In Proceedings of the 12th International Conference on Engineering Design (ICED'99), Vol. 2, ed. Lindemann, U., Birkhofer, H., Meer-kamm, H. and Vajna, S., Munich, August 24-26, 1999. Garching, Germany 1999, Technische Universität München, Konstruktion im Maschinenbau, pp. 1139-1142.

INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 99 MUNICH, AUGUST 24-26, 1999

MODELING CONFIGURABLE PRODUCT FAMILIES

Juha Tiihonen, Timo Lehtonen, Timo Soininen, Antti Pulkkinen, Reijo Sulonen, Asko Riitahuhta

Keywords: Product configuration, product modeling, conceptual analysis, case study

1 Introduction

Configurable products offer a way to efficiently manage product families with large numbers of variants. Some one-of-a-kind products can also be transformed to configurable products. Product configuration requires both systematic processes and appropriate modeling. We outline a method for modeling a product family as a configurable product. The method is based on a recently proposed object-oriented conceptualization of configuration [1]. We illustrate the conceptualization and the method through modeling rock drilling equipment.

2 Modeling on the basis of the conceptualization

2.1 Modeling concepts

Configuration knowledge categories. We distinguish between three categories of configuration knowledge: configuration solution knowledge, configuration model knowledge, and requirements knowledge. *Configuration solution knowledge* specifies a configuration. A *configuration* is a description of a product individual for an order. *Configuration model knowledge* specifies the entities that can appear in a configuration, their properties and the rules on how the entities and their properties can be combined. Thus, configuration model knowledge specifies the set of *correct* (see [2]) configurations of a product. This knowledge is represented in a *configuration model*. The configuration model is an abstraction of the real world product family specifically meant for configuration purposes. *Requirements knowledge* specifies the systematized requirements on the configuration to be constructed. Requirements knowledge can in our view be specified with the same concepts as configuration model knowledge and configuration solution knowledge, although it plays a different role in problem solving. For this reason, we explicitly discuss only configuration model knowledge and configuration solution knowledge.

Classification hierarchy. We introduce the concepts of a *type* and an *individual* to clearly distinguish between the entities that occur in configuration model knowledge and configuration solution knowledge. A configuration can contain individuals of subtypes of the following main types of configuration model knowledge: component, port, resource and function. Main types are organized in a *classification hierarchy* in the usual manner [3]. A type has a set of *property definitions* such as attribute, part and port definitions. A type *inherits* the properties of its *supertypes* in the classification hierarchy. In other words properties from each supertype of a type are “added” to those of the component type itself. A *subtype* can *refine* the property definitions it has inherited. A type is either abstract or

concrete. An individual directly of a *concrete type* is accurate enough to be used in an unambiguous configuration. An individual directly of an *abstract type* provides only partial information on the real world entity it represents. One should consider carefully which properties belong to a given type and which ones to its subtypes or supertypes.

Attributes Component, port, resource and function types can define *attributes*. These represent the characteristics of an individual of the type. Some attributes have fixed *values* and others can be given a value. Attributes can be used to parameterize components with respect to some properties. Typical attributes include physical dimensions of parametric components, surface material, color, resistance, and capacity.

Components and structure. The conceptualization directly supports generalized product structures with a varying number of mandatory, optional and alternative parts via *part definitions* in *component types*. A part definition defines a part role, which is filled by *component individual*(s) in a configuration. E.g. component type Lamp could have part definitions for roles Lampshade and a Stand. Each part definition specifies the component types whose individuals are viable alternatives for that part role.

Ports. We model connection interfaces as ports. Component types specify their connection possibilities by *port definitions*. A *port type* is a definition of a connection interface. A *port individual* represents a "place" where in a component individual some other port individual can be connected. The connections modeled can be physical or logical. A port type has a *compatibility definition* that defines a set of port types whose port individuals can be connected to the port individuals of that port type. Ports are especially suitable for modeling compatibility expressed as interfaces, which is often the case for modular products. Quite often connections require connecting components like cables or pipes. It can be difficult to decide whether a connecting component needs to be modeled or if it can be abstracted away.

Resources. Resource-oriented concepts model the production and use of some more or less abstract entity, for example power or expansion slots. The underlying idea is that some component individual(s) produce a *resource* and other component individual(s) use it. Resource production and use must be either *satisfied*, in which case the quantity of resource produced must be at least equal to the quantity of the resource used or *balanced*, in which case the quantity of resource produced must be equal to the quantity of the resource used.

Functions. Function-oriented concepts represent the functionality that a product individual provides to the customer, the user of the product or the environment. The basic concepts are *function type* and *function*. Functions can be divided to sub-functions.

Constraints. *Constraints* provide a general mechanism for specifying the interdependencies of entities in the configuration model. A constraint is a formal rule that specifies a condition that must hold in a correct configuration. Constraints are used when the other concepts do not capture the intended meaning adequately or conveniently.

2.2 Modeling guidelines

A configuration model is based on analyzing the product. Therefore the modeler should have a good understanding of the product. We consider object-oriented analysis and modeling [3] a good method for configuration knowledge modeling with the conceptualization. In the first phase the primary concern is to recognize the relevant entities. These are subtypes of the concepts in the conceptualization. After that, and to some extent in parallel with the previous

stage, relationships between the entities and their properties are identified. Again, the conceptualization provides the relevant properties and relations. In parallel with the previous steps, classification hierarchies of component, resource, port and function types are constructed to capture common characteristics. Constraints are identified and modeled.

3 Example

We use a heavy rock-drilling machine produced by Tamrock Corp. as an example of a configurable product. A Ranger consists of a body, a tracked crawler base, a boom and drilling equipment. The body is divided into power unit, cabin, fuel oil tank, and hydraulic oil tank. The Ranger product family has three variations in the main functional property (drilling dimension) and numerous variations of secondary properties. Altogether there are over 200 000 possible variants, but Tamrock Corp. regards 72 of these as substantially different — this variation is also presented in the developed model.

In Figure 1 a part of the configuration model of Ranger is presented. The figure does not show port classification and compatibility. The product has been modeled mainly with part definitions. This means that model defines the roles in the product structure and what types of components are needed in these roles.

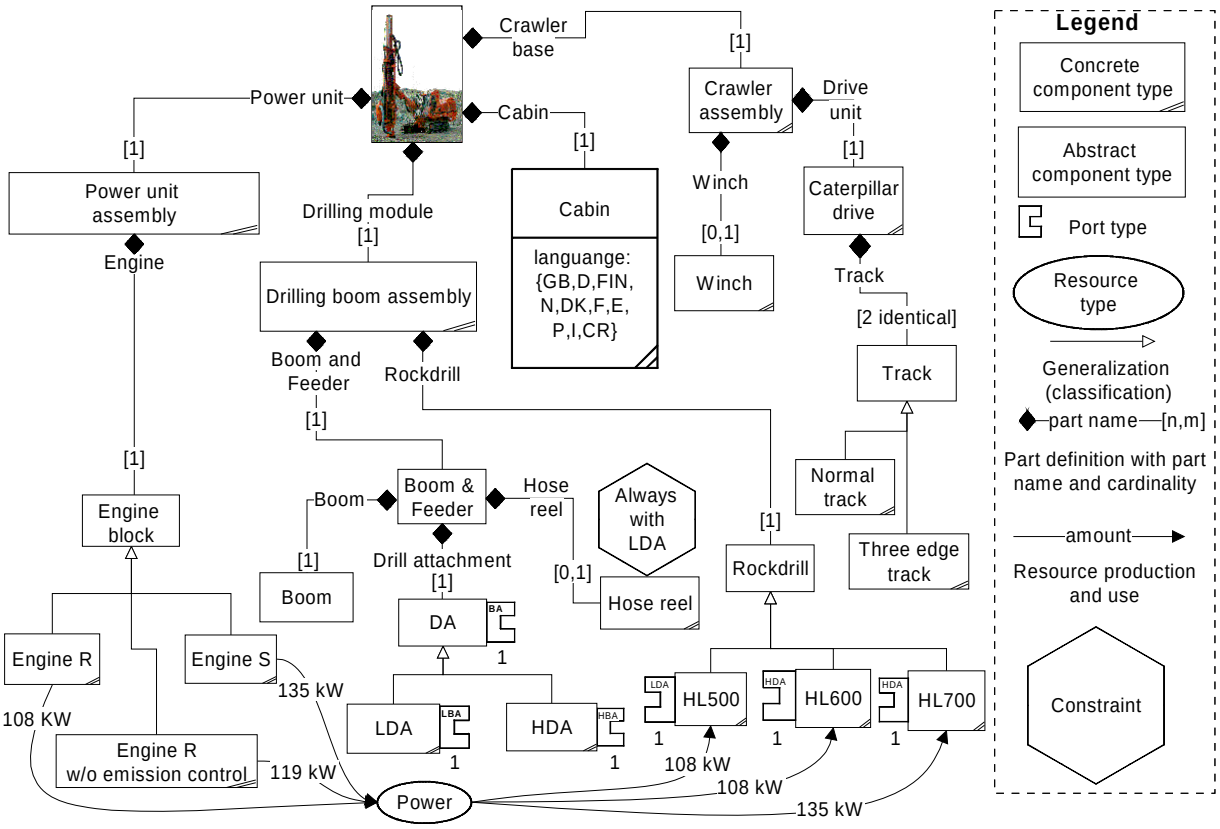

Figure 1. Part of the Ranger configuration model

As an example of part definition, Ranger has part role Power unit, which is fulfilled with an instance of component type Power unit assembly. The role Engine of Power unit assembly is filled with an instance of component type Engine block or one of its subtypes. Here the abstract component type Engine block is modeled as a supertype of the three concrete component types

Engine R, Engine S, and Engine R w/o emission control. These types inherit the properties of Engine block, but differ in some aspects. The type of feeder is dependent on the type of Rockdrill. This dependency is modeled using port types and their compatibilities, as the Rockdrill needs to be connected to the Boom. The power requirement of rockdrills varies. This is modeled using resources. The engines produce different amounts of Power. The rockdrills use this resource, represented in the bottom of Figure 1. The use of Power must be satisfied, i.e. it must be produced in at least the amount it is used.

4 Conclusions

Managing product families consisting of a large set of product variants as configurable products requires defining a configuration model. We briefly described a conceptualization for configuration. The conceptualization was evaluated by modeling a case product and was found to cover the relevant modeling needs fairly well. However, several improvements to the conceptualization were identified. For example, more elaborate mechanisms for connection modeling would be useful. The concepts and modeling guidelines for them should be extended, refined, and further validated. Modeling with the conceptualization sets new requirements for the designer. In addition to having a good understanding of the product, a designer should be familiar with object oriented modeling. Information system support is necessary for modeling real-world-sized products. We feel that the conceptualization can also be used in the product development process. The main benefit there would be improving communication within the product development team and to other functions of the company.

We thank Technology Development Centre Finland and Helsinki Graduate School in Computer Science and Engineering for financial support. We also thank Tamrock Corp.

References

[1] Soininen, T., Tiihonen, J., Männistö, T. and Sulonen, R. Towards a General Ontology of Configuration. In *AI EDAM (Artificial Intelligence for Engineering Design, Analysis and Manufacturing)*, Vol. 12, No. 4, pages 357-372, 1998.

[2] Peltonen, H., Männistö, T., Soininen, T., Tiihonen, J., Martio, A. and Sulonen, R. Concepts for Modelling Configurable Products. In *Proceedings of European Conference Product Data Technology Days 1998*, pages 189-196. Quality Marketing Services, Sandhurst, UK, 1998.

[3] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. *Object-Oriented Modelling and Design*. Prentice-Hall, Inc., Eaglewood Cliffs, 1991.

Juha Tiihonen,Timo Soininen,
Reijo Sulonen
Helsinki University of Technology
TAI Research Centre
P.O. Box 9555
FIN-02015 HUT
Finland
tel . +358-9-451 3242
fax. +358-9-451 4958
{jti, tso, shs}@cs.hut.fi

Timo Lehtonen, Antti Pulkkinen,
Asko Riitahuhta
Tampere University of Technology
Machine Design Laboratory
P.O. BOX 589
FIN-33101 Tampere
Finland
tel. +358-3-365 2627
fax. +358-3-365 2307
{tle, pulkkine, aor}@me.tut.fi