Towards an Event-Driven Workplace for Knowledge Integration

Alexander Schatten, Stefan Biffl
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Vienna, Austria
{schatten,biffl}@ifs.tuwien.ac.at

Abstract

A contemporary office or knowledge worker has to deal with an ever increasing number of information channels and associated flows of events (i.e., software applications using varying terminologies and access procedures). The events from different sources in varying terminology need suitable interpretation in the local context for (a) determining basic relevance, (b) understanding context-specific semantics, and (c) making and maintaining semantic connections between information demands from different sources. Without (tool) support the office-worker has to use her brain for this overhead activity instead of focusing on the actual work.

This paper introduces concepts on how to standardize "business events" to allow unified treatment of information flow regardless of the specific information channel they come from. These concepts of unified meta-data extraction, meta-data annotation, and unified user-to-storage interface (i.e., a non-intrusive personal assistant) relieve the user from unnecessary interpretation work load. Further they allow tool support to build up new knowledge based on aggregated information from "relevant semantic connections" between information units coming from any channels.

1. Introduction

The importance of knowledge management (KM) and well-designed communication infrastructure in knowledge-driven businesses as in Software Engineering has become evident during the last decade. Several publications stressed the importance in this field and suggested various solutions [7, 18]. While the KM topic has also been used as buzzword to promote a broad range of different IT infrastructure projects [17], conclusions can be drawn from previous approaches: e.g, that a mixture of different IT systems in approaching a KM infrastructure installation should be avoided. Unfortunately this is often not easily possible. As KM is seen as a way towards unification of user interfaces and integration of various information systems (particularly for unskilled workers [16]), our research aims to design systems that work even in heterogeneous environments. Particularly as contemporary knowledge or office workers have to deal with an ever increasing number of information channels and events that are relevant for the daily business. (see also figure 1). But it is not only about usability issues — knowledge workers want to aggregate information, but contemporary IT systems limit the necessary interoperability and the semantic connection of related entities. Hence the primary goal of this paper is to develop concepts for the individual worker that offer:

- Standardized access to data sources and events that occur at the typical workplace like emails and instant messages and on this (technical) basis: access using a unified user (query) interface to these entities.
- The capability to annotate information sources and events (e.g., incoming user requests need to be rated for importance or annotated to note the status of the request)
- The functionality to connect those entities with semantic meaning (e.g., a user request per email may be connected to an issue in the bug tracking database).
- Tool(s) that help to capture the implicit knowledge from daily work.
- Support for traceability and longevity of digital information (seen as migration step [12])

The goal is to achieve *relevant semantic connection* of information and events. The result of this process is

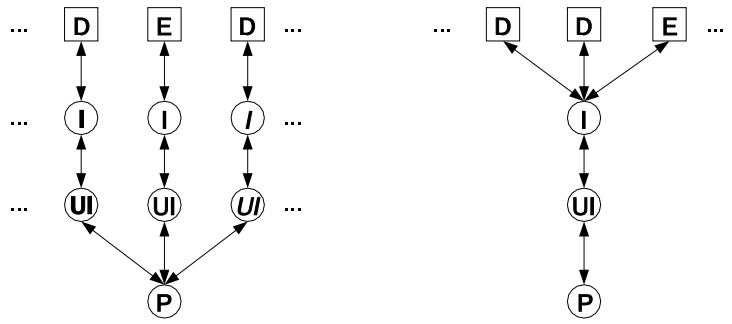

Figure 1. Various flows of events and information occur in daily knowledge work. The illustration on the left shows the current workplace situation where a person (P) has to deal with each data/event source (D), (E) using an individual application interface (I) and user interface (UI). The desired situation is illustrated on the right side with an integrated (user) interface. However, standardized interfaces (I) also support also the connection to other (backend, analysis) systems, not only to user interfaces.

business knowledge that is really relevant for the individual knowledge worker and potentially also for the company.

A personal assistant application that accompanies the user allowing to access the the knowledge repository and enrich it with semantic meaning as well as tracking down *implicit knowledge* — that is a result of human/application interaction — will be described in the next sections. It will also be a foundation to seamlessly build up future knowledge networks in the company (institution) in a *holistic* approach, *not only* bringing people together and building networks [10, 11] or collecting documents (although both are part of a necessary functionality).

A personal assistant application could use this implicit knowledge to make connections between similar entities automatically, or by supplying the knowledge worker with context sensitive information out of the knowledge repository that was built. As all KM approaches should be *non-intrusive*, that is, the user should not be bothered by the process of knowledge acquisition, the KM infrastructure has to be *highly integrated* into the usual daily work process. Moreover, the concept described in this paper, can be seen as an abstract expression of the knowledge management approach described in [15, 16].

This paper introduces (1) how to derive/model business events from user interoperation with IT systems, (2) how to standardize these events, (3) how to create a unified interface to these information resources with the help of a personal assistant application, and (4) finally a scenario that will illustrate a typical workflow in the software engineering domain.

2. Managing Nescience as Resource for Knowledge

In recent papers [15, 16] we figured out, that nescience — meaning: the *things you do not know*, the *problems*, that occur, the *errors* made can be seen as a powerful source for new, and more important for *relevant, necessary* knowledge. As Willke analyzes [19]: *The capability to treat nescience in a constructive way will become a main factor for success in the knowledge-driven economy.*

The question-based KM approach (QB KM) [16] was our first idea to treat nescience as knowledge resource and has been implemented for research purpose [13]. First experiments indicate, that this approach can have a significant impact on system integration and KM issues.

Nevertheless QB KM turned out to be a special case of a more fundamental concept — to get familiar with the new idea of *managing nescience* and transforming it to *knowledge*. While deepening the QB idea we found the possibility to express the basic concept in a more general way. We found out, that even smaller and more abstract *seed crystals* for knowledge than questions may exist, namely (interconnected) *business events*, which are the atomic units of business workflows, as will be defined later. A consequence from the more fundamental approach is the need to distinguish clearer between support for *groups of collaborators* and *individual users*.

In this paper we analyze in a first step (also for the sake of clarity) the individual workplace where the key functions are *search*, *semantic connection (aggregation)* and *annotation* of business events. The back-

end system of this workplace is based on defining business events and implementing a meta-data extraction framework. This supports the "normal" workflow of the individual worker. Nescience is detected and transformed to solutions and in the end to *knowledge* by search and similarity detections of the personal agent application. Knowledge is then a result of special semantic and aggregation operations. The event-based workplace concept supports the individual knowledge worker and increases important production factors like *productivity*, *traceability*, *simplicity* and *unified access to information*.

3. Events from Knowledge Work

This section first of all defines the terms information and (atomic) business events. Then we introduce the concept of generating and standardizing business events from the modeling and technical point of view. Finally, we describe the different event types and operation modes of the system.

3.1. Event Awareness Radius

Knowledge workers, like developers in the software industry, have to deal with an ever increasing amount of different types of (business) events. The highly networked society increases the "awareness radius" of information sources again. In many organizations, some events are already identifies as such building blocks, vital for business workflows and decisions, but a major issue is to channel all theses events to allow an effective management of the different business events.

Additionally all these events may come over various types of channels like telephone, email, fax, personal meetings, printed paper, and so on. The sources of all mentioned events can be categorized following a *shell* concept: the core events are such coming from inside the organization (like messages from developers or from management, ...), the next layers are events from partners and customers. The outer most layer contains events from external sources, e.g., from web feedback forms or questionnaires, non-customer-related emails and so on.

The concept suggested here is based on the idea, that many events can be extremely valuable, particularly when it is possible to handle them in a unified way. Such events are one of the reasons for the high quality of many open-source software products [6]. Once those events are seen as a valuable resource for innovation and new products, not as a distraction *besides* the "real" work, a clear consequence is to improve the creation, handling, and management of this new type of resource[1].

3.2. Context and Needs

The core idea of this paper is, that during daily work not only questions arise but on a finer level of granularity many other events occur, coming directly from inside of the company/institution and from outside (e.g., from customers). But different applications are required to handle these information streams and some are even seen as disturbing factors, often because of the lack of a clear logic how to handle them correctly and efficiently (table 1 gives an exemplary list of events that typically occur in the daily routine of the software developer/engineer). Some of them are mainly active events like email, SMS, or log messages from a server, that can be used immediately as a source of action. Others are passive in the sense, that events can be the result, when they are queried in the appropriate context (e.g., a section in a tutorial in a PDF document can trigger the solution of a problem, which is discussed between developers via email).

Still semantic connections between these events from many sources (made accessible through a standardized interface) are only available *in the brain of the knowledge worker*. Thus the first steps towards a personal assistant that allows a holistic work that includes these entities explicitly are to (1) detect and specify *atomic business events* and (2) to *extract* them in a *standardized* way from the *different information systems* using *abstract meta-data interfaces*.

3.3. Definition of Information Sources and Atomic Business Events (System Infrastructure)

In this paper, the term (business) event means — on an abstract level — entities that (a) occur during daily work in the knowledge-oriented business, (b) have the smallest reasonable boundaries in time, space and meaning and (c) are atomic in that sense. This includes information chunks that are already in the form of events (e.g., email) and information resources that are more static (like file systems): The latter have to be transformed into events from which meta-data can be extracted.

---

1 In fact, successful open source software projects were able to build communities with a clear strategy to define and handle business events as described here. This makes the system suggested very interesting for a contemporary software engineering setup, even in non-open source projects!

	from customer, developer	from internal sources	mixed
structured information	feature requests, bug reports,	open issues, statistical events (log files, development costs),	calendar events, schedules, to-do lists
semi-/unstructured information	specifications, general feedback, support requests	knowledge bases, "hostile events" (hacker, virus, distributed denial of service attacks), documents (repositories, shared filesystems)	communication media (email, forums)

Table 1. Examples for Events that may occur e.g., in the context of software development.

The transformation of all entities that are potentially relevant for knowledge management to atomic events is essential as it allows to see these events as building blocks that start business processes.

Atomic means, that it is not reasonable to derive smaller parts without destroying the inner context of the event. For example: One email can be seen as one event; if the email is fragmented into headers, and the content into words, the inner context is destroyed. On the other hand: the email might be a part of a discussion: Nevertheless in the first inner shell of context (hence the atomic business event) is the email as is, the second, all emails that belong to the discussion, and so on. A telephone call or an email is seen as atomic business event too (rather unstructured though), or a feature request/bug report or an order (more structured events). For examples see also table 1. The source of these events may be inside or outside the company/organizational unit, may be produced by humans or machines, as long as they are relevant for the business process.

To visualize the idea with a metaphor: Currently, different information systems are holding various data sources and the connections between the extracted information chunks has to be done mainly by the user herself. A vital flow of information between systems is hardly possible. In our concept, the events are the *digital nerves* of the company, whereas the access to the information is triggered by these nerves (which are abstract and standardized messages, in a technical sense). The system has to support a healthy nerveous system of event flows, that (1) avoids loss of important events and (2) helps to direct the events to the correct target.

3.4. Designing Standardized Event Sources

Unfortunately data and event sources are very heterogeneous by nature. Moreover some events will not appear in a structured way at all as bug-lists or structured feature requests do and many unnecessary events like multiple detections of the same bug do occur. Thus a major step towards a succsessful management strategy is to develop and setup tools that allow a formation of communities with the capability to distinguish the mentioned groups (internal, customer, external, ...). Such communities arise when a wise combination of groupware tools are used, and ideally are integrated into one user interface. For productive work in the knowledge-based society users need to interact with a broad range of different information and event sources on a regular basis. Integration or at least, access through a common standardized interface is desired for systems like:

- Discussion Forum
- Bug/Issue Tracking
- Mailing List or Mailing Notification functionality
- Content Management System (such as Wiki)
- Versioning of Documents and Sources

These information systems already are a valuable source for the individual knowledge worker (like a software developer or helpdesk worker) as they are, but it is hard to find connections between related events in different sources. E.g., a specific software problem might be announced in a bug tracking system, discussed in a discussion forum and the solution might be written in a Wiki system. Hence the knowledge worker would need to query different systems with various retrieval mechanisms! In fact currently three steps are necessary to create unified results: (1) Query various resources (2) pooling of relevant results (3) reformatting results to a unique format. These steps are an integral procedure in our described setup.

Although some strategies have been researched to handle such semi-structured sources as well as huge distributed database systems, like data warehouse, OLAP

or active databases [4, 5], those approaches usually work *a posteriori*, often used for strategical planning. To support the daily work in an optimal way, *a priori* strategies are recommended, as this allows a seamless immediate interaction with those data source. Moreover the data quality is usually higher in the latter case.

Besides, the data/query-result quality of the knowledge base is higher in the *a priori* strategy, as the user constantly interacts with events; connects, annotates, and ranks them as they come. The knowledge repositories built by these operations are of better quality than automatically performed analysis afterward, as direct (intelligent) user interaction is the reason for connection and semantic, not statistical ("artificial intelligent") analysis afterwards.

The next sections will describe, how to conclude similar events and how to assist the knowledge worker during daily work processes.

3.5. Generating and Treating Events

Generation of events from the sources is depending on the type of the event. For the design of the framework three types are distinguished: (1) natural events, (2) triggered events, (3) user interaction events.

Some systems generate events directly (*natural events*), for example the email system. The concrete implementation could either implement the necessary meta-data extraction and event generation step into the mail-server. Integration of other available systems (*triggered events*) like document management, file systems will require to implement scheduled/triggered extraction mechanisms, e.g., file indexing using search engines like Apache Lucene. In the case of databases, mechanisms like *database triggers or rules* could be used to generate generic events from database operations. The third event type (*user interaction events*) are events, that are a result of users interacting with the knowledge repository, typically using the personal assistant application. Details will be described later on.

In either case, a standardized interface is required to abstract services like the once mentioned above and extracts standardized meta-data, e.g., Dublin Core[2] conformant [8]. The extracted meta-data-message (the event) — ideally with a platform neutral representation of the original data — is sent to the central event handler (server) that stores the events and provides a unified query/access interface. This interface can be used by various applications like those explained in the next section (Fig. 2 illustrates this concept).

However several issues have to be taken into consideration: the original data might be removed from the source system. If only meta-data is stored, the user might find the event, but the original data is lost. If this problem should be avoided, also the original data or a platform independent representation should be stored in the "meta-data" store. As the events are enriched with information about locale and time, this would additionally allow to create a history of events. Considering the enormous growth of the size of hard-disk and other storage media, this strategy could become feasible. In the last consequence a kind of "event diary" would be the result. This would also allow to solve the problem of *data aging*: This means, that information might be outdated after some time. Even here, the combination of keeping all (even historical data) combined with the concept of meta-data-facets allows to retrieve data of precisely the desired period. Still there persists the problem of multiple versions of similar resources and the danger to loose overview in such huge repositories. It is difficult to estimate the significance of these problems right now; solutions will be subject to further research.

3.6. Active and Passive Modes

A further advantage of the concepts presented here shall only be noted briefly and is subject to further research: As the business events are identified, generalized and captured in a event repository, basically two further processing types are possible: active and passive processing. In this paper mainly passive processing is described. That is, a client application (personal assistant, analysis application...) queries the event repository.

But a second powerful option exists: the event repository could be extended and trigger actions following rules that are stored in the repository. This would be the second operation option: the active mode. E.g., when specific events arrive to the event repository, or even a combination of events rules might detect this and perform further processing steps, e.g., invoking other systems.

4. From Events to Knowledge

This section focuses on the user interaction with the system. First of all, the personal assistant application is introduced, that allows to work with the knowledge repository, and furthermore generates specific events dur-

2 The Dublin Core initiative standardizes the nomenclature of typical meta-data keywords used in the description of resources like documents; that are: title, creator, subject, language and so on.

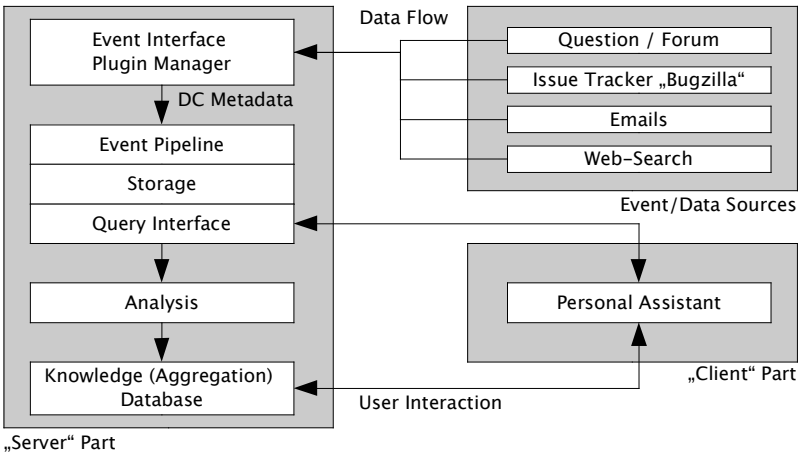

Figure 2. Dublin-Core Meta-data Events plus time-stamps are generated or extracted from conventional IT services (event sources) like email or web-searches (as examples). Those events are stored and access is provided for various applications and services. The knowledge database (knowledge proxy) contains relations between events (aggregation mechanism) and quality feedback.

ing interaction with the user. Then the steps from extraction of semantics out of various information sources towards powerful query mechanism, annotation, aggregation and knowledge generation are explained.

4.1. Personal Assistant and Implicit (User Interaction) Events

The extraction of events and the building of a event repository was described in the previous sections and is illustrated in figure 2.

After the conception of the back-end, as next step, a personal assistant application is desired, that allows to access this event-repository in different ways (see also figure 3 and 4) and offers functionality like:

- "Normal" query of the event repository.
- Pose questions, if normal query is not successful (workflow as described in [16]).
- Assistant functionality in cooperation with other applications.
- Ability to annotate, connect and rank events.

The user should be supported in conventional work processes by this personal assistant. This could be a standalone application or in the domain of software engineering, a plugin into an IDE like Eclipse [9] as a step towards further integration of development tools. To give an example: If the user performs a web-search, an analogous second search is performed by the assistant and parallel to the the web-search results, the assistant presents, e.g., documents or discussion topics to the keywords.

Besides such obvious events the assistant application has to detect and use non-obvious events: A more complex scenario would be: the application or the user interaction creates events while at the same time might query the knowledge repository using the assistant (user interaction events). Such a use case could be the email application: The user might read or write emails (event generation!) and parallel to reading and writing emails, the personal assistant presents similar information and knowledge from the database. An software engineering example could be a developer, who programs using a specific library. The agent presents documentation, forum entries or wiki pages according to this topic.

To conclude the main idea: the personal non-intrusive assistant application should try to capture events: e.g., a web search, a development activity in the IDE. The web-search/email/... triggers two succeeding actions: first of all, the system tries to detect the context (e.g., the pages finally retrieved, probably with quality feedback) and store these facts to a knowledge database. In parallel to the first action, the assistant queries the knowledge repository searching for similar information and presents this information in an non-intrusive way to the worker. In the example above, the assistant could present a list of highly ranked results of previous web-searches, resembling emails sent and received, or fitting documents on the local hard-drive.

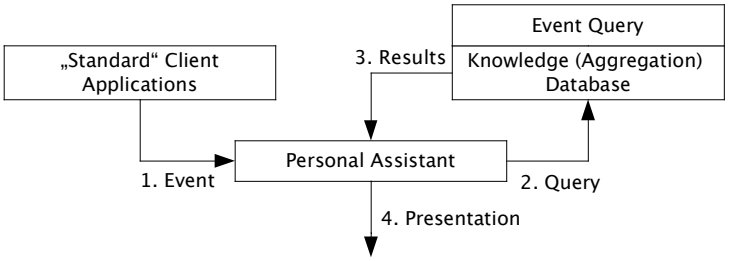

Figure 3. The personal assistant application offers manual query functionality as well as automatic queries according to generated business events (also in interaction with other applications). Query is sent to event-knowledge base and results are displayed parallel to "normal" work.

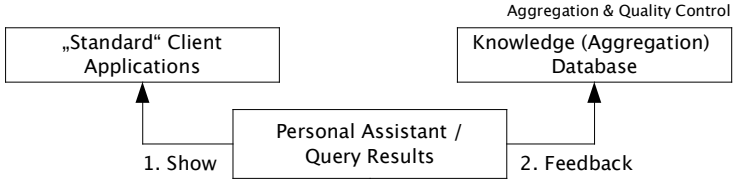

Figure 4. After the query (see fig. 3, the assistant application displays the results, allows to display the potential relevant events and asks for feedback (aggregation and ranking).

4.2. Semantics of Events and Similarity Facets

We assume, that the building of knowledge can be tested on three criteria that are a direct result of the operations described here: Knowledge hence is (1) *relevant* (2) *semantic* (3) *connection* of events or aggregated events. The first, and probably most crucial problem is the question about semantic meaning of events. E.g., the question arises, how the system might detect whether there exist events similar to the event the user is currently generating, or similar to an event a user has selected from a query. While this problem will require a lot of future research, some solutions can be proposed already: First of all, the event interface forces the generation of Dublin Core (DC) meta-data [8]. Consequently already *simple* searches will produce more relevant results than a search just based on conventional full-text-indexing.

The second step is, that during the work with the system, the user makes annotations (in a structured way) and connects and ranks events. This functionality is provided by the personal assistant application and is added to the event repository associated with the affected events. For example, if a user adds a connection between a problem described in the Wiki and a topic in the discussion forum, and there was already a connection between this topic in the discussion forum and some emails, the user just created a new level of aggregation. This aggregation can be understood as similarities between events.

Additionally the user should be able to fine-tune the similarity criteria using a concept we introduce, namely *Dublin Core meta-data-facets*. That is, the user may select one DC type like *author* or *subject* or even a combination of DC types to specify the desired view on the meta-data on the one hand, and on the other to define the criterion of similarity. For example: in one specific context, similarity is seen on the basis of *author* and *date*, and emails, forum entries and Wiki articles according to these meta-data are grouped. In another context, similarity might be seen from the viewpoint of *keywords*, (*resource*) *type* and format. This will turn out to be a very transparent approach even for unskilled users. Of course, the similarity module can be extended with further mechanisms later on, that might offer more powerful features.

	Process	Estim. Time (min)
I	query of different sources	1-10
II	pooling results for completeness	10-30
III	unification of results	40-90
IV	Event-based Knowledge Workplace	1-10

Table 2. Conventional search, pooling, and unification steps (I-III) versus the described event-driven knowledge workplace approach (IV).

4.3. Knowledge Generation

In this paper knowledge is defined as context-relevant information enriched with context relevant meta-information and aggregated to groups of similar entities (according to the meta-data facet of the context), which is provided either by the user manually or semi-automatically. As this assistant application accompanies the user during many day-to-day tasks, it is not necessary for the user to collect or enter information or knowledge pro-actively, as this is mostly done semi-automatically. The data collection and annotation steps are also very natural ones, e.g., when a user follows a suggested link from the assistant it is very little effort to note, if this reference was useful or not. Hence the connection between events are created (aggregation step). The ranking of specific events on the other hand is a manifestation of aggregation levels and meta-data-facet settings. The next steps need to be a connection of the individual assistant applications to build a connected environment, which will be described in the next section, and will be the main step towards integrated knowledge management.

5. A Software Engineering Workplace Scenario

Traditional software engineering assumed, that it is feasible to plan a particular piece of software like other industrial goods. However the advent of modern programing languages, fast compilers, integrated development environments (IDE) with extensive refactoring support created an interesting feedback loop: It becomes more and more possible to refactor, hence change the requirements of the software late in the production cycle; on the other hand, this created the demand among customers to expect, that software fits very dynamic into fast-changing "biotops". Consequences from these findings were also new software engineering methodologies like extreme programming [2].

Besides, the strong networked society, seen from a technological as well as from a sociological point of view, demands faster and faster production and change cycles (as can be seen in the necessity of fast security patches of networked applications). The consequence is, that software producers need not only to react very fast in developing patches for such problems, but also updating/patching system has to be performed in an efficient manner, which requires not only sophisticated logistic, but also cross connecting information from various sources to solve complex problems. Hence considering our approach in supporting these processes, one could find a scenario like this:

A customer has a specific software problem or a potential bug; this issue is sent by the customer to the bug-tracking system. Now an expert tries to solve the problem. In searching the knowledge base or writing emails to other colleagues, the personal assistant constantly tries to query already existing events in all resources, e.g., in the intranet discussion forum, where similar problems have been discussed by other groups. So the expert dealing with this particular customer request hopefully will be pointed to this potential solution of the problem. Already the initial search/query scenario is far more efficient with our approach (as is illustrated in table 2). But solving the problem of the customer is just the first step: in the second step, this might generate new events, e.g., a bug description, and in the third step, the expert will annotate, that the customer query event is related to this discussion forum entry. Hence the next (probably another) developer who has to deal with a similar or even identical problem, will be aware of these solution steps with a higher probability, because of the increasing amount of connections between events.

This increasing connection is a kind of aggregation, so working with different event types of similar features leads to aggregated, "crystallized" and bigger knowledge units. Moreover during complex problem-solution steps, a developer could remember the fact, that a specific other developer is expert in the area, and fine-tune the DC meta-data facet in a way, that the specific author is added as similarity criterion.

In the long run, connected event databases of many (or all) users in a company combined with the knowledge database is building up a kind of organizational

memory that supports future projects and allows to connect people with similar fields of research or similar problems to solve and helps keeping knowledge persistent. But this is future work and subject to additional research. Ultimately, the combination of IT support with a new way of dealing with business events and management issues might lead to a more flexible and new way of an *event-triggered software engineering.*

6. Outlook

> "A software organization's main asset is its intellectual capital, as it is in sectors such as consulting, law, investment banking, and advertising. The major problem with intellectual capital is that it has legs and walks home every day. At the same rate experience walks out the door, inexperience walks in the door.
> [...]
> KM is unique because it focuses on the individual as an expert and as the bearer of important knowledge that he or she can systematically share with an organization." *Rus et.al. [14]*

The advantage of the described system is, that it should generate the mentioned win-win situation for even a single user. Nonetheless, the system is already prepared to be extended for usage in a collaborative setup. As a consequence of the unification and standardization efforts on the back-end as well as on the user interface level, these individual workplaces can be connected to build a complete knowledge management infrastructure and to work as organizational memories:

> "In the long run, we feel an organizational memory should also support knowledge creation and organizational learning. [...] an OM must be more than an information system but must also help to transform information into action." *Abecker et.al. [1]*

Additional benefits can be expected from such a system, as various analysis scripts and applications can easily be connected to the meta-data interfaces (to extract meta-information) and management applications can use these meta-information e.g., for human resource management or strategical planning. Scaling from individual workers to complete companies/institutions is in the long run the basis for turning nescience — that is part of every production step — into a source of knowledge and also into a better manageable risk! However, such a successful approach must fulfill certain additional conditions:

- As soon, as the system is used in a work-group environment, the usage of the KM system must be *motivating* for all user groups.
- Knowledge detected, stored and managed has to be *relevant knowledge*!
- In implementing KM often traceability is a concern too: It may help creating meta-knowledge or detect problems or reasons for failure, when company processes are traceable.
- Events that currently "hit" only the individual user could be routed at a central point, hence a better management of business events will be feasible.

Moreover, the event-based nature of the system is an ideal foundation to initiate business actions out of (a combination) of business events (usage of event-repository active mode, as described above). E.g., specific management facilities: Managers could retrieve statistical information about event flows, events managed, problems, nescience and about connections between problems and users as well as solutions and users. This might assist human resource management as well as strategical planning. But it also poses new problems (as mentioned above) in terms of privacy and control. These problems will be subject of further research.

7. Further Work and Conclusion

The work described in this paper lays the foundation to a broad range of future applications. First of all (as described here), the individual worker will be supported by the basic setup. The next issues will be to connect the individual workplaces to a knowledge forum/marketplace that builds up an organizational memory. A first specific implementation of this abstract concept is already implemented, a system called "Question-based Knowledge Management". A detailed description can be found in a previous publication [16]. This system integrates various data sources, queries them following a unified query interface and connects and aggregates those results according to user interaction in a knowledge database. However, the system described here is far more generic and allows more than this query based access and allows to integrate a broader range of business events. So this specific implementation has to be re-engineered to support the more generic approach described here.

This again will be the basis for additional research in the areas of decision support and event-driven management. Moreover all applications like the personal assistant need to be developed under constant usability

monitoring to ensure the non-intrusiveness, but still efficiency of the approach in the daily routine work. In further work this idea should be tested according to several typical use-cases.

References

[1] Andreas Abecker, Ansgar Bernardi, Knut Hinkelmann, Otto Kuhn, and Michael Sintek. Toward a technology for organizational memories. *IEEE Intelligent Systems*, 13(3):40–48, 1998.

[2] Kent Beck. *Extreme Programming Explained. Embrace Change.* Addison-Wesley Professional, October 1999.

[3] Frederick P. Brooks. The computer scientist as toolsmith. *Communications of the ACM*, 39(3):61–68, March 1996.

[4] Surajit Chaudhuri and Umesh Dayal. An overview of data warehouse and olap technology. *ACM Sigmod*, 26(1):65–74, 1997.

[5] Surajit Chaudhuri, Umesh Dayal, and Venkatesh Ganti. Database technology for decision support systems. *IEEE Computer*, 34(12):48–55, December 2001.

[6] Chad Davis and Coskun Bayrak. Open source development and the world wide web: a certain tension. *ACM SIGSOFT Software Engineering Notes*, 27(5):93–97, 2002.

[7] Rose Dieng, Olivier Corby, Alain Giboin, and Myriam Ribiere. Methods and tools for corporate knowledge management. Technical Report RR-3485, INRIA Sophia-Antipolis, 1998.

[8] Dublin core metadata initiative. http://dublincore.org (as of January 2003), 2003.

[9] Eclipse. http://www.eclipse.org, 2004.

[10] Michael Heiss and Jared Jankowsky. The technology tree concept - an evolutionary approach to technology management in a rapidly changing market. In *Proceedings of the International Engineering Management Conference*, pages 37–43. IEEE, 2001.

[11] Georg Kubasa and Michael Heiss. Distributed face-to-face communication in bottom-up driven technology management - a model for optimizing communication topologies. In *Proceedings of the International Engineering Management Conference*, pages 234–238. IEEE, 2002.

[12] David M. Levy. Heroic measures: reflections on the possibility and purpose of digital preservation. In *Proceedings of the third ACM conference on Digital libraries*, pages 152–161. ACM Press, 1998.

[13] Open science workplace. http://www.oswp.info, 2004.

[14] Ioana Rus and Mikael Lindvall. Knowledge management in software engineering. *IEEE Software*, pages 26–38, May/June 2002.

[15] Alexander Schatten, Stefan Biffl, and A Min Tjoa. Closing the gap, from nescience to knowledge management. In *Proceedings of the Euromicro Conference*. IEEE, 2003.

[16] Alexander Schatten, Franz Inselkammer, and A Min Tjoa. System integration and unified information access using question based knowledge management strategies. In *Proceedings of the International Conference on Information Integration, Web-Applications and Services*, Vienna, 2003. Austrian Computer Society.

[17] Israel Spiegler. Knowledge management: a new idea or a recycled concept? *Communications of the AIS*, 3(4):2, 2000.

[18] L. Steels. Corporate knowledge management. In *Proceedings of ISMICK 1993*, pages 9–30. Compiegne France, 1993.

[19] Helmut Willke. *Dystopia, Studien zur Krisis des Wissens in der modernen Gesellschaft.* Suhrkamp Taschenbuch Wissenschaft, erste edition, 2002.