Ontology-Enhanced User Interfaces: A Survey

Heiko Paulheim and Florian Probst

SAP Research Center Darmstadt

{heiko.paulheim,f.probst}@sap.com

Abstract. Ontologies have been increasingly used in software systems in the past years. In many of those systems, however, the ontologies are hidden “under the hood”. While a lot of useful applications of ontologies on the database and business logic layer have been proposed, the employment of ontologies in user interfaces has been gaining comparatively little attention so far. For providing a deeper understanding of that field as well as assisting developers of ontology-enhanced user interfaces, we give an overview of such applications and introduce a schema for characterizing the requirements of ontology-enhanced user interfaces. With this article, we present a state of the art survey of approaches and identify promising research directions.

This is a draft version of the article “Ontology-Enhanced User Interfaces: A Survey”, which has appeared in the International Journal on Semantic Web and Information Systems (IJSWIS), 6(2), 2010.

Ontology-Enhanced User Interfaces: A Survey

Heiko Paulheim and Florian Probst
SAP Research Center Darmstadt

Abstract

Ontologies have been increasingly used in software systems in the past years. In many of those systems, however, the ontologies are hidden "under the hood". While a lot of useful applications of ontologies on the database and business logic layer have been proposed, the employment of ontologies in user interfaces has been gaining comparatively little attention so far. For providing a deeper understanding of that field as well as assisting developers of ontology-enhanced user interfaces, we give an overview of such applications and introduce a schema for characterizing the requirements of ontology-enhanced user interfaces. With this article, we present a state of the art survey of approaches and identify promising research directions.

Introduction

During the past years, ontologies have been used in information sources for numerous purposes, such as annotating resources for better information retrieval, integrating data from different sources and systems, and automatically coupling intelligent agents. In most of those fields, ontologies are used on the information source and the business logic layer, and thus hidden "under the hood".

One of the most complete surveys of using ontologies in software engineering is probably given by Ruiz and Hilera (2006). The authors have analyzed more than 50 possibilities of employing ontologies in software engineering, only two of which target at user interfaces. Heitmann, Kinsella, Hayes, and Decker (2009) have presented a survey on applications using semantic web technology and, not much surprisingly, they found out that more than 90% of those applications come with a user interface. The survey, however, contains only little information about *how* the employment of semantic web ontologies influences the provided user interfaces.

In this article, we want to shed light at this area and take a closer look at the various possibilities of enhancing user interfaces with ontologies. We have reviewed various projects and identified a number of purposes for which ontologies can be used on the user interface layer, e.g., adapting UIs to a user's needs, or providing input assistance. Each of those purposes poses particular requirements to the ontologies and their use in the application. So far, no structured review of those requirements and approaches has been performed.

To summarize these approaches, we prefer the more general notion *ontology-enhanced user interface* instead of *ontology-driven user interface*, as sometimes used (e.g., Paton et al. (1999); Visser and Schuster (2002)), since ontologies may also be employed to provide one single functionality in a larger user interface (and thus *enhance* the user interface) without being the key element *driving* the user interface. We propose the following definition:

Definition. *An* ontology-enhanced user interface *is a user interface whose visualization capabilities, interaction possibilities, or development process are enabled or (at least) improved by the employment of one or more ontologies.*

According to this definition, we have looked at projects where the development process and/or the usability of a user interface has been improved by employing ontologies. Applications such as pure ontology editors or viewers thus are out of scope here, since the ontologies do not improve the user interface in these cases – in these applications, ontology engineering itself is the *purpose*, not a *means to enhance* the user interface's capabilities.

While this definition is quite broad, it does not encompass every application using an ontology. There are many applications that use ontologies internally – e.g., for integrating different information sources, or for enabling information exchange with other systems – where the fact that an ontology is used in a particular place does *not* have any effect on the application's user interface. Furthermore, there are applications providing functionality which are implemented with ontologies, and the applications' user interfaces grant access to that functionality – however, in these cases, the user interface as such is not directly influenced (let alone improved) by the employment of an ontology. In contrast, we concentrate on applications of ontologies that *directly* improve user interfaces or their development.

We have carefully studied the current state of the art of improving user interfaces with ontologies. To that end, we have looked at numerous projects which use ontologies in the development of user interfaces. From that overview, we have derived a number of criteria for both the ontologies and the mode of their employment which are relevant for characterizing ontology-enhanced user interfaces. This characterization serves two purposes: it allows for a better understanding ontology-enhanced user interfaces, and it supports developers who want to use ontologies for a certain purpose in a user interface by pinpointing the relevant requirements. Furthermore, the survey in this article helps identifying new interesting research directions. The survey of approaches we present is purely descriptive, as a detailed discussion of whether each of the improvements addressed could also or even better be achieved without ontologies is out of scope of this article.

The rest of this article is structured as follows. The next section outlines existing classifications of ontologies in software systems in general. Based on these classifications, we present our own characterization framework, which is especially tailored to ontology-enhanced user interfaces. Next, we present a representative selection of approaches and apply our framework to discuss their characteristics. Following our definition, these approaches are classified in three categories: *improving the user interface's appearance*, *improving the interaction with the user interface*, and *improving the development process of the user interface*. We conclude with a summary and a discussion in which we point to possible future trends by identifying research areas within ontology-enhanced user interfaces that are currently underrepresented.

Existing Classifications of Ontologies in Software Systems

Although no classification of ontology-enhanced user interfaces and of the use of ontologies in user interfaces has been proposed so far, a number of classifications for using ontologies in software systems and in software engineering *in general* exist, which may to a certain degree be applied to the area of ontology-enhanced user interfaces.

An often-cited classification schema (Heijst, Schreiber, & Wielinga, 1997; Gómez-Pérez, Fernández-López, & Corcho, 2004) distinguishes the domain modeled in an ontology and the ontology's complexity. It distinguishes four types of domains (domain, application, representation, and generic ontologies) which vary in their level of detail and reusability within a domain and across domains, and three levels of complexity (lexicons, information ontologies, and knowledge modelling ontologies).

Guarino (1998) proposes a classification for using ontologies in information systems. This classification distinguishes three levels of generality (top level, domain/task level, and application level), two types of usage time (development and run time), and three system layers in which the ontologies are used (database, application logic, and user interface). While there are some examples for using ontologies in user interfaces, such as directly browsing an ontology for a certain domain or improving textual input, no further detailed analysis is performed.

Happel and Seedorf (2006) introduce a classification schema more targeted at using ontologies in software engineering, which is a blend of the two schemas above. The authors distinguish two types of domains: the real world domain for which the software is built (e.g., banking, travel, etc., confusingly called *software* in their work), and the domain of software systems (called *infrastructure* in their work). In addition, they distinguish between using the ontology at development time and and run time of the software system.

Gruninger and Lee (2002) identify three relevant categories for the usage of ontologies in computer systems in general. Ontologies may be used for *communication* (between people, between systems, and between people and systems.), for *inter-operability* (between systems), and for *improving the development process.* This is one of the most comprehensive descriptions of what ontologies can be used for in computer systems (not necessarily only software systems). Improving the user interface is covered by communication between people and systems. Approaches for improving the development of user interfaces can be subsumed in the last category.

Uschold and Jasper (1999) give a more detailed classification, using criteria such as the role of the ontology (used for operational data (L_0), for classification of that data (L_1), or as a language for defining other ontologies (L_2)), the actors working with it (ontology author, data author, application developer, application user, and knowledge worker), the supporting technology (i.e., languages, programming frameworks, etc.) and the maturity level of the ontology (from experimental to commercially used). The approaches in this article most often deal with L_0 and L_1 ontologies. The ontology users are implicitly reflected in our classification of approaches: improving the appearance and interaction of user interfaces target at application users, improving UI development targets at application developers.

The already mentioned survey by Ruiz and Hilera (2006) distinguishes various cases of using ontologies in software engineering, based on the traditional phases of the software engineering process. For each phase, they list different possibilities of how ontologies may be employed, and suggest ontologies that may be used for that purpose.

All of those classifications contain valuable criteria to characterize the use of ontologies in user interfaces. However, as they do not concentrate on user interfaces in particular, they are often too general for that specific area. In the following section, we propose a set of criteria suitable for analyzing ontology-enhanced user interfaces.

Characteristics of Ontology-Enhanced User Interfaces

As discussed, there are various classifications of ontologies and their employment, each following a certain purpose and thus having a particular bias. We adopt some of the criteria proposed in those works and enhance them with criteria which are useful for categorizing user interfaces.

Since the development of user interfaces is a sub-area of software engineering, we follow Happel and Seedorf (2006) and take the ontologies' domain and the time of its employment into account. Furthermore, we adopt the ontology complexity criterion from Heijst et al. (1997) and Gómez-Pérez et al. (2004), as this is a useful criterion for tackling the requirements when it comes to decide for an ontology language and a matching programming framework.

We augment these three criteria by two additional ones that target specifically at user interfaces, based on the fact that user interfaces usually serve two main purposes: a) presenting information to the user, and b) allowing interaction with a system. For ontologies in user interfaces, we therefore propose the following two additional criteria: a) how the ontology is presented to the user, and b) how the user can interact with the ontology.

Thus, we end up at a characterization schema comprising five criteria, as depicted in Fig. 1. In the rest of this section, we discuss each of those criteria in detail.

Criterion 1: What Domain is the Ontology About?

The first criterion regards the ontology's domain, i.e., what the ontology is about. Happel and Seedorf (2006) distinguish two types: the real world domain for which the software is build, and the domain of software systems. We introduce a third domain, namely users and the roles they have (Kagal, Finin, & Joshi, 2003). Roles are most often roles a person takes in the real world. However, roles may also affect the way that person can interact with a system (e.g., whether that person has certain rights within a system) or is supposed to perform certain tasks. Thus, they cannot be clearly assigned to any of the two domains introduced by Happel and Seedorf. Therefore, we propose users and roles as a domain of its own, ending up with the following three domains:

Real world. The ontology characterizes a part of the real world, typically the one that the application is used in (e.g., banking, travel, etc.). The goal is to identify the central concepts and their relations.

IT system. The IT system itself is formalized in the ontology. Such an ontology may contain categories such as SOFTWARE MODULE, WEB SERVICE, and so on. Application system ontologies can be further divided in hardware and software ontologies.

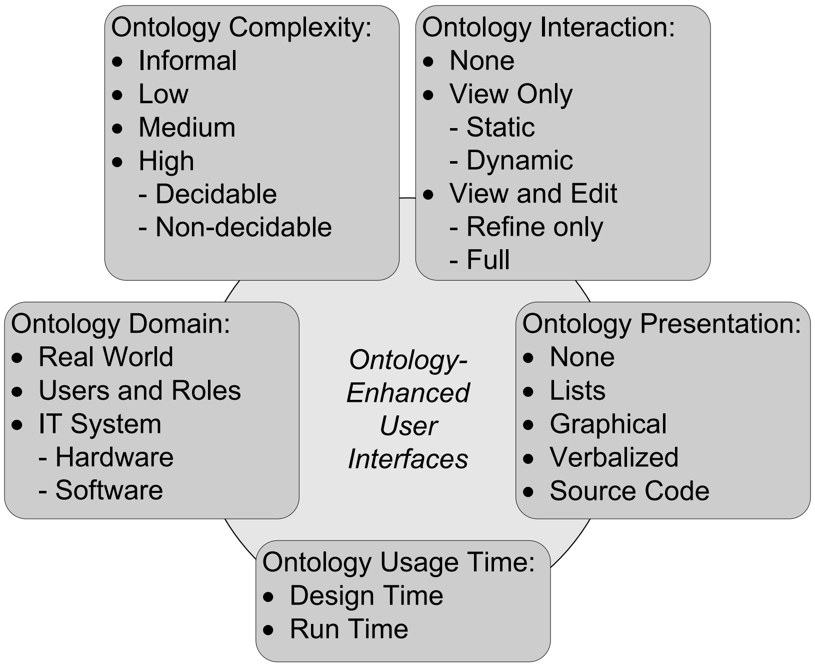

Figure 1. Characterization schema for ontology-enhanced user interfaces

Users and roles. The ontology characterizes users, their preferences, their roles, and/or the rights and possibilities they have in using a system.

Ontologies covering more than one of those domains can be used in one system, and it is technically possible to mix those different domains in the same ontology. Nevertheless, it is useful to separate them into different ontology modules even if they are needed in the same system. Especially system and real world domain ontologies should be separated, both for reasons of separation of concerns and of flexibility when reusing the ontologies (Klien & Probst, 2005).

Criterion 2: How Complex is the Ontology?

Ontologies come in various complexities, starting from simple glossaries up to sets of logical statements and constraints (Smith & Welty, 2001; Uschold & Gruninger, 2004; Rebstock, Fengel, & Paulheim, 2008). For each degree of complexity, different languages may be used. For our purpose, we distinguish the following degrees of complexity:

Informal. Informal ontologies are collections of definitions which have no further structure, such as a glossary. Such ontologies do not require any sophisticated language, they can be expressed, e.g., in XML, XML Schema, or RDF.

Low. Ontologies of low complexity consist of class hierarchies and subclass relations. Such ontologies are typically expressed in RDF-S or OWL Lite.

Medium. Ontologies of medium complexity also contain relations other than the subclass relation. Like for ontologies with low complexity, RDF-S and OWL Lite can be used.

High. Highly formal ontologies are further augmented with constraints, rules, and so on. Typically, languages such as OWL DL and OWL Full are used for defining ontologies with constraints. Rules can be expressed, e.g., by combining OWL with SWRL, or by using F-Logic.

The first three categories are often referred to as "lightweight ontologies", the latter as "heavyweight ontologies" (Gómez-Pérez et al., 2004). Since reasoning plays an important role in many ontology-based systems, it is beneficial to further classify highly formal ontologies into decidable and non-decidable ontologies (Antoniou, Franconia, & van Harmelen, 2005).

Criterion 3: When is the Ontology Used?

When developing user interfaces, ontologies may be used at different points of time in the development process. Following Happel and Seedorf (2006), we distinguish the following two:

Design time. Ontologies are used while developing the system.

Run time. Ontologies are used when the system is executed.

In the first case, the ontologies typically are not a part of the user interface which is developed, and they are used to assist the developer. In the second case, the ontologies most often become a part of the user interface and fulfil a certain function in that user interface.

Criterion 4: How is the Ontology Presented to the User?

In ontology-enhanced user interfaces, the ontologies used and the information contained therein may be, at least partially, presented to the user. Therefore, the question *how* the ontology is presented is particularly interesting.

No presentation. The ontology is completely hidden.

Lists. Lists of categories from the ontology are shown without making relations between those categories visible. Those lists may be actual selection lists as well as the names of single concepts from the ontology appearing as text blocks in some places of the user interface.

Graphical. Relations between concepts are visualized, e.g., in form of trees (most often showing the taxonomy) or graphs (including non-taxonomic relations). A detailed survey of ontology visualization techniques is given by Katifori, Halatsis, Lepouras, Vassilakis, and Giannopoulou (2007), encompassing various graphical representations in 2D and 3D.

Verbalized. A textual representation of the axioms contained in the ontology is provided.

Source code. The ontology's source code, e.g., in OWL or F-Logic, is shown to the user.

At first glance, it may be questionable whether we are still talking about ontology-enhanced user interfaces if those ontologies are completely hidden, or whether the ontologies are rather used in the business logic or data layer when they are not visible in the user interface. However, we will show some examples in the next chapter where hidden ontologies can directly influence the user interface and thus conform to our definition.

Criterion 5: How can the User Interact with the Ontology?

There may be different types of interaction with the ontologies used in an ontology-enhanced user interface. In some cases, the ontologies may only be viewed, in others, the user can also extend and alter them. We distinguish three types of interaction with ontologies:

No interaction. The user cannot interact with the ontology.

View only. The user can view the ontology or selected parts thereof. The view can be static (e.g., in form of pictures) or dynamic (i.e., allowing to browse between concepts, zoom in and out, etc.)

View and edit. The user can modify the contents of the ontology. Such modifications can either be limited to refining the ontology (i.e., adding new subcategories or relations), or allow full modification, such as changing and deleting existing concepts.

In most cases where ontologies may be altered, this functionality is only accessible by a selected group of users, such as administrators.

In summary, the five criteria in the characterization schema presented in this section are mostly independent from each other, although some cross-dependencies exist. For example, graphical visualization requires at least a taxonomy of categories, and extending and altering ontologies is only possible when ontologies are not invisible, at least for the group of users who can perform these tasks.

Other Criteria

Besides the criteria above, there is a number of others which could be used, but which we do not consider relevant for characterizing ontology-enhanced user interfaces for the following reasons:

Storage. Ontologies can be stored centrally or distributed, locally or on the web, in flat files or triple stores. This is an engineering decision which does not affect the user interface.

Free floating vs. grounded. Ontologies are often grounded in foundational ontologies such as DOLCE (Masolo, Borgo, Gangemi, Guarino, & Oltramari, 2003). This allows engineers to better understand those ontologies and compare different ontologies. For the end user, this distinction is often not particularly relevant.

Modularity. Likewise, ontologies can be built as a single large ontology or as a number of interconnected small ontologies. Although we strongly encourage the use of modular ontologies, this is merely an engineering decision without any direct impact on the user interface.

Size. Ontologies can significantly vary in size, ranging from a dozen categories to a few million. As we have found no examples where a certain minimum or maximum size is required for a certain application, we have decided to neglect this criterion as well. However, size is an important issue when it comes to the performance and scalability of an ontology-based application.

Survey of Purposes and Approaches

Following our initial definition, there are three main purposes for which ontologies can be employed in ontology-enhanced user interfaces, namely

1. improving the visualization capabilities,
2. improving the interaction possibilities, and
3. improving the development process

of a user interface. Based on this categorization, we will give an overview on approaches for pursuing these purposes and present some *representative examples* for each approach.

Each of those approaches is analyzed and characterized based on the schema introduced in the previous chapter, and the relevant requirements for implementing the approaches are pointed out. A table at the end of the chapter will compare and summarize the approaches and give hints for the identification of interesting research gaps.

Using Ontologies for Improving the Visualization Capabilities

Ontologies may be used for improving the appearance of the user interface, i.e., the way information is presented to the user. This means that given a set of information items, the transformation into visual items on the screen is influenced by ontologies. The determination of the initial set of information items, however, is not a task performed in the user interface, but in the underlying business logic. Therefore, approaches such as ontology-based information filtering are out of scope here, unless combined with an ontology-enhanced user interface.

Information Clustering. When retrieving information, it is possible that a large number of information items is returned. Therefore, the user may want some assistance in handling the query results. One possible approach is *information clustering*: Here, information items are subsumed to groups which stand for a set of single items. Each of these groups is represented by one visual item on the screen (Herman, Melançon, & Marshall, 2000).

One example for ontologies-based information clustering is the *Cluster Map* project (Fluit, Sabou, & Harmelen, 2003). In Cluster Map, search results (e.g., from a web or database search) are visualized as connected sets (painted as balloons), which contain small dots representing the individual results. The user can navigate the ontology and get an idea of the number of results in each category. The *Courseware Watchdog* project (Tane, Schmitz, & Stumme, 2004), an e-learning resource management system, uses ontologies to provide an intuitive way to access e-learning resources. Here, taxonomic and non-taxonomic relations between classes can be displayed as well, allowing the user a flexible way of navigating through the set of resources. In the *SWAPit* project (Seeling & Becks, 2003), clusters of documents are computed from the metadata assigned to them, and the documents belonging to different categories in a domain ontology are marked in different colours in those clusters.

In each of those cases, ontologies help finding resources by creating new visualizations apart from simple lists.

Ontologies can be used for information clustering at run-time to provide the groups and their labels for information clustering. By subsuming each information item to a class in the ontology's class hierarchy, those items can be clustered, and the classes are visualized on the screen. Thus, the ontology is characterizing the real world domain that the information items refer to, and it has to consist of at least a class hierarchy (although additional relations may be visualized as extra guidance). Both lists and graphical visualizations are possible, where in the latter case, the user may also navigate through the visualization interactively.

Characterization of examined information clustering approaches	
Ontology domain:	Real World
Ontology complexity:	Low or higher
Ontology visualization:	Lists or graphs
Interaction with the ontology:	View only
Ontology usage time:	Run time

Text Generation. Information contained in ontologies is usually encoded in machine-readable languages that only experts can understand. When trying to make that information available to the user, it can be beneficial to transform it into a language that can be understood by the end user. To that end, different ontology verbalizing algorithms have been developed (Kaljurand & Fuchs, 2007).

An example for using verbalized ontologies is the *MIAKT* project (Bontcheva & Wilks, 2004): in this project, information about medical cases gathered from an information base in the form of RDF triples is turned into human-readable reports. The RDF triples and the definitions from the corresponding ontology are verbalized and turned into a small text describing a medical case.

There are other examples where not full text, but only text fragments are generated from information contained in ontologies. One promising strategy is to use texts that already exist and augment them with additional information taken from ontologies. In the *COHSE* project (Carr, Hall, Bechhofer, & Goble, 2001), text documents are augmented with links to related documents and categories as lists of links added to the texts. Here, ontologies defining the documents' domain of discourse can help identifying the correct anchor terms in the document, and ontology-based information retrieval may be one step in finding relevant documents to link to the anchor terms. Depending on whether the documents are directly linked to the anchor terms or link lists based on categories from the ontologies are generated, the ontologies are either invisible or visualized as lists. One appealing vision of applying this approach is to have a large information base, such as Wikipedia, interlinked and enhanced by using ontologies (Völkel, Krötzsch, Vrandecic, Haller, & Studer, 2006). In this scenario, the user can even enhance and alter the ontology in a collaborative setting.

For verbalizing knowledge with ontologies, those ontologies have to be at least fairly complex, otherwise, no reasonably interesting text can be produced (only short sentences such as "a cat is an animal" can be generated from less complex ontologies). Reasoning over more complex ontologies can also be useful to provide better natural language representation, since statements which are not explicitly contained in the ontology can also be added to the text (Carr et al., 2001). Text generation may be performed both at design-time (in cases where a set of documents is compiled prior to using the system) as well as run-time (in cases where reports etc. are produced on demand). Although editing verbalized ontologies in collaborative settings is possible, most approaches only foresee viewing ontologies.

Characterization of examined text generation approaches	
Ontology domain:	Real world
Ontology complexity:	Medium or higher
Ontology visualization:	Verbalized
Interaction with the ontology:	View, edit is possible
Ontology usage time:	Design and run time

Adapation of User Interface Appearance. Apart from the information displayed in a user interface, the appearance of the parts of a user interface itself may also be directed by ontologies. Most commonly, the user interface is adapted to the user's needs. If both the user interface elements and the user's profile are defined by using highly formal ontologies, a reasoner can determine the appearance which fits the user's needs best and thus personalize the user interface.

One particularly interesting application is the adaptation of user interfaces for users with special needs, such as in the *Semantic Life* project (Karim & Tjoa, 2006) or the W3C's *WAI ARIA* initiative (W3C, 2009). Such adaptation includes adjusting font size, colours (for partly colour-blind people), and the organization of contents on the screen. In the Semantic Life project, the ontology models users' impairment and appropriate visualizations, i.e., it covers parts of the user as well as the system domain, so a reasoner can determine appropriate visualizations. In WAI ARIA, only the system parts are modeled in an ontolgy, and the selection of appropriate visualizations is left up to an agent (which is not specified any further). In both approaches, the underlying ontologies are evaluated at run-time for adapting the user interface, but they are not visible to the user.

Ontologies may not only be used for software adaptation, but also for *hardware adaptation*. One example is the *Context Studio* application developed by Nokia (Korpipää, Häkkilä, Kela, Ronkainen, & Känsälä, 2004). Here, the user may define rules how mobile devices are supposed to provide information to the user in given situations (e.g., "if the display is turned down, do not ring aloud"). To this end, an ontology of the mobile phone and context situations is defined, which can be utilized by the user to create rules on how they want their mobile devices to behave (and thus extend the underyling ontology). The ontology is presented to the user when defining rules in the form of selection lists (although graphical assistance in defining those rules would also be possible), but is invisible outside the rule definition mode. In the case of mobile phones, the user's context may additionally be acquired, e.g., from GPS sensors of the mobile device, thus allowing adaptation of mobile applications based on the user's spatial context (Lee, 2010).

Another approach for hardware adaptation is shown in the *MaDoE* project (Larsson, Ingmarsson, & Sun, 2007): here, user interfaces are distributed across different hardware devices. An ontology of devices and their capabilities is combined with a set of rules to make sure that each user interface component is displayed by a hardware component that has the required capabilities.

In the examples above, the rules for adapting user interfaces have been actively defined by the developer or the user, either at design or at run-time. Another approach is observing the user and learning those rules from the user's behavior during run-time, or off-line from log files. Such an approach is described by Schmidt, Dörflinger, Rahmani, Sahbi, and Thomas (2008). Here, learned rules, based on ontologies characterizing both the IT system and the real world domain, are used to optimize the information presentation in a portal. Information items that are more likely to be relevant for the user are given more prominent positions on the screen than others.

Another adaptation mechanism is the selection of visualization modules (software widgets, algorithms, or even hardware devices) that are suitable for a given information. This requires an ontology of those modules (i.e., the system domain software and/or hardware) in a degree of formality which allows reasoning to answer the queries for appropriate visualization modules. Such an approach is shown by (Potter & Wright, 2006), where input and output devices are selected at run time given the current information and visualization needs.

Ontologies for adapting user interfaces can be used both at design and at run time, depending on whether the user interfaces is supposed to adapt dynamically, or if pre-adapted user interfaces are to be shipped. It may take place in the background, with the ontologies completely hidden from the user, or visibly. The user may also be granted the possibility to express their own rules for adaptation. Since the reasoning employed for adaptation is fairly complex, the ontologies also have to be highly expressive. Information about the real world, the IT system to adapt, and the user may be included for adaptation.

Characterization of examined approaches to adapation of UI appearance	
Ontology domain:	Real world, IT system and user
Ontology complexity:	High
Ontology visualization:	None, lists, or graphs
Interaction with the ontology:	View or extend
Ontology usage time:	Design and run time

Using Ontologies for Improving the Interaction Possibilities

Besides visualizing data, the second important purpose of a user interface is to provide an access point to a system's functionalities. To this end, the user has to interact with the system by entering data, selecting items, issuing commands, etc. There are several approaches for employing ontologies to improve the interaction with IT systems.

Ontology-based Browsing. While we concentrated on information visualization as a run-once task so far (i.e., a given set of information items is visualized in an optimal form), the user usually interacts with these visualizations by browsing from one information item to another. Ontologies can be used to improve this browsing process. Hyvönen, Styrman, and Saarela (2002) use the term "semantic browsing"; however, we prefer the term "ontology-based browsing" since any kind of browsing information always requires a semantic understanding of that information (which, on the other hand, does not need to be formalized in an ontology).

One example of a semantic browser is *Ozone* (Burel, Cano, & Lanfranchi, 2009). Ozone uses RDF embedded in web pages to provide so-called “semantic overlays”, i.e., textual and/or graphical information provided on demand when selecting an information item. For example, city names in texts annotated with concepts from an ontology defining their geographical position can be displayed on a map when selected.

Embedded RDF, however, is currently rather rare on the web. The browser *Magpie* (Dzbor, 2008) uses overlays like Ozone, but does not rely on embedded RDF. Instead, HTML pages are pre-processed, and the annotations are generated automatically. This is an approach also followed by *PiggyBank* (Huynh, Mazzocchi, & Karger, 2005): here, so-called “screen scrapers” automatically transform web page contents into RDF. This RDF can then be used to provide additional information on the web page’s contents. Piggy Bank also allows the user to store and share the RDF information gathered from scraping the web.

Another popular approach for ontology-based browsing is *faceted browsing*. Here, data (such as product information, or websites) is annotated with ontologies. The user can then select from different lists of ontological categories and orthogonal property values to view filtered lists of relevant information (Heim, Ziegler, & Lohmann, 2008). While the basic faceted browsing approaches only use list-based views, there are some interesting extensions. The *SearchPoint* web search interface presents a graph of topics (which can be filled from an ontology) and their relations, and the user can click on any point in the graph to view the search results which belong to the topics which are displayed closest to the selected point (Pajntar & Grobelnik, 2008). The faceted browsing approach introduced by Hussein and Münter (2010) allows for defining arbitrary as well as domain-specific interaction elements (called “facet decorators”) for different facet types, e.g., sliders for numeric values, maps for geographic values, etc.

Ontology-based browsing cannot be performed on the web alone. The *Semantic Desktop* (Sauermann, Bernardi, & Dengel, 2005; Cheyer, Park, & Giuli, 2005) allows for browsing data contained in different applications on a single desktop computer. For example, there may be papers from authors (stored as text files), some of the authors may also be email contacts (stored in an email client program), and their web pages are bookmarked in a standard web browser. Semantic Desktop software links this data and explicitly presents these links to the user, so that the user can navigate from one resource to another, even if they are stored in different programs.

When resources are annotated with ontologies, an ontology-based browser can point to relevant other resources. Unlike finding related documents by measuring text similarities or access correlations, the user can also be informed about *how* the documents are related. Ontologies of low complexity (in the simplest case topic lists) about the documents’ real world domain are sufficient for providing ontology-based browsing, but more sophisticated browsing functionality requires at least medium complexity. Visualization can range from invisible ontologies (in case related documents are proposed with help of the ontology, but without showing the ontology itself) to lists of concepts and more sophisticated graphical visualization.

Characterization of examined ontology-based browsing approaches	
Ontology domain:	Real world
Ontology complexity:	Informal, higher complexity possible
Ontology visualization:	None, lists, or graphs
Interaction with the ontology:	View only
Ontology usage time:	Run time

User Input Assistance. In most kinds of interfaces, the user has to enter data into the system at a certain point, either in the form of textual input or by selecting values from predefined lists. In a form-based user interface, not every combination of input options is feasible. An ontology formalizing knowledge about the domain of the objects whose data is entered in a user interface may be employed to provide plausibility checking on the user's input.

It may be argued that plausibility checking is a functionality of the business logic, not of the user interface. Nevertheless, when being applied *during* the input process and not *afterwards*, plausibility checking can help *reducing the complexity of input forms.* Thus, plausibility checking can improve the usability of a user interface.

An example is introduced by Liu, Chen, and He (2005). In a flower shop application, the customer has to step through an input wizard consisting of different forms. If a male person is selected as the recipient of a bouquet, then options such as "mother's day greeting card" are not included in subsequent selection lists. Here, an ontology reasoner can decide that a male person cannot be a mother and will therefore not receive a mother's day greeting card. Another example for a real world setting is e-government, where ontologies can be used to ease input in a large amount of forms, as shown in the *SeGoF* project (Stadlhofer & Salhofer, 2008). Another setting where filtering selection lists is useful are user interfaces for configuring complex products, as demonstrated in the *CAWICOMS* project (Ardissono et al., 2002).

Feasible input options can also be extracted from context information at run-time. In the semantic e-mail client *Semanta* (Scerri, Davis, Handschuh, & Hauswirth, 2009), for example, the content of an e-mail is extracted at run-time (e.g., it contains a meeting request), and possible answers (e.g., accept or decline the request) are determined by a reasoner and shown to the user in a selection list for automatically generating an answer mail. Here, the complex input task of writing an e-mail is simplified and reduced to selecting a template and filling the gaps.

In addition to supporting the user in selecting from pre-defined values, ontologies may also be employed *improving textual input.* In the case of searching for information, as shown, for example, in *OntoSeek*, the number of search results can be increased by not only searching for documents containing one term, but also for those containing a term related to the one the user entered, where those related terms are defined in an ontology (Guarino, Masolo, & Vetere, 1999).

Another way of improving textual input is the provision of *autocomplete functionality*, where the user types the beginning of a word, and the system proposes completion based on terms defined in an ontology (Hildebrand & Ossenbruggen, 2009). This ensures that the user only enters terms that the system understands and follows the idea of using ontologies as a shared vocabulary (Gruber, 1995). In this scenario, the ontology again contains domain knowledge, but it requires only little formalization, since no reasoning is applied, and it is invisible for the user.

In many ontology-based systems, large knowledge bases are created, e.g., in the field of life sciences (Mendes, McKnight, Sheth, & Kissinger, 2008). While querying those knowledge bases is very useful, it is a difficult task for the end user who is not familiar with querying languages such as SPARQL. Therefore, assistance in *query construction* is a desirable feature in those applications. This approach requires an ontology which models the real world domain the knowledge base is about, and its complexity can range from informal (when only single query terms are selected) to high (when complex queries are issued). In most cases, lists of concepts are shown to the user, although more sophisticated visual interfaces are possible (Spahn, Kleb, Grimm, & Scheidl, 2008).

In the simplest case, query construction can be done by selecting terms from a list. More sophisticated query construction interfaces let the user specify additional properties and relations of the selected concept. For example, the user selects the concept "employee" and then restricts the query by adding the relation "worksFor" and providing a value for the relation, thus querying for all users working for a given company, as shown in the projects *QuizRDF* (Davies, Weeks, & Krohn, 2004) and *TAMBIS* (Paton et al., 1999). A similar mechanism is used in the online news portal *PlanetOnto* (Domingue & Motta, 1999): here, the user may also store their queries and get notifications when there are new results. Other systems such as *SEWASIE* try to construct the first cut of the query from natural language input and allow further graphical refinement of that query (Catarci et al., 2004).

The ontologies are usually not visually presented to the user. As it is typically information from the real world domain (e.g., customer data) which is entered into a form, the ontology used for plausibility checking at run-time has to model that domain in a highly formal way to enable reasoners to decide whether an input combination is plausible. While plausibility checking may also take place in the background without any visualization, ontologies are often presented in the form of lists (e.g. selection lists in forms) or graphs, as in the case of query construction support.

Characterization of examined user input assistance approaches	
Ontology domain:	Real world
Ontology complexity:	Informal, higher complexity possible
Ontology visualization:	None, lists, or graphs
Interaction with the ontology:	View only
Ontology usage time:	Run time

Providing Help. More general approaches of user assistance provide help for the user, especially in form of help texts, which can range from tool tips of a few words to large, interlinked help documents. Examples are projects such as *SACHS* and *CPoint* (Kohlhase & Kohlhase, 2009), where ontologies are used to generate user assistance. That process is called "Semantic Transparency" by the authors. User assistance *making the system transparent* can be created in the form automatically generated help documents, small tooltips displayed on demand (Paulheim, 2010), and graphical visualizations of how the system works.

Gribova (2007) presents an approach which accounts for the user's context. Besides the system and its real world domain, the ontologies have to model the user's possible tasks as well. By observing the user's activities and the system's state, the system determines the user's current task. Based on this information, context-sensitive help can be provided on demand, thus freeing the user from searching the relevant help text sections themselves.

Approaches for providing help for the user employ ontologies characterizing both the system itself as well as parts of its real world domain and present them to the user in verbalized form. As discussed above for text generation, the provision of useful help texts requires at least a medium level of formality and can be done at design-time or at run-time.

Characterization of examined approaches to providing help	
Ontology domain:	Real world, IT system, and user
Ontology complexity:	Medium or higher
Ontology visualization:	Verbalized or graphical
Interaction with the ontology:	View only
Ontology usage time:	Design or run time

User Interface Integration. When performing a complex task, the user often does not only use one application, but several ones in parallel. This generates additional workload for the user for coordinating work with the different applications, such as finding related information stored in different applications, or copying and pasting data from one application to another.

To provide *unified views on data* stored in different applications, portals or mash-ups displaying different applications as *portlets* or *mashlets* at the same time can be used. A few projects exist in which ontologies are used in building web portals. The approach described by Dettborn, König-Ries, and Welsch (2008) uses semantic web services to retrieve related data in different applications unified in one portal. Thus, applications can display data which is relevant according to the data displayed in other portlets. A similar approach for mashup development is shown by Ankolekar, Krötzsch, Tran, and Vrandecic (2007).

Integrated user interfaces can ease those tasks by *facilitating cross-application interactions*, i.e., interactions that span more than one application, such as highlighting related data in different applications or dragging objects from on application to the other. Thus, the integrated applications are extended with new interaction possibilities. Existing solutions for integration on the user interface layer, however, suffer from significant shortcomings when trying to implement such interactions (Daniel et al., 2007).

While ontologies have been used to integrate applications on the database and business logic layer (Doan & Halevy, 2005; W3C, 2004, 2005), using ontologies for user interface integration is a new and widely unexplored field. Díaz, Iturrioz, and Irastorza (2005) discuss an approach which allows pre-filling form fields in different applications based on data from other applications. The input and output data are annotated by using ontologies defining the real world domain, and based on those annotations, data pipes between the different forms are defined. A more flexible approach facilitating integration at run time is shown by

Paulheim (2009). Here, not only the real world data, but also the systems to be integrated are defined in ontologies. A reasoner can be used at run-time to determine the possible cross-application interactions and facilitating event exchange between the applications. In this approach, the ontology itself is not visible for the user, but used as an underlying mechanism to enhance the user interface's functionality.

In all of those approaches, the ontologies are evaluated at run-time for coordinating the behavior of integrated user interfaces. The ontology is not visible to the user, in fact, the user may not even know that the integration is based on ontologies. Both the technical components as well as the real world information they process have to be formally characterized in an ontology to automate the coordination of user interface components.

Characterization of examined user interface integration approaches	
Ontology domain:	Real world and IT system
Ontology complexity:	Informal, higher complexity possible
Ontology visualization:	None
Interaction with the ontology:	None
Ontology usage time:	Run time

Using Ontologies for Improving the Development of User Interfaces

Ontologies can be used at different stages in the software development process; starting from requirements engineering and ending up with maintenance (Hesse, 2005; Calero, Ruiz, & Piattini, 2006). This also holds for user interface development. Whereas in most of the approaches shown above, the ontologies are used at the system's run time for supporting the user, the following approaches most often employ ontologies at a system's design time, and the end user of the system does not see the ontologies, nor interact with them.

Identifying and Tracking Requirements. The task of *identifying the requirements* of a software system includes modelling the real world domain to a certain level. In the case of a form-based user interface, for example, it is necessary to define the real-world concepts for which data is entered, the ranges of their relations, etc. The approach described by Furtado et al. (2002) how such ontological models can be used in user interface development. Such an approach helps decoupling domain knowledge and user interface code and allows reuse of the domain ontology when developing different user interfaces for related applications in the same domain.

After a user interface has been developed from the requirements originally identified, those requirements may change, e.g., due to adaptation of business processes. In that case, parts of the system, including the user interface, have to be rewritten, and it is not always trivial to *track the requirements* and identify those parts. The approach presented by Sousa (2009) uses ontologies for modelling both business processes and the user interfaces. Based on these ontologies, rules describing the dependencies between elements in the user interfaces and in the business processes can be defined. In case of business process changes, those rules are used to find the spots where the user interfaces need to be adapted.

Adaptations of a user interface may not only be triggered by changes in business processes, but also by changes in technical requirements, e.g., support of additional platforms.

Such *system migrations* require decisions on which components and widgets of a platform to use for a given component. Modelling the system as well as the target platforms in formal ontologies can help assisting developers with those decisions (Moore, Rugaber, & Seaver, 1994) and thus improve the migration process.

In all the cases discussed, reasoning is employed to assist the developer, therefore, formal ontologies are required. To model requirements, both the real world and the IT system have to be taken into account.

Characterization of examined approaches to identifying and tracking requirements	
Ontology domain:	Real world and IT system
Ontology complexity:	High
Ontology visualization:	None
Interaction with the ontology:	None
Ontology usage time:	Design time

Generating User Interfaces. An area which has been increasingly gaining attention is the automatic generation of user interfaces from ontologies. This can be seen as a special case of model driven architectures (MDA), where software is generated from models (not necessarily ontological models).

Similar to the problem of system migration described above, platform independent models in MDAs can also be used in conjunction with rules defining the mappings of UI components to specific widgets and components on different target platforms (Calvary et al., 2003). Similarly, combining the system ontology with ontologies of users and user preferences, personalized versions of user interfaces can be created, like shown in the *OntoWeaver* project (Lei, Motta, & Domingue, 2003).

For being able to automatically derive a software system from an ontology, this ontology has to model the system as well as parts of its the real-world domain. Although reasoning is not necessary, the ontology has to be complex enough to model the system on a reasonable level of detail (Liu et al., 2005; Sergevich & Viktorovna, 2003).

Characterization of examined UI generation approaches	
Ontology domain:	Real world and IT system
Ontology complexity:	Medium or higher
Ontology visualization:	None
Interaction with the ontology:	None
Ontology usage time:	Design time

Reusing User Interface Components. The development of a system's user interfaces consumes about 50% of the overall development efforts (Myers & Rosson, 1992). Thus, reusing existing user interface components is especially desirable.

Reusing software components requires *finding suitable components* as a first step. Similar to defining and retrieving web services in the field of semantic web services, those components are annotated with ontologies describing their functional and non-functional properties. Base on these annotations, developers can find components suitable for their specific problems based on more precise queries then text-based search. One example for a component repository (not specifically UI components) implementing these ideas is *KOntoR* (Happel, Korthaus, Seedorf, & Tomczyk, 2006).

Once user interface components have been retrieved, they have to be integrated into one coherent user interface. This requires communication between those components and a language for composing those components (Daniel et al., 2007). Ontologies can serve these purposes by modelling the components as well as the operations that can be performed with them and the data objects they use. Based on these ontologies, integration rules can be defined, which, when interpreted at run-time by a reasoner, implement an integrated user interface. Such rules may either be triggered by events (Paulheim, 2009), or evaluated when assembling a user interface at run-time, e.g., a complex web page built from different components (Pohjalainen, 2010).

User interface components are not restricted to software components. Besides standard input and output components (keyboard, mouse, screen), more specialized and advanced devices have recently been developed. From a more general point of view, selecting suitable components for developing a user interface therefore also includes choosing the right hardware devices. There are ontologies which can be used to define hardware devices, such as the *GLOSS ontology* (Coutaz, Lachenal, & Dupuy-Chessa, 2003) and the *FIPA device ontology* (Foundation for Intelligent Phyiscal Agents, 2002). With the help of those ontologies, the process of *selecting suitable hardware devices* can be supported efficiently as a larger number of highly specialized devices becomes available.

Similarly to components, design practices and patterns can also be stored in annotated databases and retrieved by developers. Ontology-based pattern languages have been proposed to formalize those patterns (Gaffar, Javahery, Seffah, & Sinnig, 2003). With the help of such ontologies, developers can query a knowledge base of patterns in order to find patterns suitable for their problems and find additional information such as related patterns, patterns that can be combined, etc. (Henninger, Keshk, & Kinworthy, 2003).

To provide a model of a user interface component, both the component itself as well as the real world information it processes has to be modeled in the ontology. For finding components in a repository, a hierarchy of concepts is sufficient, although more complex ontologies can be employed to specify more concise queries and retrieve more accurate results.

Characterization of examined approaches to reusing UI components	
Ontology domain:	Real world and IT system
Ontology complexity:	Low or higher
Ontology visualization:	None
Interaction with the ontology:	None
Ontology usage time:	Design time

Summary of Survey Results

Table 1 summarizes the approaches discussed and their characterization in our classification scheme. In the following section, we will discuss some of the findings from this survey in more detail.

Table 1: Summary of approaches. Legend: Ontology Domain (W=Real World, S=IT System, U=Users & Roles), Ontology Formality (I=Informal, L=Low, M=Medium, H=High, + indicates a minimum level of formality required), Ontology Visualization (N=None, L=Lists, G=Graphical, V=Verbalized), Interaction with the ontology (N=None, V=View only, E=Edit), Usage Time (D=Design Time, R=Run Time)

.

Purpose	**Approach**	**Domain**	**Complexity**	**Visualization**	**Interaction**	**Usage Time**
Improving Visualization Capabilities	Information Clustering	W	L+	L,G	V	R
	Text Generation	W	M+	V	V,E	D,R
	Adaptation of UI Appearance	W,S,U	H	N,L,G	V,E	D,R
Improving Interaction Possibilities	Ontology-based Browsing	W	I+	N,L,G	V	R
	Input Assistance	W	I+	N,L,G	N,V	R
	Providing Help	W,S,U	M+	V,G	V	R
	UI Integration	W,S	L+	N	N	D,R
Improving Development	Identifying & Tracking Requirements	W,S	H	N	N	D
	Generating UIs	W,S	M+	N	N	D
	Reusing UI Components	W,S	L+	N	N	D

Discussion and Conclusion

With this article, we have given a definition for ontology-enhanced user interfaces and presented a characterization schema for such interfaces and the requirements for their implementation. Those criteria encompass the ontology's domain and formality, the type of visualization of the ontology, the possibility of interaction with the ontology, and the time at which the ontology is employed.

We have given a survey of approaches how ontologies can be used to improve the usability, appearance and development of user interfaces. The variety of approaches shows that ontology-enhanced user interfaces are a vivid, promising research area. Regarding the placement of the approaches in our classification, some interesting gaps can be identified:

Alterable ontologies

Most approaches use static ontologies; approaches where ontologies can be altered or at least extended by the user are rather rare. This finding is also supported by the analysis in Heitmann et al. (2009), which states that authoring interfaces are the least common feature in semantic web applications. The reasons may be two-fold – the developers either see no useful scenario for modifiable ontologies; or they consider the users not to have the necessary experience to edit ontologies. The latter may also be a hint at the need for more intuitive user interfaces for editing and manipulating ontologies, as pointed out by García-Barriocanal, Sicilia, and Sánchez-Alonso (2005).

Advanced visualization

Among those approaches that visualize ontologies, lists are by far the most often used form of visualization. Visualization as source code is only rarely used for providing expert users' interfaces; this is probably due to the fact that in most cases, non-expert (w.r.t. knowledge about ontologies) users are addressed. Although advanced forms of visualization could be applied in many of the approaches discussed in this article, they are used in a relatively small number of actual projects, despite the large variety of techniques available in that field (Katifori et al., 2007). This finding, which is supported by the analysis done by Heitmann et al. (2009), is surprising, because graphical visualizations are considered helpful for the user, and in many of the approaches analysed in this article, the complexity of the ontologies was large enough (more than taxonomies), so that a graphical visualization would have indeed been beneficial. One reason for the small number of approaches using such visualizations could be the developing efforts (compared to, e.g., implementing a list visualization). However, with a growing number of graphical visualization solutions which can be used off-the-shelf, this aspect should gain more attention in the future.

Highly formal ontologies

Regarding ontology formality, many approaches discussed do not use more than a class hierarchy; some also work with a collection of categories without any relations. This means that in these cases, the full power of using ontologies – providing an expressive description of a domain and being able to use reasoning – is often not utilized in user interfaces. It is possible that a larger number of approaches using highly formal ontologies will emerge in the near future, allowing for more sophisticated applications of ontologies in user interfaces.

On the other hand, recent development in the linked open data community shows many applications based on ontologies which are not too rigid and formal (Hausenblas, 2009). Therefore, it is still an open question whether more applications based on rather formal ontologies will emerge, or whether "a little semantics will go a long way" (Hendler, 2003).

Further findings

There are two more remarkable findings. First, many approaches deal with ontologies encompassing more than one domain. Especially the IT system domain and the real world domain are often modeled in ontologies within the same approach. For creating long-living, reusable ontologies, it is important to separate those concerns and model each ontology in a domain of its own (Klien & Probst, 2005). Thus, modular ontologies (Stuckenschmidt, Parent, & Spaccapietra, 2009) are essentially important in ontology-enabled user interfaces.

Second, many of the approaches discussed use ontologies at run-time, often involving reasoning. Since for user interfaces, reaction times are essential to assure good usability, performance marks of reasoners and ontology programming frameworks are important in the area of ontology-enabled user interfaces. Thus, carefully designing software architectures is essential to create software which is accepted by the end user (Paulheim, 2010).

Summary

The characterization schema presented in this article serves two main purposes: first, it helps in designing new solutions by pointing out the relevant requirements to be met by the designer. Second, it allows a clear analysis of the approaches that exist and identifies interesting research directions and promising application areas. There are clear research gaps in using alterable ontologies, advanced visualizations, and exploiting the possibilities of highly formal ontologies in user interfaces. Due to those potentials and the growing impact of ontologies in software development, we are confident that the employment of ontologies in user interfaces will add interesting features and possibilities to future software.

Acknowledgements

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

References

Ankolekar, A., Krötzsch, M., Tran, T., & Vrandecic, D. (2007). The Two Cultures: Mashing Up Web 2.0 and the Semantic Web. In C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, & P. J. Shenoy (Eds.), *WWW* (pp. 825–834). ACM.

Antoniou, G., Franconia, E., & van Harmelen, F. (2005). Introduction to Semantic Web Ontology Languages. In N. Eisinger & J. Maluszynski (Eds.), *Reasoning Web* (Vol. 3564, p. 1-21). Springer.

Ardissono, L., Felfernig, A., Friedrich, G., Jannach, D., Zanker, M., & Schäfer, R. (2002). A Framework for Rapid Development of Advanced Web-based Configurator Applications. In F. van Harmelen (Ed.), *Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002* (p. 618-622). IOS Press.

Bontcheva, K., & Wilks, Y. (2004). Automatic Report Generation from Ontologies: The MIAKT Approach. In *In Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems* (pp. 324–335).

Burel, G., Cano, A. E., & Lanfranchi, V. (2009). Ozone Browser: Augmenting the Web with Semantic Overlays. In C. Bizer, S. Auer, & G. A. Grimnes (Eds.), *5th Workshop on Scripting and Development for the Semantic Web.*

Calero, C., Ruiz, F., & Piattini, M. (Eds.). (2006). *Ontologies for Software Engineering and Software Technology.* Springer.

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., & Vanderdonckt, J. (2003). A Unifying Reference Framework for multi-target user interfaces. *Interacting with Computers*, *15*(3), 289-308.

Carr, L., Hall, W., Bechhofer, S., & Goble, C. (2001). Conceptual linking: ontology-based open hypermedia. In *WWW '01: Proceedings of the 10th international conference on World Wide Web* (pp. 334–342). New York, NY, USA: ACM.

Catarci, T., Dongilli, P., Mascio, T. D., Franconi, E., Santucci, G., & Tessaris, S. (2004). An Ontology Based Visual Tool for Query Formulation Support. In R. L. de Mántaras & L. Saitta (Eds.), *ECAI* (p. 308-312). IOS Press.

Cheyer, A., Park, J., & Giuli, R. (2005). IRIS: Integrate. Relate. Infer. Share. In *Workshop on the Semantic Desktop: Next Generation Personal Information Management and Collaboration Infrastructure.*

Coutaz, J., Lachenal, C., & Dupuy-Chessa, S. (2003). Ontology for Multi-surface Interaction. In *Proceedings of IFIP INTERACT03: Human-Computer Interaction* (p. 447-454). IFIP Technical Committee No 13 on Human-Computer Interaction.

Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., & Saint-Paul, R. (2007). Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing*, *11*(3), 59-66.

Davies, J., Weeks, R., & Krohn, U. (2004). QuizRDF: Search Technology for the Semantic Web. *Hawaii International Conference on System Sciences*, *4*, 40112.

Dettborn, T., König-Ries, B., & Welsch, M. (2008). Using Semantics in Portal Development. In *Proceedings of the 4th International Workshop on Semantic Web Enabled Software Engineering* (p. 109-110).

Díaz, O., Iturrioz, J., & Irastorza, A. (2005). Improving portlet interoperability through deep annotation. In *WWW '05: Proceedings of the 14th international conference on World Wide Web* (pp. 372–381). New York, NY, USA: ACM.

Dix, A., Hussein, T., Lukosch, S., & Ziegler, J. (Eds.). (2010). *Proceedings of the First Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS).*

Doan, A., & Halevy, A. Y. (2005). Semantic Integration Research in the Database Community: A Brief Survey. *AI Magazine*, *26*(1), 83–94.

Domingue, J., & Motta, E. (1999). A Knowledge-Based News Server Supporting Ontology-Driven Story Enrichment and Knowledge Retrieval. In *EKAW '99: Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling and Management* (pp. 103–120). London, UK: Springer-Verlag.

Dzbor, M. (2008). Best of Both: Using Semantic Web Technologies to Enrich User Interaction with the Web and Vice Versa. In V. Geffert, J. Karhumäki, A. Bertoni, B. Preneel, P. Návrat, & M. Bieliková (Eds.), *SOFSEM 2008: Theory and Practice of Computer Science, 34th Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 19-25, 2008, Proceedings* (Vol. 4910, p. 34-49). Springer.

Fluit, C., Sabou, M., & Harmelen, F. V. (2003). Supporting User Tasks through Visualisation of Light-weight Ontologies. In *Handbook on Ontologies in Information Systems* (pp. 415–434). Springer.

Foundation for Intelligent Phyiscal Agents. (2002). *FIPA Device Ontology Specification.* (`http://www.fipa.org/specs/fipa00091/index.html`)

Furtado, E., Furtado, J. J. V., Silva, W. B., Rodrigues, D. W. T., Silva Taddeo, L. da, Limbourg, Q., et al. (2002). An Ontology-Based Method for Universal Design of User Interfaces. In *Task*

Models and Diagrams For User Interface Design (TAMODIA 2002).

Gaffar, A., Javahery, H., Seffah, A., & Sinnig, D. (2003). A Pattern Framework for Eliciting and Delivering UCD Knowledge and Practices. In *Proceedings of the Tenth International Conference on Human-Computer Interaction* (p. 108-112). Lawrence Erlbaum Associates.

García-Barriocanal, E., Sicilia, M. A., & Sánchez-Alonso, S. (2005). Usability evaluation of ontology editors. *Knowledge Organization*, *32*(1), 1-9.

Gómez-Pérez, A., Fernández-López, M., & Corcho, O. (2004). *Ontological Engineering.* Springer.

Gribova, V. (2007). Automatic Generation of Context-Sensitive Help Using a User Interface Project. In V. P. Gladun, K. K. Markov, A. F. Voloshin, & K. M. Ivanova (Eds.), *Proceedings of the 8th International Conference "Knowledge-Dialogue-Solution"* (Vol. 2, p. 417-422).

Gruber, T. R. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies*, *43*(5-6), 907–928.

Gruninger, M., & Lee, J. (2002). Ontology Applications and Design. *Communications of the ACM*, *45*(2), 39–41.

Guarino, N. (1998). *Formal Ontology and Information Systems.* IOS Press.

Guarino, N., Masolo, C., & Vetere, G. (1999). OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems*, *14*(3), 70-80.

Happel, H.-J., Korthaus, A., Seedorf, S., & Tomczyk, P. (2006). KOntoR: An Ontology-enabled Approach to Software Reuse. In K. Zhang, G. Spanoudakis, & G. Visaggio (Eds.), *Proceedings of the Eighteenth International Conference on Software Engineering & Knowledge Engineering (SEKE)* (p. 349-354).

Happel, H.-J., & Seedorf, S. (2006). Applications of Ontologies in Software Engineering. In *Workshop on Semantic Web Enabled Software Engineering (SWESE) on the 5th International Semantic Web Conference (ISWC 2006), Athens, Georgia, November 5-9, 2006.*

Hausenblas, M. (2009). Exploiting Linked Data to Build Web Applications. *IEEE Internet Computing*, *13*, 68-73.

Heijst, G. van, Schreiber, A. T. G., & Wielinga, B. J. (1997). Using explicit ontologies in KBS development. *Int. J. Hum.-Comput. Stud.*, *46*(2-3), 183–292.

Heim, P., Ziegler, J., & Lohmann, S. (2008). gFacet: A Browser for the Web of Data. In S. Auer, S. Dietzold, S. Lohmann, & J. Ziegler (Eds.), *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW'08)* (Vol. 417, p. 49-58).

Heitmann, B., Kinsella, S., Hayes, C., & Decker, S. (2009). Implementing Semantic Web Applications: Reference Architecture and Challenges. In E. F. Kendall, J. Z. Pan, M. Sabbouh, L. Stojanovic, & Y. Zhao (Eds.), (Vol. 524).

Hendler, J. (2003). *On Beyond Ontology.* `http://iswc2003.semanticweb.org/hendler_files/v3_document.htm`. (Invited Talk at the International Semantic Web Conference 2003)

Henninger, S., Keshk, M., & Kinworthy, R. (2003). Capturing and Disseminating Usability Patterns with Semantic Web Technology. In *CHI 2003 Workshop: Concepts and Perspectives on HCI Patterns.*

Herman, I., Melançon, G., & Marshall, M. S. (2000). Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, *6*(1), 24–43.

Hesse, W. (2005). Ontologies in the Software Engineering Process. In R. Lenz, U. Hasenkamp, W. Hasselbring, & M. Reichert (Eds.), *Proceedings of the 2nd GI-Workshop on Enterprise Application Integration (EAI)* (Vol. 141). CEUR-WS.org.

Hildebrand, M., & Ossenbruggen, J. van. (2009, February). Configuring Semantic Web Interfaces by Data Mapping. In *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW2009).*

Hussein, T., & Münter, D. (2010). Automated Generation of Faceted Navigation Interfaces Using Semantic Models. In A. Dix, T. Hussein, S. Lukosch, & J. Ziegler (Eds.), *Proceedings of the*

First Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS).

Huynh, D., Mazzocchi, S., & Karger, D. R. (2005). Piggy Bank: Experience the Semantic Web Inside Your Web Browser. In Y. Gil, E. Motta, V. R. Benjamins, & M. A. Musen (Eds.), *International Semantic Web Conference* (Vol. 3729, p. 413-430). Springer.

Hyvönen, E., Styrman, A., & Saarela, S. (2002). Ontology-Based Image Retrieval. In E. Hyvönen & M. Klemettinen (Eds.), *Towards the semantic Web and Web services. Proceedings of the XML Finland 2002 Conference* (p. 15-27). HIIT Publications.

Kagal, L., Finin, T., & Joshi, A. (2003). A policy language for a pervasive computing environment. In *Proceedings of the IEEE 4th International Workshop on Policies for Distributed Systems and Networks.*

Kaljurand, K., & Fuchs, N. E. (2007). Verbalizing OWL in Attempto Controlled English. In C. Golbreich, A. Kalyanpur, & B. Parsia (Eds.), *Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions, Innsbruck, Austria, June 6-7, 2007* (Vol. 258). CEUR-WS.org.

Karim, S., & Tjoa, A. M. (2006). Towards the Use of Ontologies for Improving User Interaction for People with Special Needs. In K. Miesenberger, J. Klaus, W. L. Zagler, & A. I. Karshmer (Eds.), *ICCHP* (Vol. 4061, p. 77-84). Springer.

Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., & Giannopoulou, E. G. (2007). Ontology visualization methods - a survey. *ACM Comput. Surv.*, *39*(4).

Klien, E., & Probst, F. (2005). Requirements for Geospatial Ontology Engineering. In F. Toppen & M. Painho (Eds.), *8th Conference on Geographic Information Science (AGILE 2005)* (pp. 251–260). Estoril, Portugal.

Kohlhase, A., & Kohlhase, M. (2009). Semantic Transparency in User Assistance Systems. In *Proceedings of the 27th annual ACM international conference on Design of Communication. Special Interest Group on Design of Communication (SIGDOC-09), Bloomingtion,, IN, United States.* ACM Press.

Korpipää, P., Häkkilä, J., Kela, J., Ronkainen, S., & Känsälä, I. (2004). Utilising context ontology in mobile device application personalisation. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia* (pp. 133–140). New York, NY, USA: ACM.

Larsson, A., Ingmarsson, M., & Sun, B. (2007). A Development Platform for Distributed User Interfaces. In *Proceedings of the Nineteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2007), Boston, Massachusetts, USA, July 9-11, 2007* (p. 704-709). Knowledge Systems Institute Graduate School.

Lee, A. (2010). Exploiting Context for Mobile User Experience. In A. Dix, T. Hussein, S. Lukosch, & J. Ziegler (Eds.), *Proceedings of the First Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS).*

Lei, Y., Motta, E., & Domingue, J. (2003). Design of customized web applications with OntoWeaver. In *K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture* (pp. 54–61). New York, NY, USA: ACM.

Liu, B., Chen, H., & He, W. (2005). Deriving User Interface from Ontologies: A Model-Based Approach. In *ICTAI '05: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence* (pp. 254–259). Washington, DC, USA: IEEE Computer Society.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., & Oltramari, A. (2003). *WonderWeb Deliverable D18 – Ontology Library (final)* (Tech. Rep.). Laboratory For Applied Ontology, Trento, Italien. (`http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf`)

Mendes, P. N., McKnight, B., Sheth, A. P., & Kissinger, J. C. (2008). TcruziKB: Enabling Complex Queries for Genomic Data Exploration. In *ICSC '08: Proceedings of the 2008 IEEE International Conference on Semantic Computing* (pp. 432–439). Washington, DC, USA: IEEE Computer Society.

Moore, M. M., Rugaber, S., & Seaver, P. (1994). Knowledge-Based User Interface Migration. In *ICSM '94: Proceedings of the International Conference on Software Maintenance* (pp. 72–79).

Washington, DC, USA: IEEE Computer Society.

Myers, B. A., & Rosson, M. B. (1992). Survey on user interface programming. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 195–202). New York, NY, USA: ACM.

Pajntar, B., & Grobelnik, M. (2008). SearchPoint - a New Paradigm of Web Search. In *WWW 2008 Developers Track.*

Paton, N. W., Stevens, R., Baker, P., Goble, C. A., Bechhofer, S., & Brass, A. (1999). Query processing in the TAMBIS bioinformatics source integration system. In *Eleventh International Conference on Scientific and Statistical Database Management* (p. 138-147).

Paulheim, H. (2009). Ontologies for User Interface Integration. In A. Bernstein et al. (Eds.), *The Semantic Web - ISWC 2009* (Vol. 5823, p. 973-981). Springer.

Paulheim, H. (2010). Efficient Semantic Event Processing: Lessons Learned in User Interface Integration. In L. Aroyo et al. (Eds.), *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 31-June 4, 2010, Proceedings* (Vol. 6089, p. 60-74). Springer.

Pohjalainen, P. (2010). Self-configuring User Interface Components. In A. Dix, T. Hussein, S. Lukosch, & J. Ziegler (Eds.), *Proceedings of the First Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS).*

Potter, R., & Wright, H. (2006). An Ontological Approach to Visualization Resource Management. In G. J. Doherty & A. Blandford (Eds.), *Interactive Systems. Design, Specification, and Verification, 13th International Workshop, DSVIS 2006, Dublin, Ireland, July 26-28, 2006. Revised Papers* (Vol. 4323, p. 151-156). Springer.

Rebstock, M., Fengel, J., & Paulheim, H. (2008). *Ontologies-based Business Integration.* Springer.

Ruiz, F., & Hilera, J. R. (2006). Using Ontologies in Software Engineering and Technology. In C. Calero, F. Ruiz, & M. Piattini (Eds.), (p. 49-102). Springer.

Sauermann, L., Bernardi, A., & Dengel, A. (2005). Overview and Outlook on the Semantic Desktop. In S. Decker, J. Park, D. Quan, & L. Sauermann (Eds.), *Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference.*

Scerri, S., Davis, B., Handschuh, S., & Hauswirth, M. (2009). Semanta - Semantic Email Made Easy. In L. Aroyo et al. (Eds.), *The Semantic Web: Research and Applications* (Vol. 5554, p. 36-50). Springer.

Schmidt, K.-U., Dörflinger, J., Rahmani, T., Sahbi, M., & Thomas, L. S. S. M. (2008). An User Interface Adaptation Architecture for Rich Internet Applications. In S. Bechhofer, M. Hauswirth, J. Hoffmann, & M. Koubarakis (Eds.), *The Semantic Web: Research and Applications. Proceedings of the 5th European Semantic Web Conference, ESWC 2008.* (p. 736-750).

Seeling, C., & Becks, A. (2003, July). Exploiting metadata for ontology-based visual exploration of weakly structured text documents. In *Proceedings. Seventh International Conference on Information Visualization, 2003. IV 2003.* (p. 652-657).

Sergevich, K. A., & Viktorovna, G. V. (2003). From an Ontology-Oriented Approach Conception to User Interface Development. *International Journal "Information Theories and Applications"*, *10*(1), 89-98.

Smith, B., & Welty, C. (2001). FOIS introduction: Ontology—towards a new synthesis. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems* (pp. 3–9). New York, NY, USA: ACM.

Sousa, K. (2009). Model-Driven Approach for User Interface - Business Alignment. In *Proceedings of The 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS'2009)* (p. 325-328). ACM Press.

Spahn, M., Kleb, J., Grimm, S., & Scheidl, S. (2008). Supporting business intelligence by providing ontology-based end-user information self-service. In *OBI '08: Proceedings of the first international workshop on Ontology-supported business intelligence* (pp. 1–12). New York, NY, USA: ACM.

Stadlhofer, B., & Salhofer, P. (2008). SeGoF: semantic e-government forms. In *Proceedings of the 2008 international conference on Digital government research* (pp. 427–428). Digital Government Society of North America.

Stuckenschmidt, H., Parent, C., & Spaccapietra, S. (Eds.). (2009). *Modular Ontologies - Concepts, Theories and Techniques for Knowledge Modularization.* Springer.

Tane, J., Schmitz, C., & Stumme, G. (2004). Semantic resource management for the web: an e-learning application. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* (pp. 1–10). New York, NY, USA: ACM.

Uschold, M., & Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. *SIGMOD Record*, *33*(4), 58–64.

Uschold, M., & Jasper, R. (1999). A framework for understanding and classifying ontology applications. In *Proceedings of the IJCAI99 Workshop on Ontologies* (pp. 16–21).

Visser, U., & Schuster, G. (2002). Finding and Integration of Information - A Practical Solution for the SemanticWeb. In J. Euzénat, A. Gómez-Pérez, N. Guarino, & H. Stuckenschmidt (Eds.), *Proceedings of the ECAI-02 Workshop on Ontologies and Semantic Interoperability.*

Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., & Studer, R. (2006). Semantic Wikipedia. In *WWW '06: Proceedings of the 15th international conference on World Wide Web* (pp. 585–594). New York, NY, USA: ACM.

W3C. (2004, November). *OWL-S: Semantic Markup for Web Services.* (http://www.w3.org/Submission/OWL-S/)

W3C. (2005). *Web Service Modeling Ontology (WSMO).* (http://www.w3.org/Submission/WSMO/)

W3C. (2009). *WAI-ARIA Overview.* http://www.w3.org/WAI/intro/aria.