erschienen in: Norbert Gronau, Hrsg., 4th Conference on Professional Knowledge Management - Experiences and Visions, volume 2, S. 119 - 126, GITO-Verlag Berlin, 2007.

Configurable Contexts for Experience Management

Mirjam Minor, Daniel Schmalen, Ralph Bergmann
University of Trier, Trier, Germany
surname@uni-trier.de

Andreas Koldehoff
sci-worx GmbH, Hannover, Germany
andreas.koldehoff@sci-worx.com

Abstract. *Process oriented experience is valuable knowledge for companies. A documentation of this experience includes the context in which processes have taken place. This paper addresses context-aware experience management by a new approach of configurable, ontology-based context models that can be transferred from process to process and from project to project. We illustrate our approach in a sample application from the chip design domain. Our work is a result from the close collaboration of the University of Trier with the chip design company sci-worx, Hannover*[1].

1. Introduction

Context-awareness is a recent research paradigm in knowledge and experience management, e.g. in pervasive approaches [KoM05], in process oriented knowledge management [EAM01], [Sch03], or in intelligent recommender systems [CBS06]. The challenges of context-awareness for knowledge and experience management systems are how to model, acquire, and maintain context information, how to use it in order to support the knowledge workers, how to transfer it to other systems, and how to reason about context.

In this paper, we present a new approach of configurable, ontology-based context models for the management and transfer of experience on processes within projects. Our work is motivated by an application scenario from the chip design with a focus on digital design processes. The approach is also applicable for other domains with flexible contexts, e.g. for agile software development projects.

The remainder of this paper is organized as follows: Section 2 describes the goals and requirements of a context model for the digital design domain.

[1] The authors acknowledge the Federal Ministry for Education and Science (BMBF) for funding the URANOS project under grant number 01M3075. We thank the involved chip designers from sci-worx, AMD and Infineon.

Section 3 presents the context model and its owl representation. Section 4 contains the basic concepts of the context acquisition tool. Finally, we discuss our approach and draw a conclusion in Section 5.

2. Context in digital design projects

Our work on context relies on Dey's definition of *context* [Dey01] as "any information that can be used to characterize the situation of an entity". The entities that we consider in the chip domain are the digital design projects at a certain stage of their life-cycle. They are characterized by sets of context factors. We consider factors from both the application and the design context. The *application context* concerns the requirements of the customers and the end customers, e.g. the users of a mobile phone, whereas the *design context* concerns the realisation of the design by tools and designers.

Goals of the context model

The goals of acquiring a context model are manifold in chip design: First, the model is used in order to *document factors* that have a significant impact on the design processes. All design processes follow a standard design flow which describes the design steps of a project ordered in a particular control flow. The context model is integrated with the design flow in order to realize the documentation of context factors as follows:

- The context model provides a central repository for important features like the description of pins and registers within a project. Typos in pin descriptions have been one of the main sources of error in past digital design projects.
- Changeable factors like the market development are reviewed regularly during the design flow and adapted to the current situation.
- The development of particular factors can be monitored for the assessment of projects.

At the moment, the context model is acquired manually but there is a certain potential for automating this documentation task in future, e.g. by importing data or by observing factors. As a second goal, the context model is used for *automatic analysis and reasoning*:

- The analysis of risk factors provides support for the risk management.
- The dependency analysis of context features is useful for the change request handling.
- We are planning to apply case-based reasoning for authoring support when design processes have to be adapted to changes.

Requirements analysis

The above goals have been figured out during a systematic requirements analysis including interviews and discussions with several digital design experts, our analysis of the chip domain by means of sample projects, and a literature research on context models.

The discussions with the chip designers have shown that the design processes of different types of chips depend on different sets of context factors. For instance, some chips in the automotive area are designed for multiple applications which might not even be known in advance. The application context for the design of such multiple-purpose chips is restricted to general quality and security aspects, i.e. the application context is not important enough to be modelled in detail. On the other hand, chips for the customized area are highly depending on the application context: The design processes are affected by recent market development issues, for instance like the number of pixels for a camera that is en vogue. Changes cause major adaptations of the design itself as well as of the ongoing design process. Thus, the acquisition of application context factors is crucial for the design process of chips from the customized area. The results from the discussion with the experts led us to the decision to develop a *configurable context model*. For each type of chip, the context model can be tailored to the appropriate set of context factors.

Furthermore, our analysis of sample documents from the chip domain has yielded several interdependencies between context factors. For instance, the output pins of a chip segment have a direct impact on the input pins of a successor chip. We have chosen to organize the context in an ontology-based model in order to *describe interdependencies*.

Kopipää et al [KMK03] (cited according to [BDR06]) identified requirements for ontology-based context models that hold also for our approach:

- *Simplicity*: The used expressions and relations should be as simple as possible to simplify the work of applications developers.
- *Flexibility* and *extensibility*: The ontology should support the simple addition of new context elements and relations.
- *Genericity***:** The ontology should not be limited to special kind of context atoms but rather support different types of context.
- *Expressiveness*: The ontology should allow to describe as much context states as possible in arbitrary detail.

The requirement of flexibility and extensibility includes the above requirement of configurable contexts that we derived from our analysis of the chip domain.

3. The context ontology

In order to realize the manifold goals and requirements described in Section 2, we developed a top-level ontology for our context model that is depicted in Figure 1. We distinguish two areas for the definition and the application of context factors: *FactorCategory*, *FactorDefinition* and all its descendant classes belong to the pool of context factors. The pool contains the definitions of the factors including the value types and default values. *Project*, *ProjectUnit* and *Factor* with its descendants are for the assignment of factors to project units and their unit-specific values. A *FactorDefinition* has a *FactorCategory*. A *Factor* is defined by a *FactorDefinition* and is a context factor of a *ProjectUnit* which belongs to a *Project*. A *Factor* may have an impact on other *Factor*s. The non-hatched classed in Figure 1 are fix for all context models; the hatched classes and can be adapted by the users in order to configure the context models.

A context factor is defined by an instance of one child of the *FactorDefinition* classes. For instance, the factor definition *Algorithms risk definition* is specified as an instance of the factor definition class *RiskFD*. Each factor definition class on the left hand side has a partner factor class on the right hand side. *BooleanFD* has *BooleanFactor*, *DateFD* has *DateFactor*, and so on. A context factor is applied by an instance of the partner child class of *Factor* with a link to the factor definition (*hasDefinition*) and to the project (*isContextFactorOf*). For example, the algorithms risk factor of the unit MC in the MSVE project is specified by *MSVE__MC__Algorithms risk* as an instance of *RiskFactor* with *MSVE__MC__Algorithms risk* [*hasDefinition->>Algorithms risk definition*] and *MSVE__MC__Algorithms risk* [*isContextFactorOf->>MSVE*]. The default value for algorithms risk factors is stored in *Algorithms risk definition*. The actual value of the algorithms risk of the unit *MC* in *MSVE* is stored in *MSVE__MC__Algorithms risk*.

The hatched classes are for user-defined pickup lists, e.g. for the electronic design automation tools (EDA tools) that are used within a design project. The factor definition for EDA tools requires its own data type class *PickupListMultipleValue__EDA tools__FD*. The applied EDA tools factors are instances of *PickupListMultipleValue__EDA tools__Factor*, e.g. *MSVE__MC__EDA tools*, *MSVE__DCT__EDA tools* etc.

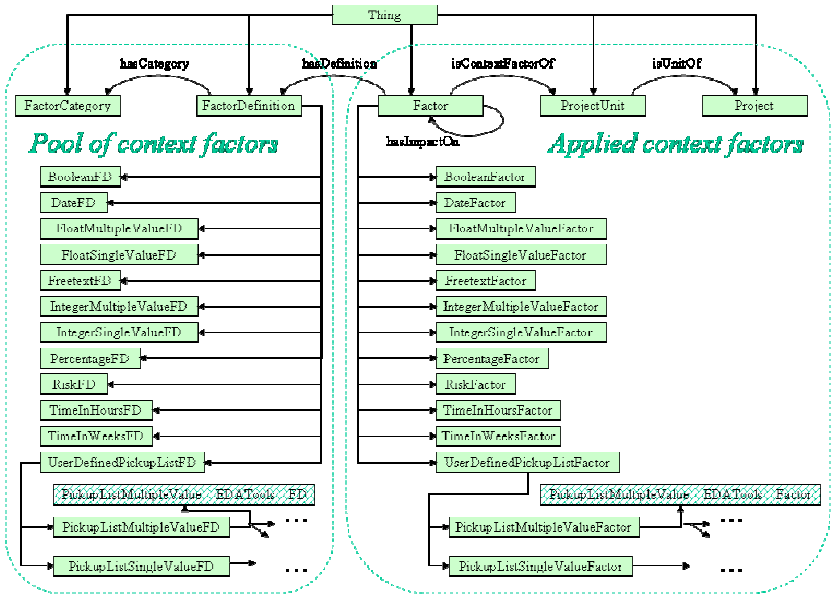

Fig. 1: The top level ontology of the context exchange format.

We use Protégé conform owl as a context exchange format. The users are able to exchange either the pool of factors only or in combination with the actual application of this pool to projects. We have selected owl because it extends RDF and XML by expressions in description logics (DL) that we use for some properties and restrictions of the relations, e.g. for the property that *hasDefinition* is functional and for the existence restriction concerning *hasDefinition* in *Factor*. These two sample expressions mean that *hasDefinition* must be specified for each instance of *Factor* with exactly one instance of *FactorDefinition*. For reasons of simplicity of the model, we have not specified syntactically that the partner instances of *FactorDefinition* and *Factor* should be from the partner children, e.g. from *RiskFD* and *Risk*; we leave this to the commonsense of the users or external tools that specify context models in owl that can be imported to our application. The users of our context acquisition tool are neither bothered with the separation of factor definitions and applied factors nor with the owl format as described in the following section.

4. The context acquisition tool

The context acquisition tool supports the users in modelling context factors. In the GUI, it distinguishes the pool of factors view from three other views for the application of context factors:

- *Pool of factors* is for the definition of context factors including their value types and default value. The names of the factor definitions are the same as the names of the factors applied to the projects. The tool automatically maps the names on unique ones when it stores them as instances of *FactorDefinition* and *Factor* in the owl file.
- *Project configuration* is for assigning factors to project units (see Fig. 2). When the user assigns a factor to a project unit, the tool automatically creates a new instance of *Factor* in the owl file.
- *Value assignment* is for the assignment of values. Initially, a new factor inherits the default value from the factor definition.
- *Dependencies* is the view for modelling dependencies between factors.

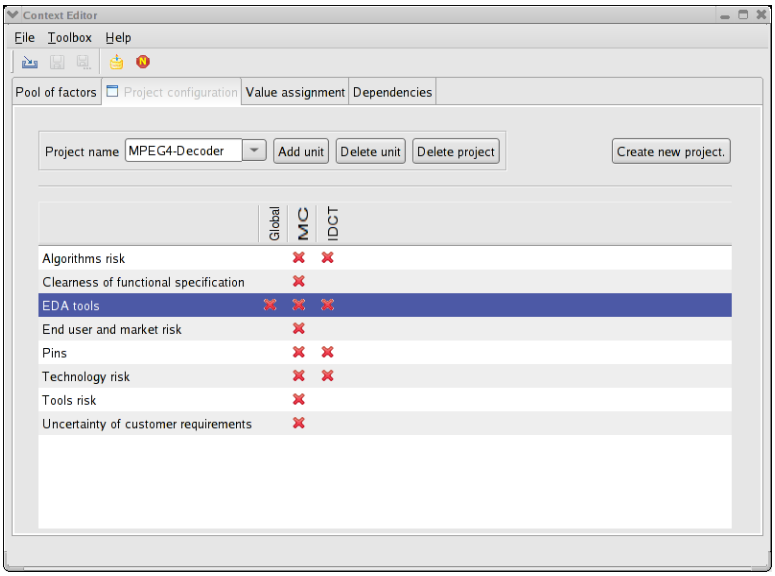

Fig. 2: Configuration of a project by assigning context factors to project units.

The context acquisition tool provides two analysis utilities for risk analysis and dependency analysis in order to demonstrate the usability already as a stand-alone tool. The deeper integration with the design processes and the

reasoning capabilities are outside the focus of the stand-alone tool. We are planning to do this by the integration with a workflow management tool that has a case-based authoring support facility.

5. Discussion and conclusion

We have presented a configurable, ontology-based context model which fulfils the goals and requirements that we proposed in Section 2: The *simplicity* is kept as the models consist of sets of context factors and dependencies between context factors which both can be specified in an easy manner. The models are *flexible and extensible* as the assignment of projects and values is agile. A certain degree of *genericity* is given by the configurable pool of context factors. The *expressiveness* of the model is restricted by the fact that we have only two levels of contexts and sub-contexts namely for projects and their units. Furthermore, the data types for context factors have to be selected from the given set of the thirteen data type classes in the top-level ontology. On the other hand, we allow an arbitrary composition and size of the sets of context factors in order to characterize the projects and project units. The simple exchange format in owl enables the experience transfer from project to project or even from organization to organization. We have illustrated our approach in a real-world application from the chip design domain.

Schwarz [Sch05a] applies context models for the retrieval of information items in order to satisfy the knowledge worker's potential information needs. The retrieval is for the rather personal view of a single knowledge worker who performs a short-term task of a workflow, e.g. to select a folder for a document to be stored in. In opposite to this, our approach bundles project-related knowledge for several users within one context model; the retrieval is performed for long-term processes rather than single tasks. Unlike Schwarz who observes context information, we do not yet need to assess the confidence of contexts, as the context factors are manually modelled.

Mendes de Araujo et al. [MSB04] discuss a complex context model with seven levels for the individual, the role, the team, the task, the project, the organization, and the domain area. They propose to retrieve contextual information from the organizational memory for group support in software development. This application area is very similar to the chip design area that we have considered. For reasons of simplicity, we restrict our context model to the project level only.

In conclusion, we have addressed the management and transfer of experience that is documented in context models for processes. Our

configurable, ontology-based context model balances the requirements of simplicity and expressiveness. The evaluation in a case study for a real-world application that will take place at the beginning of 2007 will show to what extend our approach is useful in order to support the chip developers in their design processes.

Literatur

[BDR06] Baldauf, M., Dustdar, S., Rosenberg, F.: A Survey On Context-Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing, Inderscience Publishers, 2006, (forthcoming).

[CBS06] Coyle, L., Balfe, E., Stevenson, G., Neely, S., Dobson, S., Nixon, P., Smyth, B.: Supplementing Case-Based Recommenders with Context Data. Workshop Proceedings: 8th European Conference on Case-Based Reasoning, Workshop "CBR and Context Awareness (CACOA 2006)", September 4-7, 2006, Ölüdeniz/Fethiye, Turkey, Universität Trier, Dept. of Business Information Systems II, S. 260 -272.

[Dey01] Dey, A. K.: Understanding and Using Context. Personal and Ubiquitous Computing 5/1, Springer-Verlag, 2001, S. 4-7.

[EAM01] van Elst, L., Abecker, A., Maus, H.: Exploiting User and Process Context for Knowledge Management Systems, Workshop on User Modeling for Context-Aware Applications at the 8th International Conference on User Modeling, July 13-16, 2001, Sonthofen, Germany.

[KMK03] Kopipää, P., Mäntyjärvi, J., Kela, J., Keränen, H., Malm, E.-J.: Managing context information in mobile devices. IEEE Pervasive Computing 2/3, 2003, S. 42 – 51.

[KoM05] Kofod-Petersen, A., Mikalsen, M.: Context: Representation and Reasoning – Representing and Reasoning about Context in a Mobile Environment. Revue d'Intelligence Artificielle 19, 2005, S. 479-498.

[MSB04] Mendes de Araujo, R., Santoro, F. M., Brezillon, P., da Silva Borges, M. R., Pereira da Rosa, M. G.: Context Models for Managing Collaborative Software Development Knowledge. Proceedings of the KI-04 Workshop on Modeling and Retrieval of Context, Ulm, September 20-21, 2004.

[Sch05a] Schwarz, S.: A Context Model for Personal Knowledge Management. In Proceedings of the 2nd International Workshop of Modelling and Retrieval of Context (MRC 2005) in conjunction with IJCAII 2005.

[Sch03] Schütt, P.: The post-Nonaka Knowledge Management, Proceedings of I-KNOW '03, Graz, Austria, July 2-4, 2003, S. 290-301.