Computers in Industry 63 (2012) 471–481

Contents lists available at SciVerse ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Definition and evaluation of product configurator development strategies

Anders Haug [a,*], Lars Hvam [b], Niels Henrik Mortensen [c]

[a] Department of Entrepreneurship and Relationship Management, University of Southern Denmark, Engstien 1, 6000 Kolding, Denmark
[b] Department of Manufacturing Engineering and Management, Technical University of Denmark, Building 425, 2800 Kgs. Lyngby, Denmark
[c] Department of Mechanical Engineering, Technical University of Denmark, Building 404, 2800 Kgs. Lyngby, Denmark

ARTICLE INFO

Article history:
Received 11 February 2011
Received in revised form 18 November 2011
Accepted 1 February 2012
Available online 29 February 2012

Keywords:
Product configuration
Product configurator
Knowledge acquisition
Knowledge engineering
Expert systems

ABSTRACT

Product configurators represent one of the most successful applications of artificial intelligence principles. Product configurators are a subtype of software-based expert systems with a focus on the creation of product specifications. The use of product configurators has resulted in many positive effects in engineering-oriented companies such as reduced lead times, fewer errors, shorter learning periods for new employees, etc. Unfortunately, many configuration projects also fail because the task of developing the configurator turns out to be much more difficult and time-consuming than anticipated. Thus, it is crucial to apply the appropriate strategy. However, the literature does not discuss different strategic alternatives in a detailed manner; it only provides generalised recommendations of single strategies. To deal with this issue, this paper defines and compares seven different strategies for the development of product configurators. The relevance of the defined strategies is supported by seven named case studies.

1. Introduction

Many companies experience an increasing demand for customer-specific products while competition on prices and delivery times is hard [1,2]. Mass customization is a manufacturing paradigm that focuses on satisfying individual customer requirements while keeping manufacturing costs and delivery times close to those of mass-produced products [3]. Mass customization can be seen from two very different perspectives, depending on whether the company that pursues a mass customization strategy is an engineering-to-order company or a mass production company [4]. This paper focuses on engineering-to-order companies, in which the use of product configurators in many cases has produced significant benefits.

Product configurators represent one of the most successful applications of artificial intelligence principles [5–7]. A product configurator is a subtype of software-based expert systems (or knowledge-based systems) with a focus on the creation of product specifications. A product configurator can be defined as "a software-based expert system that supports the user in the creation of product specifications by restricting how predefined entities (physical or non-physical) and their properties (fixed or variable) may be combined" [8]. In the context of engineering-oriented companies, the use of product configurators has resulted in a range of benefits such as shorter lead times, improved quality of product specifications, preservation of knowledge, use of fewer resources for specifying products, optimised products, less routine work, improved certainty of delivery, and less time needed for training new employees (e.g., [9–11]).

Although many engineering-oriented companies achieve several benefits from the use of configurators, many companies also experience great difficulties. Unfortunately, the literature provides detailed descriptions of successful cases only, which does not give an accurate picture of reality. According to the wide experience of the authors who have studied numerous configurator projects and through discussions with people from industry and academia, many initiated configurator development projects aimed at complex products and multiple users do not result in the creation of a configurator that is as widely used by the organisation as expected. In fact, projects are often abandoned even before the configurator is completed. On the other hand, it is often the case that, although a configurator project fails, a project with a similar focus but a different organisation, scope, or choice of software succeeds later.

There are two basic challenges to overcome in order to achieve success in a configurator project. First, the project needs to keep its momentum until the configurator is developed to a point where it can be utilized. If a project becomes significantly more costly than anticipated or the project fails to produce prototypes that indicate a probability of success, such projects may be abandoned in order to reduce potential losses. Second, even when a project leads to the development of a configurator, the configurator still has to be accepted and used by the organisation. In this context, there are

* Corresponding author. Tel.: +45 65501350; fax: +45 65501357.
E-mail addresses: adg@sam.sdu.dk (A. Haug), lhv@man.dtu.dk (L. Hvam), nhmo@mek.dtu.dk (N.H. Mortensen).

0166-3615/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2012.02.001

several challenges such as ensuring that the configurator covers an adequately large part of the products produced, it can produce sufficiently extensive outputs, and its precision is satisfactory. If such demands are not fulfilled, the use of the configurator may soon be abandoned. This, for instance, implies resources are used for continuous maintenance of the configurator knowledge base so that it reflects changes in the product assortment.

There are a number of different ways to carry out the project of developing a product configurator. The chosen approach will, of course, impact cost, development time, and the quality of the configurator. However, in the literature a clear definition of various ways to carry out the development of a configurator does not exist; instead, only proposals of specific approaches are provided without much consideration of alternatives or contexts in which the specific approach is best suited. Thus, this paper answers two important questions: (1) what are the different strategies for developing product configurators? and (2) what are the advantages and drawbacks of choosing one of these strategies? Thus, the paper provides an improved understanding of configurator projects for future research and a better basis for companies engaging in such projects. Although the paper focuses on product configurators, its contribution may also be relevant in other expert system development contexts. This, however, is beyond the scope of this paper.

The remainder of this paper is structured as follows: Section 2 reviews relevant concepts and literature. Based on the discussion in the previous section and cases studied by the authors, Section 3 defines the different strategies for configurator development. Section 4 discusses the defined strategies in relation to the effects of choosing them, and which case characteristics make some strategies more suitable than others. The paper ends with a conclusion in Section 5.

2. Literature base

To establish a basis for identifying the different strategies for the development of product configurators, definitions of some basic concepts are needed. This section reviews the concepts of knowledge acquisition and knowledge representation in a configuration context, followed by a description of different perspectives on the configurator development process.

2.1. Knowledge acquisition

In order to obtain the many possible benefits of implementing a configurator in the daily operations of a company, the right information has to be implemented in the configurator. The information that forms the basis for the creation of a configurator is provided by the relevant product experts of a company (i.e., those who have expertise related to design, engineering, manufacturing, sales, etc.). The experts responsible for retrieving and formalizing product expert information into conceptual models are called ‘knowledge engineers’. The process of retrieving information from product experts and transforming it into representations suitable for implementation in an expert system is referred to as ‘knowledge acquisition’. Knowledge acquisition is a process by which (1) the knowledge engineer uses communication techniques to elicit information from relevant experts (knowledge elicitation), (2) the knowledge engineer interprets this information to draw conclusions about probable underlying knowledge and reasoning processes of the product experts, and (3) the knowledge engineer uses his/her conclusions to direct the construction of a model and its implementation in an expert system shell or language [12]. In the 1980s, the development of expert systems was often seen as a transfer process based on the assumption that the required knowledge already existed, for which reason the task of creating an expert system was perceived as the process of collecting and implementing this knowledge. Today, however, there is overall consensus that the process of building an expert system is better perceived as a modelling activity [13–15]. This modelling view of the knowledge acquisition process has, according to Studer at al. [14], the following consequences: (1) models are only approximations of reality and, in principle, the modelling process is infinite since it is an incessant activity with the aim of approximating the intended behaviour; (2) the modelling process has a cyclic course since new observations may lead to a refinement, modification, or completion of the already built-up model; and (3) the modelling process is dependent of the subjective interpretations of the knowledge engineer, for which reason the process is typically faulty, and evaluation of the model with respect to reality is indispensable in order to create an adequate model.

2.2. Knowledge representation

Knowledge representation literature focuses on the use of symbol systems to represent (simulate) the relevant knowledge of some domain. Davis et al. [16] argue that the essence of a knowledge representation can be best understood by the five distinct roles it plays: (1) a surrogate (a substitute for the thing itself, used to enable reasoning about the world); (2) a set of ontological commitments (definition of in which terms the world should be described); (3) a fragmentary theory of intelligent reasoning (expressed by three components: a fundamental conception of intelligent reasoning, a set of inferences sanctioned, and a set of inferences recommended); (4) a medium for pragmatically efficient computation (the computational environment in which thinking is accomplished); and (5) a medium of human expression (a language for describing aspects of the real world).

In configurator development projects, two diagramming techniques are frequently used for the capture and representation of product information, namely the product variant master (PVM) technique (sometimes named ‘Product Family Master Plan’) and class diagrams. PVMs are targeted at representing knowledge about a product assortment, and they describe classes, their relationships and properties, and constraints that determine how classes and properties may be combined. A PVM consists of two generic sections. The part-of section, placed in the left side of a PVM, defines the classes that a given product family can comprise. The kind-of section, placed in the right side of a PVM, describes the variation of a part, i.e., different types with common characteristics. In the course of time, different definitions of the PVM notation have been proposed, most notably by Mortensen et al. [17], Harlou [18] and Haug [8]. Based on the latter definition, which represents the most extensive and formalized of these definitions, a principal example of the PVM technique is seen in Fig. 1. In the example, the guillemets (<<...>>) indicate that the class ‘Body-Assembly’ is a class of the type ‘assembly’.

In contrast to PVMs, class diagrams have not been created as a language aimed at describing product information but, instead, are created for general software development. A class diagram describes object classes, their properties, and their relationships. Class diagrams are part of the UML (Unified Modelling language), which in version 2.0 consists of thirteen diagram types [19]. Fig. 2 shows a principal example of a class diagram with some of the most common elements as if applied in a product analysis context (extended version of the model in Fig. 1, without pictures). The relationship types shown in Fig. 2 are generalization (generalization-specialization structure), composition (whole-part structure in the form of strong aggregation, which means that the ‘part’ is controlled by the ‘whole’ during all its lifetime), association (classes that are related