Why UI Standards Should Come Together with Formal Ontologies

Position Paper

Heiko Paulheim and Florian Probst
SAP Research CEC Darmstadt
Bleichstrasse 8
64283 Darmstadt, Germany
{heiko.paulheim,f.probst}@sap.com

ABSTRACT
Formal ontologies have been successfully used in various places in the domain of user interface development, e.g. in the integration of heterogeneous user interface components. Besides providing a sound basis for comparing interface components from distinct developers, a central advantage is that ontologically enhanced user interfaces allow controlling interactions between user interface components at run-time. For making these advantages available on a large scale, we strongly plead for taking principals from ontology engineering into account when standardizing user interface models.

INTRODUCTION
Software in general, and user interfaces in particular, are only seldom developed from scratched. Instead, components are reused, which may be home-grown, open source, or commercial, or a combination of those. Therefore, the task of *integrating* software components is an important part of software engineering. In this paper, we show how this task can be improved by using ontologies serving as formal models of user interfaces.

We use the term *user interface integration*, or *UI integration for short*, to denote the activity of assembling a user interface from different UI components. With traditional approaches to UI integration, especially when using heterogeneous UI components which do not come with matching screws and bolts, the developer has to acquire a lot of knowledge about the components' internal functionality and perform quite a few hacks to end up with a seamlessly integrated user interface. Typically, the result will contain lots of code-tangling and not be very modular and easily maintainable [7].

We introduce an approach for integrating user interface components, using formal ontologies for modeling those components. We present examples showing the advantages of using formal ontologies for user interface integration, and discuss the necessity for bringing together user interface standards and formal ontologies.

APPROACH
In our previous work, we have developed an approach which uses ontologies for integrating heterogeneous UI components [6, 7, 8]. Our current research prototype is implemented in Java, using the OntoBroker [1] reasoner.

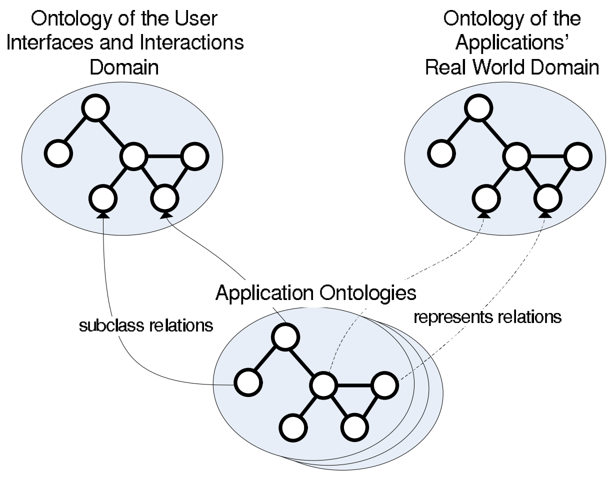
Ontology of the User Interfaces and Interactions Domain
Ontology of the Applications' Real World Domain
Application Ontologies
subclass relations
represents relations

Figure 1. Different types of ontologies are used in our framework. The domain ontologies describing the user interfaces and interactions domain and the real world are not interconnected.

In this approach, we use two types of ontologies which are shared by all developers of components and provide a common conceptualization: an ontology of the user interfaces and interactions domain, and an ontology of the real world domain the system is built for (e.g. banking, travel, etc.). Based on these two ontologies, one application ontology is defined per integrated application. This ontology reflects a formal model of the applications' UI components, the interactions that are possible with them, and integration rules that define the possible interactions with other applications. Fig. 1 shows an overview of the three types of ontologies.

It is noteworthy that the real world domain ontology and the ontology of the user interfaces and interactions domain are distinct and not connected with each other. This account allows a strict separation of concerns, and the framework becomes universally applicable: when developing systems for a different domain, the real world domain ontology can be exchanged for a

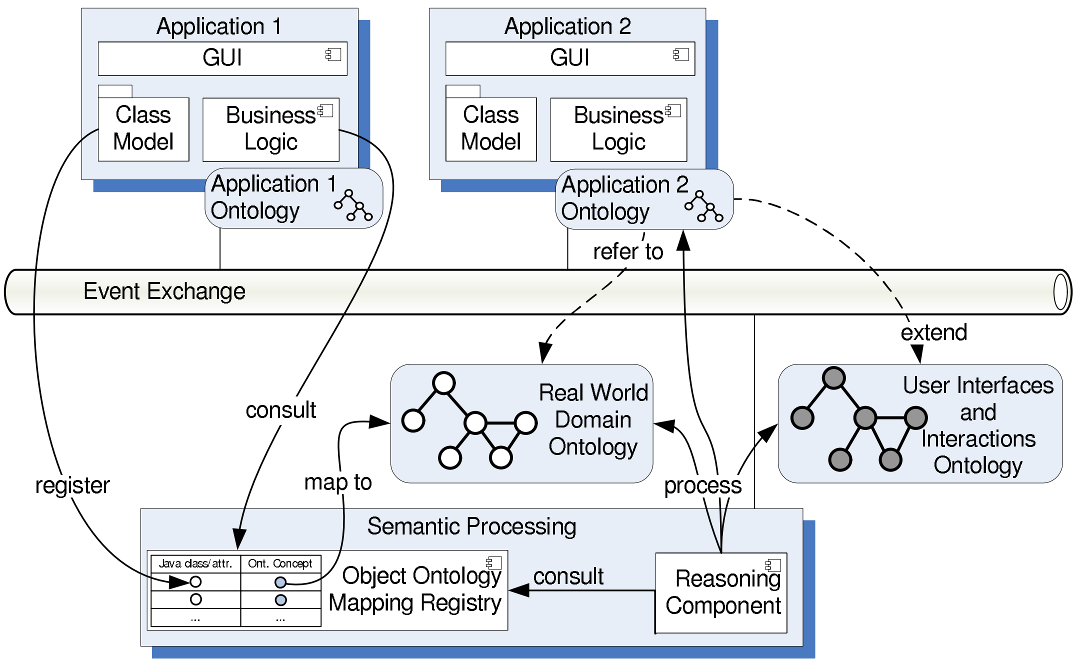

Figure 2. Architecture of our Integration Framework. UI components and information objects are described in ontologies, and a reasoner evaluates events based on integration rules at run-time, thus facilitating dynamic integration of user interface components.

different one without having to change the underlying framework for user interface integration.

In terms of MDA, the ontology of the user interfaces and interactions domain corresponds to the meta model, as it contains the definitions of things that can potentially exist in the domain of user interfaces. The application ontologies correspond to models in MDA terms, as they define one specific type of user interface, as argued in [11].

With those ontologies, all messages sent between components as well as the information objects contained therein are annotated. Assuming that the two domain ontologies refelct a generic but widely shared conceptualization of the real world domain the application is developed for as well as the *inner workings* of user interfaces, these annotations lead to a mutual understanding between components. While annotating static components of a user interface enables developers to generally assess the interoperability between interface components, events are annotated at run-time. These annotated events, raised by UI components, are processed centrally by the reasoner based on the integration rules defined in the application ontologies, facilitating a mechanism for semantic event processing [10]. Thus, a clear and complete decoupling between components is possible with our approach. This decoupling leads to an integrated system where single parts can be easily exchanged for others, and code-tangling is avoided.

Fig. 2 shows the basic architecture of our integration framework. All UI components that are integrated (which might have an underlying business logic and data storage, so they can be regarded as full-fledged applications) and are annotated with one *application ontology*. All components are connected via a message exchange bus. A reasoner which processes all the ontologies described above is also connected to the message bus. It reads events from the bus, computes reactions to those events, and then puts the corresponding events on the bus. For analyzing objects contained in an event, an annotation repository is used.

Using a reasoner for analyzing the integrated components and the events they raise at run-time requires a formal ontology. Based on this formal ontology, the reasoner can compute deductions on how the integrated components can work together. As ontologies are well-grounded in formal logics, standard reasoners can be employed to perform those computations. Furthermore, the implicit assumptions underlying the individual models are made explicit in the domain ontologies, thus, the models can be communicated more easily, without the need to deeply understand the internals of all integrated components.

CONTRIBUTIONS TO THE WORKSHOP

Our main focus lies on the integration of existing applications, especially on the user interface level. Just like MDA, our ontology-based approach for UI integration would heavily benefit from standardizing user interface models. Even more so, if user interfaces were delivered with a standardized formal description (sometimes called ontology), they could be integrated at run-time in an ad hoc fashion.

Therefore, we are very interested in driving the standardization of UI models forward. For the reasons discussed in the previous section, we are confident that ontologies are a useful enhancement when standardizing user interfaces. We therefore argue for shaping UI modeling standards in a ways that ontology-based approaches are possible. We are confident that the use of ontologies in standardized UI models will improve their reusability and communicability. Furthermore, a vast amount of ontology editors, programming frameworks etc. already exists. Thus, a standardized model built on such well-established foundations will experience increased acceptance.

With our use case of user interface integration, we have shown an approach that relies on formal ontologies as models for user interfaces. There are other interesting fields where formal ontologies are used at run-time for improving user interfaces, such as the automatic generation of help on an application [2, 5], adapting user interfaces at run-time, based on the users' needs [4], or providing user assistance when filling forms [9, 3].

With these application areas in mind, we encourage to foresee a possibility (e.g. an attribute) for linking UI models with a formal ontology of user interfaces and interactions, i.e. for *semantically annotating* UI models. In a further standardization step, the ontology itself can be subject for standardization. In the long run, we think that meta models of user interface standards and domain ontologies of the user interfaces and interactions domain can converge, thus ending up with both, a standardized domain ontology of the user interfaces and interactions domain, and user interface standards that can be coupled with ontological engineering approaches.

Acknowledgements
The work presented in this paper has been partly funded by the German Federal Ministry of Education and Research under grants no. 01ISO7009 and 01IA08006.

REFERENCES

1. S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman, Z. Tari, and S. M. Stevens, editors, *Database Semantics - Semantic Issues in Multimedia Systems, IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics (DS-8), Rotorua, New Zealand, January 4-8, 1999*, volume 138 of *IFIP Conference Proceedings*, pages 351–369. Kluwer, 1999.

2. V. Gribova. Automatic Generation of Context-Sensitive Help Using a User Interface Project. In V. P. Gladun, K. K. Markov, A. F. Voloshin, and K. M. Ivanova, editors, *Proceedings of the 8th International Conference "Knowledge-Dialogue-Solution"*, volume 2, 2007.

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

4. S. Karim and A. M. Tjoa. Towards the Use of Ontologies for Improving User Interaction for People with Special Needs. In K. Miesenberger, J. Klaus, W. L. Zagler, and A. I. Karshmer, editors, *ICCHP*, volume 4061 of *Lecture Notes in Computer Science*, pages 77–84. Springer, 2006.

5. A. Kohlhase and M. Kohlhase. 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 Special Interest Group for Design of Communication, ACM Press, 2009.

6. H. Paulheim. Ontologies for User Interface Integration. In A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, editors, *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 973–981. Springer, 2009.

7. H. Paulheim and A. Erdogan. Seamless Integration of Heterogeneous UI Components. In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2010)*, 2010. To appear.

8. H. Paulheim and F. Probst. Improving UI Integration with Formal Semantics. In A. Dix, T. Hussein, S. Lukosch, and J. Ziegler, editors, *Proceedings of the Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS)*, 2010.

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

10. K. Teymouriana and A. Paschke. Towards semantic event processing. In *DEBS '09: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, pages 1–2, New York, NY, USA, 2009. ACM.

11. G. W. Uwe Aßmann, Steffen Zschaler. *Ontologies, Meta-models, and the Model-Driven Paradigm*, chapter 9, pages 249–273. Springer, 2006.