SoKNOS – Using Semantic Technologies in Disaster Management Software

Grigori Babitski[1], Simon Bergweiler[2], Olaf Grebner[1], Daniel Oberle[1], Heiko Paulheim[1], and Florian Probst[1]

[1] SAP Research
{grigori.babitski,olaf.grebner,d.oberle,heiko.paulheim,f.probst}@sap.com
[2] DFKI GmbH
simon.bergweiler@dfki.de

Abstract. Disaster management software deals with supporting staff in large catastrophic incidents such as earthquakes or floods, e.g., by providing relevant information, facilitating task and resource planning, and managing communication with all involved parties. In this paper, we introduce the SoKNOS support system, which is a functional prototype for such software using semantic technologies for various purposes. Ontologies are used for creating a mutual understanding between developers and end users from different organizations. Information sources and services are annotated with ontologies for improving the provision of the right information at the right time, for connecting existing systems and databases to the SoKNOS system, and for providing an ontology-based visualization. Furthermore, the users' actions are constantly supervised, and errors are avoided by employing ontology-based consistency checking. We show how the pervasive and holistic use of semantic technologies leads to a significant improvement of both the development and the usability of disaster management software, and present some key lessons learned from employing semantic technologies in a large-scale software project.

1 Introduction

Disaster management software deals with supporting staff in catastrophic and emergency situations such as earthquakes or large floods, e.g., by providing relevant information, facilitating task and resource planning, and managing communication with all involved parties.

Current situations in disaster management are characterized by incomplete situation pictures, ad-hoc reaction needs, and unpredictability. First of all, these situations require collaboration across organizations and across countries. Second, they expose an ad-hoc need to resolve or prevent damage under extreme time pressure and non-planned conditions. Third, disasters as such are unpredictable by nature, although preventive planning for disasters can be taken.

Besides the ability to adapt to the current situation, software needs to adapt to the end users' needs as well. Members of disaster management organizations

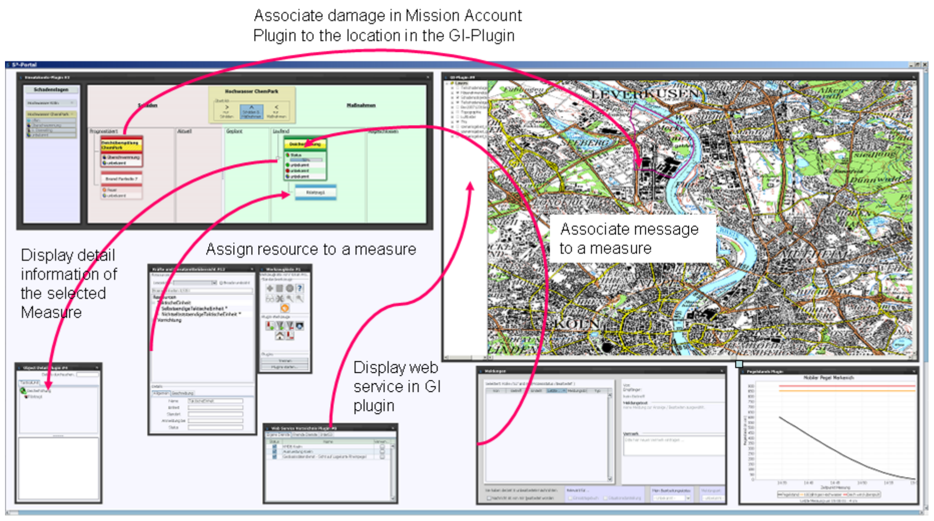
Associate damage in Mission Account Plugin to the location in the GI-Plugin
Display detail information of the selected Measure
Assign resource to a measure
Associate message to a measure
Display web service in GI plugin
LEVERKUSEN
KÖLN

Fig. 1. Screenshot of the SoKNOS prototype [1].

are to a large extent non-IT experts and only accustomed to casual usage of disaster management software, since large incidents luckily occur rather infrequently. This casual usage implies that, for example, users may not always have the right terminology at hand in the first place, especially when facing information overload due to a large number of potentially relevant information sources. A large incident poses a stressful situation for all involved users of disaster management software. Users often need to operate multiple applications in a distributed and heterogeneous application landscape in parallel to obtain a consistent view on all available information.

The SoKNOS[1] system [1] is a working prototype (see Fig. 1) for such software using semantic technologies for various purposes. In SoKNOS, information sources and services are annotated with ontologies for improving the provision of the right information at the right time. The annotations are used for connecting existing systems and databases to the SoKNOS system, and for creating visualizations of the information. Furthermore, the users' actions are constantly supervised, and errors are avoided by employing ontology-based consistency checking.

A central design decision for the system was to ensure that any newly created information as well as all integrated sensor information is semantically characterized, supporting the goal of a shared and semantically unambiguous information basis across organizations. In this sense, semantic technologies were used in a holistic and pervasive manner thought the system, making SoKNOS a good example for the successful application of semantic technologies. Fig. 2 shows an overview of the ontologies developed and used in the SoKNOS project.

[1] More information on the publicly funded SoKNOS project can be found at `http://www.soknos.de`

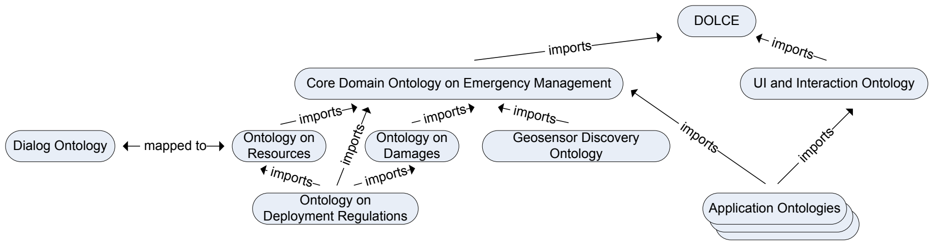

Fig. 2. Ontologies developed and used in SoKNOS.

The central ontology is a core domain ontology on emergency management, aligned to the foundational ontology DOLCE [2], which defines the basic vocabulary of the emergency management domain. Specialized ontologies are used for resources and damages, and deployment regulations defining the relations between resources and damages. Those ontologies have been developed in a close cooperation with domain experts, such as fire brigade officers.

Furthermore, for the definition of system components, ontologies of user interfaces and interactions as well as geo sensors have been developed. Specialized application ontologies can be defined for each application used in the disaster scenario, based on the aforementioned ontologies. For supporting speech based interaction for finding resources, a specialized dialog ontology has been developed, which has a mapping to the resources ontology.

The remaining paper is structured as follows. In section 2, we first give an overview on six use cases where we applied ontologies and semantic technology in the SoKNOS support system. Second, we then detail for each of these use cases the implemented functionality and its benefits for end users (e.g., the command staff in disaster management) and software engineers. As this is a survey on the different usages of semantics in SoKNOS, we only briefly highlight the benefits of applying ontologies, and we refer to other research papers for details on the implementation and evaluation where applicable.

In section 3, we present the lessons learned of the software engineering process. We point out successes and potential for improvements of the semantic technologies employed. Finally, we conclude in section 4 with a summary of the presented use cases and lessons learned.

2 Use Cases and Ontology-based Improvements in the SoKNOS Disaster Management Application

Six core use cases for ontologies and semantic technologies in disaster management turned out to be of particular interest during the SoKNOS project. We have classified the use cases according to two criteria. First, the use cases provide functionality applicable either during design time (before or after an incident) or during run time (during the incident, including the chaos phase). Second, the functionality is either used by the end users (firefighter, emergency management

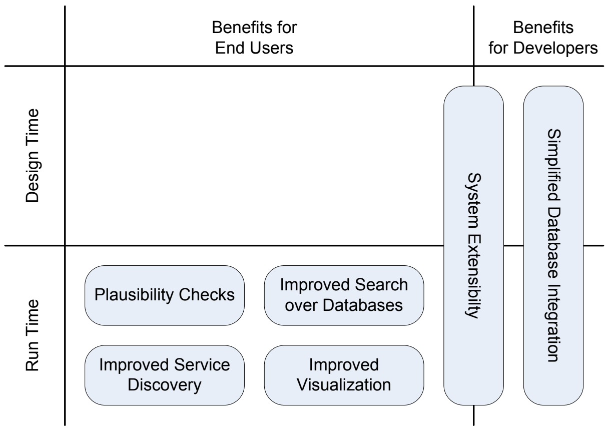

Fig. 3. Six use cases for semantic technologies covered in the SoKNOS project.

staff, etc.) or by software engineers. Fig. 3 shows an overview on these use cases identified in SoKNOS, classified according to the two criteria.

Each of the use cases presented in this paper has been implemented in the SoKNOS support system [3]. In this section, we illustrate the benefits of ontologies for the end users in each of these use cases. Furthermore, we explain how this functionality has been implemented using ontologies and semantic technology.

2.1 Use Case 1: System Extensibility

In disaster management, users typically encounter a heterogeneous landscape of IT systems, where each application exposes different user interfaces and interaction paradigms, hindering end users in efficient problem resolution and decision making. For efficiently dealing with emergency situations, *systems have to be extensible* so that applications can be added or integrated with little effort. The SoKNOS prototype features *ontology-based system extensibility* that allows the integration of new system components with comparatively little effort.

For saving as much of the engineering efforts as possible, we integrate applications on the user interface layer, thus being able to re-use as many software components as possible, and confronting the end users with familiar user interfaces. From a software engineering point of view, integration on the user interface level can be tedious, especially when heterogeneous components (e.g., Java and Flex components) are involved. From a usability point of view, we enable the end user to make use of the various existing applications which serve different purposes, such as managing resources, handling messages, or displaying digital maps – some of them may be domain-specific, such as resource handling, others may be domain-independent, such as messaging or geographic information software.

In SoKNOS, we have developed a framework capable of dynamic integration of applications on the user interface level based on ontologies. For the integration, each application is described by an application ontology, based on the SoKNOS set of ontologies (see Fig. 2). Integration rules using the concepts defined in those application ontologies express interactions that are possible between the integrated applications. Integrated applications communicate using events annotated with concepts from the ontologies. A reasoner evaluating those event annotations based on the ontologies and integration rules serves as an indirection between the integrated applications, thus preventing code tangling and preserving modularity and maintainability of the overall system. Details on the integration framework can be found in [4].

Interactions between integrated applications encompass the highlighting of related information in all applications when an object is selected, or the triggering of actions by dragging and dropping objects from one application to another. Moreover, the user is assisted, e.g., by highlighting possible drop locations when dragging an object. This allows users to explore the interaction possibilities of the integrated system. Figure 1 depicts some of the integrated applications in SoKNOS, together with example interactions.

A central innovation is that all components can expect certain events to come with some well-known annotations since all events exchanged between components are annotated using the ontologies from the SoKNOS ontology stack. Therefore, the *integration rules* used to control cross-application interactions can be defined based on those annotations and react to all events based on those annotations. By mapping annotations of events to methods of the integrated components, new components can therefore be added at run-time without having to change the system, as long as the annotations of the events remain constant, which in turn is ensured by using shared ontologies. For adding a new application to SoKNOS, an appropriate application ontology has to be written, and the mapping from events to methods has to be defined.

Although a reasoner is involved in processing the events exchanged between applications, a sophisticated software architecture of the framework ensures that the event processing times are below one second, thus, the user experience is not negatively affected by the use of semantics. Details on the architecture and the performance evaluation can be found in [5].

2.2 Use Case 2: Simplified Database Integration

Numerous cooperating organizations require the integration of their heterogeneous, distributed databases at run time to conduct efficient operational resource management, each using their own vocabulary and schema for naming and describing things. As discussed above, having the relevant information at the right place is essential in disaster management. Such information is often stored in databases. One typical example in SoKNOS are databases of operational resources (e.g. fire brigade cars, helicopters, etc.), maintained by different organizations, such as local fire brigades. Since larger incidents require ad hoc cooperation of such organizations, it is necessary that such databases can be integrated even at run-time.

In the SoKNOS prototype, we have used OntoBroker as an infrastructure for integrating databases. As discussed in [6], OntoBroker can make instance data stored in relational databases or accessed via Web service interfaces available as facts in the ontology. The SoKNOS "Joint Query Engine" (JQE) uses that infrastructure to connect different heterogeneous resource databases in the disaster management domain owned by different organizations.

A simple graphical user interface allows the user to pose queries against the different databases. The JQE processes the query by unifying different names of input-concepts like, e.g. "helicopter", defined in the SoKNOS resources ontology, which is mapped to the different underlying databases' data models. The query is then translated to a number of queries to the connected databases, and the results are unified and returned to the querying application. In [5], we have shown that directly passing the queries to underlying data sources is faster than materializing the facts from the databases in the reasoner's A-box.

For more sophisticated use cases, such as plausibility checking (see section 2.5), the JQE also provides an interface for reasoning on the connected data sources. The query for the integrated reasoning engine must be formulated in frame-logic (F-Logic), a higher order language for reasoning about objects [7]. The user interfaces, however, abstract from the query language and provide simple, graphical access.

For establishing the mappings between the resources ontology and the different databases' data models, SoKNOS provides a user interface for interactively connecting elements from the data model to the resources ontology. We have enhanced the interface described in [6] by adding the SAP AutoMappingCore [8], which makes suggestions for possible mappings based on different ontology and schema matching metrics. Internally, the mappings created by the user are converted to F-Logic rules which call database wrappers. Thus, a unified interface to heterogeneous databases is created using semantic technologies.

2.3 Use Case 3: Improved Search

As discussed in the previous section, one of the challenges in disaster management is to quickly and reliably find suitable and available operational resources to handle the operation at hand, even under stressful conditions. The challenge in SoKNOS was to combine a spoken dialog system and the Joint Query Engine (JQE), described in section 2.2, in a multilevel process in order to arrive at a more intuitive semantic search. This approach enables domain experts to pose a single query by speech that retrieves semantically correct results from all previously integrated databases.

Domain experts formulate queries using flexible everyday vocabulary and a large set of possible formulations referring to the disaster domain. Natural spoken interaction allows a skillful linguistic concatenation of keywords and phrases to express filtering conditions which leads on one hand to more detailed queries with more accurate results and on the other hand shortens the conventional query interaction process itself. For example, the spoken query "Show me all available helicopters of the fire fighters Berlin" may result in the display of the two relevant available helicopters in Berlin along with an acoustical feedback "Two available helicopters of the fire fighters Berlin were found". The ontology-based search approach developed to that end improves conventional search by mouse and keyboard interactions through the addition of spoken utterances. Implementation details on the dialog system can be found in [9].

The core components of the spoken dialog system are a speech recognizer and a speech interpretation module. The recognized speech utterances are forwarded to the speech interpretation module, which decomposes the speech recognizer result into several sub-queries. The speech interpretation module relies on an internal ontology-based data representation, the so called dialog ontology, which defines domain knowledge and provides the basic vocabulary that is required for the retrieval of information from natural language. Based on this dialog ontology, the speech interpretation module can resolve ambiguities and interpret incomplete queries by expanding the input to complete queries using the situational context. For processing the spoken queries with the JQE, the dialog ontology is mapped to the SoKNOS resources ontology, as shown in Fig. 2.

The spoken dialog system translates the natural language query into combined F-Logic expressions which the JQE processes to deliver search results (see above). According to the task of giving an incident command, the development of the dialog system was focusing on rapid and specific response to spoken domain-specific user input, rather than on flexible input allowing for a broad common vocabulary.

Simple measurements for the project work, carried out on a standard desktop PC, have shown that the complete processing times of all integrated parsing and discourse processing modules are in the range of milliseconds. This examination shows that the processing of speech input will take place within the dialog platform in real time. However, with increasing complexity of the knowledge domain, the vocabulary and thus the complexity of the generated grammar also increase, which in turn affects the runtime of the developed module. Detailed examinations of the influence of the complexity of the knowledge domain on the runtime behavior will be a subject of future research.

2.4 Use Case 4: Improved Discovery of External Sensor Observation Services

Semantically correct, hence meaningful integration of measurements, for example a system of water level sensors in a river, is especially problematic in situations with high time pressure and low familiarity with a (sub) domain and its terminology. In such cases, the crisis team member might be in need for additional information and is just missing the appropriate search term. The SoKNOS support system addresses this need by providing an ontology-based service discovery mechanism.

The crisis team member benefits from this functionality by being able to integrate specific sensor information quickly, e.g., the water level of a river or the concentration of a pollutant.

In our approach, Web services designed according to the SOS[2] specification, are semantically annotated. To this end, we have developed a geo sensor discovery ontology which formalizes both observable properties (for example wind speed, substance concentration etc.) and the feature of interest (e.g., a particular river, a lake or a city district). The annotation is performed by extending the standard service description with URLs pointing to the respective categories in the ontology.

To facilitate discovery, we have established a way to determine the observable property of interest, based on the ontology. The crisis team member specifies a substance or geographic object (e.g., river) to which the observable properties may pertain (e.g., water level, or stream velocity). The latter are then determined through the relation between an observable property and its bearer, as formalized in the ontology. To get sensor data, the end users finally specify the area of interest by marking this area on a map, provided by a separate module in the SoKNOS System, and by specifying the desired time interval. Details of the implementation can be found in [10].

2.5 Use Case 5: Plausibility Checks

In an emergency situation, the stress level in the command control room is high. Therefore, the risk of making mistakes increases over time. Mistakes in operating the system can cause severe problems, e.g., when issuing inappropriate, unintended orders. Therefore, it is important to double check the users' actions for adequacy and consistency, e.g. by *performing automatic plausibility checks*. Missing plausibility checks in disaster management solutions further increase stress and hence errors on end user side. For example, the system checks the plausibility of an assignment that a crisis team member issues and warns the user if the assignment of tactical unit (e.g., a fire brigade truck) to a planned task (e.g., evacuating a building) does not appear plausible.

Plausibility checks have been considered very useful by the end users, however, they do not want to be "over-ruled" by an application. Therefore, it is important to leave an open door for doing things differently – the system should therefore warn the user, but not *forbid* any actions explicitly. As emergencies are per definition unforeseeable, the system has to provide means for taking unforeseeable actions instead of preventing them.

While such checks may also be hard-coded in the software, it can be beneficial to perform them based on an ontology, such as proposed in [11]. From an engineering point of view, delegating consistency checking to an ontology reasoner reduces code tangling, as consistency checking code may be scattered way across an application. Furthermore, having all statements about consistency in one ontology eases maintainability and involvement of the end users.

In SoKNOS, the ontology on deployment regulations contains the information about which operational resource is suitable for which task. Based on this

[2] The Sensor Observation Service Interface Standard (SOS) is specified by the Sensor Web Enablement (SWE) initiative of the Open Geospatial Consortium (OGC); `http://www.opengeospatial.org`

ontology, a reasoner may check whether the assigned unit is suitable for a task or not. The domain knowledge involved in this decision can be rather complex; it may, for example, include information about the devices carried by a tactical unit, the people operating the unit, the problem that is addressed by the task, and so on.

The implementation is quite straight forward: when the user performs an assignment of a resource to a task, an F-Logic query is generated and passed to the JQE (see above), which asks whether the resource is suitable for the respective task. Based on the knowledge formalized in the ontology, the reasoner answers the query. The processing time of the query is below one second, so the user can be warned instantly, if required, and is not interrupted in her work.

2.6 Use Case 6: Improved Information Visualization

In a typical IT system landscape, information is contained in different IT system. Finding and aggregating the information needed for a certain purpose is often a time consuming task.

In SoKNOS, we have used the semantic annotations that are present for each information object for creating an *ontology-based visualization* of the data contained in the different systems. Each IT system integrated in SoKNOS has to offer its data in a semantically annotated format, comparable to Linked Data [12]. Those different annotated data sets can be merged into a data set comprising the information contained in all the systems.

Based on that merged data set, an interactive graph view is generated by the *Semantic Data Explorer (SDE)*. The user can interact with the SDE and the existing applications' interfaces in parallel, thus allowing a hybrid view for data exploration[3]. User studies have shown that for complex information finding tasks in the domain of emergency management, such as finding resources that are overbooked, the Semantic Data Explorer leads to significant improvements both in task completion time and in user satisfaction. Details on the architecture and the user study can be found in [13].

3 Lessons Learned – Disaster Management Applications and Ontologies

In the three years of research and development within SoKNOS, we have implemented the use cases sketched in this paper, employing semantic technologies in numerous places. We present our lessons learned from that work in three parts: the ontology engineering process, the software engineering process, and the usage and suitability of ontologies in the disaster management domain.

[3] A demo video of the Semantic Data Explorer can be found at `http://www.soknos.de/index.php?id=470`.

3.1 Ontology Engineering Process

Involving end users. The early and continuous involvement of end users actually working in the disaster management domain was a core success factor.

On the one hand, discussions with end users to *gather and verify the domain understanding* are at the core of both the ontology engineering process and the application development. In our case, we built the ontology and applications based on multiple workshops with the German firefighting departments of Cologne and Berlin. Additionally, using real-life documents such as working rules and regulations and user generated knowledge sources such as Wikipedia supported the re-construction and formalization of the knowledge to a great extent.

On the other hand, *evaluating the ontologies* with end users helped to consolidate and validate the engineered ontologies. As a result of frequent evaluation workshops, we were able to consolidate the overall ontology and tighten the representation of the domain.

Establishing the role of an ontology engineer. The ontology engineering task as such *usually cannot be executed by end users.* The person taking the role of an ontology engineer needs to work in close cooperation with the end users as well as the application's business logic developer. Knowledge exchange with end users is important to gather a correct and complete understanding of the domain, while software developers need support in understanding the ontology's concepts.

Working with end users, the role requires knowledge in both ontology engineering and the respective domain that is to be formalized. We found that the task of formalizing the domain terminology cannot be done by end users as it requires ontology awareness, for example in the form of taking modeling decisions and obeying to modeling conventions. Without this awareness, no formally correct ontology will emerge, and the ontology cannot be used for reasoning tasks. Working with software developers, the ontology engineer can communicate the usage of the ontology in the application and thus develop an awareness where and how to place semantic annotations.

We had good experiences with an *ontology engineer in a dedicated role* embedded in the project team. The ontology engineer worked with both end users and software developers. This way, we could map the end users' domain knowledge efficiently into the application's business logic.

Finding the right tools. Current tools are rarely designed with end users, i.e., laymen with respect to ontological modeling, in mind. In fact, ontology editors are especially weak when it comes to complex ontologies [14].

The complex terminology modeling involved in ontology engineering is hard to comprehend for domain experts in case of existing modeling artifacts. We experienced domain experts in the disaster management as not been trained in any modeling environment.

Ontology editors need improvement in their "browsing mechanisms, help systems and visualization metaphors" [14], a statement from 2005 which unfortunately still holds true. Better ontology visualization helps to quickly gain an understanding of the ontology and better browsing mechanisms helps editors get better suited to modeling laymen and domain-knowledgeable end users.

Details on the ontology engineering process in SoKNOS can be found in [15].

3.2 Software Engineering Process and Ontologies

Developing new mechanisms for semantic annotations. In SoKNOS, we have relied on semantic annotation of all data within a SoKNOS system. To this end, data models and ontologies need to be interrelated, so that each data object instance can be semantically annotated. Based on these annotations, various useful extensions to the system, as sketched in section 2, have been implemented.

Current approaches for interrelating class models and ontologies most often assume that a 1:1 mapping between the class model and the ontology exists. With these approaches, the mapping is *static*, i.e. each class is mapped to exactly one ontological category, and each attribute is mapped exactly to one ontological relation. Moreover, most annotation approaches are implemented in an *intrusive* fashion, which means that the class model has to be altered in order to hook it up with the annotation mechanism, e.g., by adding special attributes and/or methods to classes. In SoKNOS, we have learned that both these premises – *static* annotation and *intrusive* implementation – are not practical in a real world software engineering setting.

The goals of ontologies and class models are different: class models aim at simplification and easy programming, while ontologies aim at complete and correct formal representations. Thus, we cannot assume that a 1:1 mapping always exists, and static approaches are likely to fail. A typical example from SoKNOS is the use of one common Java class for predicted as well as for actual problems, distinguished by a flag. While this is comfortable for the programmer, the class cannot be statically mapped to a category in the ontology, because predicted and actual problems are very different ontological categories.

Semantic annotations of data objects must feature a non-intrusive implementation for non-alterable legacy code or binary packages where source code and class model cannot be altered. Class models often come as binary packages or are created by code generators which do not allow changes to the way the classes are generated, or licenses of integrated third-party components forbid altering the underlying class models. In these cases, intrusive implementations for semantic annotation are bound to fail.

As a consequence, we have developed a novel approach for semantic annotation of data objects in SoKNOS. This approach is dynamic, i.e. it allows for computing the ontological category a data object belongs to at run-time, and it is implemented in a non-intrusive way: The rules for annotating objects are stored separately from the class models, and an annotation engine executes those rules by inspecting the objects to annotate, using mechanisms such as Java reflection. Thus, class models which cannot or must not be altered can be dealt with [16].

Addressing performance. An essential requirement for user interfaces in general is high reactivity. On the other hand, introducing a reasoning step in the event processing mechanism is a very costly operation in terms of run time. As reaction time of two seconds for interactions is stated as an upper limit for usability in the HCI literature, high reactivity of the system is a paramount challenge.

We have evaluated different architectural alternatives with respect to run time performance and scalability to larger integrated systems, such as centralized vs. decentralized event processing, or pushing vs. pulling of dynamic instance data into the reasoner's A-box (assertion component). With our optimized system design, we are able to keep the reaction times on a reasonable level, even for a larger number of integrated applications used in parallel, with a relatively high number of integration rules (where each integration rule controls one type of cross-application interaction). Details on the evaluation can be found in [5].

3.3 Ontology Usage and Suitability

Finding the right modeling granularity. Both domain experts and end users face problems in dealing with concepts needed due to ontology formalisms. We observed that end users were irritated by the concepts and methods imposed by the use of a strictly formal top level ontology such as DOLCE, which were not part of their colloquial language, but needed for formally correct modeling. We found two sources of "problematic" concepts where domain experts and end users had problems with.

First, *domain experts were not used to concepts needed to create a formally correct ontology*, e.g., as induced by using top level ontologies. Using reference ontologies, like in our case the DOLCE top level ontology, requires complying with a certain structure and formalism. For example, the SoKNOS ontologies are based on DOLCE which uses concepts like "endurant" and "perdurant" to cater for semantic interoperability between information sources. However, from a domain expert's point-of-view, this terminology is complicated to understand compared to normal, colloquial language usage [17]. In our case, professional firefighters as the domain experts were irritated by these concepts.

Second, *end users were irritated by modeled domain terminology that was not part of their colloquial language.* There are concepts that firefighters don't use in their colloquial language but which are needed for a formally correct modeling of the ontology, e.g., to satisfy reasoning requirements. For example, resources are categorized in the ontology via their usage characteristics. In the given example, the class "rescue helicopter" is sub-class of the classes "equipment", "means of transportation", "motorized means of transportation", "flying motorized means of transportation", "helicopter" and "ground-landing helicopter". Except for the term "equipment", all other terms are not part of a firefighter's colloquial language but are needed have a formally correct ontology and support useful reasoning.

In SoKNOS, we have found that this question cannot be answered trivially, as a heuristic for identifying an optimal proportion between "every day concepts" and "top level concepts" in the ontology is missing. Having a dedicated ontology engineer, as discussed above, involved in the ontology engineering session helped the domain experts understand the need and the intention of top level concepts.

Finding the right visualization depth. Offering only a class browser with a treelike visualization of the ontology's extensive OWL class hierarchy caused confusion among end users. The SoKNOS inventory management application visualized the modeled OWL class hierarchy directly in a class browser. Here, the end user for example can browse resources like cars, trucks and aircrafts. However, due to the numerous concepts and the extensive class hierarchy, the user actions of selecting and expanding nodes were often too complicated for the end user. In case that the end user doesn't know exactly where in the class hierarchy the desired concept is located, browsing involves a high error-rate in the exploration process when the explored classes and sub-classes do not contain the concept the end user looks for. As shown in the example above, the concept "rescue helicopter" has six upper classes. An end user needs to select and expand in this example six times the right node to finally select the concept "rescue helicopter" as resource in this class browser due to the direct OWL class hierarchy visualization. In sum, we found the simple browser visualization of an OWL class hierarchy as not sufficient for an end user interface.

In SoKNOS, we have addressed this challenge by hiding top level categories in the user interface. Only concepts defined in the core domain ontology are used in the user interface (but are still available to the reasoner); the foundational categories from DOLCE are not. Thus, the end user only works with concepts from her own domain. As a further extension, immediate categories that do not provide additional value for the end user, such as "motorized means of transportation", can be suppressed in the visualization.

Finding the right visualization. Various ways of visualizing ontologies and annotated data exist [18]. In the SoKNOS Semantic Data Explorer discussed above, we have used a straight forward graph view, which, like the standard OWL visualization, uses ellipses for instances and rectangles for data values. The user studies have shown that even that simple, straight forward solution provides a large benefit for the end user. Thus, slightly modifying Jim Hendler's famous quote [19], we can state that *a little visualization goes a long way.*

4 Conclusion

In this paper, we have introduced the SoKNOS system, a functional prototype for an integrated emergency management system which makes use of ontologies and semantic technologies for various purposes.

In SoKNOS, ontologies have been used both at design-time and at run-time of the system. Ontologies are used for providing a mutual understanding between developers and end users as well as between end users from different organizations. By annotating information objects and data sources, information retrieval, the discovery of relevant Web services and the integration of different databases containing necessary information, are simplified and partly automated. Furthermore, ontologies are used for improving the interaction with the system by facilitating user actions across application borders, and by providing plausibility checks for avoiding mistakes due to stressful situations.

During the course of the project, we have employed ontologies and semantic technologies in various settings, and derived several key lessons learned. First, the ontology engineering process necessarily should involve end users from the very beginning and foresee the role of dedicated ontology engineers, since ontology engineering is a non-trivial task which is significantly different from software engineering, so it cannot be simply overtaken by a software engineer. Tool support is currently not sufficient for letting untrained users build a useful ontology.

Second, current semantic annotation mechanisms for class models are not suitable. Those mechanisms are most often intrusive and require a 1:1 mapping between the class model and the ontology. When dealing with legacy code, both assumptions are unrealistic. Thus, different mechanisms for semantically annotating class models are needed. Furthermore, relying on a programming model backed by an ontology and using reasoning at run-time imposes significant challenges to reactivity and performance.

Third, it is not trivial to find an appropriate modeling and visualization depth for ontologies. While a large modeling depth is useful for some tasks, the feedback from the end users targeted at the need for simpler visualizations. In SoKNOS, we have addressed that need by reducing the complexity of the visualization, and by providing a straight forward, but very useful graphical visualization of the annotated data.

In summary, we have shown a number of use cases which demonstrate how the employment of ontologies and semantic technologies can make emergency management systems more useful and versatile. The lessons learned can also be transferred to projects with similar requirements in other domains.

Acknowledgements

The work presented in this paper has been partly funded by the German Federal Ministry of Education and Research under grant no. 01ISO7009.

References

1. Paulheim, H., Döweling, S., Tso-Sutter, K., Probst, F., Ziegert, T.: Improving Usability of Integrated Emergency Response Systems: The SoKNOS Approach. In: Proceedings "39. Jahrestagung der Gesellschaft für Informatik e.V. (GI) - Informatik 2009". Volume 154 of LNI. (2009) 1435–1449
2. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: WonderWeb Deliverable D18 – Ontology Library (final) (2003) `http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf`. Accessed August 2nd, 2010.
3. Döweling, S., Probst, F., Ziegert, T., Manske, K.: SoKNOS - An Interactive Visual Emergency Management Framework. In Amicis, R.D., Stojanovic, R., Conti, G., eds.: GeoSpatial Visual Analytics. NATO Science for Peace and Security Series C: Environmental Security, Springer (2009) 251–262
4. Paulheim, H., Probst, F.: Application Integration on the User Interface Level: an Ontology-Based Approach. Data & Knowledge Engineering Journal **69**(11) (2010) 1103–1116

5. Paulheim, H.: Efficient Semantic Event Processing: Lessons Learned in User Interface Integration. In Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T., eds.: The Semantic Web: Research and Applications (ESWC 2010), Part II. Volume 6089 of LNCS., Springer (2010) 60–74
6. Angele, J., Erdmann, M., Wenke, D.: Ontology-Based Knowledge Management in Automotive Engineering Scenarios. In Hepp, M., Leenheer, P.D., Moor, A.D., Sure, Y., eds.: Ontology Management. Volume 7 of Semantic Web and Beyond. Springer (2008) 245–264
7. Angele, J., Lausen, G.: Ontologies in F-Logic. In Staab, S., Studer, R., eds.: Handbook on Ontologies. International Handbooks on Information Systems. 2nd edition edn. Springer (2009) 45–70
8. Voigt, K., Ivanov, P., Rummler, A.: MatchBox: Combined Meta-model Matching for Semi-automatic Mapping Generation. In: Proceedings of the 2010 ACM Symposium on Applied Computing, New York, NY, USA, ACM (2010) 2281–2288
9. Sonntag, D., Deru, M., Bergweiler, S.: Design and Implementation of Combined Mobile and Touchscreen-based Multimodal Web 3.0 Interfaces. In Arabnia, H.R., de la Fuente, D., Olivas, J.A., eds.: Proceedings of the 2009 International Conference on Artificial Intelligence (ICAI 2009), CSREA Press (2009) 974–979
10. Babitski, G., Bergweiler, S., Hoffmann, J., Schön, D., Stasch, C., Walkowski, A.C.: Ontology-Based Integration of Sensor Web Services in Disaster Management. In: Proceedings of the 3rd International Conference on GeoSpatial Semantics. GeoS '09, Berlin, Heidelberg, Springer (2009) 103–121
11. Liu, B., Chen, H., He, W.: Deriving User Interface from Ontologies: A Model-Based Approach. In: ICTAI '05: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence, Washington, DC, USA, IEEE Computer Society (2005) 254–259
12. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems **5**(3) (2009) 1–22
13. Paulheim, H., Meyer, L.: Ontology-based Information Visualization in Integrated UIs. In: Proceedings of the 2011 International Conference on Intelligent User Interfaces (IUI), ACM (2011) 451–452
14. García-Barriocanal, E., Sicilia, M.A., Sánchez-Alonso, S.: Usability evaluation of ontology editors. Knowledge Organization **32**(1) (2005) 1–9
15. Babitski, G., Probst, F., Hoffmann, J., Oberle, D.: Ontology Design for Information Integration in Catastrophy Management. In: Proceedings of the 4th International Workshop on Applications of Semantic Technologies (AST'09). (2009)
16. Paulheim, H., Plendl, R., Probst, F., Oberle, D.: Mapping Pragmatic Class Models to Reference Ontologies. In: 2nd International Workshop on Data Engineering meets the Semantic Web (DESWeb). (2011)
17. Hepp, M.: Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies. IEEE Internet Computing **11**(1) (2007) 90–96
18. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.G.: Ontology Visualization Methods - A Survey. ACM Comput. Surv. **39**(4) (2007)
19. Hendler, J.: On Beyond Ontology. `http://iswc2003.semanticweb.org/hendler_files/v3_document.htm` (2003) Invited Talk at the International Semantic Web Conference 2003.