Electronic Commerce: The End of the Beginning

13th International Bled Electronic Commerce Conference

Bled, Slovenia, June 19-21, 2000

A Customization Approach for Structured Products in Electronic Shops

Armin Stahl, Ralph Bergmann, Sascha Schmitt

University of Kaiserslautern, Department of Computer Science
Artificial Intelligence – Knowledge-Based Systems Group
67653 Kaiserslautern, Germany
Phone: ++49-631-205 3362, Fax: ++49-631-205 3357
{Stahl, Bergmann, SSchmitt}@informatik.uni-kl.de

Abstract

Customers of electronic shops find more and more support for the search and selection of products in the sales systems. Unfortunately, most of the shops do not provide additional support with parameterizable or configurable products. Such products could be further customized. One of the major problems most customization techniques suffer from is that they require large knowledge acquisition effort, which leads to problems in the rapidly changing e-Commerce scenario. In this paper, we present a new approach to customization that is particularly suited to e-Commerce applications. It assumes that products can be structured hierarchically into sub-components. Customization is achieved by incrementally replacing unsuitable sub-components through recursively finding best-matching alternative sub-components, using Case-Based Reasoning technology for this search process. The presented approach avoids huge portions of the knowledge acquisition effort. The approach is implemented as a prototypical system.

1. Introduction

A sales system that offers products that can be modified to some degree has to provide its customers with the possibility to further customize the base products, i.e. to enable the customer to tailor the base products according to her or his wishes. Examples of appropriate products, i.e. customizable products, are technical equipment like computers [12], designs for electrical engineering [10], holiday trips, service products like insurances or investment plans, etc. For this paper, we limit our considerations to the customization of complex technical systems.

Customization is especially important when complex products with a large number of possible variants must be supported during sales [7]. Product customization can be realized by the known customization techniques. However, the applicability of the different techniques strongly depends on the kind of products to be supported, in particular on the number of different variants. A general problem of most approaches to customization is that they require a large effort for acquiring the customization knowledge, to be represented, e.g., by rules or by operators [7]. This turns into a real problem when products have many different variants or if the product spectrum changes rapidly so that the customization knowledge must be updated. Other approaches require a complete problem solver for product recommendation and sufficient knowledge for this problem solver. Such approaches also suffer form intractability, both in terms of computational efficiency

and knowledge acquisition effort. Hence, for customizing complex products with a large number of possible variants, the existing customization approaches are often not suitable in practice.

In this paper, we present a new approach to customization that is particularly suited to electronic commerce applications. This approach avoids huge portions of the knowledge acquisition effort of the previous approaches. It assumes products that are structured into sub-components, possibly in a hierarchical manner. The knowledge required is knowledge about available pre-configured complete products as well as knowledge about available sub-components. Both kinds of knowledge are easily available in an electronic commerce setting. After retrieving the best pre-configured product with respect to the customer's requirements, the product is customized by incrementally replacing sub-components by more suitable sub-components. These new sub-components are determined by recursively applying CBR, i.e. similarity-based product retrieval, on the level of the sub-components [1, 7]. In the remainder of this paper, this approach is described in more detail. We will first describe a typical e-Commerce scenario and analyze the shortcomings of existing approaches. Then the general idea of recursive CBR is introduced before it is specialized for the purpose of product customization. Finally, we report on the current state of the implementation.

2. Product Customization in e-Commerce Applications

Within the set of possible products, a continuum of products can be identified (see Fig. 1) [12]. It classifies different products according to their ability to be customized by a customer. Generally, we distinguish between constant and variable products. Constant products are products which cannot be modified by the customer. The product is fixed in such a case. Variable products may be customized via product parameterization or product configuration. We differentiate between different kinds of such a customization.

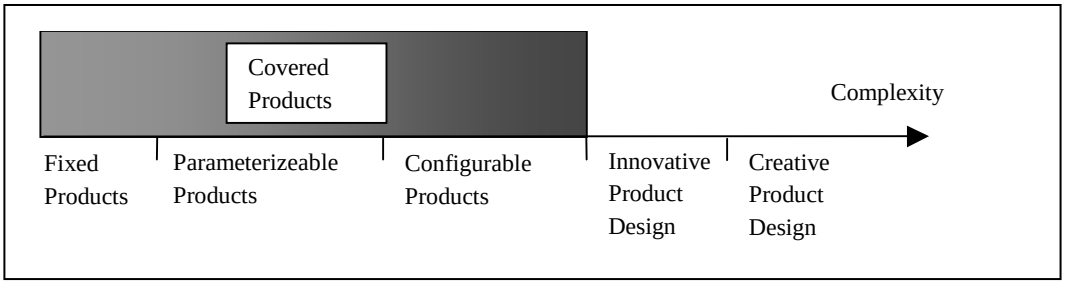

Fig. 1. The Continuum of Products.

2.1 General Product Categories

Fixed Products: At the lower end of this continuum we have fixed products. The product cannot be modified and as a result, the sales assistant cannot customize the product. Examples are music CDs, books, integrated circuits, etc., or a single computer monitor, where the product is completely fixed.

Parameterizeable Products: Next on the continuum, we find products which are parameterizeable by certain values. These values may be discrete, like the color of a good and also they may be continuous values, like the capacity of a storage device. The sales assistant may calculate these values or he may use given or existing ones during the sales process. However, the product may only be modified by the instantiation of one or more parameters concerning the product.

Configurable Products: Next, we have configurable products which consist of a set of predefined components and the knowledge about how components can be connected. Further, all available components must be known and also all the knowledge how components are allowed to be connected. During the sales process the sales assistant has to configure the product for the customer.

We will not cover innovative product design which produces an artifact significantly different from existing ones or even creative product design products which produce a new type of artifact. Examples for such tasks are the design of complete assembly lines, a one-family house, or other complex products which do not have a very similar prototype in the past.

2.2 Case-Based Product Recommendation: The Scenario

During recent years, the technology known under the term Case-Based Reasoning (CBR) has become a successful tool to realize product customization. The main idea behind CBR is the assumption that it should be possible to use experiences of the past, called cases, to solve actual problems. In an e-Commerce scenario, the cases are represented by the available products or their descriptions respectively. The actual problem that has to be solved is given through a set of customer demands, i.e. the customer formulates his needs and wishes on a searched product. We consider the scenario in which a customer enters a virtual shop to buy a complex technical system. Complex technical systems (e.g., PCs) can usually be decomposed into a set of different components. We suppose that the shop offers a set of pre-configured standard systems as well as the different individual components.

If the shop provides an intelligent product recommendation agent the first step of the sales process is a demand acquisition phase in which the customer states his individual demands on the searched product. The result of this demand acquisition phase can be formulated in form of an incomplete product description (on a technical level) which we call *query*. This query is then used to start a similarity-based retrieval in order to determine one of the available pre-configured standard systems that fulfills the demands as well as possible. However, because of the large number of possible product variants, the retrieved product does usually not fulfill the demands exactly. Hence, to be able to present the customer a satisfying result, it is necessary to customize the product, e.g., by replacing some components by more suitable ones [9]. This is the task of the adaptation phase of CBR. In the following, we review several well-known approaches to adaptation in CBR and discuss whether they are appropriate for the customization of complex products.

2.3 Existing Approaches

Customization Rules and Customization Operators. The knowledge how existing solutions can be adapted on actual problems is encoded explicitly. Adaptation rules [3] consist of a set of preconditions and a set of actions. Dependent on the evaluation of the preconditions, the actions are able to modify a retrieved case to a new target case with respect to the given query. Customization operators [7] are very similar to adaptation rules. While the rules will be executed after the retrieval automatically, the operators provide more control by the customer. Therefore, the retrieved case with an additional set of applicable operators will be presented to the customer. If the retrieved product does not fulfill his demands, he can repeatedly choose operators to change the given product until he gets a satisfying result.

The general problem of both approaches is the necessity to define every possible case modification that may occur explicitly in the form of preconditions and actions. Therefore, even simple domains often require a large number of customization rules or operators to cover all customization possibilities. For really complex domains, the huge amount of necessary rules or operators prevents the application in practice.

Configuration From-Scratch. The configuration of a product can also be performed from-scratch by classic configuration systems [4] without using CBR. Applying such a system for the customization of products in an e-Commerce application often leads to some disadvantages. First, if it is impossible to configure a product that fulfills all customer demands exactly, such a system rarely finds a suitable alternative solution. Second, in classic configuration systems it is often difficult to handle optimality criteria. Therefore, the system can only present any solution (if existing), but a high quality of this solution cannot be guaranteed in general. An additional problem of the from-scratch configuration is the time-critical aspect, i.e. the configuration process usually takes a lot of calculation time. This is often not acceptable in e-Commerce applications.

Derivational Adaptation. The common property of customization rules and operators is that they are used to transform an existing solution to a new, hopefully better, solution. Thus, this approach is also called *transformational adaptation* [13]. Another approach is known under the term *derivational adaptation*. Here, the cases do not represent concrete solutions. Instead, they contain a description of the problem solving trace which describes how a concrete solution is generated. This known solution trace can then be used by a from-scratch problem solver to generate a solution very quickly. Generally, this approach is coupled with the same disadvantages as configuration from-scratch, except for better performance. A description of configuration systems using derivational adaptation can be found in [6, 5].

The described approaches for realizing configuration of complex products in e-Commerce applications are all coupled with more or less big problems or significant disadvantages. In the remainder of this paper, we will present an alternative approach for realizing product customization in complex and highly structured domains. A classic example for such a domain is the configuration of personal computers.

3. Product Customization by Incremental Similarity-Based Adaptation

To illustrate the functionality of our approach we consider the configuration of personal computers. However, the presented approach can be applied for the customization of any structured products.

3.1 Domain Representation

Structure of Products. In general, it is possible to represent complex structured products by an object-oriented domain model with an additional aggregation hierarchy [11]. In the following, we assume a special form of such a compositional structure like shown in Fig. 2.

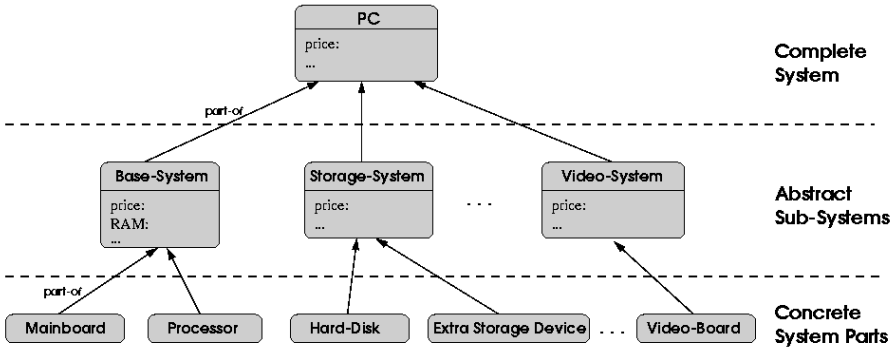

Fig. 2. An Example Compositional Structure.

In this structure, the root node represents the whole technical system to be configured, i.e., in our example, a complete personal computer. The leaf nodes represent the concrete parts that a PC consists of, like the hard-disk, the processor, the CD-ROM, and so on. In a concrete PC, these parts are realized by concrete technical components. For example, the processor part can be realized by a Pentium-III with 500MHz. The inner nodes of the compositional structure (if existing) do not correspond to concrete parts of the technical system. They rather represent abstract nodes used for aggregating related parts into kinds of abstract subsystems (e.g., Base-System or Storage-System).

Query. The starting point of the configuration process is the query represented by an incomplete instantiation of the compositional structure. When looking at the example query shown in Fig. 3, we can interpret the root note as our actual problem, i.e., we are searching a PC with a set of special properties. To reach this goal it is necessary to select appropriate components that fulfill the properties of the respective part-queries. In our example, one part-query states that the PC

should have a hard-disk with 12GB of capacity. To fulfill this demand we can, e.g., integrate the concrete hard-disk ``Maxtor91303D6" for the hard-disk part in the PC. Thus, we can interpret the different leaf nodes of the query as sub-problems that must be solved to solve the overall problem, i.e., the configuration of the required PC. If we have found suitable sub-solutions, i.e. suitable components, for every part-query, we have to combine these sub-solutions to a final solution for the overall configuration problem.

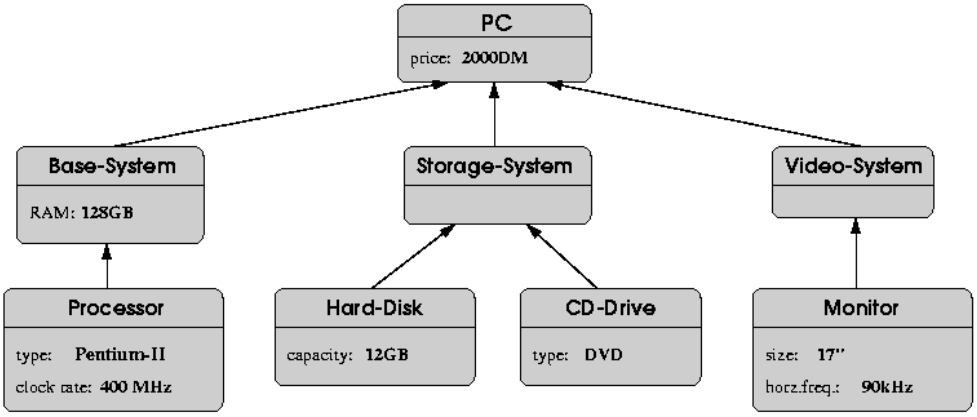

Fig. 3. An Example Query.

Similarity Measure. To retrieve appropriate PCs from a product database (in the CBR context the case base), the assumed CBR-system uses special domain-dependent similarity measures. To compute the similarity between a query and a given product we assume a bottom-up strategy using the defined compositional structure. First, it is necessary to compute the similarities between the part-queries and the corresponding product parts. These local similarities are used for determining the global similarity between the query and the product. A more detailed description of such a similarity computation can be found in [2].

Dependencies between Components. Generally, the decomposition of a configuration problem like discussed above is coupled with a basic problem. If we decompose a configuration problem into the sub-problems of finding suitable components it is not sufficient to handle these sub-problems absolutely isolated. The reason are dependencies between the different sub-problems or their solutions respectively. In the described application domain such dependencies occur in form of technical constraints between the different components. For example, it is impossible to combine every kind of processor with any mainboard model because the respective interfaces must fit together. To be able to handle these technical restrictions during the configuration process, the domain model must include a formal description of the existing constraints. For our approach, we suppose the existence of a special constraint system that is able to check whether the constraints of a given configuration are fulfilled or not.

3.2 The Configuration Framework

We now describe an approach to realize product customization by a kind of incremental similarity-based adaptation. Fig. 4 shows the respective process model that we will discuss in the following.

Basically, the whole configuration process can be subdivided into two major steps, called base product retrieval and adaptation cycle. The task of the first step is the similarity-based retrieval of the best available base product from the respective product case-base. The second step is an iterative procedure which performs the necessary adaptation of the retrieved base-product if it does not fulfill all customer demands.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.