary file and path naming conventions, including guidelines for the utilization of version numbers, status flags, file creation dates, filenames, and information regarding the person or system used to create a file. For example, in many of our collaborative industrial projects typical file names are of the following signature: 'CB_Spec_driveShaft_OE_draft_V9.doc', which represents the following information: the customer is CB, the document includes the draft specification of a drive shaft, the author initials are OE, and the document version number is 9. As mentioned before, such guidelines are proprietary and usually not strictly enforced. Presently, the only consistent aspect of file naming that represents at least a minimum of semantic information regarding file generating source and content is that of document type identifiers such as .dwg, .doc, .pdf, etc. For example, .dwg stands for a drawing created with AutoCAD, and .doc stands for a text file created with Microsoft Word.

Sharing traditional file systems among several users normally leads to a number of inherent problems. Users often have to guess or anticipate the correct words employed by other users to denote pathnames and filenames. For example, a particular *partA.dwg* used for a *machineTybeB* sold to *customerC* might be stored in the directory /customerC/machineX/partA.dwg or in the directory /machineX/customerC/partA.dwg, depending on the path and file naming conventions of the user who created and stored the file. As a result, users need to remember not only the correct keywords associated with a document but also the mapping of these keywords to a particular directory structure. If the subdirectories 'customerC' and 'machineX' appear at the root directory, users have to browse both subdirectories to locate the drawing file being sought. From a logical perspective, the file should appear in both directories, which simply is not possible in traditional file systems due to redundancy avoidance conventions. The definition of guidelines for organizing the path hierarchy and naming files resolves this problem only partially.

Recently, engineering tools have been developed that allow describing meta-information in files in order to improve the effort of document retrieval. For example, in the area of Model-Based Systems Engineering (MBSE) [2] and multi-view modeling [30], a number of standards and formats for describing meta-information of a system or a class of systems to be designed are used. Examples include model descriptions in SysML [7] or Modelica [8], to mention just a few.

All the before-mentioned meta-information-based efforts address the same important goal of improving file access and retrieval. However, the main problem with the existing meta-information-based approaches and their corresponding meta-information is the fact that not all files are described completely since the primary mandatory file description is still given by pathname and filename. Therefore, file information is separated into two areas that cannot be used together in a single efficient file retrieval method. Furthermore, efficient retrieval facilities require meta-data to be stored in a central database instead of distributed across thousands of files within the file system.

The utilization of semantically annotated ontology has become an established approach in numerous information retrieval tasks, and semantic annotation has emerged as an important research area. Semantic annotation can be described as both meta-data and the process of generating meta-data. Semantically annotating a document provides, as output, meta-data related to the document. On the other hand, ontology is a formal model describing some knowledge domain and consists of concepts, such as types and properties, and the relationships between concepts. Together, semantically annotated ontology within the scope of integrated computer-aided product design provides a formal methodology for the extraction of meta-data from diverse information sources such as domain-specific product design files coupled with their ontological relationships. Adopting the concept of semantically annotated ontology from Semantic Web applications to the field of computer-aided product design has shown to be very promising for design-related retrieval tasks. Therefore, research approaches for semantics-based engineering document analysis and retrieval as well as ontologies for Engineering Information Retrieval (EIR) are becoming more prominent [20,21].

1.2. Scope of work and contribution

In this paper, we propose a semantic file system, SIL (Semantics Instead of Location), which allows engineers to access data based on semantic information rather than storage location information thereby facilitating enhanced search and retrieval capabilities as needed for integrated, cross-disciplinary computer-aided product development processes.

In contrast to previous approaches such as [20,24], SIL does not attempt to realize yet another standard retrieval system with slightly more sophisticated retrieval mechanisms. Instead, the goal of SIL is to provide a semantic-enabled intelligent file system that provides the search mechanisms necessary for both effective and efficient retrieval on a software layer that may be used as a common platform for the enhancement of future integrated computer-aided product design systems and other related engineering data retrieval systems such as PDM, CAD, and CAE tools. In other words, instead of developing yet another point solution, our long-term goal is to create retrieval mechanisms based on a standard software architecture that may be embedded (or 'plugged') into future computer-aided product development applications. An example of a similar idea within the context of standard office software is the opportunity to install a single PDF software module that automatically enhances all installed office applications (Word, Excel, Access, PowerPoint, etc.) with the same additional functionalities and capabilities. Again, we would like to stress that our idea is aimed at providing a far-range solution of significant impact on an entire field rather than yet another stand-alone approach.

In order to represent file semantics and to provide associative access to a semantic file system, meta-data information is required. SIL represents property-based semantics and content-based semantics. The proposed file system paradigm is based on a set of keywords that describe the content-based semantics of data files. This information is based on two sources: user input and content analysis. Content analysis is realized by the extraction of meta-information contained within the data files. In the above outlined engineering context, semantic information of documents is almost always available. For example, semantic annotations for

product lifecycle management can be used as keywords as well as meta-data, which are automatically added through text processing when a data file is modified and saved. Extracting the meta-data from files and storing it in a central database enables efficient retrieval facilities to evaluate retrieval queries with a very short execution time.

In the proposed approach, such annotations are used to identify and locate files. In contrast to traditional tree-structured file systems, these keywords are not ordered and classified in a hierarchy. In contrast to other related approaches for semantic file systems, tree-structured file systems are completely replaced by a new structure of keywords specifically tailored to the requirements of collaborative design activities across technical disciplines. By adding ontology technologies, information search in the semantic file system is related to information search in the Semantic Web or other advanced information management systems, which can be integrated more easily than traditional file systems.

In this paper, we introduce a formal definition of a new semantic file system. In addition, a corresponding programming interface along with a graphical user interface used as a file browser is presented. Further, we present the benefits of utilizing such a semantic file system for product data management in the field of integrated design of mechatronic systems.

In summary, the presented semantic file system, SIL, provides the following features, which collectively are not available in any traditional file systems known to the authors. A detailed discussion of these features follows in Section 3.

- **Multi-path accessing**:
  The concept of multi-path accessing allows many paths leading to a file.
- **Enabling of efficient retrieval facilities**:
  A central database stores meta-data of the file and user specification collectively. Indexing of this database allows the evaluation of queries in a very fast execution time and makes possible efficient retrieval facilities.
- **Tag-based browsing**:
  Tag-based browsing provides the benefits of subdirectories of hierarchical file systems working with subset of files.
- **Formal model of semantic file systems**:
  The formal model of the semantic file system provides a precise definition of the file system behavior.
- **Automatic tag extraction**:
  User tags are completed by tags extracted automatically from the document itself. This rather specific information allows a more precise specification and improved retrieval facilities.

2. Related work

In general, the mechanism of tag-based retrieval is widely used in a variety of information systems that are required to process large quantities of data. Examples range from software applications for engineering data analysis and computer-aided product development to more general information systems that manage text, audio [17], video, photo [19], and mail files. In this section, we provide a brief overview of previously developed approaches for and elements of tag-based retrieval that are of relevance to the work presented in this paper.

Mendeley Desktop [25] is well known software for indexing and organizing Portable Document Format (PDF) documents and research papers into a personal digital bibliography. Document details are gathered from PDFs allowing one to effortlessly search and organize a bibliography. Similarly, indexing and tag-based technology is also used by many other retrieval systems for engineering-related documents [20,24]. Unfortunately, these approaches are isolated software solutions tailored to very specific and narrow contexts and application fields. Although many tag-based approaches share the central goal of efficient document retrieval, the approach presented in this paper addresses the problem from a different perspective. Instead of developing another isolated software system that would only lead to more compatibility issues, a new file system that may be embedded (or plugged) into engineering software applications via a software layer is proposed. Therefore, most of the following literature review focuses on approaches and extensions of traditional file systems.

In a number of approaches, associative access to documents is integrated into a tree-structured file system through the concept of a virtual directory ([10,26]). These virtual directories are interpreted as queries and provide flexible associative access to files and directories. In contrast to separating between real and virtual directories, in the approach presented in this paper tree-structured directories are replaced by a structure of keywords that do not distinguish between pathnames and filenames.

A file system that provides both simultaneous name and content-based access to files is presented in [11]. The concept of a semantic directory is introduced, associating every semantic directory with a query. Whenever a user creates a semantic directory, the file system automatically creates a set of pointers to the files in the file system satisfying the query associated with the directory.

In [27], the topic of file semantics is split into three classes: property-based semantics, content-based semantics and context-based semantics. Property-based semantics is defined as general content and context independent information related to files, such as owner, creation date, last modified date, and the type of file. In many approaches, these semantics are used for all files and shared for all applications and users. In the content-based semantics class, the internal organization of file content is represented. Roles are used to specify text and provide text-based searching. [27] defines context-based semantics as the information about the interrelated conditions in which a file exists. Therefore, context-based semantics express relationships of files with other subjects, such as other files, concepts, persons, etc.

The first archival file system that integrates semantic storage and retrieval capabilities is presented in the system Sedar [22]. Sedar introduces several novel file system features such semantic-hashing to reduce storage consumption, virtual snapshots of the namespace, and conceptual deletions of files and directories.

Connections [31] is a file system search tool that combines traditional content-based search with context information gathered from user activity. Connections can identify relationships between files by tracing file system calls and use them to expand traditional content search results.

Desktop search tools such as Google Desktop [12] and Glimpse [23] use hierarchical file systems to create an index of the stored files. The index provides a fast, associative text-based file search. In contrast to the approach presented here, support of these tools is limited to searching and does not support the creation and management of the file system taxonomy.

In [6], Semantic Web technologies are used to replace the hierarchical file system for file management. Semantic classification is used to support the transition of semantic file system interfaces. Additionally, a file system path construction is presented using semantic web ontologies and Resource Description Framework (RDF) data to build generic file system interfaces.

The proposed approach focuses on property-based and content-based semantics of files in order to share meta-information between several users and to use it in enhanced search facilities. Information relevant only to individuals should not be stored. In these regards, approaches for personal information management can be found under the topic of semantic desktop research [34].

Another well-known approach that uses file meta-data to organize and retrieve documents from relational database systems is

WinFS by Microsoft [13]. WinFS was planned to become a part of an upcoming version of Microsoft Windows, but has not yet been realized. In WinFS, a SQL database system that supports powerful queries is integrated into the operating system. Attributes are used to tag a data item's meta-data, and relationships are used to associate data items together. The semantic file system presented here focuses on meta-data and does not provide capabilities such as file relationships. However, the presented file system takes previously developed meta-data concepts a step further by providing automatic meta-data extraction, meta-data indexing or multi-path accessing, which leads to robust and efficient retrieval mechanisms with fast performance.

[5] develops and evaluates several new approaches to automatically generate file attributes based on context. Context-based attribute assignment is realized by introducing approaches from two categories: application assistance and user access patterns.

Other systems using semantics in conjunction with their data include the Semantic Web and the Semantic Desktop [27]. The World Wide Web Consortium (W3C) promotes Semantic Web technologies that enable people to share content beyond the boundaries of applications and websites [1]. The Semantic Web relies on formal ontologies that structure underlying data and specifies a representational vocabulary for a shared domain of discourse [28]. The core idea of Semantic Desktops is to bring Semantic Web technologies to the desktop. People are enabled to use their desktop computers like a personal Semantic Web where applications integrate and ideas are connected through ontologies [29].

3. A semantic file system for integrated product data management

In this section, the new semantic file system SIL (Semantics Instead of Location) is introduced. First, the approach is presented from the perspective of an engineer working with the system. The associated concept of tag-based browsing is then presented in Section 3.2, along with a demonstration of how the system can be used in file retrieval facilities. Second, a formal mathematical model that specifies the properties of the file system is defined in Section 3.3. Finally, the important issue of identifying document tags by extracting them from existing documents is addressed in Section 3.4.

3.1. The semantic file system – semantics instead of location (SIL)

The presented file system paradigm to store and search for files is based on a set of tags that describe the content-based semantics of the file. These keywords are used to identify and locate files. In this context, *tagging* means to *tag* files with additional information that describes the files.

According to the application of relations in other semantic representation structures, e.g. [32], semantics are represented by relationships. In conceptual graphs or ontologies, the concept of role types is used to describe interactions of instances. However, in contrast to these formalisms, roles do not connect classes or individuals in SIL. Instead, roles assign keywords that describe the data file to which they are attached.

A *tag* consists of two components:

- optional role name
- mandatory keyword

The role name is used to clarify the semantics of the tag and its corresponding keyword. Further, roles make the semantics of the associated keywords more precise and meaningful. When the optional role name is missing, the corresponding keyword is attached without additional description. The notation of *keyword* is used as a synonym to *attribute* of related approaches. The keyword defines the value of the tag. The type of value in regards to the role name and keyword is defined as a string.

The following information is relevant and mandatory for *all* documents and is, therefore, stored separately and not modeled by tags:

- Creation date
- Last modified date
- Version number
- Document size
- Document type

Mandatory information is defined as property-base semantics within the meta-model itself (see Section 4) and can be identified automatically by the system. All other optional information is modeled by tags, which provide a more flexible way of defining information relevant for single documents.

As an example, a component within a mechatronic system may include a large number of attributes needed for its identification. Examples include component id number, functional descriptor, machine or assembly in which the component is used, design date, and so forth as well as models or meta-models describing templates of system components or functions. In contrast to standards that may define predefined attributes for specific file types, SIL is not limited to specific file types and their attributes.An example of a CAD file of a drive shaft designed in 2008, used in a machine of type A, and sold to customer B is summarized in Fig. 1. This figure shows two tags using a role to describe the semantics of the corresponding keywords, and two keywords without a role showing a general characterization of the drawing file's content.

In contrast to tree-structured file systems storing the drawing, for example, in the file '/2008/DriveShaft/CustomerB/machine.dwg', these keywords are not ordered. When a user searches a drive shaft by giving the customer name, all drive shafts ever used in any machine sold to that customer are selected. Alternatively, when the user gives a design year, all drive shafts designed in the given design year are presented. In traditional directory structures, the drive shaft is stored in several subdirectories. This property of SIL is called *multi-path accessing* in [27].

3.2. Tag-based browsing

When presenting a semantic file system that does not provide hierarchical directory structure to engineers, we were often told that the concept of subdirectories is missing. Subdirectories provide the ability to group a huge amount of files in subgroups and allow work with files in a more manageable amount of documents. In this section, it is shown that SIL allows for utilizing subgroups in a much more flexible way. Using tag-based browsing actually provides a visualization of the semantic file system related to the tree-structure of traditional file systems.

Role	Keyword
Design Year	2008
Part Name	Drive Shaft
	Customer B
	Drive Shaft of Machine A

Fig. 1. A tagging example of a CAD drawing.

When searching for stored documents in file systems, generally two search facilities are possible:

- Searching by query
- Searching by browsing

First, users can create a query describing the documents for which they are searching. A search engine evaluates the submitted query and returns the set of files containing matching documents. This way of searching for documents demands that the user knows what he/she is asking for. When the user does not know the keywords of the wanted document, a browsing search method is necessary in presenting the user tags of documents and letting him/her choose these keywords that best describe the wanted document. SIL not only provides both search procedures separately, but also in combination. Therefore, it is possible to search by browsing on the results of a query or to ask a query on a selected subgroup selected by a browser. Since the first search facility can be realized by simple input fields of a user interface, in this paper we focus on the second, more interesting facility.

The presented approach to tag-based browsing allows users to browse files in a search tree similar to a traditional directory structure. Since tags in the SIL system show a structure similar to directory structures, the selected keywords can be presented in a search structure similar to directory names in traditional file systems. This leads to a new search procedure called *tag-based browsing*. A browsing tool can use this structure to give users the chance to browse documents and tags when searching for documents if the set of keywords are unknown. At the beginning of the search procedure, no keywords are selected and therefore all files with all their available keywords are presented. Therefore, at the root of the structure, all keywords are shown that are used to describe the stored files. Additionally, all documents that are tagged by all selected keywords can be shown.

In contrast to subdirectories in hierarchical file systems, the 'explorer tool' shows tags that are associated with the selected files and which are qualified to limit the number of selected documents by additional keywords. These tags are referred to as *option tags*, while the selected keywords are simply named *selected tags*. The set of files selected by the selected tags are called *selected files*.

When an additional tag is selected, the following happens:

- The set of selected tags is added by the new tag
- The set of selected files is reduced to the subset of files that are tagged with this keyword
- The set of option tags is reduced to the subset of tags appearing in the selected files

The property, that the number of optional tags and the number of selected files decreases when the user adds keywords to his set of selected keywords, ensures that the search process leads to the files being sought. In the next section, our formal model of tag-based browsing provides a formal specification of the system behavior and proves this important "conjunctive" property of selected file reduction.

In Fig. 2, a visualization of a browser realized in an application similar to current file explorers is shown. A ' + ' indicates that the diagram is expanded at this node, a '-' indicates that it is not expanded. A mouse click on an expanded node contracts the graph; a mouse click on a contracted node expands it. At the root of the browsing tree, all keywords that are used to tag files are presented and all files of the file systems are selected.

At first glance, tag-based browsing appears to be very similar to traditional tree browsing. Therefore, the differences of the presented approach need to be emphasized. Tag-based browsing provides multi-path access, which means that when a document

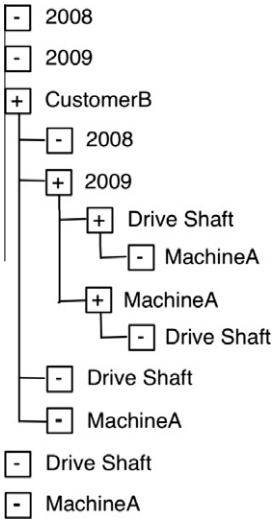

Fig. 2. Tag-based browsing.

appears in the browsing structure in A/B, it also appears in B/A where A and B are directories. In the example of Fig. 2, the drawing of the drive shaft is located in '2009/CustomerB/MachineA/DriveShaft' as well as in 'CustomerB/2009/MachineA/DriveShaft' or 'CustomerB/MachineA/DriveShaft/2009', meaning that the result of the query is independent from the order in which tags are selected. Additionally, we emphasize that no separation between filename and directory name exists even when the browsing diagram shows a graph similar to a directory tree.

Even this simple example clearly highlights some of the advantages of SIL compared to traditional file systems. Additionally, when a document is stored in 'DriveShaft/CustomerB/2009' in hierarchical file systems and a user searches in /2009, he will not find the document since he is searching in a false directory subtree. However, a document's location and its name are not relevant in SIL. With computer-aided engineering tools, the complexity and amount of data is significant and, therefore, the benefits of SIL have much more impact on the search process. Regardless of these differences in the underlying structure, the browser appears very familiar to users, and first tests of the prototype shown in Section 4 have shown that users are working easily with the browsing structure.

3.3. Formal model for semantic file systems

In this section, a formal mathematical model of the concept of tag-based browsing is provided. In order to make the formal model easier to understand, an informal textual specification of tag-based browsing was presented in the previous section. The formal model defines the behavior of the semantic file system, which prevents inconsistencies and incompatibilities between different implementations of the semantic file system's specification.

Let R be a set of roles, K a set of keywords, and F a set of files. The set of tags is defined as $T = R \times K$.

The mapping g is defined as g: $F \rightarrow \wp(T)$, where g maps a file to a set of tags that are associated to the corresponding file[1].

S is defined as the set of tags selected by a user in a search process (*selected tags*). We have: $S \subseteq T$. At the beginning of the search process, we assume $S = \varnothing$.

[1] In ontologies, conceptualization is a set of conceptual relations defined on a domain space [14,33]. Since objects are attached to keywords in the presented approach, a function is used to define roles here.

Let F_S be defined as the set of files that are conjunctively associated to all tags of the set S.

$$\textbf{Definition} : F_S = \{f \in F \mid S \subseteq g(f)\} \quad (1)$$

$$\textbf{Proposition 1} : S1 \subseteq S2 \Rightarrow F_{S2} \subseteq F_{S1} \quad (2)$$

Proof: Let us assume the contrary, i.e.: $S1 \subseteq S2$ and $F_{S2} \not\subseteq F_{S1}$, i.e., $\exists f \in F_{S2}, f \notin F_{S1}$. According to (1) we have $S2 \subseteq g(f)$. $f \notin F_{S1} \Rightarrow \exists t \in S1 : t \notin g(f)$, i.e. we have $S1 \not\subseteq S2$ and a contradiction. Let $OS \subseteq T \setminus S$ be the set of *option tags* that allows a more precise selection on the selected files.

$$\textbf{Definition} : O_S = \{t \in T \setminus S \mid \exists f \in F_S : t \in g(f)\} \quad (3)$$

$$\textbf{Proposition 2} : F_{S2} \subseteq F_{S1} \Rightarrow O_{S2} \subseteq O_{S1} \quad (4)$$

Proof: Let us assume the contrary, i.e.: $F_{S2} \subseteq F_{S1}$ and $O_{S2} \not\subseteq O_{S1}$, i.e. $\exists o \in O_{S2}, o \notin O_{S1}$. $o \notin O_{S1} \Rightarrow \neg\exists f1 \in F_{S1} : o \in g(f1)$. According to (3) we have $\exists f2 \in F_{S2} : o \in g(f2)$ and a contradiction to $F_{S2} \subseteq F_{S1}$.

$$\textbf{Proposition 3} : S1 \subseteq S2 \Rightarrow O_{S2} \subseteq O_{S1} \quad (5)$$

Proof: Transitivity of (2) and (4). Proposition 3 verifies the decrease of selected files and option tags when additional tags are selected.

3.4. Extracting file content

Detailed and comprehensive semantic information are a premise for efficient retrieval facilities with complete retrieval results. The quality of the meta-information of files has a great impact on the effectiveness of the entire semantic file system. In general, tags that classify files based upon a set of keywords have to be entered manually. These keywords have to be selected carefully since they are used to identify files and are the only way to get the files back from the file system. Even if this activity is very time-consuming, the file creator has the best understanding of their files and is able to describe his/her file precisely. User input is often regarded as a drawback since users worry about locating their file again after storing it in SIL. However, it has to be remembered that when using traditional file systems, files are lost as well when the user doesn't remember the pathname and filename of his/her document.

While filenames and pathnames are mandatory in traditional file systems, the SIL system has to enforce that files are adequately specified. For this reason, we describe below how content analysis is realized by extracting tags automatically from the document.

Meta-data information can be used in search tools to provide enhanced search facilities. However, search tools using this meta-data are restricted to single standards and users have to switch between several tools when searching for files of different types. Extracting file meta-data to SIL makes this hidden information visible and provides a uniform search facility to all meta-data. Meta-data can be used in retrieval facilities with good performance when the meta-data is stored in a central database and not in thousands of files itself. In database systems, indexing allows the evaluation of queries on meta-data in a very fast execution time, which makes efficient retrieval possible at all.

Additional to one's manual specification of meta-data, the SIL approach considers the following automatic tagging techniques.

- Extraction of meta-data stored in the files themselves
- Extraction of meta-data from file content
- Reuse file or directory names of files stored in traditional file systems as tags

One technique to collect file content without manual input is to extract meta-information stored in the file itself. This meta-data provides higher level information about the files concerned. Values of this meta-data can be processed by many software systems. Each meta-data entity is called a property, and properties are grouped into administrative, descriptive and rights related properties.

In the architecture of a prototype of the presented approach, a meta-data extractor is introduced that extracts different meta-data into the semantic file system (Fig. 3). Since the meta-model of the file system is generic and not limited to certain attribute names, meta-information of all standards can be transformed into the system. A standard-specific filter has to be provided for each meta-data standard in order to integrate the meta-data into the file system.

The extraction of meta-data from file content requires intelligent mechanisms for analyzing the content of specific file types. An example of a system doing this extraction for PDF files is the system Mendeley [25], which extracts references of research PDF papers.

4. Methan – SIL prototype

The presented approach to realizing a new semantic file system, SIL, has been implemented as a prototype system called *Methan* in order to prove the concept, validate it and show its applicability. Methan is a Java-based application providing flexible associative access facilities by a programming interface, a shell command interface and a graphical explorer. The architecture of Methan is depicted in Fig. 4.

The graphical design of the Methan explorer, shown in Fig. 5, is similar to that of other explorers within current operating systems in order to facilitate a wider acceptance among users. The explorer consists of a browser that shows the structure shown in Fig. 1 and allows searching for documents by expanding and contracting nodes of the graph. Even if the structure presents no tree, the structure is called a *tag tree*. Additionally, an extended search is provided by text input. When keywords are typed into the input field, the structure is expanded automatically according to the searched keywords. This allows users to input known keywords and subsequently browse in the tag tree.

A functional filesystem is a userspace program that can be implemented easily using FUSE [9]. This module allows non-privileged users to create their own file systems without editing the kernel code and is available for several operating systems. The shell command interface of Methan expands the UNIX shell commands by instructions for operating with SIL. UNIX commands like *cd* or *ls* operating on hierarchical file systems are replaced by commands that operate on SIL. Here commands are added like *addTag* to add a tag, *remTag* to remove a tag, *lsfiles* to list files with the given keywords, or *lsopTag* to list option tags.

The extraction of ID3-meta-data of audio documents is realized in the Meta-data Extractor by the Jaudiotagger User API v1.0.9 [15]. IPTC-data of photos is extracted using Imagero 3.0 programming interface [18].

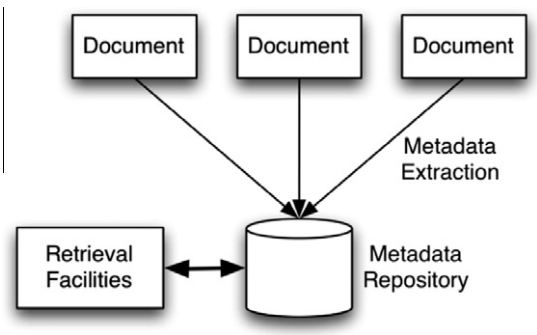

Fig. 3. Metadata Extraction.

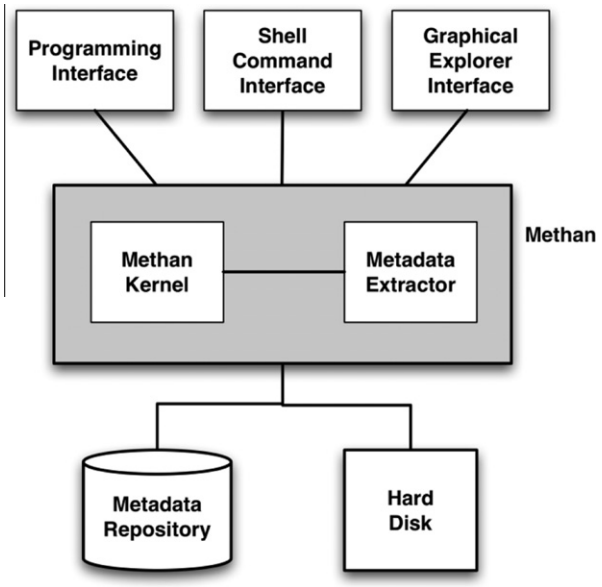

Fig. 4. Methan – a prototype of SIL.

Accessing documents from the hard disk is realized on top of the NTFS file system. This is of course a temporary solution since the goal of the file system is to replace current file systems. But as a first prototype, our implementation provides a feasible solution for the purpose of evaluating the benefits of our proposed semantic file system. Surprisingly, in spite of this rather inefficient implementation, the efficiency of the file system exceeded expectations even with a huge amount of data used. The Meta-data Repository is realized by the H2 database system [16]. H2 is a very fast, open source JDBC-API, which allows SQL-scripts to run in embedded mode. An evaluation of several database systems showed that the JDBC-API, a small but fast database, is more adequate to fulfill the requirements of Methan than fat client server database systems.

Fig. 6 shows the entity relationship model of the H2 database system representing the meta- model of SIL. Property-based semantics are represented by the attributes *CreateDate*, *LastModDate*, *MmeType*, *Size* and *VersionNr*. Documents are identified not only by their keywords, but also by a version number. This version number supports the integrated product data management working on several versions of documents. Each time a file is modified and closed, a new version of the file is produced. Similar to [22], meta-data is not versioned. Therefore, version information and tags are represented in different entites in the model. Attributes *PhysPath* and *PhysName* describe the location of files that are stored in the NTFS file system. When the semantic file system is not realized on top of a hierarchical file system, these attributes have to be replaced by attributes indicating the physical storage location on the hard disk. In order to speed up retrieval of data, attribute *Keyword* of entity *Tagvalue* and attribute *Rolename* of entity *Role* are indexed in the H2 database system. As first tests have indicated, these indexes allow for excellent retrieval performance and thus make the retrieval performance one of the main benefits of the Methan system.

5. Closing

The presented semantic file system SIL is applicable to all operating systems and all fields of application. The advantages of the file system become important when users are required to work with a large amount of complex documents that relate to significant amounts of meta-data. Therefore, fields such as computer-aided engineering and design, PDM, PLM, etc. are predestined for applying the semantic file system. However, in order to demonstrate the prototype system without having to gain access to a large number of CAE-related data files, Methan was evaluated and tested while managing approximately 5000 photos and audio files with great success. This demonstrates that SIL is highly transformative in nature and actually can be utilized in any application domain for which meta-information can be extracted from files. Feedback from alpha testers confirms that SIL is very easy to use and that a significant increase in efficiency with regard to search tasks can be achieved.

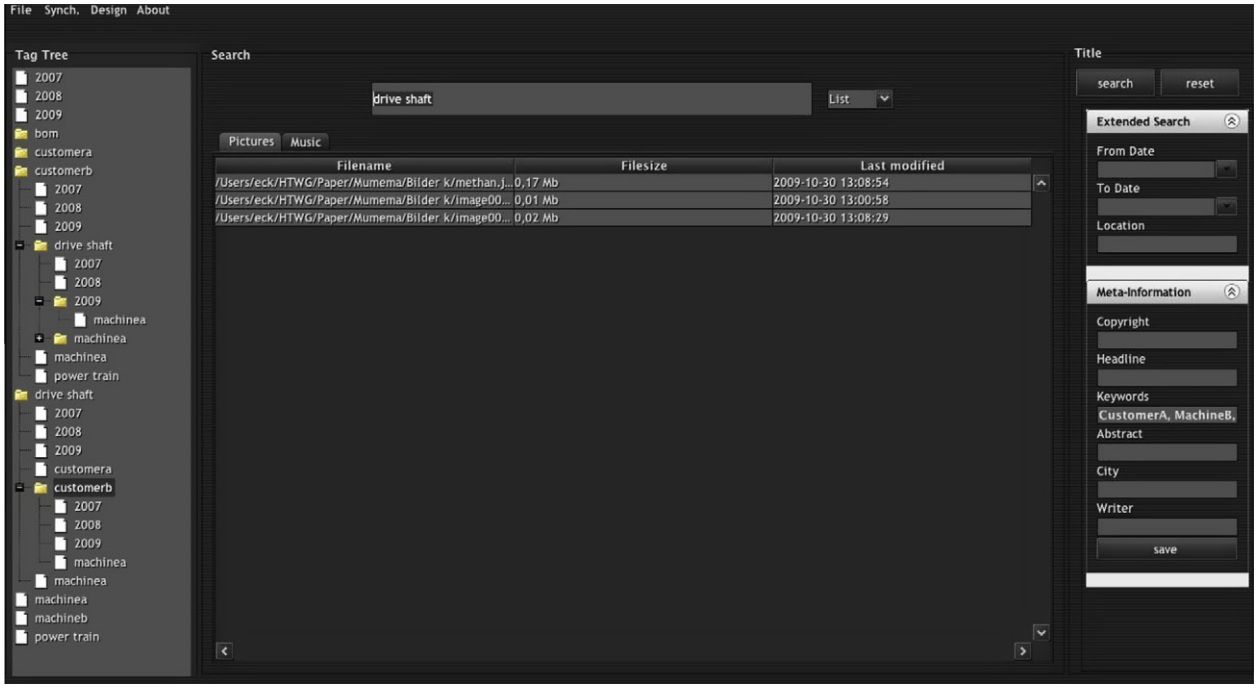

Fig. 5. The graphical explorer interface of Methan.

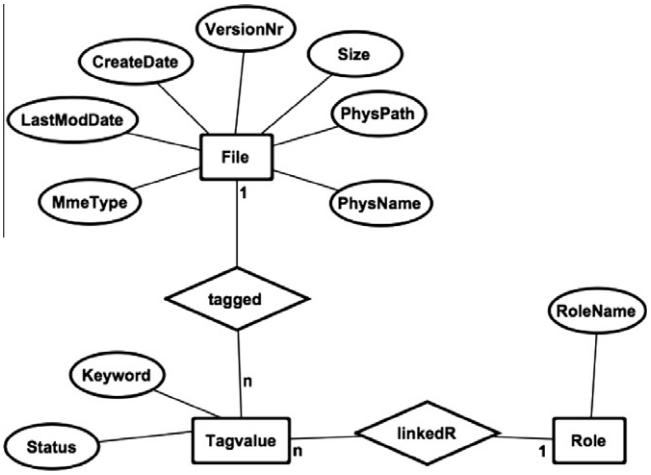

Fig. 6. The meta-model of the Meta-data Repository.

In order to get deeper semantics in describing digital objects and enable to share meanings of terms to overcome barriers between several engineering departments, it is planned to integrate ontologies into our semantic file system. Related works have shown great promise for integrating ontologies into semantic file systems [27]. The goal is to use ontologies to enhance knowledge sharing by specifying property-based semantics, content-based semantics, and context-based semantics.

An extension our proposed semantic file system is planned to integrate context-based semantics. Context-based semantics would allow representing the fact that two documents represent audio and visual information of the same object, which would be very helpful in search facilities.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web, Scientific American, 2001.
[2] D.M. Buede: The Engineering Design of Systems: Models and Methods, Wiley, 2009, ISBN 0470164026.
[3] K. Chen, J. Bankston, J. Panchal, and D. Schaefer, A framework for the integrated design of mechatronic products", In: L. Wang and A.Y.C. Nee. (Eds.), Collaborative Design and Planning for Digital Manufacturing, Springer, ISBN 184822863, pp. 37–70.
[4] W. Cheung, and D. Schaefer, Product lifecycle management: state-of-the-art and future perspectives, In: M.M. Cruz-Cunha (Ed.), Enterprise Information Systems for Business Integration in SMEs: Technological, Organizational and Social Dimensions, (Advances in Information Management book series), IGI Global Publishing, ISBN 978-1-60556-892-5, pp. 37–55.
[5] A. Craig, N. Soules, Gregory R. Ganger, Toward automatic context-based attribute assignment for semantic file systems, Technical report CMU-PDL-04-105. June 2004.
[6] S. Faubel and C. Kuschel, Towards Semantic File System Interfaces, 7th International Semantic Web Conference ISWC2008, 2008.
[7] S. Friedenthal, A. Moore, R. Steiner, A Practical Guide to SysML, Morgan Kaufmann, 2009, ISBN 012378607X.
[8] P. Fritzson: Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, Wiley, 2004, ISBN 0471471631.
[9] http://fuse.sourceforge.net/.
[10] D.K. Gifford, P. Jouvelot, J.J. O'Toole, and M.A. Sheldon, Semantic File Systems, ACM Operating Systems Review, Oct. 1991, pp 16–25.
[11] B. Gopal and U. Manber. Intergrating Content-based Access machanisms with Hierarchical File Systems. in: Usenix OSDI. 1999. New Orleans, Louisiana, USA.
[12] Website of Google Desktop: http://desktop.google.com.
[13] Richard Grimes: Revolutionary File Storage System Lets Users Search and Manage Files Based on Content, http://msdn.microsoft.com/en-us/magazine/cc164028.aspx.
[14] N. Guarino (Ed.), Formal Ontology in Information Systems, in: Proceedings of FOIS'98, Trento, Italy, 6-8 June 1998. Amsterdam, IOS Press, pp. 3–15.
[15] http://www.jthink.net/jaudiotagger.
[16] Website of H2 database system: http://www.h2database.com.
[17] Website of ID3: http://www.id3.org/.
[18] Website of Imagero: http://reader.imagero.com.
[19] Website of the International Press Telecommunications Council: http://www.iptc.org.
[20] Z. Li, V. Raskin, and K. Ramani, 2007. Developing ontologies for engineering information retrieval. in: Proceedings of the ASME 2007 IDETC/CIE 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Las Vegas, Nevada, USA, September 4–7, 2007. pp. 1–9.
[21] S.C. Lim, Y. Liu, and W.B. Lee, 2009. Faceted search and retrieval based on semantically annotated product family ontology. in: Proceedings of the WSDM '09 Workshop on Exploiting Semantic Annotations in information Retrieval, Barcelona, Spain, February 09-09, 2009). ESAIR '09. ACM, New York, NY, pp. 15–24.
[22] Mahalingam, C. Tang, Z. Xu, Towards a semantic, deep archival file system, in: The 9th International Trends of Distributed Computing Systems (FTDCS), 2002.
[23] U. Manber, S. Wu, Glimpse: a tool to search trough entire file systems. in: Proceedings of the USENIX Winter Conference, 1994, pp. 23–32.
[24] C. McMahon, A. Lowe, S. Culley, M. Corderoy, R. Crossland, T. Shah, D. Stewart, Waypoint: An integrated search and retrieval system for engineering documents, Journal of Computing and Information Science in Engineering 4 (4) (2004) 329–338.
[25] http://www.mendeley.com.
[26] P. Mohan, V.S. Raghuraman, A. Siromoney, Semantic File Retrieval in File Systems Using Virtual Directories, 13th Annual IEEE International Conference on High Performance Computing (HiPC), Bangalore, India, 2006.
[27] Hung Ba Ngo, C. Bac, F. Silber-Chaussumier, Thang Quyet Le, Towards Ontology-based Semantic File Systems, 2007 IEEE International Conference on Research, Innovation and Vision for the Future, RIVF 2007, 5–9 March 2007 pp. 8–13.
[28] J.M. Park, J.H. Nam, Q.P. Hu, H.W. Suh, Product Ontology Construction from Engineering Documents, International Conference on Smart Manufacturing Application, 2008. ICSMA 2008.
[29] L. Sauermann, A. Bernardi, and A. Dengel, "Overview and Outlook on the Semantic Desktop," Proceeding of Semantic Desktop Workshop, 2005.
[30] A. Shah, D. Schaefer, C. Paredis, Enabling Multi-View Modeling with SysML, International Conference on Product LIfecycle Management, July 6–8, University of Bath, UK, 2009.
[31] A. Craig, N. Soules, Gregory.R. Ganger, 'Connections: Using Context to Enhance File Search', Proceedings of the 20th ACM Symposium on Operating Systems Principles 2005, Brighton, UK, 2005 (pp. 119–132).
[32] J.F. Sowa, Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, MA, 1984.
[33] C. Tziviskou1 and C. Maria Keet, A Meta-Model for Ontologies with ORM2, Springer Berlin/Heidelberg, 2009.
[34] H. Xiao and I.F. Cruz, A multi-ontology approach for personal information management, in: Proceeding of 1st Workshop on The Semantic Desktop, 2005.