01

KNOWLEDGE ACQUISITION FOR BUILDING AND INTEGRATING PRODUCT CONFIGURATORS[1]

A. Felfernig[*], G. Friedrich[*], D. Jannach[*], M. Zanker[*], and R. Schäfer[+]
[*]*Computer Science and Manufacturing Research Group, Universität Klagenfurt, Klagenfurt, Austria*
[+]*DFKI, Saarbrücken, Germany*

Nowadays configuration systems are typically standalone systems not supporting supply chain integration of configurable products and services. The goal of the EU-funded project CAWICOMS is the development of an integration platform for such systems that supports a personalized, distributed configuration process. One of the key tasks of such a platform is the effective support of configuration knowledge acquisition and interchange, which is a prerequisite for enabling communication among configuration systems. In this paper we present the principles of the CAWICOMS Knowledge Acquisition Component which supports the design of configuration knowledge bases and the integration of heterogeneous knowledge bases using a configuration domain-specific language based on the Unified Modeling Language (UML). Configuration models represented in this language can be imported into the CAWICOMS environment and furthermore be integrated with configuration models stemming from other supplier configuration systems.

1. INTRODUCTION

In today's rapidly changing, globalizing markets, traditional mass production paradigms appear anachronistic. Mass production is increasingly replaced by *customer-individual production of variant products*. Companies have to differentiate their product spectrum for fulfilling the customers' individual needs. The additional costs for offering individual products must be minimized in order to be competitive. *Mass Customization* (PineII, Victor and Boynton, 1993) is a new paradigm representing the trend towards the production of highly variant products under mass production time and pricing conditions. This paradigm imposes increasing demands on the development and maintenance of software supporting processes related to variant products. This software must be able to handle rapidly changing, complex constraints on products and corresponding business processes supporting quotation, order processing, production, delivery, and maintenance. *Configuration systems* (configurators) are an important prerequisite for the effective implementation of processes supporting a mass customization business strategy. The application of configuration systems leads to reduced response times to customer requirements by effectively supporting quotation and order processing. Furthermore, configuration

[1] This work takes place with the financial support of the IST Program of the European Union under contract IST-1999-10688.

technology avoids invalid orders which consequently decreases time between sales and delivery/installation of the product.

Business processes are no longer restricted to single enterprises, but transcend companies' boundaries along the value chain of products and services. In this context, supply chain integration of specialized solution providers is an important issue. In the telecommunication industry, solution providers configure enterprise networks based on switching hardware solutions. Add-on products for these integrated enterprise configurations are ordered from sub-suppliers (e.g. add-on applications providing *voice-over-ip* functionality). Products of sub-suppliers must themselves be configured. Since there are dependencies between the parts provided by different suppliers, a distributed configuration approach must be supported in order to enable the calculation of a consistent distributed solution. For privacy reasons as well as for reasons of different knowledge representation formalisms, a centralized approach for solving configuration tasks is not feasible. However - in order to allow the calculation of a distributed configuration solution - parts of the configuration knowledge must be shared between the involved configuration systems.

In the following sections, we show how knowledge acquisition and knowledge interchange is realized in CAWICOMS[2], which aims to support a distributed and personalized configuration process for configurable products and services. Figure 1 shows the overall architecture of the CAWICOMS environment.

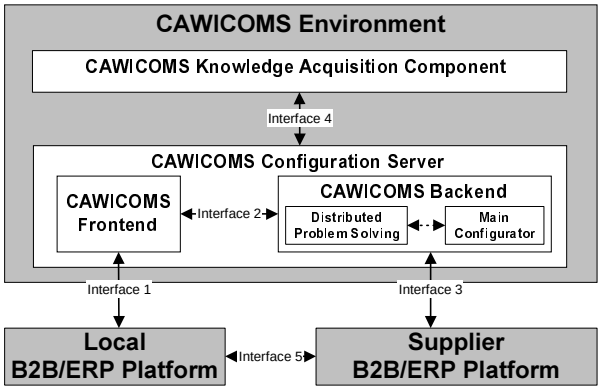

Figure 1: Overall CAWICOMS Architecture

This architecture is based on the following scenario. The user of a B2B/ERP platform[3] navigates through a product catalog. When the user selects a configurable product (e.g. an integrated telecommunication solution), the CAWICOMS environment is activated and the distributed configuration process is initiated via the Frontend (Interface 1). Parameters representing user requirements are exchanged between Frontend and Backend via Interface 2. The distributed configuration

[2] CAWICOMS is the acronym for Customer-adaptive Web Interface for the Configuration of products and services with Multiple Suppliers.

[3] B2B/ERP platform denotes a kind of online store, electronic marketplace, or ERP system.

process is coordinated by the CAWICOMS Backend via Interface 3, i.e. Interface 3 supports the communication between the involved configuration systems. Within this scenario, the *Frontend* is responsible for personalizing the presentation of a configurable product. Personalization is done according to the user's needs. Here, it is also possible to take into account company specific rules which reflect the company's policies. For example, it could be specified which sub-suppliers should be preferred when configuring a product. The *Backend* is responsible for coordinating the distributed configuration process, whereby the *Distributed Problem Solving* component is responsible for coordinating the communication with remote configurators, and the *Main Configurator* component is responsible for calculating local solutions and forwarding requirements to remote configurators[4]. Interface 4 enables inclusion of configuration models stemming from the Knowledge Acquisition Component into the CAWICOMS Configuration Server. Finally, Interface 5 represents established B2B channels between sub-suppliers and integrated-solution providers[5]. The CAWICOMS *Knowledge Acquisition Component* is responsible for supporting an effective design process for configuration models and for integrating partial supplier configuration models for knowledge sharing purposes.

The paper is organized as follows. Section 2 discusses knowledge representation concepts of the CAWICOMS Knowledge Acquisition Component, which supports effective knowledge acquisition and knowledge sharing in heterogeneous configuration environments. On this basis, Section 3 discusses the knowledge acquisition process inside the CAWICOMS Knowledge Acquisition workbench. Section 4 sketches the application of this workbench for realizing knowledge sharing between different configuration environments. Section 5 and Section 6 contain related work and conclusions.

2. KNOWLEDGE ACQUISITION IN CAWICOMS

One goal of the CAWICOMS project is to provide a knowledge acquisition environment which supports the design of configuration knowledge bases for different configuration systems. Such a flexible modeling environment must be based on an integrated extensible configuration ontology. Similar to the standardization of object-oriented modeling languages which resulted in the formulation of the Unified Modeling Language (UML) (Rumbaugh, Jacobson and Booch, 1998), our approach is to trigger a standardization process resulting in a standardized configuration language we call the *Generic Configuration Language* (GCL). The introduction of such a language has the following advantages.

- The integration of PDM (Product Data Management) standards with configuration technology is still an open issue. UML 2.0 is intended to be synchronized with the EXPRESS language, which is the standard language for defining STEP (ISO, 1994) application protocols (STEP standards). Using UML in this context alleviates the integration of configuration and PDM

[4] The CAWICOMS Frontend as well as the Backend have further subcomponents – in this paper only the Backend components are shown in further detail.

[5] Note that the interfaces shown in Figure 1 represent a simplified subset of the interfaces defined within the CAWICOMS environment.

technology since the same basic representation language is used for representing configuration knowledge and PDM knowledge.

- UML is widely applied in industrial software development as a standard design language supporting the whole software development process starting with the requirements analysis phase up to the implementation phase.
- UML is extensible for domain-specific purposes, i.e. the semantics of the basic modeling concepts of the language can be further refined in order to be able to provide domain-specific modeling concepts, which allow a more intuitive construction of the corresponding models.
- The Object Constraint Language (OCL) (Warmer and Kleppe, 1999) is a built-in constraint language supporting a formal definition of constraints on the models, which were built using the diagrammatic notations of UML.
- Finally, we have made excellent experiences in using UML designs for validation by technical experts.

Before we start to sketch the ideas behind GCL, we give a short introduction into the ideas behind UML 2.0. A first version of the UML 2.0 standard is announced for the end of 2001. Within this standardization process the goal of the precise UML (pUML) group (see Evans and Kent, 1999) is to introduce a meta-modeling language (MML) with clear semantics, which can be used for constructing modeling languages for specific application domains[6]. Consequently - compared to the extension mechanisms available in the version 1.4 of UML – in UML 2.0, extension mechanisms will be the building blocks of the language. (Clark, Evans, Kent, and Sammut, 2001) point out that the current semantics definition of UML uses a semi-formal notation (OCL – Object Constraint Language) to represent syntactic rules on the meta-model and natural language text for the rest. (Clark, Evans, Kent, and Sammut, 2001) present the Meta Modeling Language (MML), which should be central part of UML 2.0 and which should be used to define the semantics of UML 2.0[7]. Extensibility in MML is provided by the notion of class-based inheritance and the notion of package extension which is derived from the extension mechanisms of Catalysis (D`Souza, Wills 1998). A prototype implementation of MML is already available and is tested within the CAWICOMS project for the definition of the first version of GCL.

GCL consists of a set of UML 2.0 packages which represent modeling concepts of state-of-the-art configuration systems. We identified a set of useful modeling concepts widely used in the configuration domain. These commonly used concepts are contained in a basic GCL configuration package. This basic package is extended with concepts of particular configuration languages - within CAWICOMS the first step was to integrate the ILOG JConfigurator (Junker, 2001) representation language. Such configurator-specific concepts are integrated into specializations of the basic GCL package.

The left hand side of Figure 2 contains an MML definition of the *CClass* (abbreviation for *configuration class*) modeling concept defined in the GCL *Basic* package. This modeling concept represents basic entities constituting a product structure (e.g. component types or function types). In addition to the attributes

[6] In the following we will show how this can be done for the configuration domain.

[7] An executable version of OCL is a subset of this language.

defined for *CClass*, a set of constraints (denoted as well-formedness-rules in the actually available UML versions) is defined which restrict the usage of the modeling concept *CClass*. The left hand side example in Figure 2 contains one constraint, which states the fact that if a *CClass* instance is the root of a component hierarchy *(isroot=true),* then this *CClass* instance must not have any superclasses (*SuperClasses->size=0*) and this instance must not be part of any other *CClass* instance (*PartOfs->size=0*).

```
Package "Basic"
...
// A Configuration Class
 class CClass
  name : String;
  stereotype : String;
  id : String;
  isroot : Boolean;
  SuperClasses : Set(basic.CClass);
  SubClasses : Set(basic.CClass);
  Attributes : Set(basic.CAttribute);
  PartOfs : Set(basic.CAssociation);
  HasParts : Set(basic.CAssociation);
  UsedClasses : Set(basic.CAssociation);
  Constraint : String

  inv
   "RootProperty"
   self.isroot = true implies
        self.SuperClasses->size=0 and
        self.PartOfs->size=0
 end;
...
```

```
Package "JConfig"
...
class CClassJCONFIG extends basic.CClass
  iIoInstancesCount : Integer;
  scope : String

  inv
   "StereotypeProperty"
   self.stereotype = "Class"
 end;
...
```

Figure 2: GCL Package definitions

On the right hand side of Figure 2 a configurator-specific extension of the basic *CClass* concept named *CClassJCONFIGURATOR* is introduced. Additional configurator-specific attributes are added and further constraints are specified which only hold in the context of the considered configuration environment. Class-based inheritance is used to specify the new class *CClassJCONFIGURATOR*. The building blocks of the JConfigurator configurator are classes, where the attribute *stereotype* is set to "*Class"* – this is assured by the well-formedness-rule *StereotypeProperty*.
Note, that the MML prototype presented in (Clark, Evans, Kent, and Sammut, 2001) allows automatic checking of instances of a GCL package against the definitions on the meta-level. The class definitions of Figure 3[8] represent *CClass* instances corresponding to the definition shown in Figure 2[9]. This configuration model is syntactically correct w.r.t. the definitions given in Figure 2.

3. KNOWLEDGE ACQUISITION PROCESS

The following tasks constitute the knowledge acquisition process defined for the CAWICOMS *Knowledge Acquisition Component*:

[8] The TeCOM class represents the central component type of an integrated telecommunication solution.
[9] The attributes *id* and *isroot* are hidden in Figure 3.

- *Design the Configuration Model:* In order to design a configuration knowledge base for a certain configuration environment, a subset of GCL is used, i.e. those concepts are used, which are contained in the package of the corresponding configuration system (e.g. package *JConfigurator*). Figure 3 contains an example configuration model representing the basic structure of an integrated telecommunication switch. In the current version of the CAWICOMS Knowledge Acquisition Component constraints on the product model are represented as ILOG JConfigurator-specific business rules. In future versions this language will be replaced by OCL (Warmer, Kleppe, 1999).
- *Check the syntactic usage of the used modeling concepts* The usage of the modeling concepts must correspond to the well-formedness-rules[10] of the meta-model. This check can be done automatically using the MML interpreter provided by (Clark, Evans, Kent, and Sammut, 2001). In our case the complete TeCOM model (see Figure 3) is checked against the corresponding package definitions (e.g. package *JConfigurator*).
- *Generate a valid XML instance:* The UML model of the configurable product is translated into an XML (W3C, 1999) instance[11]. The XML instance represents the basic configuration model, which is further used by the CAWICOMS Frontend in order to add additional personalization information. For reasons of space limitations an example XML instance is omitted here.
- *Generate the configuration knowledge base:* Within CAWICOMS, the XML instance is translated into the ILOG JConfigurator representation. This translation into the ILOG-specific representation is realized using an XSLT stylesheet (Kay, 2000). Such a stylesheet must be provided for each configuration environment where a corresponding knowledge base generation should be supported.

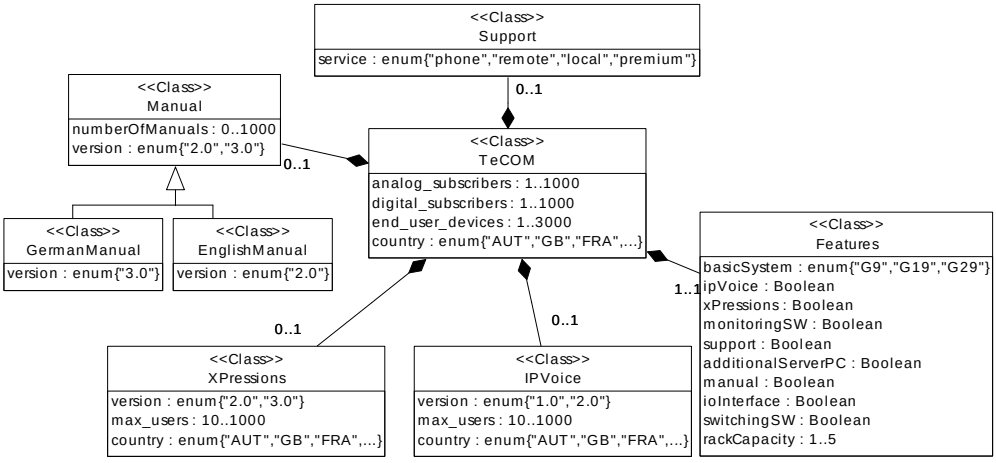

Figure 3: Configuration Model of Telecommunication Solution Configurator

[10] Rules defining restrictions on the syntactic usage of the modeling concepts.

[11] The configuration model is translated into an XML representation in order to alleviate further transformation operations (e.g. XSLT transformations).

4. KNOWLEDGE SHARING FOR DISTRIBUTED CONFIGURATION

Up to now we have discussed basic representation concepts and tasks supported by the CAWICOMS Knowledge Acquisition Component. As stated in the introduction, this component should support knowledge sharing between different configuration environments in order to enable effective communication between the underlying configuration systems. The CAWICOMS Configuration Server Backend (see Figure 1) is responsible for coordinating the distributed configuration process. Partial product models imported from remote configuration knowledge bases are integrated into the local configuration model of the Main Configurator. These imported product models can be seen as a functional view on the complete configuration model only seen by the supplier configurator. In order to support their integration into the CAWICOMS environment, the functional description must be formulated in GCL. This approach is similar to the integration of simple product catalogue information (Fensel, Ding, Omelayenko, Schulten, Botquin, Brown, Flett, 2001) in electronic marketplace environments, where proprietary supplier catalogs are translated into the representation of the marketplace environment. In CAWICOMS this basic functionality is also provided for configurable products.

5. RELATED WORK

In recent years several fields of AI focused research on the improvement of the inter-operability of knowledge-based systems by developing and sharing ontologies. In (Chandrasekaran, Josephson and Benjamins, 1999) an ontology is defined as a theory about the sorts of objects, properties of objects, and relationships between objects that are possible in a specified domain of knowledge. In (Soininen et. al, 1998) such an ontology is defined for the configuration domain. This ontology is based on the frame ontology of Ontolingua (Gruber, 1992) and represents a synthesis of resource-based, function-based, connection-based, and structure-based approaches to represent configuration problems. Similar concepts are contained in the CAWICOMS configuration knowledge representation language, whereby precise semantics for the used concepts are given by the formulation of rules for translating the UML representation into the representation of the corresponding configuration system. The proposed approach to represent configuration knowledge using techniques from Software Engineering is very similar to currently developed knowledge representation techniques in the context of the Semantic Web (Fensel, van Harmelen, Horrocks, McGuiness, and Patel-Schneider, 2001). Within CAWICOMS, we currently compare both approaches w.r.t. their applicability to support Web-based distributed configuration. First results indicate that Semantic Web approaches provide a well defined semantics for the included modeling concepts and support a complete representation of ontologies using basic XML techniques (which will also be the case in future versions of UML).

6. CONCLUSIONS

State-of-the-art configuration systems are typically standalone systems not supporting supply-chain integration of configurable products and services. Those systems are based on proprietary knowledge representation formalisms which

complicate knowledge acquisition tasks and the integration of those systems in order to support distributed configuration processes. The goal of the CAWICOMS project is the development of an integration platform for such systems which supports a personalized, distributed configuration process. In this paper, we focused on the discussion of the CAWICOMS Knowledge Acquisition Component which is based on the extension concepts provided by the Unified Modeling Language. This component supports a standardized representation of configuration problems and configuration knowledge interchange between different configuration environments.

REFERENCES

1. Chandrasekaran B, Josephson J, Benjamins R. What Are Ontologies, and Why do we Need Them? IEEE Intelligent Systems, 14,1:20-26, 1999.
2. Clark T, Evans E, Kent St, Sammut P. The MMF Approach to Engineering Object-Oriented Design Languages, Workshop on Language Descriptions, Tools and Applications, April 2001.
3. Cranefield S and Purvis M. UML as an Ontology Modelling Language. In Proceedings of the Workshop on Intelligent Information Integration, 16th International Conference on Artificial Intelligence, Stockholm, Sweden, 1999.
4. D'Souza D, Wills AC. Object Components and Frameworks with UML – The Catalysis Approach. Addison Wesley, 1998.
5. Evans AS, Kent S. Meta-modelling semantics of UML: the pUML approach. The Second International Conference on the Unified Modeling Language. Editors: Rumpe B and France RB, Colorado, LNCS 1723, 1999.
6. Fensel D, van Harmelen F, Horrocks I, McGuiness D, Patel-Schneider P.F. OIL: An Ontology Infrastructure for the Semantic Web, IEEE Intelligent Systems, 16,2:38-45, 2001.
7. Fensel D, Ding Y, Omelayenko B, Schulten E, Botquin G, Brown M, Flett A. Product Data Integration in B2B E-Commerce, IEEE Intelligent Systems, 16,4:54-59, 2001.
8. Gruber T. Ontolingua: A mechanism to support portable ontologies. Technical Report KSL 91-66, 1992.
9. Junker U. Preference-programming for Configuration, *Proc. IJCAI'01 - Configuration Workshop*, Seattle, Wa, August, 2001.
10. Kay M. XSLT Programmer's Reference. Wrox Press, 2000.
11. Männistö T, Martio A, Sulonen R. Modelling generic product structures in STEP. Computer-Aided Design, 30,14:1111-1118, 1999.
12. PineII BJ, Victor B, Boynton AC. Making Mass Customization Work. Harvard Business Review, Sep./Oct. 1993:109-119, 1993.
13. Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual. Addison Wesley, 1998.
14. Soininen T, Tiihonen J, Männistö T, Sulonen R. Towards a General Ontology of Configuration. AIEDAM – AI Engineering Design Analysis and Manufacturing Journal, Special Issue: Configuration Design, 12,4:357-372, 1998.
15. ISO Standard 10303-1: Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles, 1994.
16. W3C. Extensible Markup Language (XML). www.w3c.org, 1999.
17. Warmer J, Kleppe A. The Object Constraint Language – Precise Modeling with UML. Addison Wesley Object Technology Series, 1999.