Ontologies for User Interface Integration

Heiko Paulheim

SAP Research
heiko.paulheim@sap.com

Abstract. Application integration can be carried out on three different levels: the data source level, the business logic level, and the user interface level. With ontologies-based integration on the data source level dating back to the 1990s and semantic web services for integrating on the business logic level coming of age, it is time for the next logical step: employing ontologies for integration on the user interface level. Such an approach supports both the developer (in terms of reduced development times) and the user (in terms of better usability) of integrated applications. In this paper, we introduce a framework employing ontologies for integrating applications on the user interface level.

1 Introduction

Applications are often described in three layers: data, business logic, and user interface. Consequently, application integration can be performed on each of those three levels [1], as depicted in Fig. 1:

- Integrating the data sources, and developing common business logic and user interface layers above the integration layer,
- integrating the business logic, and developing a common user interface above the integration layer, and
- integrating the user interfaces.

Integrating applications on the user interface level means reusing existing user interfaces or parts thereof and coupling them in a way that *a user can interact with those interfaces as if they were a single application.* Such an integration may include that the applications share a common toolbar or menu, that one application reacts to user actions performed with another one, e.g. related objects are highlighted in other applications when selected in one application, objects can be dragged and dropped from on application to the other, etc. There are some approaches such as plugin-based systems [2], portals [3], and mashups [4] that propose integration on the user interface level. However, all of those approaches are either very limited concerning cross-application interaction or require deep changes of the applications in order to facilitate such interactions [1].

With semantic database integration [5] as well as ontology-based agents [6] and semantic web services [7,8], there have been considerable efforts to using ontologies in the integration on the database and business logic layer. However, little research has been conducted on ontology-based integration on the user

A. Bernstein et al. (Eds.): ISWC 2009, LNCS 5823, pp. 973–981, 2009.

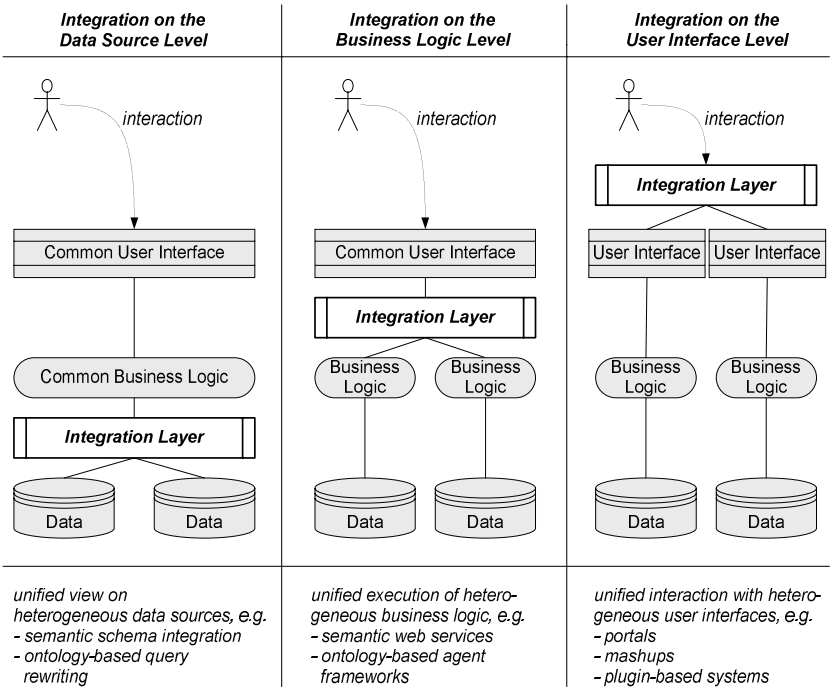

Fig. 1. Three layers of integration; based on [1]

interface level so far. In contrast to integration on the lower layers, user interface level integration has two significant advantages:

- The development of the user interface consumes up to 50% of the total efforts in developing an application [9]. Therefore, the benefit from reusing existing user interface components is significant.
- Users interacting with applications integrated on the user interface level will experience a decreased learning effort if already familiar with the applications' original interfaces, compared to interacting with a newly developed common user interface.

Yu et al. argue that user interface integration requires a description of the interfaces to be integrated that is *formal*, *human readable*, *modular*, and *simple* [10]. Ontologies perfectly meet the first two criteria and also provide the possibility for modularization [11]. As simplicity is a rather subjective criterion, and the description language must be flexible enough to cover all possible cases of integration, hence must not be too simple, we claim that ontologies are a suitable approach.

Since, as discussed above, *interaction* plays a crucial role in integrated user interfaces as well as it imposes problems in current approaches to user interface integration, the approach discussed in this paper aims at describing such

interactions with ontologies. We introduce a prototype framework that allows run-time integration of user interfaces based on ontological descriptions of the interactions they support.

2 State of the Art

There are some approaches that employ ontologies for user interface integration in portals and mashups. The approach described in [12] uses semantic web services, i.e., web services described by means of ontologies. It rather focuses on communication between a portlet and its backend system than on inter-portlet communication and on user interaction. The work described in [13] uses ontologies to annotate the contents delivered by portlets. That approach is rather data-centric and has little focus on interaction. The work described in [14] shows how ontologies can help building mashup applications to integrate contents from diverse data sources in one mashup.

The work described in [15] and [16] shows how ontologies can be used to formalize user interfaces and to generate user interfaces with a model driven approach. The approach suggested in [17] also formalizes user interfaces with the help of ontologies, but with the aim of making user interfaces accessible to people with disabilities. Other approaches, such as [18] and [19], annotate software components in general (not necessarily user interface components) with ontologies to support the developer in searching and choosing appropriate components. The work described in [20] and [21] propose ontologies for describing different types of user interfaces on a rather general level, such as characterizing different input and output devices. Such formalizations are helpful, but so far, they have not been applied to *integrating* different user interfaces.

A research direction which comes close to ontologies-based user interface integration is the Semantic Desktop [22]. Here, data encapsulated in different applications is made accessible via a central query interface. In some semantic desktop systems, existing applications may be integrated as plugins [23]. The main focus of this direction, however, is to provide an integrated access to data in different applications rather than on cross-application interaction; it can therefore be regarded as an approach to data integration rather than to user interface integration.

3 Roadmap

3.1 Prototype

So far, a first prototype has been developed that shows how user interfaces can be integrated by using ontologies [24]. Three types of ontologies are used (see Fig. 2):

- An *ontology of the user interfaces and interactions domain*, which defines basic categories for describing applications. This ontology is part of the framework discussed in this paper.

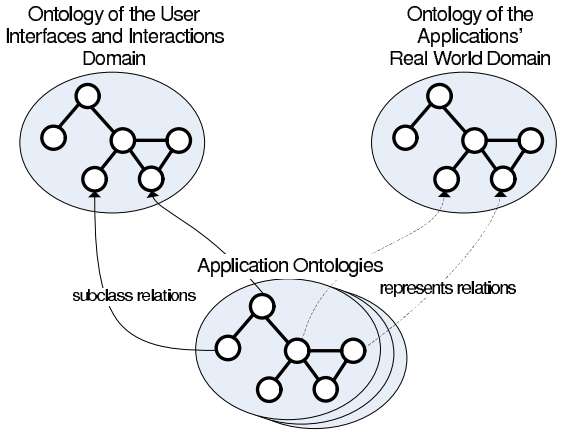

Fig. 2. Using two domain ontologies and several application ontologies for integration on the user interface level

- An *ontology of the application's real world domain*, which defines the categories of real world objects of the domain that the integrated application is built for (such as banking, travel, etc.). The information objects processed by the application *represent* those real worlds objects. When integrating applications from a given domain, an appropriate domain ontology has to be chosen or developed. A set of different real world domain ontologies may be used in case of modular domain ontologies or when developing cross-domain applications.
- One or more *application ontologies*, which use the user interfaces and interactions ontology's basic concepts to describe the applications to be integrated, and the interactions that are possible with them. The application ontologies may refer to the real world domain ontology for describing the types of objects that may be processed by the integrated applications. During the integration process, one application ontology per integrated application has to be developed.

This categorization resembles Guarino's classification [25] (without the top level layer, which may also be present in our framework, but its presence is not essential) – here, two kinds of domain ontologies are used. While the ontology of the user interfaces and interactions domain is a part of the integration framework, the real world domain ontology and application ontologies are dynamically added for each integrated application. It is particularly noteworthy that there is no direct connection from the user interfaces and interactions domain ontology to the real world domain ontology. Thus, the framework is domain independent.

Fig. 3 shows an overview of the framework prototype, which is based on Java and the OntoBroker reasoner [26]. Integrated applications consist of a class model for representing data, a user interface, and a business logic (the classical model view controller [27] components), and are described by an application ontology.

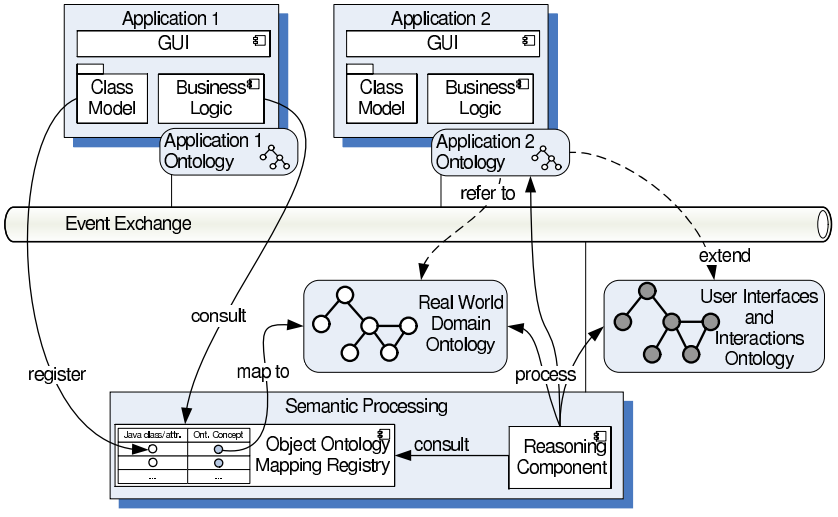

Fig. 3. Overview on the prototype's framework architecture. Applications are described by ontologies. A reasoning component evaluates those ontologies to facilitate integration at run time.

Applications communicate via events which are annotated using the respective application ontology. One key design decision is that no application sends events directly to any other application. Instead, events are processed by a reasoning component. For example, one application sends an event that a certain object is selected. The reasoner reads the event, queries the application ontologies to determine those applications which declare to react to that sort of event, and notifies the respective applications. Thus, no application has to directly react to other applications' events and only needs to process the event types defined in its own application ontology.

To allow mediation between different data models, an *object ontology mapping registry* is introduced. In this registry, each application stores annotations of its data model. Classes as well as properties may be annotated with concepts from the domain ontology. When the reasoner receives an object from an application or vice versa, the receiver consults the registry to analyse that object and convert it into a representation which can be processed by the reasoner.

3.2 Further Research Plan

While the prototype shows that the approach is valid and feasible, there are quite a few open research questions. We have shown that simple interactions (such as highlighting objects selected in different applications) are possible with our framework. More complex interactions will require extensions and refactoring of both the application ontology and the prototype implementation.

So far, we have only considered typical single-user WIMP (Windows, Icons, Mouse, Pointing) interfaces when modelling the application ontology. A more

sophisticated ontology describing interactions would be flexible enough to allow different input and output devices (such as speech interfaces, gestures, and so on) [28], as well as multi-user interaction. Coupling the application ontology with a device ontology, such as the FIPA device ontology [29], could lead to a more universal framework.

Complex interactions may have conditions under which they may be performed, e.g. the visibility of a component or the presence of an application which is able to perform a certain task. To evaluate those conditions, an application's internal state has to be exposed to a certain extent. Thus, the relevant state information has to be identified and modelled in the user interfaces ontology. Furthermore, the state information has to be made known to the reasoner, either by the reasoner dynamically querying the applications or by the applications sending updates to the reasoner. A special case of preconditions are the users' rights to perform a certain interaction. Thus, adding an ontology of users' rights, such as the one proposed in [30], would be a feasible approach to account for that requirement.

The design decision that each application can use their own data model eases the reuse of existing components, but it comes with certain challenges in data integration. As there is a large variety of heterogeneities that may occur here [31], some methods to cope with those heterogeneities is needed. Such methods will probably impose certain restrictions on the data model used (such as a 1:1 mapping between elementary data types in the data model and data attributes in the domain ontology). We aim at finding a minimal set of such restrictions in order to allow a maximum degree of freedom in the applications' data models.

The application ontologies describing the interactions possible with applications may not only be used for integration. Another way of utilizing those ontologies is the provision of user assistance, such as automatic generation of help texts, or highlighting possible drop locations in different applications when dragging an object. The latter has already been successfully demonstrated in our prototype.

Finally, the approach requires validation beyond having a running prototype. We plan to conduct case studies where different example applications are to be integrated in a way providing a given set of interactions. Here, development efforts can be measured, e.g. in lines of code, and expert interviews with developers can reveal additional insights into the feasibility of the approach. In addition, studies with end users may be conducted to demonstrate the advantage of integrated user interfaces over non-integrated side-by-side use of applications.

4 Conclusion

In this paper, we have presented the idea of using ontologies for integrating applications on the user interface level. In our framework, applications are described by application ontologies, making use of two or more shared domain ontologies.

A first prototype shows that the approach is feasible. It has been successfully used in the SoKNOS project [32], where an integrated emergency management

software has been built, consisting of twelve integrated single applications, serving purposes such as planning measures with resources, handling messages, or displaying relevant mission data on charts and geographic maps.

This prototype as well as the underlying ontology and algorithms are going to be subsequently extended. The aim is to enhance the framework in a way that it covers the most common interaction patterns between integrated applications, and that it can be enhanced in cases where more unusual interactions are to be implemented.

In summary, we believe that such a framework for integrating applications on the user interface level is a useful complement to existing integration efforts on the data and business logic level.

Acknowledgements

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

References

1. Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. IEEE Internet Computing 11(3), 59–66 (2007)
2. Birsan, D.: On plug-ins and extensible architectures. ACM Queue 3(2), 40–46 (2005)
3. Wege, C.: Portal Server Technology. IEEE Internet Computing 6(3), 73–77 (2002)
4. Abiteboul, S., Greenshpan, O., Milo, T.: Modeling the Mashup Space. In: WIDM 2008: Proceeding of the 10th ACM workshop on Web information and data management, pp. 87–94. ACM, New York (2008)
5. Doan, A., Halevy, A.Y.: Semantic Integration Research in the Database Community: A Brief Survey. AI Magazine 26(1), 83–94 (2005)
6. Sycara, K.P., Paolucci, M.: 17. International Handbooks on Information Systems. In: Ontologies in Agent Architectures, pp. 343–364. Springer, Heidelberg (2004)
7. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic Markup for Web Services (November 2004), `http://www.w3.org/Submission/OWL-S/`
8. Lausen, H., Polleres, A., Roman, D., de Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., König-Ries, B., Kopecky, J., Lara, R., Lausen, H., Oren, E., Polleres, A., Roman, D., Scicluna, J., Stollberg, M.: Web Service Modeling Ontology, WSMO (2005), `http://www.w3.org/Submission/WSMO/`
9. Myers, B.A., Rosson, M.B.: Survey on user interface programming. In: CHI 1992: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 195–202. ACM, New York (1992)
10. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M.: A framework for rapid integration of presentation components. In: WWW 2007: Proceedings of the 16th international conference on World Wide Web, pp. 923–932. ACM, New York (2007)

11. Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing, Duluth, MN, USA, vol. 43, pp. 907–928. Academic Press, Inc., New York (1995)
12. Dettborn, T., König-Ries, B., Welsch, M.: Using Semantics in Portal Development. In: Proceedings of the 4th International Workshop on Semantic Web Enabled Software Engineering (2008)
13. Díaz, O., Iturrioz, J., Irastorza, A.: Improving portlet interoperability through deep annotation. In: WWW 2005: Proceedings of the 14th international conference on World Wide Web, pp. 372–381. ACM, New York (2005)
14. Ankolekar, A., Krötzsch, M., Tran, T., Vrandecic, D.: The Two Cultures: Mashing Up Web 2.0 and the Semantic Web. In: WWW 2007: Proceedings of the 16th International Conference on World Wide Web, pp. 825–834. ACM, New York (2007)
15. Sergevich, K.A., Viktorovna, G.V.: From an Ontology-Oriented Approach Conception to User Interface Development. International Journal Information Theories and Applications 10(1), 89–98 (2003)
16. Liu, B., Chen, H., He, W.: Deriving User Interface from Ontologies: A Model-Based Approach. In: ICTAI 2005: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence, pp. 254–259. IEEE Computer Society, Los Alamitos (2005)
17. W3C: WAI-ARIA Overview (2009), `http://www.w3.org/WAI/intro/aria`
18. Graubmann, P., Roshchin, M.: Semantic Annotation of Software Components. In: EUROMICRO 2006: Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, Washington, DC, USA, pp. 46–53. IEEE Computer Society, Los Alamitos (2006)
19. Happel, H.J., Korthaus, A., Seedorf, S., Tomczyk, P.: KOntoR: An Ontology-enabled Approach to Software Reuse. In: Zhang, K., Spanoudakis, G., Visaggio, G. (eds.) Proceedings of the Eighteenth International Conference on Software Engineering & Knowledge Engineering (SEKE), pp. 349–354 (2006)
20. Coutaz, J., Lachenal, C., Dupuy-Chessa, S.: Ontology for Multi-surface Interaction. In: Proceedings of IFIP INTERACT03: Human-Computer Interaction, IFIP Technical Committee No 13 on Human-Computer Interaction, pp. 447–454 (2003)
21. Eick, S.G., Wills, G.J.: High Interaction Graphics. European Journal of Operational Research 84, 445–459 (1995)
22. Sauermann, L., Bernardi, A., Dengel, A.: Overview and Outlook on the Semantic Desktop. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729. Springer, Heidelberg (2005)
23. Cheyer, A., Park, J., Giuli, R.: Iris: Integrate. relate. infer. share. In: Workshop on the Semantic Desktop: Next Generation Personal Information Management and Collaboration Infrastructure (2005)
24. Paulheim, H.: Ontology-based Modularization of User Interfaces. In: Calvary, G., Graham, T.C.N., Gray, P. (eds.) Proceedings of The ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 23–28. ACM, New York (2009)
25. Guarino, N. (ed.): Formal Ontology and Information Systems. In: Guarino, N. (ed.) Proc. of the 1st Int'l Conf. on Formal Ontologies in Information Systems (FOIS 1998), Trento, Italy. IOS Press, Amsterdam (1998)
26. ontoPrise: OntoBroker Website (2009), `http://www.ontoprise.de/de/en/home/products/ontobroker.html`
27. Krasner, G.E., Pope, S.T.: A cookbook for using the model-view controller user interface paradigm in small-talk-80. Journal of Object Oriented Programming 1, 26–49 (1988)

28. van Dam, A.: Post-wimp user interfaces. Commun. ACM 40(2), 63–67 (1997)
29. Foundation for Intelligent Phyiscal Agents: FIPA Device Ontology Specification (December 2002),
`http://www.fipa.org/specs/fipa00091/index.html` (accessed, 2008-01-22)
30. Kagal, L., Finin, T., Joshi, A.: A policy language for a pervasive computing environment. In: POLICY 2003: Proceedings of the IEEE 4th International Workshop on Policies for Distributed Systems and Networks (2003)
31. Klein, M.: Combining and relating ontologies: an analysis of problems and solutions. In: Gomez-Perez, A., Gruninger, M., Stuckenschmidt, H., Uschold, M. (eds.) Workshop on Ontologies and Information Sharing, IJCAI 2001, Seattle, USA (2001)
32. Doeweling, S., Probst, F., Ziegert, T., Manske, K.: SoKNOS - An Interactive Visual Emergency Management Framework. In: Amicis, R.D., Stojanovic, R., Conti, G. (eds.) GeoSpatial Visual Analytics. NATO Science for Peace and Security Series C: Environmental Security, pp. 251–262. Springer, Heidelberg (2009)