# Architectural approach for handling semi-structured data in a user-centered working environment

Andreas Ekelhart, Stefan Fenz, Gernot Goluch, Markus D. Klemen, and Edgar R. Weippl

Secure Business Austria – Security Research – Vienna University of Technology,

Favoritenstrasse 16, A-1040 Vienna, Austria

{aekelhart, sfenz, ggoluch, mklemen, eweippl}@securityresearch.at

WWW: http://www.securityresearch.at

Research paper

**Purpose of this paper**

Today the amount of all kind of digital data (e.g., documents and e-mails), existing on every user's computer, is continuously growing. Users are faced with huge difficulties when it comes to handling the existing data pool and finding specific information respectively. We aim to discover new ways of searching and finding semi-structured data by integrating semantic metadata.

**Design/methodology/approach**

The proposed architecture allows cross border searches spanning various applications and operating system activities (e.g., file access and network traffic) and improves the human working process by offering context specific, automatically generated links that are created using ontologies.

**Findings**

The proposed semantic enrichment of automated gathered data is a useful approach to reflect the human way of thinking which is accomplished by remembering relations rather than keywords or tags. The proposed architecture supports the goals of supporting the human working process by managing and enriching personal data, e.g.

by providing a database model which supports the semantic storage idea through a generic and flexible structure or the modular structure and composition of data collectors.

**Originality/value**

Available programs to manage personal data usually offer searches either via keywords or full text search. Each of these existing search methodologies has its shortcomings and apart from that, people tend to forget names of specific objects. It is often easier to remember the context of a situation in which e.g. a file was created or a website was visited. By proposing our architectural approach for handling semi-structured data we are able to offer sophisticated and more applicable search mechanism regarding the way of human thinking.

**Keywords**: Semantic Desktop, Desktop Search, Semantic Data Storage, Event Correlation

## Introduction

Referring to the study ’How much information 2003?’ (Lyman & Varian 2004) at the University of California Berkeley in the year 2002, the whole world population produced data about 5 Exabytes. The estimate on which the study is based on considers data of the domains print, film, magnetic and optical saved data. Of these 5 EB, 440.606 Terabytes (TB) were accounted for e-mails. If we divide this data amount by the estimated internet users (600 million people), each user will statistically produce about 700 MB e-mail data each year. Considering the whole data production of all internet users, each internet user produces statistically 7 GB data a year. Even though these numbers are based on estimates and do not consider the huge amount of spam, they show the dimension of data an average internet user is faced with. It is assumed that these values are going to rise continuously for the next years and every single user is faced with the problem of a more complex and ever-growing amount of data. The main problem is not the amount of data itself but rather the organization of documents, files, e-mails and so on. More and more documents and communication data of everyday life is saved on data media. For future usage we also need a quick and precise access to this data.

One possibility are file system hierarchies. Files are stored in specific locations which depend on their primary classification or purpose. The shortcomings of this strategy are the attributation and the problem of decentralized data storage. If a file would fit in more than one folder the system starts to fail. Symbolic links as Unix and Linux provide, are a mostly insufficient workaround for large amounts of data. Another problem is the upcoming use of the Internet and decentralized storages. In such cases the information is not locally accessible and thus not integrated in the file system hierarchy.

Keyword indexing is another possibility with a few drawbacks. Allocating keywords, if done manually, is a sumptuous process and users will try to avoid this work. Besides that, choosing suitable keywords over a long period of time is not easy but crucial for feasible search results. Extracting file names to avoid manual work is usually not sufficient for good results, especially considering that file names located in the World Wide Web often do not follow any naming conventions and do not give hints about the content (e.g. *pic3.jpg*). The full text search is a common strategy, well known from Internet search engines. Users no longer have to assign keywords or maintain the storage structure. Major shortcomings of this technology are firstly that it is a priori not guaranteed that the results are in the desired context and secondly that only text-based documents or documents with text-based metadata—such as EXIF for pictures from digital cameras—can be queried. Alternative storage formats, used for video or audio for example, are left out.

All these attempts ignore the context in which data appears, cross references in terms of context relationships are not possible. But for humans, it is often easier to remember the context in which files have been created, opened, changed, etc. Such context thoughts in the mind cannot be mapped on standard query engines; therefore a new storage structure with semantic metadata as an integral part is required. Breaking storage and application borders is an important part to allow cross-reference querying. The product of the final step, which should allow enriching the gathered data with semantic metadata, is the Semantic Desktop which allows new forms of queries: *I remember that I have found a good article in the Internet but I forgot the web address. Furthermore I know that I came across the page during my last meeting with Joe.* This information should be enough to obtain a list of possible web addresses.

## Related Work

It is Vannevar Bush's vision of the *Meme*x (Bush 1945), a device storing all books, records and communications from an individual, under which a lot of projects have been started. The ultimate goal for most of them is to build up an information system managing a human lifetime. *MyLifeBit*s (Gemmell, Bell, Lueder, Drucker & Wong 2002), a Microsoft research project, *Haystac*k (Adar, Kargar & Stein 1999, Quan, Huynh & Karger 2003), *SemanticLif*e (Ahmed, Hoang, Karim, Khusro, Lanzenberger, Latif, Michlmayr, Mustofa, Nguyen, Rauber, Schatten, Nguyen & Tjoa 2004) and Gnowsis (Sauermann 2005) will be discussed in more detail in the following sections.
Sauermann (Sauermann, Bernardi & Dengel 2005) also published on Semantic Desktops. The general idea is to create "semantic glue"—when connecting resources, you need a way to express why the resources are connected. Ideally, this information would be located within the resources. Unfortunately, often one cannot or does not want to tamper with the resources themselves" (Kiesel & Sauermann 2005). Other similar projects are Nepomuk[i], a social semantic desktop; Apogee[ii], a project that tries to build a framework enterprise content management to integrate data from different vendors; and Beagle++[iii], a personal semantic search tool.

### MyLifeBits

Microsoft's *MyLifeBit*s project is an effort to implement a personal digital store with lifetime information. The original *Meme*x concept is extended to handle multimedia data types, including documents, images, sounds and videos. It strives to replace hierarchical storage by enriching data with annotations. The concept of stories goes a little bit further: Stories are described as a layout in time and space, constructed by users and offer background information on other stored data. Such stories can be implemented as slide shows, photo albums, videos or PowerPoint presentations. Besides storing strategies for such huge amounts of data, query mechanisms and the data visualization are important. Different views on the data, including timeline representation, exist. Unlike *Blackma*n this project seems to aim only for offering a desktop search engine. Semantic enrichment based on ontologies is not included.

**Semantic Life**

Like *Blackma*n the *Semanti*c *Lif*e project aims to build up an information storage system including emails, browsed web pages, phone calls, images, contacts, etc. Another similarity is the vision of semantic enrichment based on ontologies. In contrast to *Blackman*, *Semanti*c *Lif*e defines its ultimate goal as building up a Personal Information Management system over human lifetime. Hence it positions itself very close to the original *Meme*x vision, *Blackma*n in contrast is designed to be deployed and customized for specific working environments. This enables the *Blackma*n project to concentrate on solutions which support working activities and not just sophisticated data storage and search. Furthermore a prototype for the *Blackma*n project exists, including data collecting agents for multiple system parts.

**Haystack**

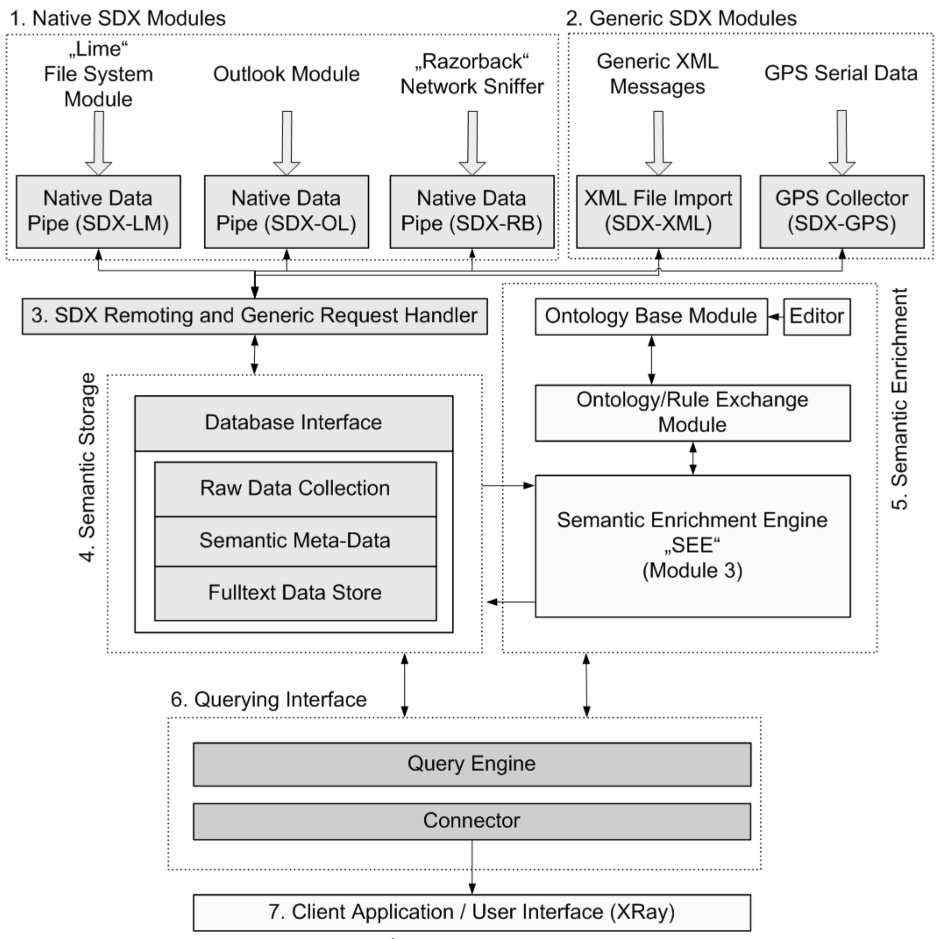
*Haystac*k is an open source personal store, working with annotations. It is designed to support individuals in managing their information, gathered from various sources by breaking application-created barriers. In comparison to *Blackma*n it was designed to run on an individual machine as single user system. Data gets personalized based on system and user-defined ontologies. The visualization process is of high importance and ontology driven. The *Haystac*k Client offers a whole working environment including email, web surfing, etc. and is based on tools like java, RDF and Adenine.

**Gnowsis**

*Gnowsis*, an open source project led by the DFKI department, is referred to as *Semantic Desktop Environment for Information Integration*. As well as *Blackman*, it aims to cross application borders by collecting data from various sources.
It offers so called *Adapter*s which are similar to *Blackma*n collectors. A beta prototype exists which includes *Adapter*s for MS Outlook, Mozilla Firefox, the Filesystem and MP3-ID3 tags. In contrast to *Blackma*n in this approach the idea of the Semantic Web and its possibilities and tools dominates. Their approach is to view the personal computer as a web server and everything stored on it as web resources. Gnowsis makes use of already well developed Semantic Web tools and environments. A local HTTP server is the heart of this solution, allowing users to use their desktop computers like a mini-semantic web. *Adapter*s collect data from applications and transform them to RDF. Gathered metadata is stored in native RDF databases and ontologies, expressed as RDF-S or OWL, are bundled with *Adapter*s to describe the managed data. Similar to *Blackman*, data structures are not changed and existing applications are not replaced but rather extended. Further users have the possibility to link the gathered information manually, automatic generated associations are not supported yet. Possible data exchange over http-based peer-to-peer systems is mentioned but not concretized. In contrast *Blackman*'s multi-user visions constitute an essential part of the concept. Another point of division is the *Blackma*n concept of offering business specific packages instead of a complete personal Semantic Web like data store.

## Architecture

*Blackma*n is a scientific project which integrates different technologies and applications in a single semantic framework. The first data gathering prototypes are implemented using C# (MS .Net Framework 2.0), MS SQL2005 and MS Office 2003. *Blackma*n currently consists of three native data collectors and two generic data collectors as shown in Figure 1.



1. Native SDX Modules
„Lime“ File System Module
Outlook Module
„Razorback“ Network Sniffer
Native Data Pipe (SDX-LM)
Native Data Pipe (SDX-OL)
Native Data Pipe (SDX-RB)
2. Generic SDX Modules
Generic XML Messages
GPS Serial Data
XML File Import (SDX-XML)
GPS Collector (SDX-GPS)
3. SDX Remoting and Generic Request Handler
4. Semantic Storage
Database Interface
Raw Data Collection
Semantic Meta-Data
Fulltext Data Store
5. Semantic Enrichment
Ontology Base Module
Editor
Ontology/Rule Exchange Module
Semantic Enrichment Engine „SEE“ (Module 3)
6. Querying Interface
Query Engine
Connector
7. Client Application / User Interface (XRay)

Figure 1. Architecture - Overview

Using MS .NET Remoting, the *Request Handler* connects the external data sources to the database and in addition notifies the semantic enrichment component in case incoming data requires an adoption of the underlying ontology. Based on the results of the project on dynamic ontologies[iv], the system modifies the ontology semi-automatically when required.

The *Semantic Storage* component stores semi-structured, highly interconnected data. This requires data models which take these characteristics of semantic environments into consideration and still yield a high performance.

The *Semantic Enrichment* module is crucial for the usability of a semantic information management system. This Engine is the main focus of the ongoing research and its capabilities are crucial for the usability and comfort that *Blackma*n offers.

**SDX Modules**

Currently there are three native SDX (*Semanti*c *Dat*a *eXchange*) modules implemented. Their task is to monitor specific parts of the users' operating system:

- file system
- organizer software (currently implemented for *Microsof*t *Outloo*k *200*3 )
- network traffic

Two generic SDX modules provide interfaces to external applications and data sources.

Lime -File System Module: The File System Module is watching user defined file system directories for create, change and deletion events. This capability is realized by watching the '*Recen*t *Docs*' folder Windows XP provides, its corresponding registry entries and event handlers for specific directories.

Outlook Module: The Outlook Module was designed to register change events at the database (.pst file) of *Microsof*t *Outlook*. In a first step it is possible to select Outlook default folders like *Delete*d *Items*, *Sen*t *Mail*, *Calendar*, *Contacts*, *Inbo*x and its sub folders for monitoring.

Razorback -Network Sniffer: The Network Module is watching traffic of user defined network protocols. Basically it is implementing a network sniffer which is able to monitor HTTP, IMAP or ICQ protocol packets and process them on different user defined detail levels.

XML File Import: There are several types of usage scenarios for this specific SDX module. For example in case of connection problems the gathered data is stored in local XML files, based on a XML scheme reflecting the *Blackma*n database scheme for optimized database import processing. The next time a connection to the *Blackma*n server is set up; the locally stored data is transported via the request handler to the data storage. Another purpose is the data import from sources, which are not monitored by existing SDX modules, like a Linux system. The gathered data is once again stored in local XML files and sent to the server via this specific module.

GPS Data Collector: GPS data, reflecting the current geographical position of a specific object, is vital for many of the system's scenarios. Therefore a GPS Module is required which collects position data, by using an attached GPS device and sending data directly to the data storage (Smartphones and Pocket PCs offer a great range of possibilities in this area) or storing them locally in XML files until the next time a connection to the *Blackma*n server is set up. A prototype for *Window*s *Mobil*e *5.*0 is implemented.

**SDX Remoting and Generic Request Handler**

The SDX Remoting and Generic Request Handler is the central interface of the *Blackma*n architecture. It handles all SDX module data requests, processes them and writes the data to the semantic storage. The interface is connected through .NET Remoting channels to all SDX modules.

**Semantic Storage**

The current implementation of the *Blackma*n architecture has a focus on the raw data collection. Shortly described its task is to store data coming from the request handler for which we use a very generic data structure (Weippl, Klemen, Linnert, Fenz, Goluch & Tjoa 2005) (see Figure 2), described in the following sections.

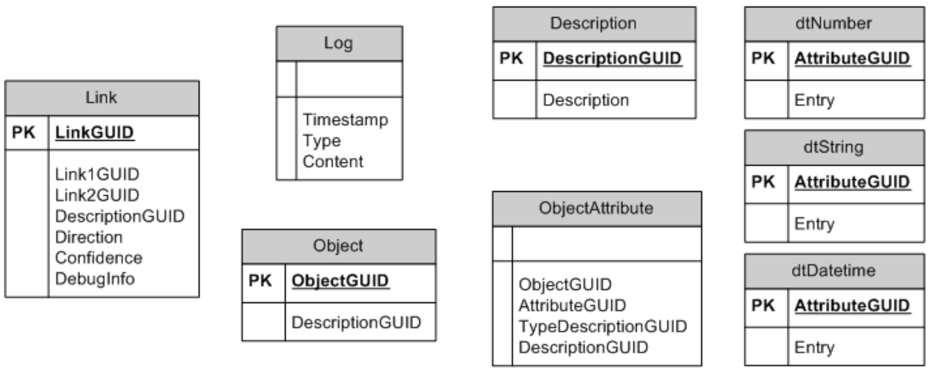


Figure 2. Database model

The Approach: For our architecture a database schema was needed which allowed describing the relationship between any two entities (like in any relational database scheme), but also allowed the modification of semantic relationships easily and without database schema modifications or reprogramming. The traditional relational model has one important drawback when relationships between already stored data objects need to be modified. Typically, in our semantic environment we would have to link many different objects. This requires many n:m relationships, which are usually realized in modern relational database systems by introducing a third table which incorporates a 1:n and 1:m relationship. Links between tables are usually realized using foreign keys. Such a database structure would allow to relate any number of objects (e.g.: documents) with events (e.g. creation of a file) if all tables and all relationships are known a priori. However, a database containing semantic information is highly volatile with regard to its relationships, which can vary often and extensively. This was the main reason we decided not to use a classic relational database scheme, but develop our own flexible data model.

The Model: The central element in our database scheme is the *Lin*k table which connects different objects (including their attributes). Table *Lin*k and *ObjectAttribut*e (see Figure 2) are responsible for connecting datasets. No foreign keys are used to maintain flexibility. As previously described, the drawback of the classical approach is that the Semantic Enrichment Module or SDX Modules cannot easily add new tables and relationships. Now new relationships can easily be added, just by adding a link between objects in the *Lin*k table. Changing an existing 1:n to n:m relationship needs no database scheme changes at all, the same applies to new relationships between two objects which had no prior link. To ensure unique IDs, called GUIDs, within the whole database, hash values (SHA-512) are used to identify datasets. The following itemization provides a short overview on the main components of our database model:

- The main task of the *Lin*k table is to connect different objects, by inserting ObjectGUIDs at *Link1GUI*D and *Link2GUI*D column. *DescriptionGUI*D references the description for the relation, at table Description. Direction is filled with a Boolean value which indicates how we have to read the relation (e.g.: 'File - User' means 'file is used by user', and 'User -File' means 'file is owned by user'). The confidence of a link is described in the corresponding column. Links that are imported by SDX modules have a confidence of 100 percent, but links created automatically by the Semantic Enrichment Module (e.g.: File and Website, based on the timestamps of the two objects) are not always confident by 100 percent. Finding the right value sets for assumptions by the Semantic Enrichment Module, indicating that the link is not 100 percent certain but contains some degree of uncertainty is part of the ongoing research.

- The *Objec*t table stores the ObjectGUID, which identifies the stored object, and *DescriptionGUID*, which references a human readable description of that object. An object may describe an event, a computer, a website, and so on. ObjectGUID is also defined as unique, and as mentioned above it is realized by using hash values (SHA-512).

- The *Descriptio*n table contains human readable information for each *DescriptionGUID*. *DescriptionGUI*D is unique and represents the SHA-512 hash value of Description, so each entry is stored only once and reused every time if needed. The usage of a unique hash value as *DescriptionGUI*D enables a fast lookup at tables. Theoretically a query could save several join statements to table Description, by building the hash value of the searched value, application based.

- The *ObjectAttribut*e table connects attributes from dt* tables to the corresponding objects. *AttributeGUI*D is once again a hash value of the human readable information at dt* and so a query application is able to look for a certain value by comparing the hash value at table *ObjectAttribute*. *Type-DescriptionGUI*D describes the table name, for instance dtNumber, where the *AttributeGUI*D is hosted.

- The actual values we want to store and retrieve at queries are finally stored in the dt* tables. Each *AttributeGUI*D describes the hash value of a specific *Entr*y field.

The following itemization summarizes the main advantages our data model provides:

1. Relationships can be added without schema modifications. This allows to easily perform operations within transactions.

2. Tables and indices can be clustered to improve the processing time of joins

with the central *Lin*k table. In the classical model many n:m relationships exist. Hence it is a priori difficult to assess how to cluster the data optimally.

3. Our approach allows retrieving relationships from the *Lin*k table without accessing the data dictionary. Since data dictionaries are vendor specific, the classical approach requires modifying the application for each database system.

4. Using hash values as unique identifiers allows quick searches and queries.

The semantic metadata sub module is needed as temporal data storage by the semantic enrichment module. The full text data store contains indexed documents to provide a full text search functionality, which can be useful in some specific contexts (e.g.: Project Management Usage, Security Monitoring Usage, etc.). The usage and implementation of the full text data store is similar to existing full text search mechanism. Detailed information about performance issues on the semantic storage could be found at (Weippl et al. 2005).

**Querying Interface**

The preceding section introduced our semantic database model. Since it is built on a relational database scheme, SQL is used to query the database. Apart from the aforementioned advantages (see Chapter *Semanti*c *Storage*) the introduced generic model has one drawback: long and complex SQL statements are required to retrieve the desired data. Due to this level of complexity, end users as well as developers who want to take advantage of the *Blackma*n database model cannot be expected to produce these statements on their own. The acceptance of applications and the data model itself also depends on an existing query language which allows the user to create queries fast and intuitive. To achieve this goal the Link Query Language (LQL) was developed.

The following requirements have been identified:

– Usability

The primary goal of LQL is to allow the users and developers to query the *Blackma*n database model in an easy and intuitive way by hiding the complex underlying SQL statements. Syntactical elements from SQL as well as OQL (ODMG 2003) are integrated, so users who are familiar with classical attempts do not have to learn a completely new language.

– Expandability

Expanding the LQL grammar later on should be possible without major changes to the existing LQL architecture.

– Performance

The translation rules, as well as the database should be optimized to perform well with long and complex SQL statements.

## LQL ARCHITECTURE

In this section the main parts of the Link Query Language architecture (see Figure 3) are explained (detailed information on this topic can be found in (Ekelhart 2005)). The phases inside the LQL Processor are geared to the recommendations in the book 'Compilers' (Aho, Sethi & Ullman 1988) and (Terry 2005).

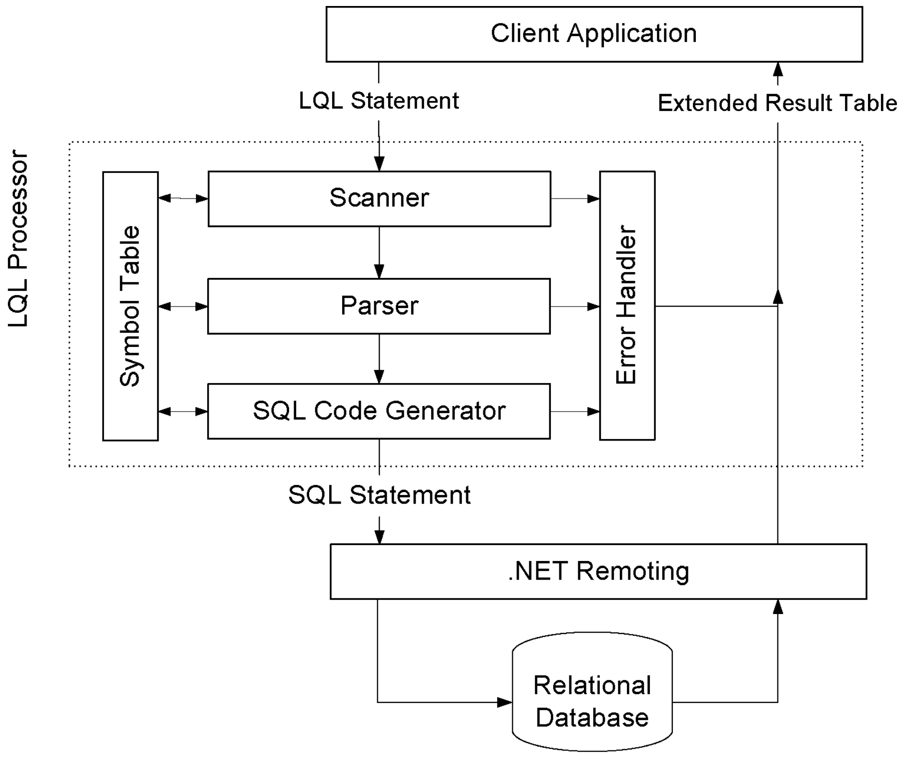


Figure 3. LQL architecture

Client Application: The client application offers functionality to the end users and needs to operate on data from the *Blackma*n data storage. Here *Lin*k *Quer*y *Languag*e statements must be created, dependent on the application and the user's needs. User initialized queries should be supported by graphical statement construction to make the process as intuitive and easy as possible.

LQL Processor: LQL statements are passed to the *LQL Processo*r which is responsible for transforming them into equivalent SQL statements. The LQL syntax is described in the Extended Backus-Naur Form (EBNF) (Wirth 1977). A *Scanne*r splits the LQL string into tokens. After this lexical analysis the parser checks for syntactical correctness of the LQL statement. The parser analyzes the tokens and stores information about the structure in the *Symbo*l *Table*. A compiler generator called 'Coco/R' is used:

Coco/R is a compiler generator, which takes an attributed grammar of a source language and generates a scanner and a parser for this language. The scanner works as a deterministic finite automaton. The parser uses recursive descent. LL(1) conflicts can be resolved by a multi-symbol look ahead or by semantic checks. Thus the class of accepted grammars is LL(k) for an arbitrary k (Terry 2005).

Occurring errors are collected and managed by the *Erro*r *Handler*. As a final transformation step the *Cod*e *Generato*r evaluates the gathered information and creates the SQL statement based on transformation rules. Results from a relational database are tables, having rows and columns as basic navigation scheme, which do not reflect the object oriented *Blackma*n data structure. A customized data table was created which allows more intuitive result interpretation by reflecting an object structure.

The LQL syntax is described in the Extended Backus-Naur Form. *LinkQueryLanguage* is the start symbol of the grammar. To keep it simple we only show the *SelectCommand* definition. Other commands like Insert, Delete, etc. can be added at this point.

```
LinkQueryLanguage
= ModifyData.

ModifyData
= SelectCommand.

SelectCommand
= "SELECT"
   SeletectedAttributes
   "FROM"
   SelectedObjects
   [ "WHERE" Condition    ]
   [ "ORDER" "BY" SortCriteria ].

SelectedAttributes
= SelectedAttribute { ',' SelectedAttribute }.

SelectedAttribute
= Object "." Attribute.

SelectedObjects
= SelectedObjects [ "AS" Ident ]
  { ( '<' Ident '>' | ',' ) SelectedObject { [ "AS" Ident ] }.

Condition
= Logical Term
  { "OR" Logical Term }.

LogicalTerm
```

```
= LogicalFactor
  { "AND" LogicalFactor}.

LogicalFactor
= ( SelectedAttribute
    ( ( ( RelationalOperation | EqualityOperation )
         ( SelectedAttribute | number | string ) )
         | ( "LIKE" string )

    )
       | '(' Condition ')'
    ).

RelationalOperation
= ( | "<" | ">" | "<=" | ">=").

EqualityOperation
= ( "=" | "!=").

SortCriteria
= SortCriterion
  { "," SortCriterion }

SortCriterion
= SelectedAttribute [ ( "ASC" | "DESC" )].
```

A sample query for emails related to events that occurred after March 30, 2007 is shown in the following:

*SELECT mail.subject, mail.from, event.datetime FROM mail <link> event WHERE mail.subject LIKE '\%IMPORTANT\%' AND event.datetime > '2007-03-30' ORDER BY event.datetime DESC, mail.subject*

Remoting: Created SQL statements must be transported to the *Blackma*n database, and results must be returned to the caller. A .NET Remoting channel is used to access the database; the same mechanism is utilized by the data collectors to write gathered data to the *Blackma*n storage (see Section *Semanti*c *Storage*).

Relational Database: The underlying database may be virtual any relational database system such as Oracle, PostgreSQL, MySQL, or SQL Server. Depending on the client application's purpose and requirements a decision on an appropriate database can be made. Our prototype is based on MS SQL Server 2005.

## Semantic Enrichment

The architecture of the Semantic Enrichment Module is shown in Figure 4.

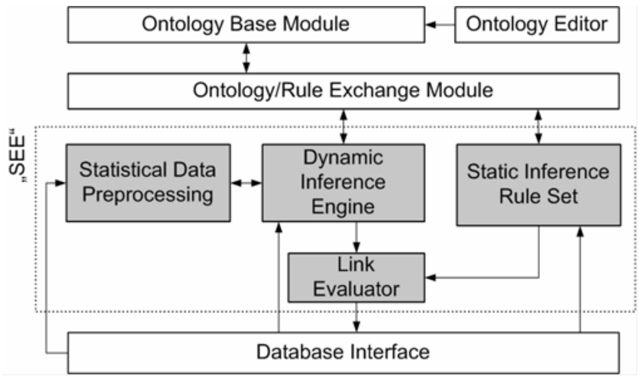


Figure 4. Semantic Enrichment Engine (SEE)

We start off with an existing light-weight ontology administered by the ontology base module. This ontology describes the relationships of objects known a priori to the system (e.g. calendar entries, contacts, etc.). A *Static Inference Rule Set* is used to create links between objects such as calendar entries and photographs. A static rule is, for example, 'If a photo is taken at the time of a scheduled meeting, the photo is related to this meeting'. *Statistical Data Pre-Processing* is used to analyze and characterize the typical usage scenarios of the user. It is necessary to know how many emails the user usually receives or sends, how many photographs he makes, etc. to subsequently detect anomalies and periods of increased activity. The *Dynamic Inference Engine* uses statistical data and modification events in the database to create new links with varying degrees of reliability. The *Link Evaluator* module checks whether links already exist or any negative rule applies.

User specific requirements to the *Blackman* system are going to be easily fulfilled by using the SEE. It's not important if the user is a single customer, who wants his own personal organizer system, which provides him semantic search on his documents, e-mails, etc. and client applications to support and organize his 'digital life', or a company with the need for a project management, workflow visualization or security monitoring tool.

## Client Application

We retrieve a lot of data directly from applications such as MS Outlook and MS Internet Explorer via the .NET Framework and therefore we collect more contextual information compared to similar projects that rely on monitoring at a generic operating system level. To navigate through the search results, users need a comfortable and innovative GUI that integrates seamlessly into existing applications. *Blackma*n offers three main GUI modules:

1. A tree view-based context specific search fully integrates into the Explorer property menu. Right-clicking any object permits to display a structured list of related objects. Adding icons into *MS Outloo*k *200*3—similar to the Smart Tags offered in *Visua*l *Studi*o *200*5 or *Microsof*t *Offic*e *200*3—allows users to see e.g. that photos are available that have been taken at the time of an appointment by some person who is listed as attending in this Outlook appointment.

2. The main browsing form is timeline-based, offering various dimensions and allowing easy change of dimensions (see Figure 5).

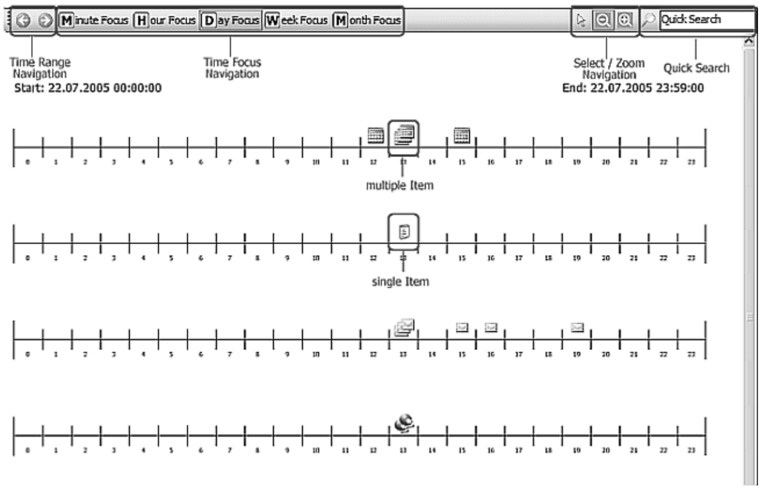


Figure 5. Timeline-based browsing

3. A passive background information form, based in Windows messaging service suggests direct links to data, which is repeatedly and strongly associated with the data of active tasks. For example, opening a word file which has often been opened with an Excel spreadsheet will trigger the software offering to open this specific spreadsheet as well. These suggestions are shortly displayed without interrupting the normal workflow—comparable to MS Outlook's 'New Mail' message (compare Figure 6).

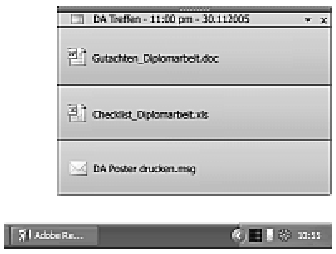

DA Treffen - 11:00 pm - 30.11.2005
Gutachten_Diplomarbeit.doc
Checklist_Diplomarbeit.xls
DA Poster drucken.msg
Adobe Re...

Figure 6. Sidebar visualization

## Conclusion

Evermore data storage enables people to save virtually their whole life digitally in various file formats or databases, e.g.: photos, videos, e-mail, address databases etc.
Available personal programs to store and manage these files usually offer searches either via file system hierarchies, keywords or full text search. Each of these existing search methodologies has its specific shortcomings and apart from that, people tend to forget names of specific objects. It is often easier to remember the context of a situation in which a specific file was created, modified or viewed, especially with regard to a timeline (*I remember I just got an e-mail from Barry when I was working on that document*).

Semantic enrichment of automated gathered data is a useful approach to reflect this human way of thinking which is accomplished by remembering relations rather than keywords or tags. The aims of the *Blackma*n project are to support and improve the human working process by managing and enriching the huge amounts of semi-structured data which users have to deal with every day. The proposed architecture supports these goals, e.g. by providing a database model which supports the semantic storage idea through a generic and flexible structure or the modular structure and composition of data collectors.

## Acknowledgements

This work was performed at the Secure Business Austria Research Center funded by the Federal Ministry of Economics and Labor of the Republic of Austria (BMWA) and the federal province of Vienna.

## References

Adar, E., Kargar, D. & Stein, L. A. (1999), Haystack: per-user information environments, *i*n ‘CIKM ’99: Proceedings of the eighth international conference on Information and knowledge management’, ACM Press, New York, NY, USA, pp. 413– 422.

Ahmed, M., Hoang, H. H., Karim, M. S., Khusro, S., Lanzenberger, M., Latif, K., Michlmayr, E., Mustofa, K., Nguyen, H. T., Rauber, A., Schatten, A., Nguyen, T. M. & Tjoa, A. M. (2004), ’semanticlife’ -a framework for managing information of a human lifetime., *i*n ‘iiWAS’. Aho, A. V., Sethi, R. & Ullman, J. D. (1988), *Compilerba*u -*Tei*l *1*, Addison-Wesley.

Bush, V. (1945), ‘As we may think’, *Th*e *Atlanti*c *Monthl*y 176(1), 101–108.

Ekelhart, A. (2005), The blackman project: Collecting and querying semi-structured data for the ’semantic desktop’, Masterthesis, University of Technology Vienna, Vienna.

Gemmell, J., Bell, G., Lueder, R., Drucker, S. & Wong, C. (2002), Mylifebits: fulfilling the memex vision, *i*n ‘MULTIMEDIA ’02: Proceedings of the tenth ACM international conference on Multimedia’, ACM Press, New York, NY, USA, pp. 235–238.

Kiesel, M. & Sauermann, L. (2005), ‘Towards semantic desktop wikis’, *UPGRAD*E *specia*l *issu*e *o*n *Th*e *Semanti*c *We*b 6(6), 30–31.

Lyman, P. & Varian, H. (2004), ‘How much information 2003?’, Available http://www.sims.berkeley.edu/research/ projects/how-much-info-2003/.

ODMG (2003), *ODM*G *OQ*L *Use*r *Manual*.

Quan, D., Huynh, D. & Karger, D. (2003), Haystack: A platform for authoring end user semantic web applications, *i*n ‘International Semantic Web Conference’, pp. 738– 753.

Sauermann, L. (2005), The gnowsis semantic desktop for information integration, *i*n ‘Proceedings of the 3rd Conference Professional Knowledge Management (WM 2005)’, Springer.

Sauermann, L., Bernardi, A. & Dengel, A. (2005), Overview and outlook on the semantic desktop, *i*n S. Decker, J. Park, D. Quan & L. Sauermann, eds, 'Proceedings of the 1st Workshop on The Semantic Desktop. 4th International Semantic Web Conference', Galway, Ireland.

Terry, P. (2005), 'The compiler generator coco/r', Available http://www.scifac.ru.ac.za/resourcekit/pdf/CocoManual.pdf.

Weippl, E., Klemen, M., Linnert, M., Fenz, S., Goluch, G. & Tjoa, A. (2005), Semantic storage: A report on performance and flexibility, in 'Proceedings of DEXA, Lecture Notes in Computer Science (LNCS)'.

Wirth, N. (1977), 'What can we do about the unnecessary diversity of notation for syntactic definitions?', Communications of the ACM 20(11), 822–823.

[i] http://nepomuk.semanticdesktop.org/xwiki/

[ii] http://www.eclipse.org/proposals/apogee/

[iii] http://beagle.kbs.uni-hannover.de/

[iv] http://dynamont.factlink.net/