A Formal Ontology on User Interfaces – Yet Another User Interface Description Language?

Position Paper

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

ABSTRACT
During the past years, a lot of user interface description languages, most of them based on XML, have been introduced. At the same time, the use of formal ontologies for describing user interfaces has been discussed for a number of use cases. This paper discusses the differences between a formal ontologies and user interface description languages and and points out how both research directions can benefit from each other.

Author Keywords
User Interfaces, Ontology, UI Description Languages, Formal Models

ACM Classification Keywords
D.2.2 Software Engineering: Design Tools and Techniques—*User Interfaces*; D.2.11 Software Engineering: Software Architectures—*Languages*; I.2.4 Artificial Intelligence: Knowledge Representation Formalisms and Methods—*Semantic Networks*

General Terms
Design, Languages

INTRODUCTION

Recently, a number of use cases have been proposed that employ ontologies for modeling user interfaces, their components and interaction capabilities. Examples are automatic generation of explanations for user interfaces, adaptation of user interfaces for different needs and contexts, and integration of user interface components [14]. Those use cases require a strongly formalized ontology of the domain of user interfaces and interactions.

In parallel, various UI description languages have been proposed, most of them XML based [7, 12]. The duality of UI description languages and formal ontologies gives rise to the question whether an additional ontology is really needed, or whether it is going to be yet another user interface description language.

ONTOLOGIES AND MODELS

Although ontologies and software models are related, they are not essentially the same. Software models and ontologies are different by nature. An ontology claims to be a *generic*, commonly agreed upon specification of a conceptualization of a domain [6], with a focus on precisely capturing and formalizing the semantics of terms used in a domain. A software model in turn is *task-specific*, with the focus on an efficient implementation of an application for solving tasks in the modeled domain [2, 16, 18]. Thus, a software engineer would rather trade off precision for a simple, efficient model, with the possibility of code generation, while an ontology engineer would trade off simplicity for a precise representation. Another difference is that in software engineering, models are most often *prescriptive* models, which are used to specify how a system is *supposed* to behave, while ontologies are rather *descriptive* models, which describe how the world *is* [1]. Figure 1 illustrates those differences.

Taking this thought to the domain of user interfaces and interactions, models are used to define particular user interfaces (e.g. with the goal of generating code implementing those interfaces), while a formal ontology would capture the nature of things that exist in the domain, e.g., which types of user interfaces exist, and how they are related.

Due to those differences, we argue that developing a formal ontology on user interfaces will not lead to yet another user interface description language, but to a formal model with different intentions and usages. In the next sections, we will discuss how the two worlds can benefit from each other.

HOW A FORMAL ONTOLOGY CAN BENEFIT FROM UI DESCRIPTION LANGUAGES

A lot of research work has gone into the development of different user interface description languages. Those research efforts can be and should be taken into account when developing an ontology of the domain.

Collection of Concepts

Most methodologies for ontology engineering foresee the capturing of key concepts and relationships as one of the first steps. This can be done by conducting interviews

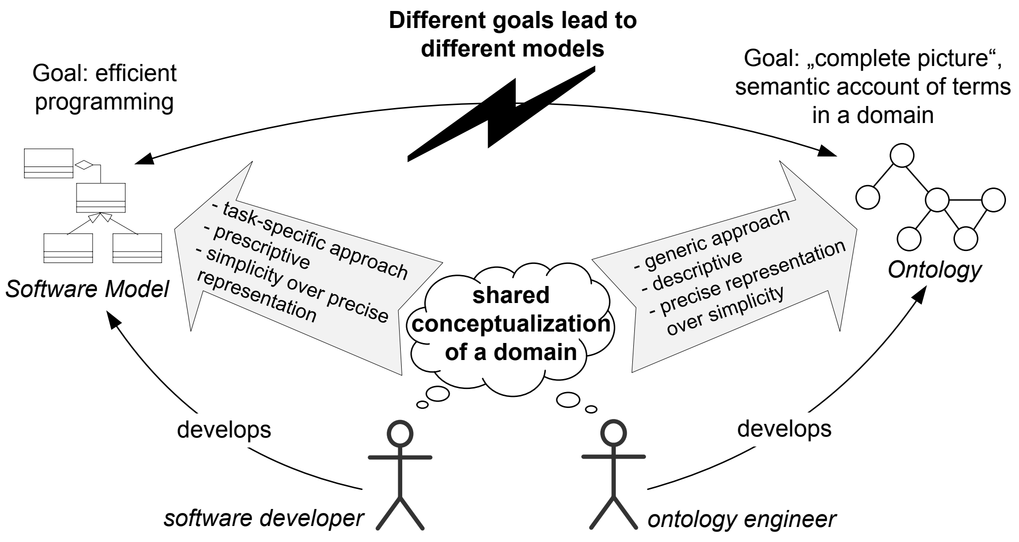
Different goals lead to different models
Goal: efficient programming
Goal: „complete picture“, semantic account of terms in a domain
- task-specific approach
- prescriptive
- simplicity over precise representation
- generic approach
- descriptive
- precise representation over simplicity
shared conceptualization of a domain
Software Model
Ontology
develops
develops
software developer
ontology engineer

Figure 1. Ontologies and modeling languages serve different purposes.

with domain experts, scanning books and other material, and/or reusing parts of other ontologies [5, 19]. At this point of ontology engineering, lots of input can be used from existing user interface description languages.

Since those languages are most often XML-based, they consist of a smaller or larger number of tags and attributes, which determine the expressivity of the language. As many of those elements define certain concepts of the domain, such as UI components or actions that can be performed with them, they are a good starting point for developing a formal ontology of the domain.

Benchmarking the Ontology's Completeness

As discussed above, ontology engineering aims at providing a complete, comprehensive formal description of a domain. However, assessing the completeness of an ontology is not always an easy task. Here, user interface description languages can once again help by providing a benchmark for the ontology's completeness.

Such a benchmark can be performed in different ways. On the meta-model level, the number of concepts contained in the meta model (e.g., tags and attributes in an XML schema) which have a counterpart in the ontology can be determined. On the model level, one can check whether given models in a user interface description language can be expressed using only the terms given in an ontology, either informally, or formally, e.g., in RDF. Thus, user interface description languages can provide a measure for the completeness of an ontology of the domain.

HOW UI DESCRIPTION LANGUAGES CAN BENEFIT FROM A FORMAL ONTOLOGY

Once an ontology of the domain of user interfaces and interactions has been created, it can be used to improve the development and usage of new and existing user interface description languages as well.

Disambiguation of Terms

In an analysis of user interface description languages, we have found that terms are often used differently in different standards. An example is the term *dialog*. In XIML, for example, a `dialog element` is defined as being "like a command that can be executed [...] It is the more concrete instantiation of a task." [15]. In contrast, XUL defines a `dialog` as an "element [which] should be used in place of the window element for dialog boxes" [10]. Such ambiguities can easily lead to misinterpretations, especially if users are trained on a particular language and switch to another one.

Mapping a user interface description language to a formal ontology capturing the *semantics* of those terms can avoid such misinterpretations. With the example term *dialog*, a formal ontology can help resolving the ambiguity by indicating that the languages imply different top-level categories such as PROCESS, PLAN, or SOFTWARE COMPONENT as super-category for DIALOG.

Facilitating Extensibility of User Interface Description Languages

XML based languages usually use a fixed set of tags. In order not to be too strictly limited for practical use, many of those languages provide some extension mechanisms such as universal general purpose tags that can be used for user-defined concepts (e.g. the `ELEMENT` tag in XIML). These extension slots are then filled with arbitrary strings.

Arbitrary strings, however, are dangerous. They lead to extensions that are incompatible with each other, interpreted differently by different people and systems

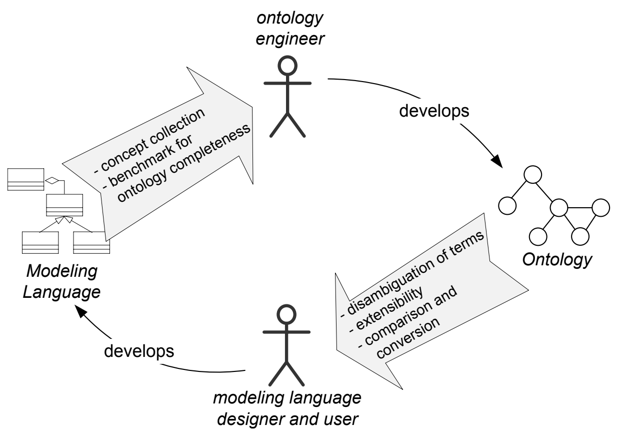
ontology engineer
develops
Ontology
- concept collection
- benchmark for ontology completeness
Modeling Language
- disambiguation of terms
- extensibility
- comparison and conversion
develops
modeling language designer and user

Figure 2. How user interface description languages and ontologies can benefit from each other

relying on different conventions and external documentations, and, in the end, foil the overall idea of having a standardized modeling language.

A formal ontology can help here by providing a standardized vocabulary which can be used to fill such extension slots. Thus, it can be assured that there is an unambiguous interpretation of the extensions.

Model Comparison and Conversion

When bringing together different development teams, information systems, or organizations, it is likely that models created with different user interface description languages already exist. Using a mediating ontology for annotating the models is a common way of establishing comparability between models, not only user interface models [4].

Once models are annotated and can be compared using a common ontology, automatic conversion of models can be long-term objective. For the moment, a common ontology can at least support developers in understanding each other's models and assist them in unambiguously transferring their contents between modeling languages manually.

Fig. 2 summarizes how modeling languages and a formal ontology can benefit from each other.

TOWARDS A FORMAL ONTOLOGY OF THE DOMAIN OF USER INTERFACES AND INTERACTIONS

With these considerations in mind, we have started to develop a formal ontology of the domain of user interfaces and interactions. The goal is to end up with an ontology that is *comprehensive* at least with respect to the expressivity of current user interface definition languages, that is universal enough to be *extendable* to future user interfaces that do not exist at the moment. Furthermore, to support valuable reasoning on user interfaces and provide meaningful semantics, the ontology should be *highly axiomatized.*

To end up with a comprehensive ontology, we have analyzed several user interface description languages in order to collect a maximum set of relevant terms. We have used UsiXML, XIML, UIML, Maria, XUL, LZX, WAI ARIA, and XForms as a basis for identifying the core concepts.

In order to build upon well-acknowledged roots, we have chosen the top level ontology *DOLCE* [9] and its extensions as a basis for our ontology. This top level ontology provides an embracing basic classification of things and has been used as a basis for building numerous ontologies. Since the top level provides a complete classification, it ensures extensibility of the ontology by design, as every new concept can be classified in some existing category. Furthermore, we have reused two core ontologies of software and software components [11], which are also built upon the foundations of DOLCE.

The ontology we have developed is divided into two parts: a *top level* which captures the semantics of the basic terms of the domain, such as *User Interface Component* and *Interaction*, while the *detail level* classifies the actual things that exist in the domain, such as types of user interface components and user tasks that can be performed with those components. The OWL version of the top level ontology consists of 15 classes, two object properties, and 75 axioms, while the detail level consists of 179 classes, eleven object properties, and 448 axioms.

CONCLUDING REMARKS

This position paper has discussed the differences between UI description languages and a formal ontology of the domain of user interfaces and interactions. Furthermore, We have given insight into the development of a comprehensive formal ontology of the user interfaces and interactions domain. In the long run, we are confident that formal ontologies and UI definition languages will both have their places, and that both will benefit from each other.

We have presented a number of potential improvements where developers employing user interface description languages could benefit from those languages being mapped to a formal ontology of user interfaces and interactions. Thus, our claim is that organizations providing user interface description languages could improve the usability and acceptance of those languages by providing such a mapping.

As a long-term objective, such a mapping could even facilitate automatic conversion between models developed with different user interface description languages. To that end, more sophisticated mapping approaches than simply relating elements form a modeling language to a category in an ontology are needed [13].

A formal ontology will not replace user interface description languages, but be a valuable enhancement. Due to the conceptual differences between software mod-

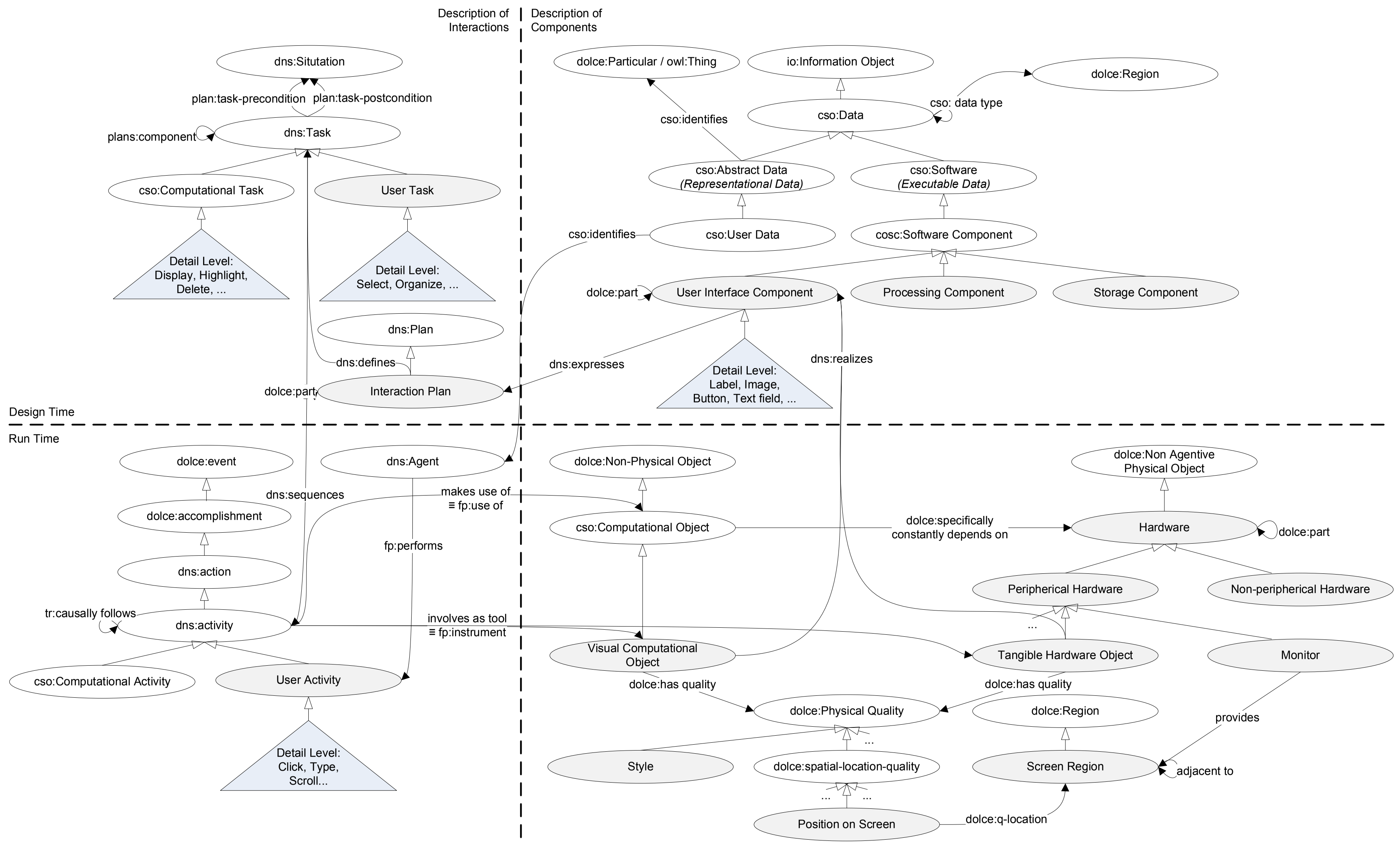

Figure 3. The top level of the ontology of the user interfaces and interactions domain. In the upper part, the design time concepts are shown, the lower part contains the run time concepts. The left part deals with interactions, the right part with components. The white ellipses denote concepts from the reused ontologies (with the following namespace conventions: DOLCE (dolce), Descriptions and Situations (dns), Plans (plans), Functional Participation (fp), Temporal relations (tr), Core Ontology of Software (cos), Core Ontology of Software Components (cosc)), the grey ellipses denote concepts from the top level ontology of the user interfaces and interactions domain. The grey triangles denote definitions carried out in the detail ontology.

els and ontologies, user interface description languages do a better job, e.g., when developing user interfaces in model based approaches. Although there have been attempts for UI code generation from ontologies [8, 17], the latter even claiming that ontologies should entirely replace existing user interface description languages, we believe that a co-existence of both is more beneficial.

Acknowledgements
The work presented in this paper has been partly funded by the German Federal Ministry of Education and Research under grants no. 01IA08006 and 13N10711.

REFERENCES

1. U. Aßmann, S. Zschaler, and G. Wagner. Ontologies, Meta-models, and the Model-Driven Paradigm. In Calero et al. [3], chapter 9, pages 249–273.
2. C. Atkinson, M. Gutheil, and K. Kiko. On the Relationship of Ontologies and Models. In S. Brockmans, J. Jung, and Y. Sure, editors, *WoMM*, volume 96 of *LNI*, pages 47–60. GI, 2006.
3. C. Calero, F. Ruiz, and M. Piattini, editors. *Ontologies for Software Engineering and Software Technology*. Springer, 2006.
4. J. Fengel and M. Rebstock. Linking Heterogeneous Conceptual Models through a Unifying Modeling Concepts Ontology. In N. Stojanovic and B. Norton, editors, *Proceedings of the 5th International Workshop on Semantic Business Process Management (SBPM 2010)*, volume 682 of *CEUR-WS*, pages 1–4, 2010.
5. M. Fernández, A. Gómez-Pérez, and N. Juristo. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In *Proceedings of the AAAI97 Spring Symposium*, pages 33–40, 1997.
6. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, Juni 1993.
7. J. Guerrero-Garcia, J. M. Gonzalez-Calleros, J. Vanderdonckt, and J. Munoz-Arteaga. A Theoretical Survey of User Interface Description Languages: Preliminary Results. In *LA-WEB '09: Proceedings of the 2009 Latin American Web Congress (la-web 2009)*, pages 36–43, Washington, DC, USA, 2009. IEEE Computer Society.
8. B. Liu, H. Chen, and W. He. Deriving User Interface from Ontologies: A Model-Based Approach. In *ICTAI '05: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence*, pages 254–259, Washington, DC, USA, 2005. IEEE Computer Society.
9. C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. WonderWeb Deliverable D18 – Ontology Library (final), 2003. `http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf`. Accessed August 2nd, 2010.
10. Mozilla. XUL. `https://developer.mozilla.org/en/XUL`, 2010. Accessed August 4th, 2010.
11. D. Oberle, S. Grimm, and S. Staab. An Ontology for Software. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, chapter 18, pages 383–402. Springer, 2nd edition edition, 2009.
12. F. Paternò, C. Santoro, and L. D. Spano. XML Languages for User Interface Models - Deliverable D2.1 of the ServFace Project. `http://141.76.40.158/Servface/index.php?option=com_docman&task=doc_download&gid=5&Itemid=61`, August 2008. Accessed August 9th, 2010.
13. H. Paulheim, R. Plendl, F. Probst, and D. Oberle. Mapping Pragmatic Class Models to Reference Ontologies. In *2nd International Workshop on Data Engineering meets the Semantic Web (DESWeb)*, 2011. to appear.
14. H. Paulheim and F. Probst. Ontology-Enhanced User Interfaces: A Survey. *International Journal on Semantic Web and Information Systems*, 6(2):36–59, 2010.
15. RedWhale Software. The XIML Specification. Available as part of the XIML Starter Kit version 1, available at `http://www.ximl.org/download/step1.asp`, 2000. Accessed August 3rd, 2010.
16. F. Ruiz and J. R. Hilera. Using Ontologies in Software Engineering and Technology. In Calero et al. [3], chapter 2, pages 49–102.
17. K. A. Sergevich and G. V. Viktorovna. From an Ontology-Oriented Approach Conception to User Interface Development. *International Journal "Information Theories and Applications"*, 10(1):89–98, 2003.
18. P. Spyns, R. Meersmanand, and M. Jarrar. Data modelling versus ontology engineering. *SIGMOD Rec.*, 31(4):12–17, 2002.
19. M. Uschold and M. Gruninger. Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, 11:93–136, 1996.