DeepaMehta — Another Computer is Possible

Jörg Richter, CTO, DeepaMehta Company
jri@deepamehta.de
www.deepamehta.de

Jurij Poelchau, fx-Institute
poelchau@fhochx.de
www.fhochx.de

August 10, 2007

Introduction **2**
- Machine Dreams 2
- The Potential of Ontologies 2
- The Trouble with Computers and their solutions 3
- The Vision of an Integrated Work Environment 6

The DeepaMehta Platform **7**
- Topic Maps Frontend 8
- System Architecture 10
- Data Model 11
- The Application Framework 13
- The DeepaMehta Unified Process 16

Customer Solution Examples **17**
- Kiezatlas, a geographical CMS 17
- amina Knowledge Platform 18

Outlook **19**
- Extended support for semantic technologies 19
- Extended collaboration features 20
- Extended standard applications 20
- Architectural Concerns 20

Related Work **21**
- Semantic Desktop 21
- Graph-based Information Visualization 21

Future Research Directions **23**
- Reanimating Soft Cybernetics 23
- Examining underlying images 24
- Research Issues 25

References **26**

Additional Reading **27**

About the Authors **28**

Introduction

Machine Dreams

A crucial experience during my time at university — computer science (with focus on AI) and linguistics — was the documentary “Maschinenträume” (1988) by Peter Krieg. It features the long-term AI project “Cyc”, in which Doug Lenat and his team try to represent common sense knowledge in a computer. When Cyc started, in 1984, it was already known that many AI projects failed due to the machine’s lack of common sense knowledge. Common sense knowledge includes e.g. that two things can’t be in the same place at the same time, or that people die, or what happens at a children's birthday party. During the night, while the researchers are sleeping, Cyc tries to create new knowledge from its programmed facts and rules. One morning the researchers were surprised by one of Cyc’s new findings: “Most people are famous”. Well, this was simply a result of the researchers having entered, beside themselves, only celebrities like e.g. Einstein, Gandhi, and the US presidents. The machine-dreaming researchers, however, were in no way despondent about this obviously wrong finding, because they figured they would only have to enter the rest of the population, too. The underlying principle behind this thought is that it is possible to model the whole world in form of ontologies. The *meaning* of the world can be captured in its entirety in the computer. From that moment the computer can know everything that humans know and can produce unlimited new insights. At the end of the film Peter Krieg nevertheless asks: “If one day the knowledge of the whole world is represented in a machine, what can humans *do* with it, the machine having never seen the world.”

Jörg Richter

The Potential of Ontologies

In order to create computer applications that suit the human way of thinking and working, one needs to think about ontologies. There are ontologies for e.g. molecular biologists, for news editors, for tourist agents, or for connoisseurs of wine. By developing an ontology people agree on the meaning (*semantics*) of certain computer codes. Every community of interest is free to create their own ontology or to co-work with like-minded people on the development of a shared ontology. This development process — the ontogenesis — is at least as important as the resulting ontology itself.

An ontology of the entire world will never exist, because the meaning of terms depends on their usage context. A wine merchant probably has different priorities than a wine connoisseur which will be reflected in their ontologies. When developing an ontology, local or global communities of interest focus on that parts of the reality that are relevant to their intended usage context. The authors see the prior potential of ontologies not in creating an one-to-one representation of reality, in order to enable automatic reasoning, but in “learn[ing] to think together by building shared structures of meaning” (Murray, 2003, p. 11). That is not in automation, but in collaboration.

We can accept that invitation not only to think about application domains and usage contexts but also to re-think the underlying concepts of the computer itself, especially the human-computer-interface. It is interesting to bring to mind that ontologies are firmly molded into every software system — long before the semantic web effort. These ontologies are made up of the concepts and their relations defined by the software architect, and according to which the user is supposed to think when using the software.

So, the concepts of e.g. “presentation”, “slide” and “masterslide” and their relations are as firmly molded into the software “Powerpoint” as the concepts “application”, “window”, “file”, and “folder” in the software “user interface”.

It would be significantly easier for the computer user if he/she could work semantically not only in a single application but if the entire computer were designed as a semantic work environment. In such an environment the user would no longer work in applications, or store files in folders, but would be confronted with the terms of his/her way of thinking and working *directly*. At the lowest level such an environment could provide the concepts “more or less structured information unit”, “relation between information units”, “view of a information context”, “shared workspace”, and “private sphere”. At the next higher level more specific concepts of yet general usage like “contact info”, “appointment”, “task”, and “project” could be provided. And at the highest level the user would find the concepts of its individual work and leisure domains, e.g. a molecular biologist and movie fan would work with the concepts “chromosome”, “gene”, “protein”, as well as “movie”, “actor”, and “cinema”.

The Trouble with Computers and their solutions

When learning how to use a computer nowadays, one understands quickly: for every purpose one needs a specific application. To write a text one uses a word processor. Emails are dealt with using an email application. To surf the internet one needs a web browser and one’s contact information is stored in the address book. Launching an application causes a window to open, showing one the application controls as well as the actual content (texts, emails, web pages, contact details). When writing a text, one needs to save it into a file. This file is located in a certain folder on the hard disc. When one wants to access one’s text later on, one has to use another application: a file browser to locate and open the respective file. Within the first weeks of learning every computer user experiences the loss of his/her text (image, etc.) — either because he/she forgot to save it or because he/she cannot find it anymore.

Working with applications, files, and folders and the importance of saving is surely not difficult to learn, and even may — as soon as one has internalized the logic of the computer — seem reasonable. However, if we turn away from the machine logic to human work situations it becomes apparent how unnatural and inefficient this mode of operation is.

A typical work situation involves a number of heterogeneous content objects. If we take, for example, the production of this book, “SWE”: there are about 20 articles addressing different topics. Each article is written collaboratively by several authors. For each author there are several contact details (email address, homepage, telephone number). Each article refers to a number of further information resources (projects, websites, other articles). During the review phase authors review each others’ articles leading to further article revisions. Eventually the editors assemble the accepted articles into the final book ready for the print shop.

By means of this work scenario two fundamental flaws of the computer and the resulting problems can be made clear:

1) Missing Relations

The meaningful relations between information resources are not represented in the computer. Thus, they neither can be displayed on-screen nor be exploited for navigation.

There is no relation between e.g. the author of a Word document and its entry in the address book application. The user has to switch to the address book and search for the contact details manually. Even within a single application, relations are not easy to represent: if an article, as well as its reviews, are created with e.g. Microsoft Word, it is not possible to represent the relations between these files within Word, but only indirectly in the file system structure, if at all.

Because the relations are not represented in the computer they can not be shared with other users. This makes collaboration inefficient, because every co-worker has to reconstruct the relations individually. To represent the relations between e.g. the articles and their reviews one could store every article in its own folder and its reviews in respective sub folders. If a review is sent to a co-worker as a mail attachment, the relations to the main article gets lost and the receiver has to reconstruct it in his/her own way. This additional effort increases with the number of co-workers, thus a lack of synchronization and various misunderstandings between the co-workers are programmed.

The problem of missing relations can be solved by releasing the contents (texts, emails, web pages, contact details) from their applications and storing them in a corporate memory using a neutral data format. Within the corporate memory, arbitrary content objects can be set into relation, and external resources can be integrated by storing reference objects in the corporate memory. The problem of inefficient collaboration can be solved by letting all co-workers operate on a central (server side) corporate memory instead of their local hard discs.

For both problems the suggested solutions are already in use through specific applications (e.g. Document Management, Content Management, Personal Information Management, Workflow Management, Groupware, and Wikis), but to the knowledge of the authors, there is no computer platform that operates system-wide according to this paradigm.

2) Missing Work Context

The greatest problem which makes using a computer so cumbersome is that the user's work context is not represented on-screen. This applies especially to the knowledge worker who deals with a number of applications and windows simultaneously. In the work context of "SWE Book", for example, a number of heterogeneous content objects (texts, emails, web pages, contact details) are involved. A book author will probably use the concept of "folders" to set up his/her computer as follows: on the hard disc there is a folder "SWE Book" to hold the article and further assets. In the web browser there is a folder "SWE Book" to hold the bookmarks of the project's website and other websites. In the email application there is a folder "SWE Book" to hold the mails of the book editors and other authors. And in the address book application there is a folder "SWE Book" to hold the contact details of the project's members. Conclusion: in the user's mind there is *one* project, "SWE Book", whereas in the computer his/her project is chopped into pieces and fragmented to at least four different work environments.

The user has to switch between several applications on a regular basis. Every switch changes the display as well the usage rules abruptly. What is displayed in one moment has vanished (resp. is obscured) in the next, the usage rules that are applicable in one moment are not so in the next. The user can, of course, recover his/her "SWE Book" folder in each application, but probably along with another 100 folders which are not related to the current work context in any way.

If the content objects belonging to the current work situation — and only these — were visible together on-screen, the cognitive load of the user would be reduced significantly.

This would free the user's mind to let him/her concentrate on his/her work instead of on the workings of the machine.

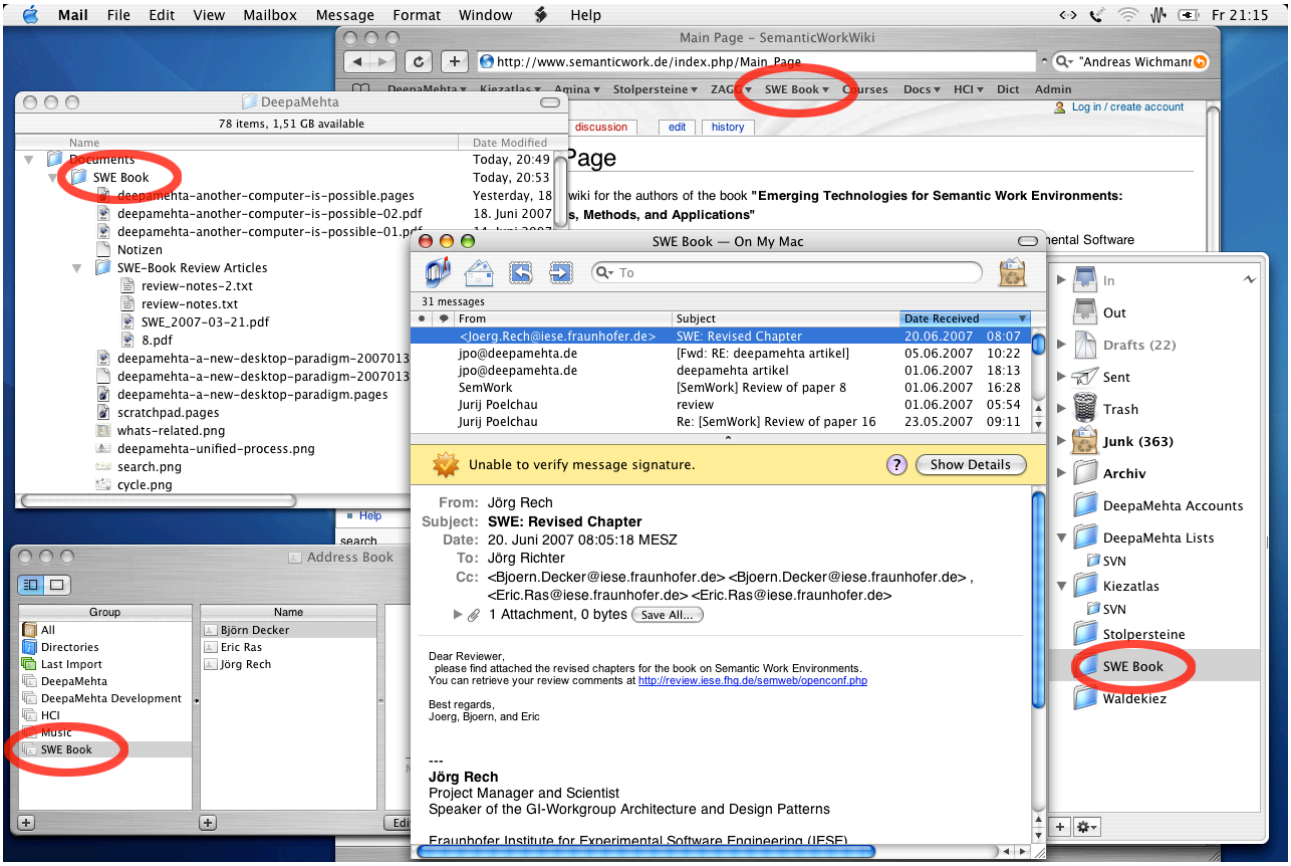

Fig. 1: *Current window-based user interface with 4 open applications: web browser, file browser, email client, and address book. Problem 1: the meaningful relations are not displayed. Information belonging to one project (here: "SWE Book") are fragmented to several work environments. Problem 2: the user's work context is not displayed. The user must switch between applications frequently and has to cope with permanent context changes.*

Within the scope of the current user interface paradigm — *every application opens another window and the application's content objects are bound to that window* — the problem of the missing work context can not be solved. The kind of navigation that results from that paradigm is called here "Go-To Navigation". To look up e.g. an email address while being in a web browser the user must *go-to* that email address (that is switching to the address book application) and cope with a complete context switch. This situation can not be abolished by the development of ever further applications because this only leads to a further fragmented user experience.

The problem of the missing work context can only be solved by establishing a new user interface paradigm in combination with a new application model. This new interface paradigm can realize a kind of navigation that is called here "Bring-To-Me Navigation". To look up e.g. an email address while reading a web page, the email address is *brought* to the user, that is, it is searched for and displayed within the current work context. The on-screen work context remains stable. Heterogeneous content objects (texts, emails, web pages, contact details) and their meaningful relations (problem 1) are displayed within the same window. The aim of Bring-To-Me Navigation is to let the interface itself disappear, and let the display become a "visual cache" (Canfield Smith et al., 1982) of the user's state of mind.

In Richter et al. (2005) the terms “Stable Views” and “Constructive Browsing” are used to describe this new interface paradigm, but here the term “Bring-To-Me Navigation” is used to express that information is brought to a single window and to contrast it to the traditional Go-To Navigation.

In order for the content objects from different applications to share the same display context, the application logic must be separated from the display. The application as such will no longer be visible to the user, but only the content objects. Because there is no application-specific user interface anymore, the content objects must provide the application-specific operations themselves, e.g. an email object must provide a “send” operation to the user. This requires a new application model. The new application model will be realized as an application framework to allow application developers to bind operations to content objects and to interfere with the shared display context.

To our knowledge there is no computer platform that offers stable on-screen work contexts system-wide.

The Vision of an Integrated Work Environment

The vision of the DeepaMehta platform is to provide the knowledge worker with an integrated environment that supports his/her work, thought, and collaboration process. The DeepaMehta platform replaces the traditional computer desktop by a semantic desktop that “push[es] the user interface farther out from the computer itself, deeper into the user’s work environment” (Grudin, 1990).

DeepaMehta envisions:

- Removing machine concepts (applications, document formats, ...) from the user interface and thus from the user’s mind. Confronting the user only with the concepts of his/her daily work and thought. Allowing the user to fully control the display in order to represent his/her work context on-screen. Proving that "a well-designed computer system can actually improve the quality of your thinking" (Canfield Smith et al., 1982)
- Breaking down the “application prison” (Nelson, 2001) and freeing the data from their applications in order to let the user navigate associatively across application-borders. Allowing the user to re-use information objects in different work contexts.
- Designing a user interface that allows the user to apply his/her natural faculties like his/her sense of orientation: “I’m located at a certain place within a stable environment”. Designing a user interface that accommodates the individual learning process.
- Designing a software architecture which allows software developers to contribute domain-specific application logic without confronting the user with applications.
- Supporting collaboration by allowing workgroups to operate on a shared information repository and to build shared structures of meaning. At the same time: providing the user with protected spaces to encourage creativity.
- Designing a work environment that supports the whole process, from a) content creation, to b) enrich content by structure, and c) enrich structured content by application logic. Designing a work environment that allows constant changes at every stage.
- Delivering computers that boot directly into DeepaMehta as the standard user interface and work environment.

The developers of DeepaMehta strive to identify generic operations that are common to all applications, e.g. editing, searching, displaying, navigating, and to build them right into the platform. Domain-specific operations may be implemented in the scope of the DeepaMehta application framework. DeepaMehta applications are running at the server side and are not in the sight of the user.

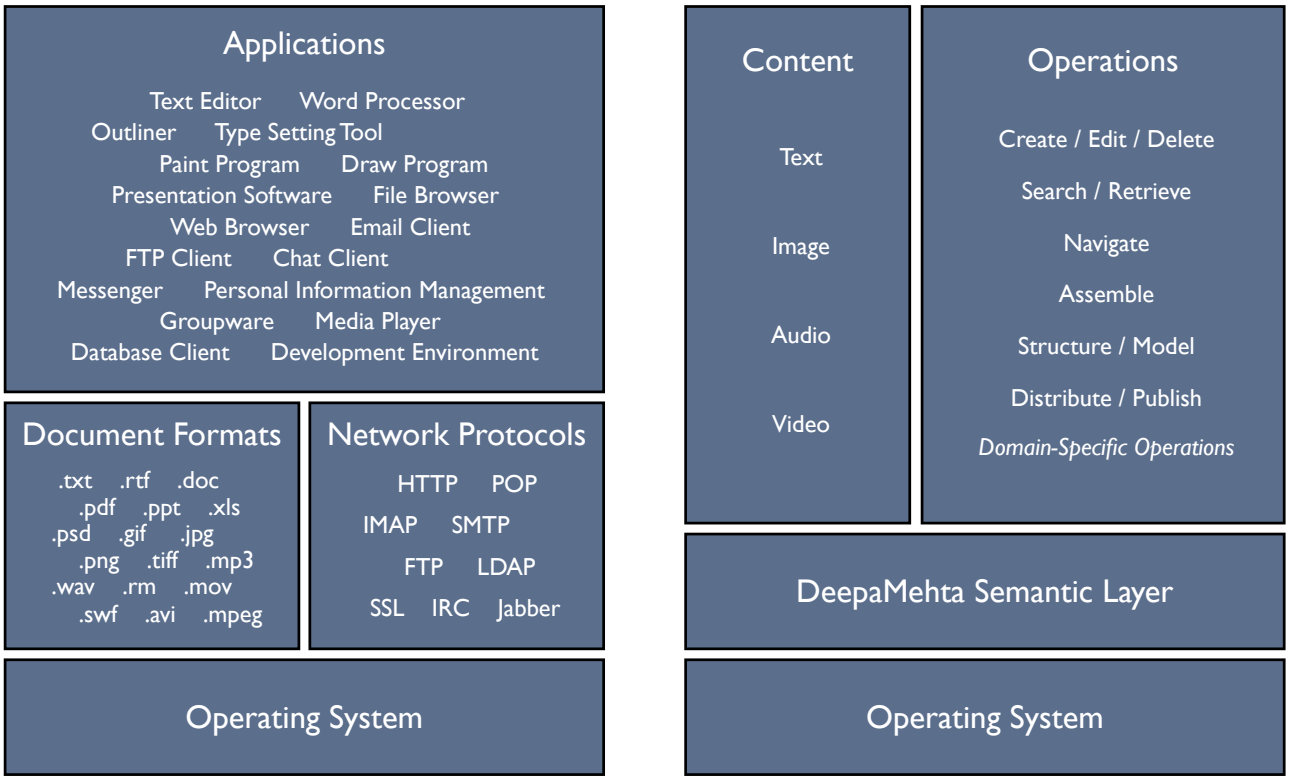

Fig. 2: *Old world — New world: The traditional computer user is bothered with operating systems, and applications which are built around document formats and network protocols (left side). In contrast, the DeepaMehta user is confronted only with content, and performs operations on the content directly — all on a semantic foundation (right side).*

The DeepaMehta Platform

The DeepaMehta software platform for collaboration and knowledge management is a comprehensive integrative concept that removes existing barriers that make current computer usage so cumbersome. DeepaMehta combines concepts and research findings in the fields of software engineering, information visualization, human computer interaction, semantic web, and creativity techniques in an innovative manner. DeepaMehta has the potential to be a great integrator:

- Mind Maps / Concept Maps

  The DeepaMehta semantic desktop combines the cognitive virtues of mind maps with the computational virtues of concept maps resp. semantic networks.

- Visualization / Workspace

  In DeepaMehta there is no separation between the graphic visualization of content structures and the actual work environment. All kinds of content is created and edited in-place. There is no separation between file-level and application-level.

- Brain Storming / Structuring / Processing

  DeepaMehta supports the whole information handling process: from creating information in a brainstorming-mode, to structure information and building models, and to processing information by implemented logic — all is performed in one environment.

- File Level / Application Level

  DeepaMehta steps away from the file-application-dichotomy and establishes a new paradigm of *content* and *operations*. Typed content objects provide the manipulative operations themselves. Generic operations are provided by the DeepaMehta platform.

- Network / Local Machine

  DeepaMehta removes the barrier between the local desktop machine and the network by establishing a uniform user interface for personal content and shared content. Working with local content and remote content results in no different user experience.

- Topic Maps / RDF

  The DeepaMehta data model is inspired by ISO 13250 Topic Maps but has extensions to incorporate RDF-like features. DeepaMehta has the potential to marry the ISO Topic Maps standard and the W3C RDF standard.

Topic Maps Frontend

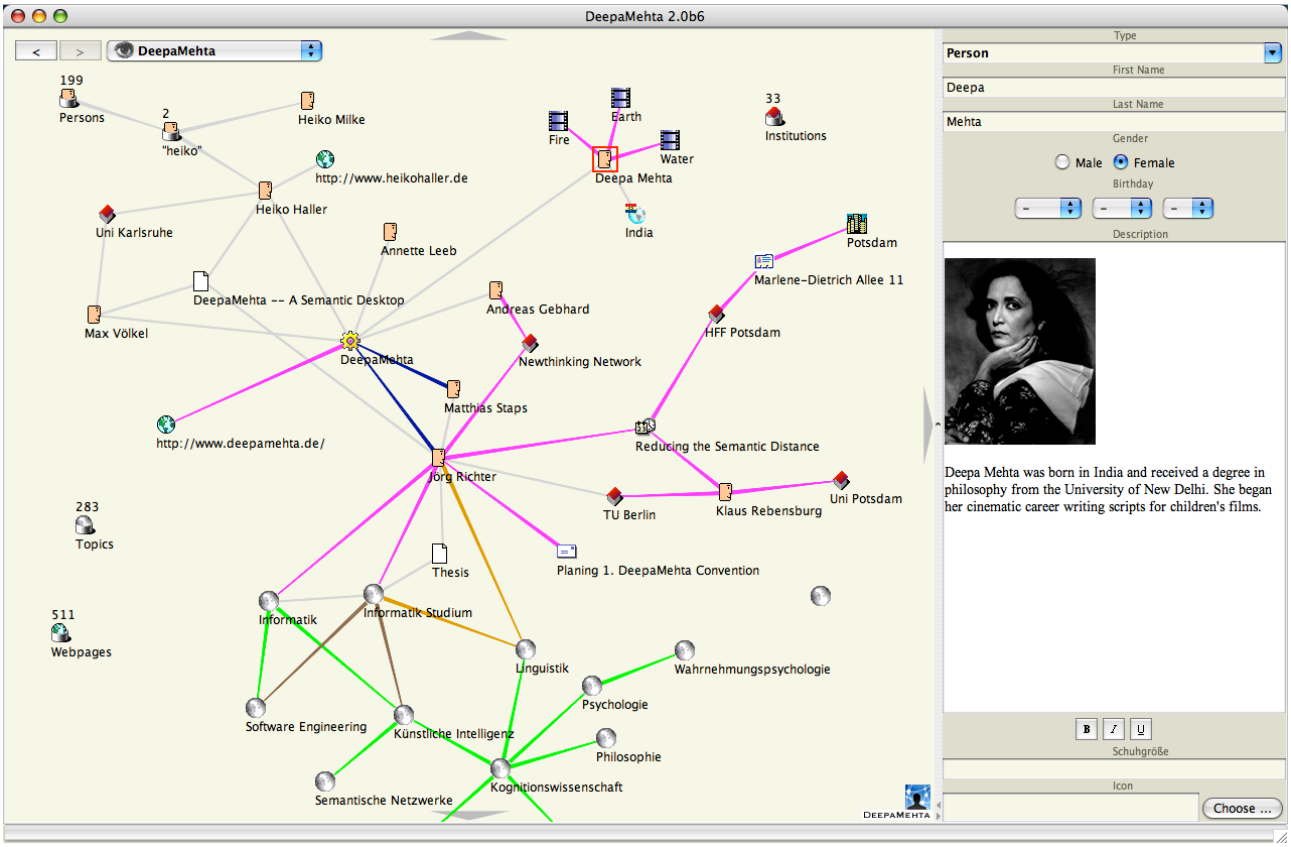

Fig. 3: *The "DeepaMehta" networked semantic desktop. The left side displays information objects of different kinds and origins as well as their meaningful relations as a topic map that represents the current work context of the user. The right side displays detail information about the selected topic, e.g. an email or a webpage. The information objects are edited and manipulated in-place, e.g. emails are written and send straight from the desktop.*

The DeepaMehta desktop is split into two areas. At the left hand side a topic map consisting of typed topics and typed associations is displayed. On the right hand side, detail information about the currently selected topic, association, or the topic map itself, is shown. This panel is called the *Property Panel*. If, for example, an email topic is selected, the email (with its “From”, “To”, “Subject” and “Text” properties) is shown in the property panel. If a webpage topic is selected, the page (with its “URL”, “Title” and “Content” properties) is rendered in the property panel. The user performs basic text editing and image-manipulation operations directly in the property panel, without the use of external applications.

To perform operations on a topic, an association, or the topic map itself, the user utilizes context menus. Generic operations such as “Hide”, “Retype”, “Delete” or “What’s Related?” and “Google Search” are provided by every topic. Depending on their type, topics may provide more specific operations, e.g. an email topic provides a “Send” operation, and a webpage topic provides a “Reload” operation. Topics and associations can be created manually or programatically. Topics and associations can result from a variety of sources such as mailservers, webservers, databases, and web-services. Associations can be created between any two topics, regardless of their origin.

Topic maps are not like documents containing contents, but always personal views to contents existing elsewhere. All topics and associations are stored in a server-side repository, called the *Corporate Memory*. The corporate memory is accessible to various users and workgroups simultaneously, exposed to an access control mechanism of course. A topic map is an individual view to extracts of the corporate memory, that are relevant in a particular working context. It’s up to the user to decide which topics and associations to retrieve, and where to show them. Topic maps are of unlimited size and are moved by mouse dragging. Topic maps that serve as common views within workgroups are published into shared workspaces.

Fig. 4: *The context menu of the person topic “Douglas C. Engelbart”. By the means of the “What’s Related?” command associated topics are retrieved from the corporate memory and displayed in the current topic map. Here, the submenu reveals Douglas C. Engelbart is associated with 2 other persons, 1 computer, 2 articles and so on.*

A query to the corporate memory is visualized graphically and the result set is displayed as a topic of the form of a ton. In order to avoid cognitive overload, DeepaMehta doesn’t reveal a search result at once if it comprises more than 7 topics (Miller, 1956). If the result does not comprise more than 150 entries, single topics may be revealed via a popup menu. Large result sets may be reduced by applying a narrower filter. A topic query may involve the topic name, topic type, topic properties and associations the topic is involved in. The result may be sorted. Result sets in DeepaMehta can be regarded as “smart folders”, i.e. the user can re-trigger the query that underlies a

ton by double-clicking it. A ton may represent the query, e.g. “All emails I’ve received from Jurij in the last 30 days”.

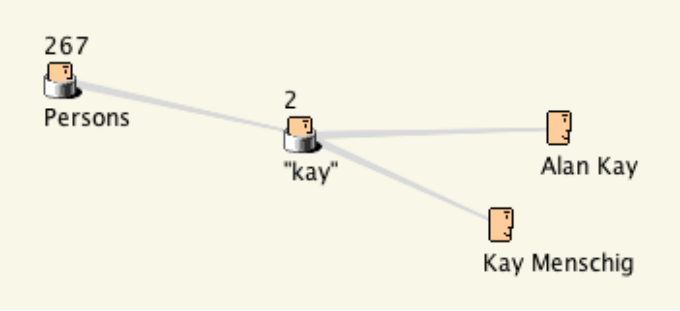

Fig 5: *In DeepaMehta large amounts of data are displayed as a ton. A ton represents a query to the corporate memory and the result set at the same time. In this example an unspecific person search results in 267 topics — too much to reveal at once. Applying a filter “kay” yields to a result set small enough to be revealed immediately.*

The user interface is fully personalized. After logging in the user finds him/herself directly on the DeepaMehta desktop, exactly as it was left in the previous session.

The DeepaMehta Platform is based on ISO 13250 Topic Maps. Topic maps consist of topics and associations, both are typed. Types are Topics too. The user can define new types by deriving them from existing types, create additional properties, and defining relations to other types. New types can be used immediately, also together in shared workspaces. Every topic can be retyped at any time without losing content. All this provides the basis for supporting the dynamic collaborative process of building new ontologies.

System Architecture

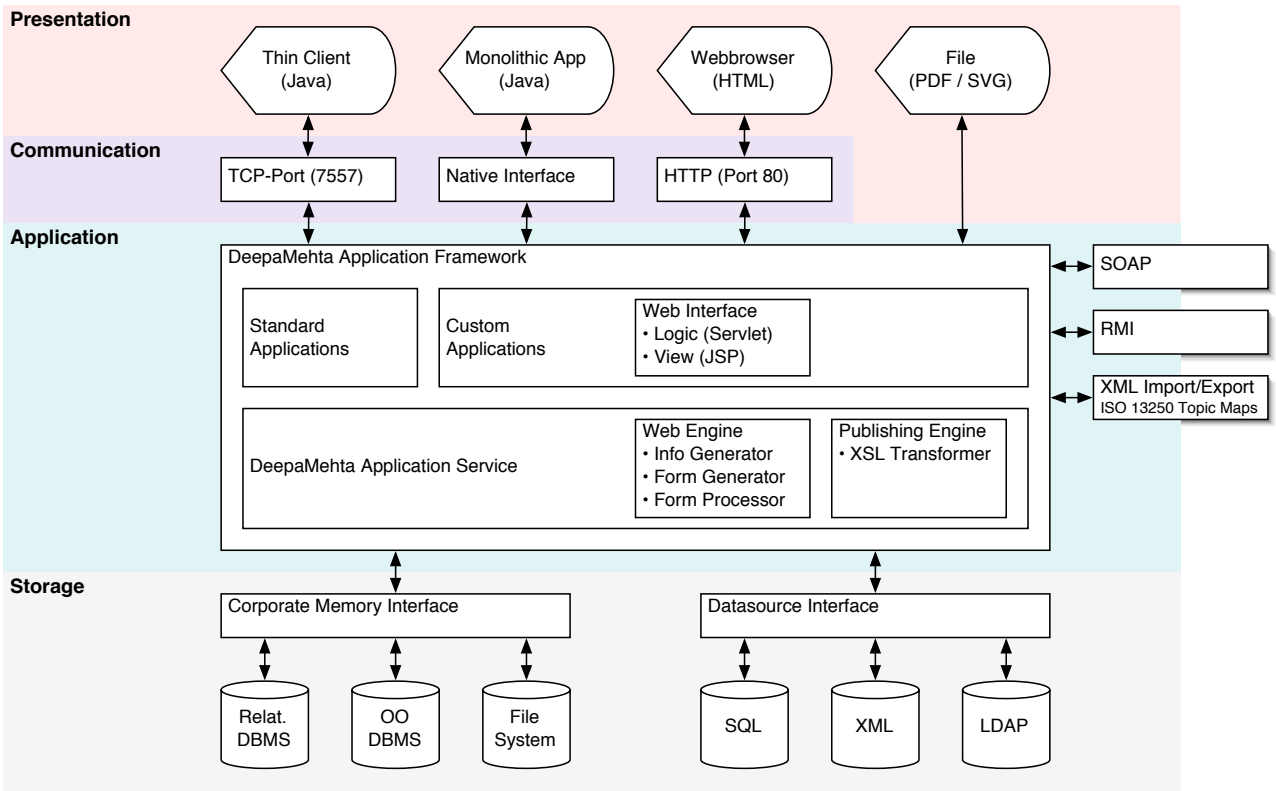

Fig. 6: *DeepaMehta is a multi-layered distributed software architecture. The heart of the DeepaMehta software architecture consists of an application server. DeepaMehta applications are developed in the scope of the DeepaMehta application framework and run at server-side. Applications can access a variety of data sources and can be served to a variety of frontends.*

Features of the DeepaMehta software architecture:

- Java-based application server
- multi-layer architecture
- Supported Semantic Web standards: ISO 13250 Topic Maps, RDF
- DB-neutral storage layer, currently MySQL and HSQL (pure Java DB)
- Variable frontends: Topic Maps GUI (Thin Client), web browser, PDA, mobile phone
- Object-oriented framework for application developers
- SOA (Service Oriented Architecture)
- Access to external web-services via SOAP
- Access to external datasources: SQL, LDAP
- XSLT-based publishing engine for dynamic SVG, PDF and XHTML generation
- Integrated mail and web support: SMTP, POP3, IMAP, HTTP

The DeepaMehta system architecture comprises all the layers of an IT system: the storage layer, the application layer, and the presentation layer. DeepaMehta is a distributed architecture, so that every layer can reside on a different machine.

The heart of the DeepaMehta software architecture consists of an application server. DeepaMehta defines a unique application model and provides a framework for application developers. A DeepaMehta application is actually a collection of topic types and association types that may be assigned to a workspace. Every type can be attached to a Java class that implements the behavior of the instances of that type. The topic type “Postal Address”, e.g. may be attached to an implementation that fetches a map from Google Maps for a particular address. New applications can be aggregated from existing types. Thus every application that deals with postal addresses will benefit from the Google Maps behavior.

Types can be assigned to shared workspaces. The user “deploys” an application just by joining a shared workspace. Once a user joins a shared workspace he/she gets access to the assigned types, and thus to their functionality.

Data Model

The DeepaMehta data model is inspired by ISO 13250 Topic Maps. On the one hand the Topic Maps standard is not fully implemented, on the other hand, there are extensions to incorporate RDF/RDFS-like features. There are a lot of proposals for the integration of the ISO Topic Maps standard and the W3C RDF/RDFS standard (Pepper et al., 2006) but DeepaMehta chooses its own ad-hoc approach because when the project started, in early 2000, these proposals did not yet exist. Furthermore in DeepaMehta there are higher-level concepts like *Query*, *Datasource*, *Workspace*, and *User* defined.

- Topic Maps concepts that are not realized resp. modified in DeepaMehta:

  n-ary Associations: For the sake of simplicity DeepaMehta supports only binary associations. This is not a serious lack because a n-ary association can be emulated by n binary associations.

  Association Role, Association Role Type: because DeepaMehta supports binary associations only, and associations are explicitly directed from node 1 to node 2 there is

no need to realize the concepts of Association Role and Association Role Type. The meaning of the involved nodes are represented in the direction of the association.

Occurrence, Occurrence Role, Occurrence Role Type: because DeepaMehta pursues an integrated GUI approach, occurrences are not realized explicitly. External resources are represented as topics, e.g. a file is represented as a topic of type Document, and a webpage is represented as a topic of type Webpage.

Facet: because occurrences are represented as topics and because DeepaMehta provides the concept of *properties* (see below), there is no need to realize the concept of facets.

Scope: not yet realized. To a great extend ambiguity may be diminished by typing topics. In DeepaMehta there would be e.g. 2 topics “Tosca”, one of type *Opera* the other of Type *Character*. Furthermore DeepaMehta provides the concepts of *Workspaces* and *Users*. To Workspaces a number of types is assigned. Users are members of workspaces and have access to the respective types.

Topic Maps: In DeepaMehta topic maps are topics too. Thus, topic maps can act as containers for other topic maps. (This is exploited by the GUI to realize the personal and shared workspaces which are in fact topic maps that serve as storage spaces for other topic maps.)

- DeepaMehta Topic Maps extensions to incorporate RDF/RDFS-like features:

  Property, Property Value: part of the type system. To a topic type or association type a number of properties can be assigned. E.g. the type Person has the properties “Birthday” and “Gender”. Property Values are a enumeration of predefined property values. E.g. for the Property “Gender” the values “Male” and “Female” are predefined.

  Relation: to define a relation between types; part of the type system. Part of the definition is the *Cardinality* and the *Association Type* to be used at instance-level. E.g. to model “City of birth”, a relation between the types “Person” and “City” may be defined, cardinality would be set to “One” and association type would be set to “City of birth”.

- Higher-level DeepaMehta concepts:

  Query: a query to the corporate memory and the corresponding result set of topics is a topic itself (and is presented at the GUI as a ton). A query can involve the Topic Name, Topic Type, Property Values (at instance level) and associations a topic is involved in.

  Datasource: external datasources like SQL databases or LDAP repositories are represented as topics too. A datasource is specified via its connection string (an URL starting with jdbc: or ldap: for example) and the assignment of topic types whose instances are created from datasource entities.

  Workspace: a shared workspace with a number of members (users). A workspace is a exchange place for topic maps. Furthermore there are topic types and association types assigned to a workspace to hold the concepts which are used in it. Also the access control mechanism is based on workspace memberships. Workspaces are topics too.

  User: a user of the platform. A user is a member of a number of workspaces. He/She has access to all the types assigned to the workspaces he/she is a member of, as well as to private types. A user has its private sphere (a topic map itself) to store private topic maps. A user is a topic too (derived from *Person*).

The Application Framework

The DeepaMehta platform defines its own application model and provides an application framework. By means of the DeepaMehta Application Framework software developers create applications running on the DeepaMehta platform. A DeepaMehta application consists of a collection of types: topic types and association types. To create, for example, a calendar application, one could define the topic types “Calendar”, “Event”, “Location”, “Person”, and the association type “Participant”.

DeepaMehta applications are not visible to the user as such but only their topic types and associations appearing in the context menus. Types can be assigned to workspaces; the user obtains access to types by joining a workspace.

The application’s data is modeled as type definitions and the application’s behavior is realized by attaching a Java class to a type.

A type definition comprises a set of properties, a derivation from a basis type, and relations to other types.

- A *Property* models an untyped single-value data field of a topic type or association type. The topic type “Event”, for example, could have the properties “Begin Date” and “Begin Time” and the topic type “Person” could have the properties “First Name”, “Last Name”, and “Gender”. For a property a list of possible values can be predefined. For the “Gender” property, for example, the values “Male” and “Female” could be defined.
- By the means of *Derivation* a type definition can be build by specialization of an existing type. The derived type inherits all the properties and behavior (see below) of the basis type.
- A *Relation* models the relation between two topic types. In order to model, for example, the place of birth one could define a relation between the topic types “Person” and “City”. A relation includes the specifications of the cardinality — one or many — and the association type to be used at instance level.

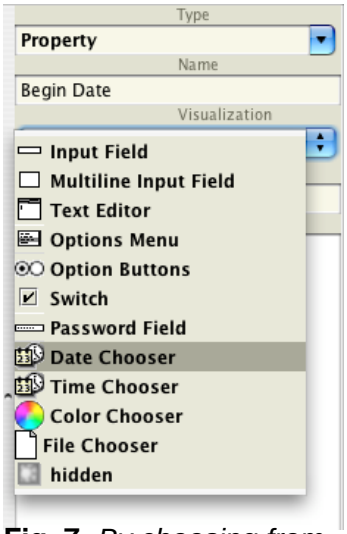

Fig. 7: *By choosing from the list of widgets one can specify how to render a property in the property panel.*

All the building blocks of the data model — topic types, association types, properties, property values, derivations, relations — are topics and associations themselves. Thus, the data model of a DeepaMehta application can be created directly in DeepaMehta as a topic map. As soon as the data model is defined the basis functions of the application can be used immediately. To create, e.g. an event and its participants the user creates an Event-topic and connects it via Participant-associations with Person-topics. Further basis functions like changing, deleting, or searching for events (e.g. all my upcoming events) and also the collaborative calendar usage is possible immediately, because standard information processing and communication functions are already built right into the DeepaMehta platform.

An application consists not only of a data model and generic data manipulation operations but also of domain-specific application logic. The calendar application, for example, could notify the participants of an event in due time via email or instant messaging. This is possible in DeepaMehta because topics and associations are not just information carriers but also provide active behavior. Every type can be attached to a Java class which

implements the behavior of the instances of the respective type. The Java class must be compliant to the DeepaMehta application framework, that is, it must be derived directly or indirectly from a certain base class ("*LiveTopic*") and override specific methods (hooks) to let the topic instance react upon certain events.

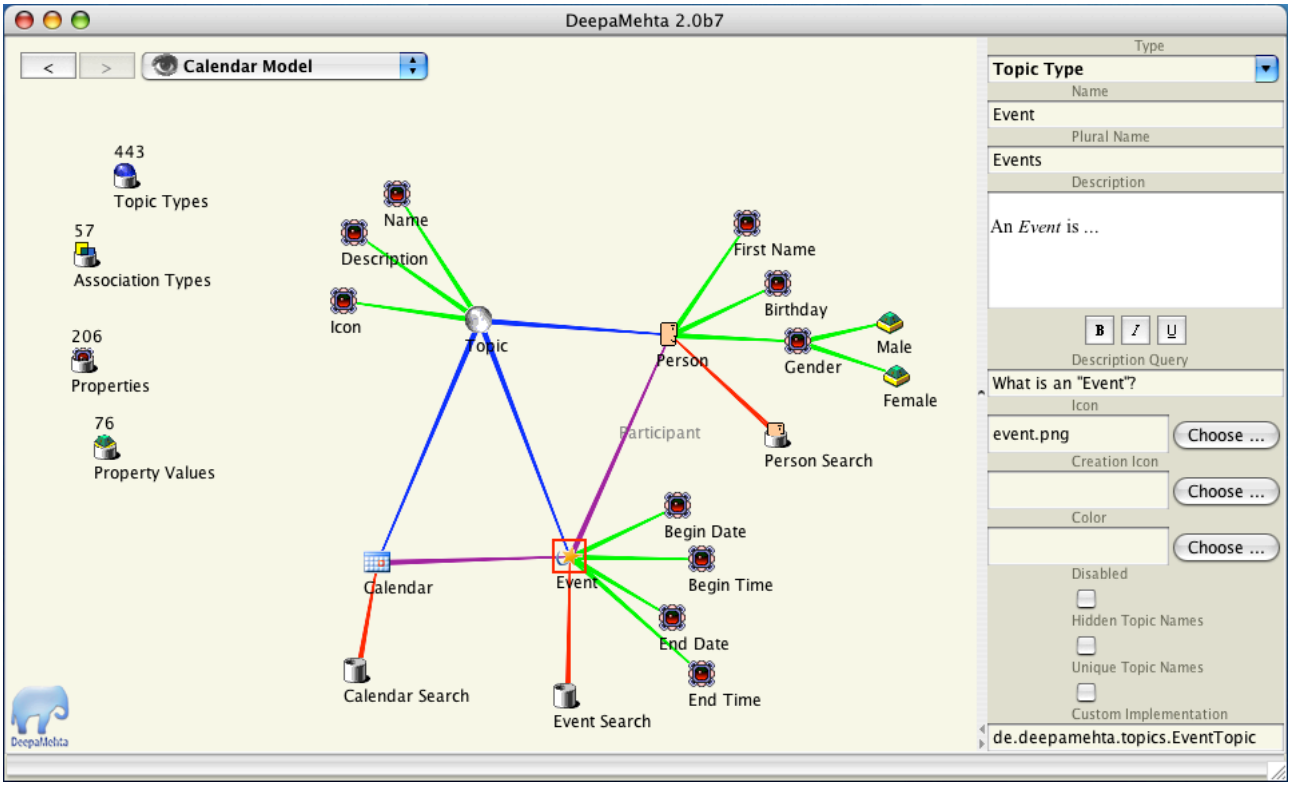

Fig. 8: *The elements of the data model — topic types, association types, properties, and property values — are topics themselves and are handled as topic maps as well. Here, the topic types of a calendar application are displayed. The topic types "Calendar", "Event", and "Person" derive basis properties from the "Topic" base class and define their own properties. To realize application-specific behavior every type can be attached to a Java class.*

The DeepaMehta application framework is primarily event driven. It defines a number of events to which a topic or an association can react. A topic, for example, can react once …

- … it is clicked. Most of the topics react by displaying their properties in the property panel (right side).
- … it is right-clicked. Most of the topics react by dynamically assembling and displaying a context menu. An "Email"-topic, for example, can provide a "Send" command.
- … a command has been chosen from its context menu. The base class provides the standard behavior for handling generic topic commands like "Hide", "Retype", "Delete", "What's Related?", and "Google Search".
- … one of its properties has been changed. An "Event"-topic, for example, could notify all the participants once an event is postponed. Furthermore, a topic can check the new property value and possibly reject it, or prohibit a property change a priori.
- … it is associated with another topic. An "Event"-topic, for example, could notify a newly assigned participant. Furthermore, both of the associated topics get the chance to specify a certain association type or to reject the association.

- … it is moved within a topic map. This can be useful, for example, if the topic map acts as a time or geo grid. Furthermore a topic can lock itself to prohibit any movement.
- … the topic map in which it is contained has been published in a shared workspace. A “Document”-topic, for example, uploads its assigned file to the server-side file repository.

The former list provides just an overview of the most important events. The DeepaMehta application framework defines about 40 events.

The event handler hooks are ordinary Java methods. Within the hooks all Java APIs can be used that are available on the server. Furthermore, the DeepaMehta application framework provides a number of service calls in order to navigate in the corporate memory, for example. All hooks are informed by the framework about the user who triggered the event. Thus the topic can impose access restrictions, for example, by disabling the menu commands for which the current user doesn’t have the required credentials.

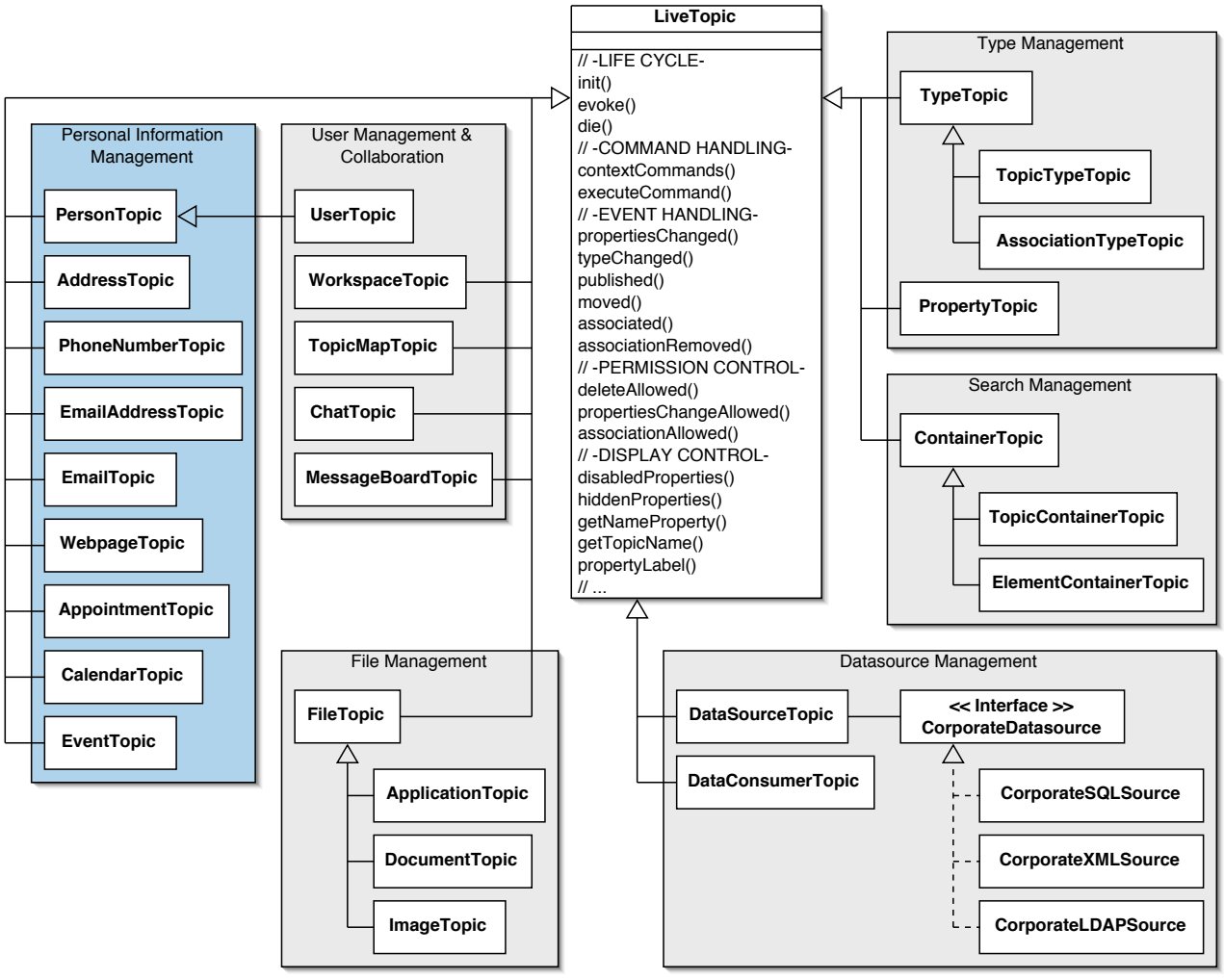

Fig. 9: *This class diagram shows a selection of the LiveTopic subclasses that come with the DeepaMehta platform. The platform’s core features like “User Management & Collaboration” (grey areas) as well as the standard applications like “Personal Information Management” (blue area) are implemented as LiveTopics. The base class “LiveTopic” provides the hooks to be overridden by application-specific topic classes, in order to react on certain events.*

The DeepaMehta Unified Process

The development process is divided into three stages. Traditionally, at every stage another tool is used. At the stage of *brainstorming* one creates text or graphic content e.g. by using text editors or dedicated brainstorming tools like MindManager. To be able to process the contents later on by the means of application logic, a *model* must be built e.g. by creating database tables. Finally, the application logic must be *coded* by a programmer. At each of the three stages, different persons with different skills are involved and different work environments are used, each with a different user interface. This makes for a slow and cumbersome development process. Content must be migrated from one stage to another, changes in one stage may require changes in the other stages. Communication between the involved persons is difficult because of the different jargons.

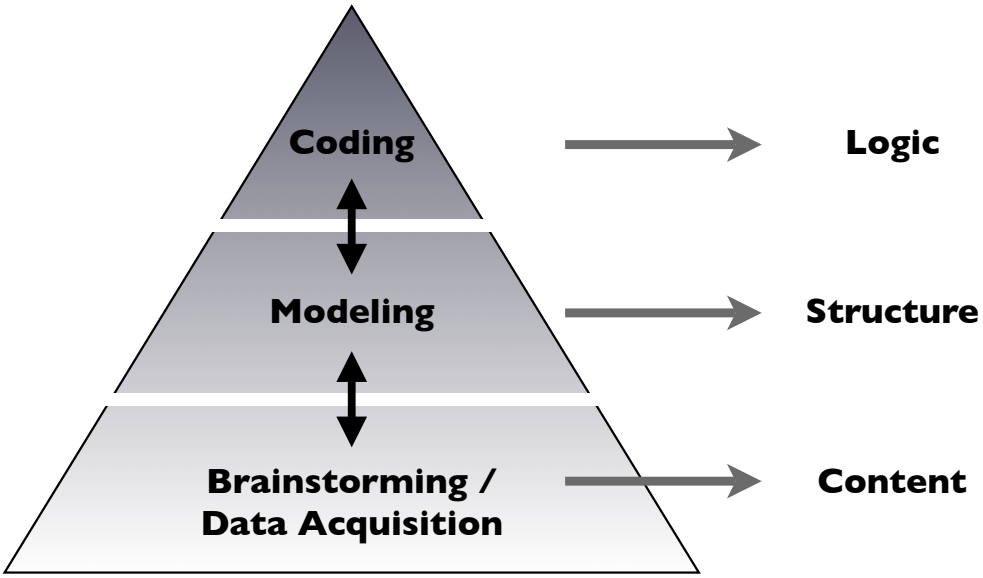

Fig. 10: *Processes (left side) and artifacts (right side) as involved in every IT-project. The DeepaMehta Unified Process accommodates constant changes in the Content, Structure, and Logic levels. The DeepaMehta platform supports the Brainstorming/Data Acquisition, Modeling, and Coding processes within one user interface.*

The DeepaMehta Unified Process enables smooth transitions between all three levels (content, structure, and logic). Content can be created *before* any structure exists, and unstructured content can later be turned into structured content. No content will be lost, even if the structure changes later. Structured content, as well as unstructured content, and even the structure itself may be subject of later brainstorming sessions. Standard logic for e.g. navigating, searching, editing, and displaying content are built right into the DeepaMehta platform. Structured content may be processed by implementing custom logic.

In order to support the communication process between the involved persons collaboration features are built into the platform. To build contents and structure collectively, the DeepaMehta platform offers shared workspaces. For the negotiation about the meaning of structures, every shared workspace provides a forum and chat as standard communication tools.

The DeepaMehta Unified Process provides a significantly more efficient development process than other existing processes because the DeepaMehta platform simultaneously copes with visual, verbal, and virtual modalities.

Customer Solution Examples

Kiezatlas, a geographical CMS

The first commercially deployed DeepaMehta application was “Kiezatlas”, a geographical content management system (CMS). It was contracted by the Verband für sozial-kulturelle Arbeit, (www.stadtteilzentren.de) the German umbrella organization of settlements and neighborhood centers. Since 2004 Kiezatlas is successfully deployed to publish (city) maps of social relevant institutions on the website www.kiezatlas.de.

Fig. 11: *The public web frontend of the Kiezatlas geographical content management system. The left side displays locations of stores and institutions in a city map. The right side shows detail information about the selected institution. Institutions can also be found by category or by entering a search term.*

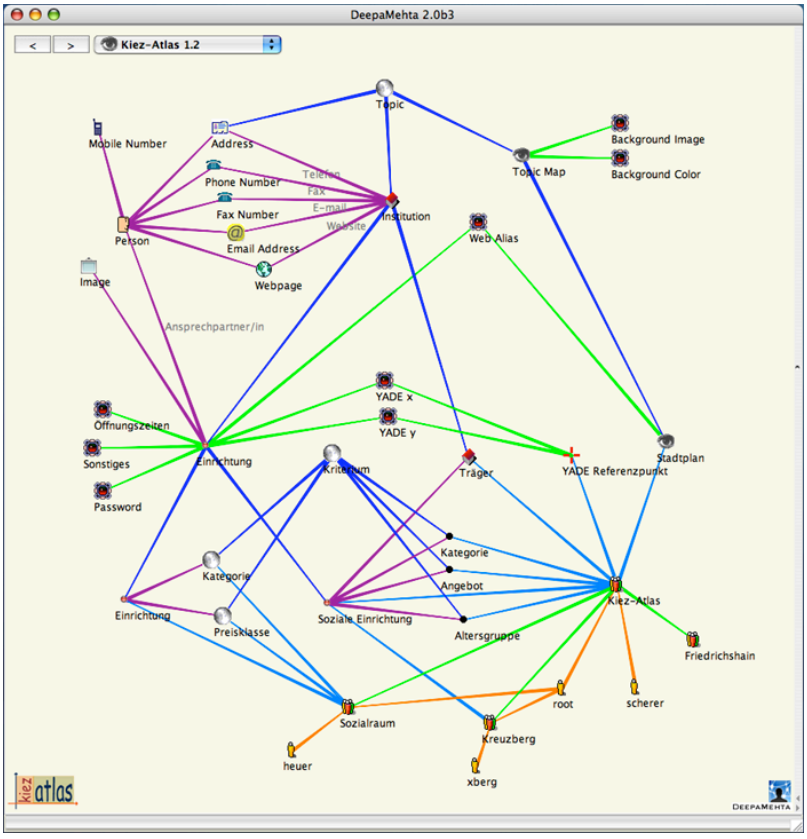

Fig. 12: *The Kiezatlas administrators use the DeepaMehta topic map frontend to define the underlying data models of city maps. In the upper area DeepaMehta standard topic types and properties are visible, e.g. “Person”, “Institution”, and “Topic Map”. In the lower area one can see how the Kiezatlas-specific topic types and properties are derived from (blue associations) and set into relation (purple associations) the DeepaMehta standard types. The topic type “Stadtplan” (city map), for example, derives its properties and behavior from “Topic Map” and adds its own property (green association). Furthermore the search criteria for the city map contents are defined via derivations and relations. Also part*

of the topic map are users, workgroups, and memberships (orange associations), to define access control (turquoise associations), i.e. the responsibilities of the various (sub)administrators. The public website (Fig. 11) as well as the editor backends for the institution owners (Fig. 13) are generated directly from this data model, e.g. if a new property or search criteria is added to the model, the website and the editor backend are updated automatically.

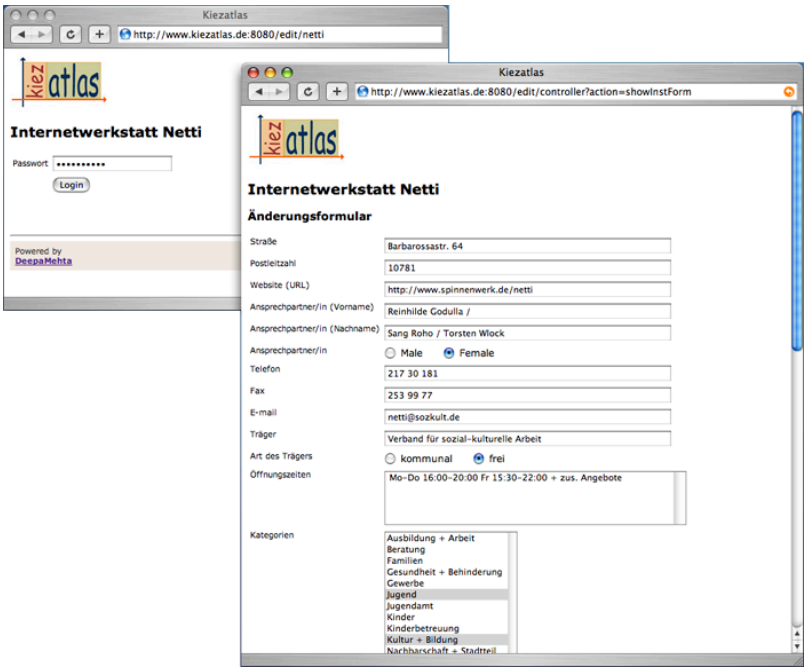

Fig. 13: *The Kiezatlas editor backend is a form-based web application by which institution owners can update institution information on their own. The institution form is generated dynamically by the DeepaMehta web engine, based on complex type definitions (Fig. 12). The contact person of an institution, for example, is modeled as relation between the "Institution" topic type and the "Person" topic type. The DeepaMehta web form generator embeds the Person form (with "First Name", "Last Name", "Gender" fields) inside the Institution form. To the editor the form looks like an ordinary HTML form, but with the DeepaMehta web engine the user input is stored as a semantic network in the corporate memory. The contact person of an institution, for example, is represented as topic of type "Person" that is associated with the "Institution" topic.*

amina Knowledge Platform

The amina knowledge platform was contracted in 2006 by the German amina foundation, (www.amina-initiative.de). The amina initiative aims to promote corporate responsibility (CR) projects by establishing a dialog between companies, universities and avant-garde thinkers. The amina knowledge platform is built on the basis of DeepaMehta. An interactive public demo is available (in German) at www.amina-wissensplattform.de. DeepaMehta is also utilized as a live-mapping tool during the various amina events.

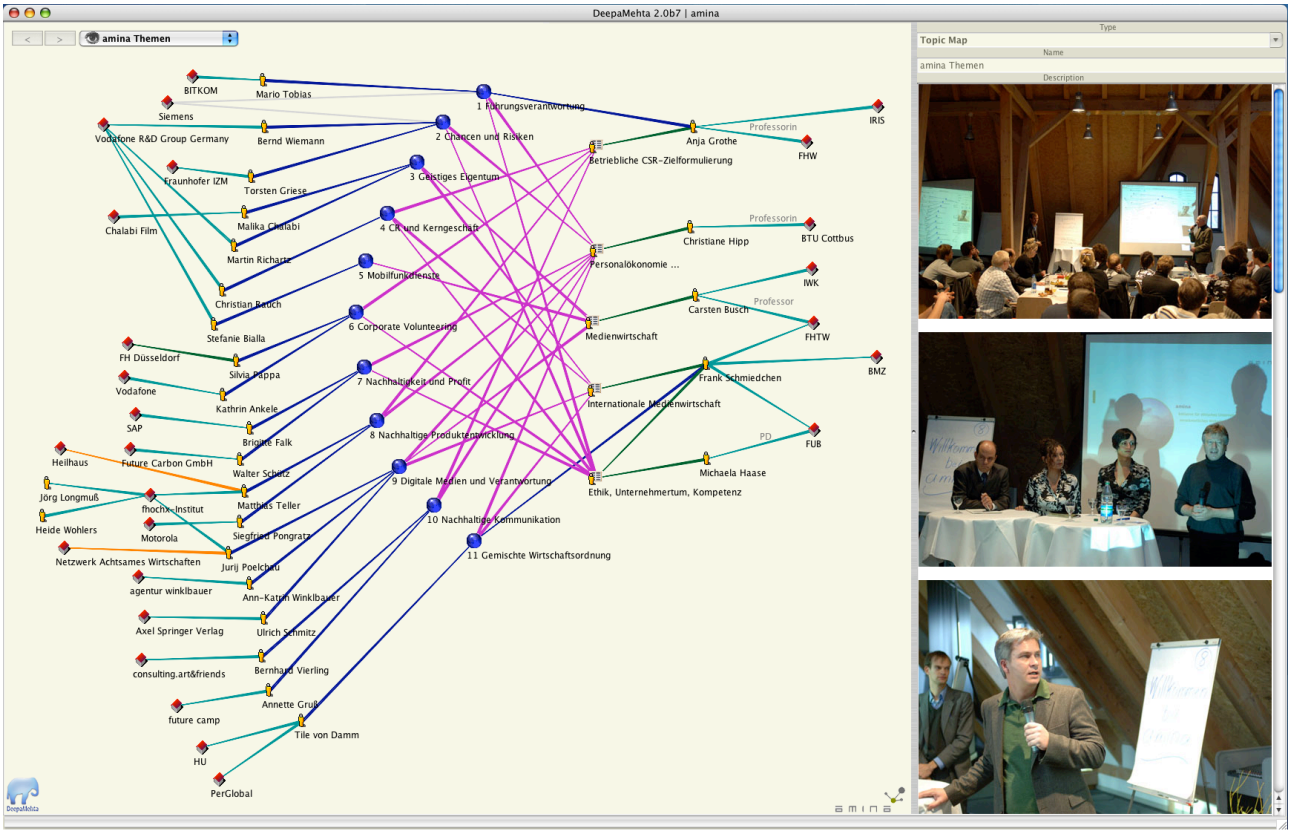

Fig. 14: *A topic map of the amina network. Relations between amina Corporate Responsibility topics (blue balls) and suitable university courses are shown in pink. amina agents and their affiliations to corporations and universities are displayed alongside their mentorships (blue associations) for the amina topics. Every amina topic is related to a shared workspace for the students and mentors collaborative work (not visible here).*

Outlook

Extended support for semantic technologies

Currently DeepaMehta can import and export topic maps in a modified XTM format. Future versions of DeepaMehta will support further semantic technologies and domain-specific applications:

- DeepaMehta type definitions will be built from RDF Schema or vice versa.
- All topics and associations will be annotated by a Subject Identifier. This allows e.g. merging or interconnecting topic maps or whole corporate memories with each other while preserving their semantics.
- DeepaMehta will import semantically enriched domain-specific data and transform and visualize them as a topic map, e.g. when dropping a RDF enabled business card (involving the "foaf", "contact" and "geo" ontologies) to the DeepaMehta desktop, it will visualize the contents as a topic map, showing the person, with all of his/her relations to friends, projects and a city map retrieved from google maps.

Extended collaboration features

Essential features of every collaboration environment are a) versioned data storage, b) a change history, c) notification, and d) access control.

- Versioning is a good method to cope with competing change requests without establishing a locking mechanism. Currently DeepaMehta contents are not versioned. Future versions of DeepaMehta will support versioning at 4 levels: topic contents, association contents, topic map geometry/visibility, and file contents.
- A change history helps to keep track of content changes performed by users: what was changed by whom and when? Changes can be presented visually like in Microsoft Word's Change-View or in Wikis. Certain changes can be reverted. Currently DeepaMehta provides no change history.
- A notification mechanism informs users of relevant actions performed by other users: what has been done by my colleagues since my last login? A crucial concern is granularity: who should be notified and in what detail? Currently DeepaMehta's notification mechanism is quite limited: members of shared workspaces are informed via messaging and email once a topic map is updated but not about what has been changed.
- An access control mechanism is crucial to protect confidential information and privacy and to enforce user hierarchies. Currently DeepaMehta provides an access control mechanism but it is not flexible enough. Basically it works on type-level and has the concept of a topic-owner. Future versions of DeepaMehta will provide access control at instance-level and even property-level.

Extended standard applications

The DeepaMehta platform comes with standard applications for Personal Information Management (PIM), Document Management and Web browsing/searching. PIM includes handling emails, contact information, appointments, and notes. All these functions are at prototype-level and can not yet replace the traditional applications. The rendering of HTML pages, in particular, is currently too limited.

Future versions of DeepaMehta will be able to replace the traditional email, web, text, and image applications and will import their data. The feature set that is used by 90% of the users will be provided. On high demand is the integration of a web engine like Gecko. Furthermore, audio and video contents will be supported.

Architectural Concerns

Currently DeepaMehta is a client-server system. The graphic DeepaMehta client is connected to one DeepaMehta server at a time. One server instance serves one corporate memory at a time. In order to fully exploit the network effect, future versions of DeepaMehta will allow one client to be connected with multiple servers resp. corporate memories at once. The requirements of associating topics from different corporate memories and of synchronization of certain parts of different corporate memories is apparent. A matching approach for topics is required. The Topic Maps concepts of Subject Indicators and Subject Identifiers would help here, but their is no obvious solution to these requirements yet.

In order to support the growth of a large-scale network of DeepaMehta servers, one can replace the client-server architecture by a peer-to-peer architecture. This opens up a lot of questions and requires in-depth discussions.

In order to integrate the DeepaMehta application server with other (non-Java) backend applications and services, the DeepaMehta server should provide the DeepaMehta applications as web-services. Because of the layered DeepaMehta architecture which clearly separates the application layer from the storage and presentation layers, it is already very close to a Service Oriented Architecture (SOA). For the moment there is no language independent interface to the application layer. Future versions of DeepaMehta will provide a SOAP interface to the DeepaMehta applications.

Related Work

Semantic Desktop

There are other semantic desktop environments, e.g. “IRIS” (www.openiris.org), “Haystack” (http://groups.csail.mit.edu/haystack/), “Gnowsis” (www.gnowsis.org), or “Nepomuk” (http://nepomuk.semanticdesktop.org). But none of them solve the problem of the missing work context in a cognitively adequate fashion (see “The Trouble with Computers and their solutions”).

Graph-based Information Visualization

There are other popular applications, for example, “TheBrain” and “Inxight StarTree”, that deploy tree or graph-based visualization to navigate in information spaces.

With “TheBrain” the user can organize his/her notes and files into a tree structure. The application displays an extract of the tree as nodes and edges. The node currently focused by the user is displayed in the middle of the screen. Around the focused node the application places all neighboring nodes. These comprise all the nodes that have a direct connection to the focused node. All other nodes are not displayed.

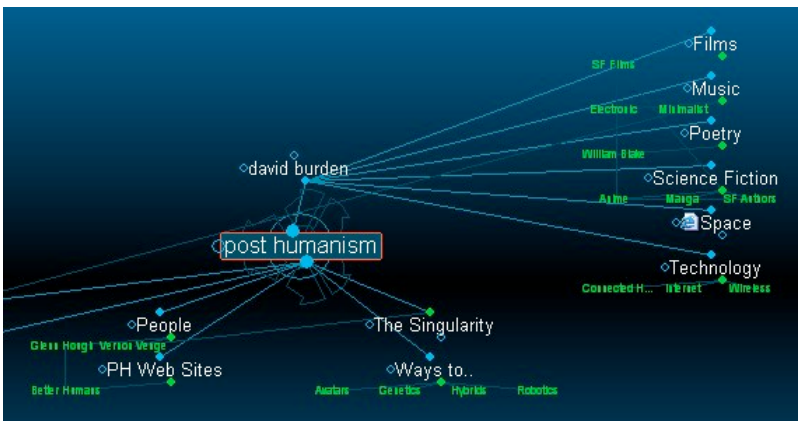

Fig. 15: *“The Brain” displays an extract of a tree structure, with the focused node in the middle of the screen. Setting another focus causes a complete rearrangement of the display.* *www.thebrain.com*

To navigate the tree structure the user clicks on a visible node. The application then moves the clicked node to the middle of the screen and again displays its direct neighborhood. The tree layout is done automatically, according to rules not transparent to the user. Therefore “TheBrain” is not cognitively adequate as it does not satisfy the criteria of free positioning. Haller (2003) points this out in his thesis “Mappingverfahren zur Wissensorganisation”:

> “The main advantage of [visual] mapping approaches is the possibility to organize information *spatially*, according to the user’s Cognitive Map (Chen & Czerwinski, 1998; Dillon et al., 1993). So, it is important to let the user position the nodes freely in order to adapt the map to his/her internal spatial model. The nodes should keep their positions (at least relatively to their neighborhood), if the map is modified. Otherwise orientation in information space by means of Cognitive Maps is hindered.”

“Inxight StarTree” allows the visualization of extensive tree and network structures on a limited display space. Like “TheBrain”, the node that is focused by the user is displayed near the middle of the screen. But unlike “TheBrain”, not only the direct neighborhood is displayed, but also deeper levels of the node hierarchy. This is possible because “Inxight StarTree” deploys a projection of a hyperbolic plane to a region of the screen. Effectively, an increasing scaling factor is used towards the screen borders.

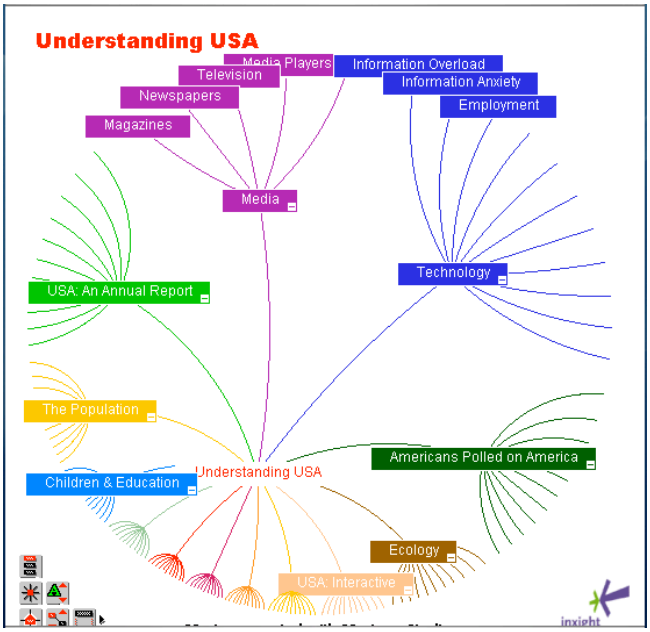

Fig. 16: *“Inxight StarTree” deploys a hyperbolic projection to display different hierarchy levels at once. While navigating, the spatial neighborhoods of the nodes are not preserved.* *www.inxight.com/products/sdks/st/*

The tree or network structure is navigated quasi by moving the screen region within the hyperbolic plane. Due to the hyperbolic projection, it can happen that one minute node A is above node B and the next minute it is the other way around. This visualization approach is also cognitively inadequate as it doesn’t allow the user to orientate him/herself within the information space.

Both, “TheBrain” and “Inxight StarTree” allow the user to navigate large information spaces in an associative manner. But their automatic layout approaches are not cognitively adequate because they do not support the process of knowledge acquisition — learning — of the user. The visualization approach does not consider the mental state of the learner. A work environment for knowledge workers can be regarded as cognitively adequate if it supports associative navigation *and* a visualization approach that enables the user to arrange, and thus create, the display.

Future Research Directions

Until now artificial intelligence and semantic web have brought no significant benefit for the computer user. This situation will not change as long as computer scientists adhere to the rationalist tradition in AI and its reduced and mechanical understanding of our environment and human nature itself. The authors propose to reanimate the principles of soft cybernetics and to examine the images that underly the current computer research and system designs.

Reanimating Soft Cybernetics

The modern computer age is in its infancy and our understanding of what computers do and how their functioning is related to human language, thought, and action is very limited. This is evident when software engineers use terms like “intelligence”, “recognition”, “meaning”, “learning”, “reasoning”, “knowledge”, and “understanding”, for example, to describe their systems. Computer science is practiced mainly by engineers. The authors propose interdisciplinary collaboration between computer scientists and humanists: psychologists, linguists, philosophers, and epistemologists, as well as with biologists and artists. To the knowledge of the authors the last crucial effort for such collaboration were the Macy conferences held from 1946 to 1953.

During the following years, computer research split into two schools of thought: hard cybernetics, which later became the AI movement, and soft cybernetics. It is the school of hard cyberneticists and the AI advocates that increasingly dominates computer science research. They receive the major funding and set the research agenda, while the soft cyberneticists seem to have disappeared from the scene.

The central points of these two schools of thought are:

- The hard cyberneticists and AI advocates believe that all things and processes, including language, understanding, learning, and social interaction are describable by the means of logic. The meaning of a thing exists independently from a person that uses that thing, e.g. the meaning of a text is hidden *in* the text itself and is independent from the reader. Cognitive aspects play no role in this belief, because cognition is a phenomenon that can be simulated by a mechanical-mathematical model.
- The soft cyberneticists believe intelligence is an evolutionary result of the environment in which it has emerged. Meaning emerges only in the moment of action. Cognitive systems are closed systems which emerge from themselves, in a process that can be called “Autopoiesis” (Maturana & Pörksen, 2002). Intelligence is inseparable from cognition and information is inseparable from its use. Computers can not be intelligent as long as they are not closed systems but dependent on a programmer.

The authors suggest a redirection of attention away from the school of hard cybernetics and AI to a focus on the school of soft cybernetics, and to study the human-machine-system and the relation of cognition and computers.

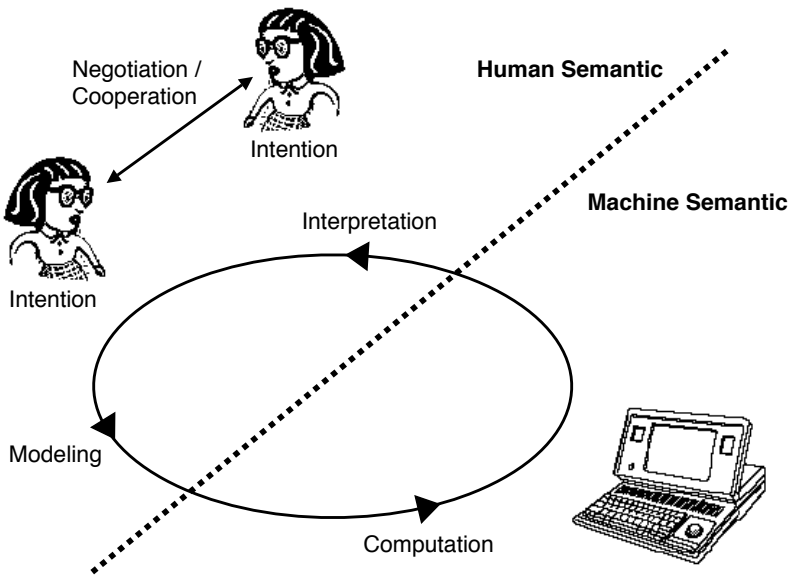

Fig. 17: *Feedback cycle of the human-machine-system. Human beings have intentions and negotiate/cooperate with each other about how to use the computer in a certain usage context. They make a mathematical model with which the machine can compute. The results are interpreted by human beings at two levels: 1) what to do with the result in the usage context? and 2) Meta-Interpretation: how did the result come about? Is the model correct?*

A soft cyberneticist does not see the computer as an entity that performs a dialog with the user, or that cooperates with the user, or that has any intention, because a computer is not a cognitive system. Even if the property of having intentions or the process of interpretation is included into a mathematical model, no new quality is established in the machine. The cycle is just delayed. In the end, it remains up to the human being to interpret the value of the generated results in respect to his/her usage context. Human beings and machines belong to different semantic realms, and the dividing line can not be removed. Interpretation can not be automatized.

Interdisciplinary collaboration would help to get a better understanding of the human-machine-system and to build computers with a greater benefit for the user.

Examining underlying images

For the direction of future computer research one needs to examine the underlying images that computer scientists and system designers have of the machine. One would foresee completely different application domains and create different user interfaces depending on whether one sees the computer, for example, as an assistant or as a tool:

- With the image of an *assistant* in mind, the computer is regarded as an intelligent agent, one who performs a dialog with the user, and understands the user's intentions.

  This is evident, for example, when the word processor asks one "A sentence must begin with a capital. Shall I do it for you?". Or when semantic web researchers try to delegate the process of semantic annotation to the machine, as if the machine could read and understand webpages on one's behalf.

- With the image of a *tool* in mind, the computer is regarded as a thing. The human being performs his/her tasks him/herself and uses the tool to augment his/her faculties.

Outside from computers, the building of tools has a very long tradition. We are far away from harnessing the digital power as we harnessed, for example, a hammer or a pencil. But if we ask the right questions about how digital tools should be built, we are on the right track. The authors propose that researchers and engineers adapt the tool image when formulating research issues and designing computer systems.

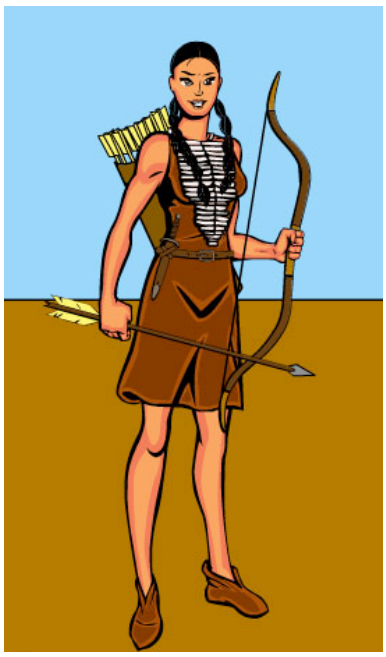

Fig. 18: *The underlying image of the machine determines the research direction and the resulting computer designs. If one sees the computer as an intelligent assistant (left side), one tries to build a machine with cognitive faculties. If one sees the computer as a tool (right side), one tries to design it to meet the demands of human beings in their environment.*

Research Issues

Computer Science / Human Computer Interaction

- What are the implications for the semantic web, if semantics is not hidden in texts but only emerges when the text is used by human beings in a certain usage context? (*Meaning as Use* as described in Wittgenstein, 1953)
- What are the implications for artificial intelligence and the semantic web if “information and its use are inseparable and actually form a single process”? (Jerzy Konorski, cited in Foerster, 1993)
- How can a collaborative semantic work environment be designed to support constant changes of content, structure, and logic?
- What images support a sustainable technology development?

Cognitive Science

- How can the process of knowledge creation be supported by a computer?
- How can a user interface be designed to exploit the inherent human cognitive faculties?

Communication Science / Linguistics

- How can the process of semantic emergence be supported by a computer?

Philosophy

- Do computers act?

Cultural Studies

- What comes after postmodernism? What could be the features of post-digitalism?

References

Canfield Smith, D., Irby, C., Kimball, R., Verplank, B., & Harslem, E. (1982). Designing the Star User Interface. *Byte, 4/1982*, pp. 242-282.

Chen, C., & Czerwinski, M. (1998). From latent semantics to spatial hypermedia — An integrated approach. *Proceedings of the 9th ACM Conference on Hypertext (Hypertext '98). June, 1998, Pittsburgh*. Retrieved August 6, 2007, from www.pages.drexel.edu/~cc345/papers/ht98.pdf

Dillon, A., McKnight, C., & Richardson, J. (1993). Space — the Final Chapter or why physical representations are not semantic intentions. In C. McKnight, A. Dillon, & J. Richardson (Ed.), *Hypertext: a Psychological Perspective* (pp. 169-191). Chichester: Ellis Horwood.

Foerster, H. v. (1993). *Wissen und Gewissen. Versuch einer Brücke*. Frankfurt am Main, Germany: Suhrkamp.

Grudin, J. (1990). Interface. *Proceedings of the 1990 ACM Conference on Computer Supported Cooperative Work* (pp. 269-278). Los Angeles, California: ACM Press.

Haller, H. (2003). *Mappingverfahren zur Wissensorganisation*. Thesis. Published at KnowledgeBoard Europe. Retrieved August 6, 2007, from www.heikohaller.de/literatur/diplomarbeit/mapping_wissorg_haller.pdf

Maturana, H. R., & Pörksen, B. (2002). *Vom Sein zum Tun. Die Ursprünge der Biologie des Erkennens*. Heidelberg, Germany: Carl-Auer-Systeme Verlag.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 1956, vol. 63*, pp. 81–97.

Murray, J. H. (2003). Inventing the Medium. In N. Wardrip-Fruin, & N. Montford (Ed.), *The New Media Reader* (pp. 3-11). Cambridge, Massachusetts: The MIT Press.

Nelson, T. (2001). Talk at ACM Conference on Hypertext (Hypertext 2001). Retrieved August 4, 2007, from http://asi-www.informatik.uni-hamburg.de/personen/obendorf/download/2003/nelson_ht01.avi.bz2

Pepper S., Vitali F., Garshol L. M., Gessa N., & Presutti, V. (2006). *A Survey of RDF/Topic Maps Interoperability Proposals* (W3C Working Group Note 10 February 2006). Retrieved August 4, 2007, from www.w3.org/TR/rdftm-survey/

Richter J., Völkel M., & Haller, H. (2005): DeepaMehta — A Semantic Desktop. In S. Decker, J. Park, D. Quan, & L. Sauermann: *Proceedings of the 1st Workshop on The Semantic Desktop. 4th International Semantic Web Conference (Galway, Ireland)*, volume 175. CEUR-WS, November 2005.

Wittgenstein, L. (1953). *Philosophische Untersuchungen*. Frankfurt am Main, Germany: Suhrkamp (2003).

Additional Reading

Semantic Web / Semantic Desktop

Decker S., & Frank, M. (2004). The Networked Semantic Desktop. In *Workshop on Application Design, Development and Implementation Issues in the Semantic Workshop at WWW2004*, New York, USA. Retrieved August 6, 2007, from http://triple.semanticweb.org/svn/stefan/DeckerFrank.pdf

Manola, F., & Miller, E. (2004). *RDF Primer* (W3C Recommendation 10 February 2004). Retrieved August 6, 2007, from www.w3.org/TR/rdf-primer/

Pepper, S. (2000). *The TAO of Topic Maps. Finding the Way in the Age of Infoglut*. Retrieved August 6, 2007, from www.ontopia.net/topicmaps/materials/tao.html

Pepper, S., & Moore, G. (2001). *XML Topic Maps (XTM) 1.0* (TopicMaps.Org Specification). Retrieved August 6, 2007, from www.topicmaps.org/xtm/1.0/

Computer System Design

Landauer, T. K. (1996). *The Trouble with Computers: Usefulness, Usability, and Productivity*. Cambridge, Massachusetts: The MIT Press.

Norman, D. A. (1986). Cognitive Engineering. In D. A. Norman & S. Draper, *User Centered System Design* (pp. 31-61). New Jersey: Erlbaum.

Suchman, L. A. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication (Learning in Doing: Social, Cognitive & Computational Perspectives)*. Cambridge, UK: Cambridge University Press.

Winograd, T., & Flores, F. (1986). *Understanding Computers and Cognition. A New Foundation for Design*. Addison-Wesley Professional.

Philosophy of Language / Semantics

Austin, J. L. (1962). *How to Do Things With Words*. Cambridge, Massachusetts: Harvard University Press (2005).

Searle, J. (1969). *Speech Acts*. Cambridge, UK: Cambridge University Press.

Cognition

Foerster, H. v., & Pörksen, B. (2006). *Wahrheit ist die Erfindung eines Lügners - Gespräche für Skeptiker*. Heidelberg, Germany: Carl-Auer-Systeme Verlag.

Maturana, H. R., & Varela, F. J. (1992). The Tree of Knowledge. *The Biological Roots of Human Understanding*. Boston: Shambhala.

Miscellaneous

Weizenbaum, J. (1978). *Die Macht der Computer und die Ohnmacht der Vernunft*. Frankfurt am Main, Germany: Suhrkamp (2003).

About the Authors

Dipl.-Inf. Jörg Richter

Born 1967 in Berlin. Develops software since 1980. CHIP programming award in 1985. 1988-1995 study of computer science (focus on AI and software engineering) and linguistics at Technical University Berlin. Fellow at Research Centers for Network Technologies and Multimedia Applications under Prof. Rebensburg. Professional software developer in the areas of knowledge management, e-learning, and learning management. Lead developer of artfacts.net, the world's largest portal for modern and contemporary art. Since 2005 CTO of the DeepaMehta company. Best-Practice awards for DeepaMehta of D21 and we make IT Berlin.Brandenburg initiatives.

Dr. rer. nat. Jurij Poelchau

Born 1963 in Berlin. 1982-1998 study of physics, mathematics and philosophy at the Technical University Berlin. Scientific assistant. Publications in the field of many-body quantum mechanics. Since 1998 researcher and consultant for sustainability strategies at Agenda-Agency Berlin (co-founder), TU Berlin, Regioconsult, and fx-Institute for Sustainable Economics (co-founder). His main field of work is to synthesize social, cultural, ecological, economic and technological innovation. Since 2007 consultant for the DeepaMehta company. Co-founder of the amina-foundation.