Framework and Conceptual Model for Reconfiguration

Tomi Männistö, Timo Soininen, Juha Tiihonen and Reijo Sulonen

Product Data Management Group
TAI Research Centre and Laboratory of Information Processing Science
Helsinki University of Technology, P.O. Box 9555, FIN-02015 HUT, Espoo, FINLAND
{Tomi.Mannisto, Timo.Soininen, Juha.Tiihonen, Reijo.Sulonen}@hut.fi

Abstract

Reconfiguration is a significant area of after-sales, especially for companies with configurable products. In this paper, the field of reconfiguration is characterised and a framework for reconfiguration is presented on the basis of experiences gathered from Finnish manufacturing companies. Although the state of the practice leaves much room for improvement, reconfiguration is an important business in several companies. Based on the experiences, an abstract conceptual model for reconfiguration is defined.

Introduction

After-sales is an increasingly important business area for many companies. There is a need to provide support to and maintenance for *product individuals* for extended periods. One important task in the after-sales is to *reconfigure* an existing product individual, i.e., to modify it to meet new requirements. Although product configuration tasks and their computer support have been actively studied recently (see e.g., [1]), reconfiguration has not been satisfactorily dealt with [2, 3, 4].

One of the main differences between configuration and reconfiguration is the existing product individual, which in reconfiguration serves as an input to the process. For example, the platform for the machinery of an elevator has holes for the ropes. In configuration these holes are output as the platform is manufactured according to the specification; in reconfiguration, the holes act as inputs and their placement is very much fixed.

In this paper, we define a simple framework for reconfiguration that outlines the characteristics of reconfiguration from the business perspective. The framework is based on our experiences on reconfiguration in two dozen industrial companies, which we briefly summarise. We then argue that reconfiguration is not always adequately supported by similar means as configuration. We define a conceptual model for the reconfiguration task and knowledge that synthesizes and generalises our experiences. Finally, the framework and model are discussed and topics for further work are given.

Our model differs from most of the approaches to configuration, which simplify the reality by assuming that a configuration model always represents the current situation—there is no history, future or changes of any kind to the configuration model. Some authors have proposed models for reconfiguration [5] or propose integration of change management to configuration models [1, 3].

A closely related area to reconfiguration is schema-evolution in databases, which addresses manipulating old individuals with a new schema [6, 7]. There is, however, a fundamental difference in the nature of schema-evolution and evolution of configuration models. When a configuration model is modified, it does not represent the same product individuals with a different model—it represents a different set of product individuals. For example, when a new component type is included in a product, this does not mean that the old individuals should be converted to have such a component. Schema-evolution also occurs in configuration models, but is more related to database management than to product evolution.

The main terminology of this paper is as follows. A product individual is manufactured according to a *configuration* that satisfies a *configuration model.* A configuration model defines, among other things, *component types*, which allows us to say a *component individual* is of a certain component type. Term *product* typically refers to a whole and *component* to a constituent of a product. Furthermore, we assume that the distinction between an individual as a physical entity and its description in an information system is clear from the context.

Reconfiguration Framework

The feasibility of reconfiguration depends on the business. In this section, we characterise reconfiguration as a business and describe some factors affecting its feasibility. We then describe different modes of reconfiguration and report our experiences with case companies.

Feasibility of Reconfiguration

Some trends make reconfiguration more desirable than before. For example, the waste of natural resources can be lessened by extending the lifetime of a product individual and tighter investment budgets make extensible products attractive.

Maintaining reconfigurability. Reconfiguration is problematic since it cannibalises the markets from new products. In addition, maintaining reconfigurability can make introduction of new features based on new technology more difficult. These arguments easily lead to trade-offs between developing new products and extending the reconfigurability of old ones.

Customers. Reconfigurable products also tend to bind customers to the company. Even a relatively minor reconfiguration may allow significant added value to the customer, thus providing wider profit margin for the manufacturer. Furthermore, after-sales is typically less

dependent on the economic fluctuations than the sales of new products.

Type of product. In addition to business aspects, the feasibility of reconfiguration is also affected by the type of the product. One-of-a-kind industrial products are typically complex products that are designed and manufactured as separate projects. Examples include power plants and paper machines. It is difficult to define systematic reusable knowledge on such products, e.g., for reconfiguration, which implies that reconfiguration must also be carried out in projects. The most relevant type of products for reconfiguration seems to be configurable products since they allow systematisation of the necessary knowledge. Modular fixed products may resemble configurable products in this sense.

A relevant factor in reconfiguration is also whether the product individual is manufactured by the reconfiguring company or by a competitor.

Product cost affects the feasibility of reconfiguration, as it can be more economical to buy a new inexpensive product individual than to reconfigure an old one.

Lifetime of product individuals is another factor that affects reconfiguration. Product individuals with long life times and high cost are typically investments, which are often modernised at some point. Reconfiguration is seldom relevant for commodity products due to short life times and low cost of product individuals.

The rate of technological change also reflects on reconfiguration—the higher the rate, the more problematic the management of reconfiguration is. A very high change rate can make reconfiguration unfeasible by quickly outdating most of the product and thus requiring the changes to almost every component. This is currently the situation for modernisation of personal computers.

Experiences and Modes of Reconfiguration

Next, we briefly describe different modes of reconfiguration and our experiences in 24 Finnish companies that manufacture configurable or partially configurable products. The experiences were gathered in the period from 1994 to 1999. We have made a configuration survey of ten companies [8], a design for configuration survey in seven companies and cooperated with some of the surveyed and several other companies in research projects addressing product data management, configuration or design for configuration.

The importance of reconfiguration varied. Fifteen companies supported reconfiguration in some form. However, only six of them considered reconfiguration a significant business. These companies operated in telecommunications, heavy machinery (4), and medical instrumentation. We next describe five modes of reconfiguration and discuss the related experiences. Not all categories are mutually exclusive, i.e., some companies belonged to more than one category.

No reconfiguration. Companies that do not support reconfiguration belong to this category. Eleven companies had no reconfiguration.

Project. Reconfiguration is a part of after-sales operations, but it is not systematically supported. Consequently, separate projects design and implement the needed modifications, which makes reconfiguration expensive. In case of expensive investment products, the costs may still be justified. This category was common for companies (9) who had some reconfiguration or manufactured one-of-a-kind products.

Reconfiguration packages. In order to reduce the amount of design work in reconfiguration, the company develops *reconfiguration packages*. The idea is to identify possibilities for reconfiguration and package the necessary modifications into reusable packages. Active marketing of reconfiguration packages implies significance of reconfiguration. Three heavy machinery companies with significant reconfiguration used reconfiguration packages and five others had some form of reconfiguration packages. For example, typical reconfiguration operations for an elevator manufacturer included modernising doors, upgrading control systems and organising separate elevators into an elevator group.

Systematic. In systematic reconfiguration, a significant proportion of new features is offered to old product individuals. This necessitates design of the product to support this principle, typically in a modular manner. The new features and other developments to the product are introduced and documented in a disciplined way. Systematic reconfiguration is based on a strategic decision. One of the companies belonged to this category. Its success relied significantly on the customers' confidence on later extension possibilities to the system. This applied even to new features not available at the time of the first purchase.

Automatic. In this special category, reconfiguration is automatic. The base product and crucial components are designed so that reconfiguration is easy or even trivial possibly because of built-in 'intelligence'. 'Plug and play' computers can be considered automatically reconfigurable with respect to expansion cards. Another small-scale example is SLR cameras with attachable accessories. The interface between lenses and the camera body has in some makes been maintained for decades, thus allowing the use of old lenses with the newest body, perhaps with some lost functionality, such as, automatic focusing. Two companies produced products that were capable of partial automatic reconfiguration.

Implications of our experiences

Configuration models change. The changes need to be managed to control and record which types of products were sold at a given time. The changes are not only additive; component types may be removed from the configuration models and some dependencies between component types may cease to be valid. A straightforward approach to the evolution of configuration knowledge is to retain versions of the whole model. A more advanced method is to time stamp versions of more detailed concepts, such as, component types and their relations.

Most approaches to configuration seem to assume that reconfiguration can be accomplished using similar configuration models as the configuration of new product individuals. In other words, assume configuration models

CM_1 and CM_2 that were valid at times t_1 and t_2, $t_1 < t_2$, and a configuration c that was configured at time t_1 according to CM_1. At time t_2 it would be enough to load c to the configurator and reconfigure it according to CM_2. We feel that this is impossible in the general case.

The companies we have worked with have decided to operate with the current configuration knowledge only and handle reconfiguration separately. A configuration model is used for configuring new products and consists of surface knowledge. This means that the configuration model abstracts away several issues that may affect the validity of possible configurations. These include the actual behaviour of current in a circuit, existence of minor components, most connections between components as well as the compatibility of different component versions. Otherwise, the configuration model would become much too complex. As a related work, Stumptner and Wotawa [5] also separate reconfiguration from "conventional configuration from scratch".

When the product evolves, the assumptions made in the abstraction may no longer hold. One needs to change some of the assumptions behind the configuration model to capture all the relevant issues. This implies that c cannot be considered to conform to CM_2 since it does not represent the same products as CM_1. The question is: "can c be changed to conform to the view of CM_2?"

Some researchers have emphasised the depth of modelling in alleviating this problem. While we believe that deeper modelling can reduce the problem, especially if resources can be found for "modelling from the first principles", the problem will come up eventually. Advances in technology will probably evolve beyond the scope of the original abstraction, unless it is based on very detailed first principles derived from physics. Low-profit or simpler products do not necessarily allow deep modelling since the effort would be too costly. A better trade-off may be found by modelling with less depth but systemizing the changes between configuration models. The conceptual model we present in the next section takes this approach.

Conceptual Model

In this section, we present an abstract conceptual model of reconfiguration tasks and the knowledge required for reconfiguration. The model does not define how exactly the reconfiguration knowledge is represented but the basic knowledge elements and their interactions are included. This generality allows extending the model to suit the needs of a particular application domain.

Reconfiguration Task

We define the *reconfiguration task* as:

> *Given an* ***existing product individual,*** *a* ***set of requirements*** *and a* ***reconfiguration model,***
> *provide a* ***modified product individual*** *fulfilling the requirements and the* ***necessary changes,***
> *such that, both the individual and the changes are* ***correct*** *with respect to the reconfiguration model.*

In addition, the modified product individual and the changes should be *optimal* in the sense that as much as possible of the existing product individual is retained and no unnecessary or unwanted changes are made. Without these requirements, the definition would allow such pathological reconfigurations as scrapping the whole existing product individual and configuring a new one from scratch. Sometimes this may be the only way to satisfy the new requirements—and even feasible, e.g., in case of software or other products with low manufacturing costs. However, the optimality criterion is needed to cover cases in which this is not desired.

Reconfiguration Knowledge

Managing configuration models and configurations over time requires extending the concepts used for representing configuration knowledge (for a unified conceptualization of configuration knowledge, see [9]). The extensions include versions of component types and other knowledge elements, effectivity of versions, and other concepts related to configuration management and product data management. In addition, it may be necessary to represent manufacturing or maintenance related knowledge such as serial numbers.

In reconfiguration representation, we assume two languages: C_R and CM_R for representing *augmented configurations* and *augmented configuration models*, respectively. The languages C and CM underlying these can be any languages used for representing configurations and configuration models. The augmented versions contain additionally the concepts for reconfiguration.

As usual, we consider a language equivalent with the set of sentences belonging to the language. Under this view, each (augmented) configuration c_r belongs to C_R and each *reconfiguration model* m_r to CM_R, formally, $c_r \in C_R$ and $m_r \in CM_R$.

A reconfiguration model m_r consists of two parts:

- A set of *reconfiguration operations,* denoted by m_{ro}, that defines the possible changes to configurations. A reconfiguration operation consists of a *precondition* and an *action*. The operation can be applied to a configuration if it satisfies the precondition. The action defines the change on the configuration. Typical actions include adding, removing and replacing component individuals or relations between them.
- A set of *reconfiguration invariants,* denoted by m_{ri}, that defines the invariant conditions that configurations must satisfy to be correct.

Intuitively, the reconfiguration operations both limit the possible changes and describe the changes. In addition, their preconditions can control the order of changes. In contrast, reconfiguration invariants specify the conditions that must hold across and between all versions of configuration models, independently of the operations.

In order to represent the operations m_{ro} and invariants m_{ri} of a reconfiguration model m_r, we define CM_R to consist of two sublanguages CM_{RO} and CM_{RI}.

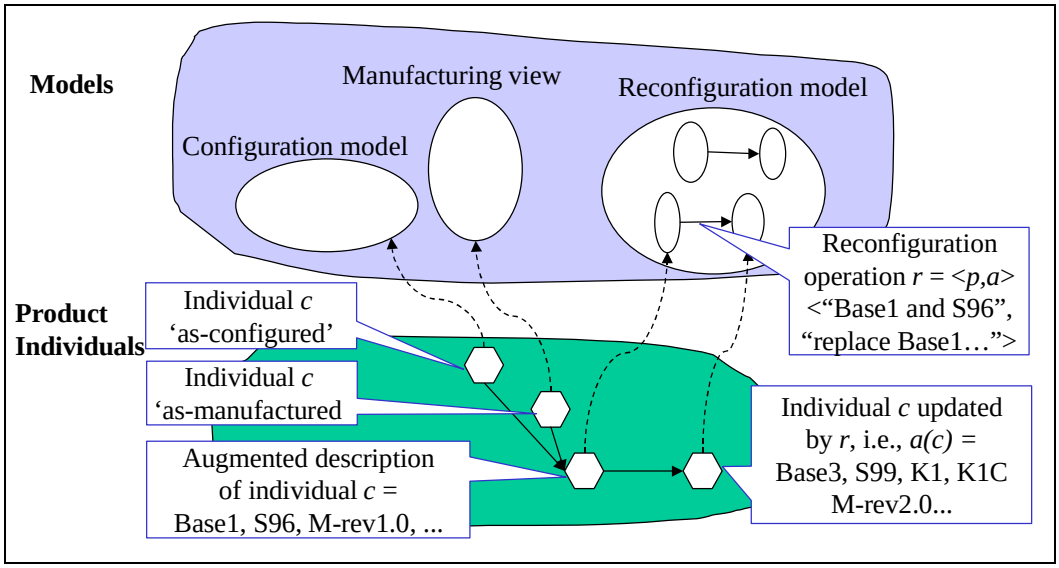

Figure 1. Sample reconfiguration operation.

- CM_{RO} consists of reconfiguration operations r of form $\langle p, a\rangle$, where precondition $p \in CM_{RI}$ and a is a deterministic action. An action a is formally a function a: $C_R \rightarrow C_R$, i.e., it maps an initial configuration to a final configuration. We denote the fact that c_{r2} is the result of applying the action a of operation r to c_{r1} by $c_{r2} = a(c_{r1})$, and use shorthand form $c_{r2} = r(c_{r1})$ for this. Note that we do not assume that reconfiguration operations are atomic. They may change the configuration in arbitrary ways as long as the resulting configuration is uniquely determined.
- CM_{RI} is any suitable language for defining conditions on a configuration. A condition may, for example, state that there is a component individual of given type and version in the configuration, maybe in a specific relation to another component individual.

As an example from the case company, we discuss here a reconfiguration operation r that adds to a product a new feature called 'record keeping'. The operation is actually divided into four similar cases. Each case has a precondition, for example, p = "the product has modules Base1 and S96". The corresponding upgrade action is a = "replace Base1 by Base3 and S96 by SA99, add K1 and K1C". The other three cases are similar describing the operation for other combinations of Base and S modules. In addition, there were invariant conditions such as "SA99 requires M version 2.0 or higher".

Figure 1 presents the same example for a sample individual c. The individual has been configured according to a configuration model and then manufactured. Based on this information the augmented representation is created. The precondition of the operation r is evaluated against c and since c satisfies the precondition, the action a can be applied. Individual c is modified accordingly and is no longer a configuration to which r can be applied.

Redefinition of reconfiguration task

In order to define the reconfiguration task more precisely, we first define when a configuration satisfies a set of conditions, when a reconfiguration operation is applicable to a configuration and the concept of a sequence of reconfiguration operations. We further define at an abstract level the optimality of reconfiguration operations.

We assume that there is a relation "satisfies" on configurations and conditions, denoted by $\models \subseteq C_R \times CM_{RI}$, usually used in infix form. This relation defines when a configuration satisfies a set of conditions according to the semantics of the language CM_{RI}. We say that a reconfiguration operation $r = \langle p, a\rangle$ is *applicable* to a configuration c_{r1} with result c_{r2} iff $c_{r1} \models p$ and $c_{r2} = r(c_{r1})$.

A *sequence of reconfiguration operations* is of form $(r_1, \ldots, r_n) = (\langle p_1, a_1\rangle, \ldots, \langle p_n, a_n\rangle)$ and is applicable (in order from r_1 to r_n) to a configuration c_{r1} with the result c_{r2}, denoted as $c_{r2} = (r_1, \ldots, r_n)(c_{r1})$, iff the following conditions hold

(i) for all $i < n$, there exists $c_i, c_{i+1} \in C_R$ s.t. $c_i \models p_i$ and $c_{i+1} = r_i(c_i)$
(ii) $c_{r1} = c_1$
(iii) $c_n \models p_n$ and $c_{r2} = r_n(c_n)$

We assume the existence of a value function v that gives a value to a combination of a sequence of operations and the change between the existing and solution configuration. This function is needed to define the optimality of a solution to reconfiguration task. Its precise definition is dependent on the application domain, but we assume for simplicity that for any $\langle c_{r1}, c_{r2}, (r_1, \ldots, r_n)\rangle$ such that $c_{r2} = (r_1, \ldots, r_n)(c_{r1})$ it provides a unique value. We note that it may not be enough to provide a solution that is optimal with respect to the sequence of operations, as it may be that the resulting configuration is not acceptable to the customer. Similarly, the sequences of changes leading to an optimally modified configuration may be unacceptable. It would seem reasonable, that the optimality criteria includes some forms of the following principles:

- as few or inexpensive things as possible should be changed, expressed as (possibly a function on) the difference between the original and result configuration,
- as few or inexpensive operations as possible should be used to accomplish the change.

We now have the ingredients for a more precise definition of the reconfiguration task:

> ***Given:*** *a configuration $c_{r1} \in C_R$, a set of requirements expressed as a set of conditions R in language CM_{RI}, a value function v on the changes to c_{r1} and the resulting c_{r2}, and a reconfiguration model m_r in language CM_R, i.e., m_r consisting of operations m_{ro} and invariants m_{ri}*
>
> ***Provide:*** *a modified configuration c_{r2} and a sequence of required changes $(r_1, \ldots, r_n)$ where each $r_i \in m_{ro}$, such that*
>
> *(i)* $c_{r2} \models R$
> *(ii)* $c_{r2} = (r_1, \ldots, r_n)(c_{r1})$
> *(iii)* $<c_{r1}, c_{r2}, (r_1, \ldots, r_n)>$ *is optimal according to v*
> *(iv)* $c_{r2} \models m_{ri}$

Note that invariants may be broken in intermediate states.

Discussion

In this section, we compare our conceptual model to the practices of a case company. We then discuss the generality and properties of the conceptual model.

Comparison to Case Company

A case company has defined reconfiguration operations each with a precondition and an action similar to our model. The case product is configured from plug-in components called modules. The reconfiguration operations cover all the allowed ways of modifying the product in a single step.

The simplest operations only add new modules, which is possible because of guaranteed compatibility of modules. Some modules can also be upgraded to provide enhanced functions. The pre-conditions for module upgrades are simple—they only refer to the version of the module.

The product also has more complex functional upgrades. Their pre-conditions refer to the existence of components of a given type and occasionally also to versions of components. The actions contain component additions, component replacements and version upgrades. Further, some reconfiguration operations require checking that simple additional dependencies are respected. These dependencies are represented by reconfiguration invariants in our conceptual model.

Generally, the reconfiguration operations of the company fit into the conceptual model. However, a few reconfiguration operations specify conditions in which it is necessary to contact the company for detailed instructions. Nevertheless, the number of these exceptions is small and reconfiguration is mostly possible without the help of product experts.

Conceptual Model

The conceptual model is general enough to include several more restricted reconfiguration scenarios as special cases. For example, in the model of Fleishandler et al. [10] it is assumed that both the existing product individual and the modified product individual must be correct with respect to temporally different configuration models. We do not assume this, allowing configurations that mix components in several versions of configuration models, as long as they respect the invariants in the reconfiguration model. However, the conditions can represent entire configuration models, in which case the reconfiguration operations can be used as conversion rules between different versions of configuration models. Stumpner and Wotawa [5] doubt the feasibility of "fixes" in modelling reconfiguration. We, however, have seen that reconfiguration operations of the form discussed in this paper are feasible in practice.

The approach of the case company where all the possible reconfiguration packages are explicitly defined is covered by our model. Another approach that seems to be included in our model is case based reasoning, where the possible modifications to cases retrieved from solution library can be encoded as reconfiguration operations. In fact, our model also subsumes configuration if all operations needed in construction of a configuration, e.g., adding a component to configuration, are defined as reconfiguration operations. The "existing configuration" would simply be empty. Our model also subsumes configuration tasks where the requirements are given using optimality criteria instead or in addition to hard criteria that must be satisfied.

In some domains, the ordering of the reconfiguration operations is crucial, such as for configuring software systems where libraries need to be compiled in the order implied by their dependencies. In complex and large products, such as paper machines, one may need to create a good plan for how to change the product. For example, physical obstructions or the need for fixing things to each other makes the ordering a non-trivial task. For some domains, the ordering is not relevant and only the set of changes is required.

We refrain from defining more precisely the conditions that an optimal sequence of reconfiguration operations or an optimal configuration should fulfil. This is due to several reasons. It is not clear that, for instance, changing the configuration as little as possible is the desired goal. Companies may want to restrict the set of different versions of components that are in the field to have better control of the product and to restrict the types of components that they need to keep in stock for servicing the products individuals. This can be accomplished by changing the products more than what is strictly required when reconfiguring them, for example by always updating certain key components or ones that tend to wear out. In addition, it is not clear which optimisation criteria should be used. These obviously include costs of labour for the reconfiguration, cost of new components, prices on the reconfiguration operations and added com-

ponents, desirability of certain functions or components, e.g., manufactured by a given company, over others, etc.

Our approach allows inclusion of other after-sales operations, such as, component replacements, under a single reconfiguration framework. All spare part replacement operations are inevitably related to reconfiguration since they in fact are simple reconfiguration operations.

In this paper, we have only discussed the evolution of product individuals and developed mechanisms to support that. The reconfiguration knowledge, however, is not static, which may necessitate consideration of this sort of 'meta-evolution'.

Our conceptual model of reconfiguration also bears strong resemblance to planning problems (see e.g., [11]). Configurations can be understood as *world states*, reconfiguration operations as *planning operations*, existing configuration as the *initial world state* and requirements as the *goal state*. The main distinction is that the basic definitions of planning problems do not usually consider the optimality of plans. Optimality of the modified configuration or its difference from the original configuration has no analogue in basic planning. However, in the representation of reconfiguration knowledge and automated or supported reconfiguration similar issues need to be addressed and hence the results and recent advances from the planning domain can be used.

Conclusions and Future Work

Reconfiguration is often a problematic area because of the uncontrolled nature of the population of product individuals. What customers do with their product individuals is inherently difficult to control since it is very much "their own business". We presented a framework that characterises products and business with respect to reconfiguration and discussed our experiences with case companies. The experiences show that despite the inherent complexity of the field, reconfiguration is a successful business for certain companies and there are systematic ways to handle it.

From the premises set out in the framework, we developed an abstract conceptual model for reconfiguration. The model contains reconfiguration operations, which each consists of a precondition that controls the applicability of the operation and an action that describes the modifications to the product individual. In addition, the model includes invariant conditions for controlling the validity of resulting configurations and a value function for selecting optimal or good operation sequences. The complexity of reconfiguration is inherently at least the same as for configuration, since configuration is a special case of reconfiguration. The requirement for optimality may lead to higher complexity.

Our conceptual model reflects the reconfiguration in a successful case company. Nevertheless, the company does not have any advanced tools supporting reconfiguration. As a future work, the model should be made more accurate and a tool for supporting it needs to be implemented. With such a tool, both the conceptual and the computational feasibility of the model could be tested on case products. In addition, trade-offs between the complexity and feasibility of the conceptual model needs to be investigated.

Acknowledgements

This research has been partly funded by the Technology Development Centre of Finland (TEKES) and Helsinki Graduate School of Computer Science and Engineering. The work has benefited greatly from the discussions with the other members of Product Data Management Group.

References

1. Sabin D and Weigel R. Product configuration Frameworks—A survey. *IEEE Intelligent Systems & Their Applications,* 13(4):42-9. 1998.
2. Franke DW. Configuration research and commercial solutions. *AI EDAM (Artificial Intelligence for Engineering Design, Analysis and Manufacturing),* 12(4):295-300. 1998.
3. Haag A. Sales configuration in business process. *IEEE Intelligent Systems & Their Applications,* 13(4):78-85. 1998.
4. Männistö T, Peltonen H, and Sulonen R. View to Product Configuration Knowledge Modelling and Evolution. In Faltings, B. and Freuder, E.C. (eds.) *Configuration—Papers From the 1996 AAAI Fall Symposium (AAAI Technical Report FS-96-03)*, AAAI Press. Pages 111-18. 1996.
5. Stumptner M and Wotawa F. Model-Based Reconfiguration. *Proceedings of the 5th Conference on Artificial Intelligence in Design '98*. 1998.
6. Banerjee J, Kim W, Kim H-J, and Korth HF. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. *Proc. of the International Conference on Management of Data (SIGMOD)*, pages 311-22. 1987.
7. Monk S and Sommerville I. Schema evolution in OODBs using class versioning. *SIGMOD Record,* 22(3):16-22. 1993.
8. Tiihonen J, Soininen T, Männistö T, and Sulonen R. State-of-the-Practice in Product Configuration—A Survey of 10 Cases in the Finnish Industry. In Tomiyama, T., Mäntylä, M., and Finger, S. (eds.) *Knowledge Intensive CAD*. London, Chapman & Hall. Pages 95-114. 1996.
9. Soininen T, Tiihonen J, Männistö T, and Sulonen R. Towards a General Ontology of Configuration. *AI EDAM (Artificial Intelligence for Engineering Design, Analysis and Manufacturing),* 12(4). 1998.
10. Fleishandler G, Friedrich GE, Haselböck A, Schreiner H, and Stumptner M. Configuring large systems using generative constraint satisfaction. *IEEE Intelligent Systems & Their Applications,* 13(4):59-68. 1998.
11. Russel S and Norvig P. *Artificial Intelligence—A Modern Approach*. Prentice-Hall. 1995.