Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 44 (2016) 204 – 209

www.elsevier.com/locate/procedia

6th CIRP Conference on Assembly Technologies and Systems (CATS)

Configuration management through satisfiability

Bryan Pearce[a], Mary E. Kurz[a*], Keith Phelan[a], Joshua Summers[a],

Jörg Schulte[b], Wolfgang Dieminger [b], Kilian Funk[c]

[a]*Clemson University, Clemson, SC 29634, United States*
[b] *BMW, Spartanburg SC United States*
[c] *BMW, Munich Germany*
* Corresponding author. Tel.: +1-864-656-4652; *E-mail address:* mkurz@clemson.edu

Abstract

In a Build-to-Order environment, a configurator relays the taxonomy of customization choices to the customer, then translates these choices into a bill of materials. Configuration Management (aka Variety Management) of the system entails validating proposed changes to the policies that govern both configurator processes. We present a satisfiability approach to the problem, in which a suite of conflict classes are developed, representing potential configurator failure modes. Satisfiability logic routines test the potential presence of each conflict class if the proposed change is adopted, using an integrated constraint set including both part allocation and customization object relationships.

Peer-review under responsibility of the organizing committee of the 6th CIRP Conference on Assembly Technologies and Systems (CATS)
Keywords: Configuration Management, Variety Management; Satisfiability

1. Introduction

Over the last few decades, global markets for manufactured goods have increasingly offered customizable products flexible to customer preferences. To remain relevant in this shifting economy, manufacturers have focused on mass customization practices that support the increase in product variety while retaining high production volumes [1][2].

Configuration management is the process of constructing and managing a domain of product variety space that meets customer needs. There are several component activities under this umbrella, including assessing customer preferences and product variant capabilities, and identifying specific product configurations that meet demand [3]. These problems are particularly difficult when the degree of customization choices is high, as each configuration management problem grows exponentially in response to each additional customization choice. Product families are a common method for managing this complexity, in which the domain of product variety space is divided into an array of independent base product platforms, each of which can be modified by the addition, subtraction, or substitution of modular options [4]. Even with a product family approach, however, maintaining product variety information remains a challenge for highly customizable products [5].

Build-to-Order production allows customers to configure their purchase personally, choosing from the domain of offerings made by the manufacturer. If product variety is small, with relatively few configuration alternatives, then the customer may be presented with a catalog enumerating each fully-configured offering. If product variety is high, however, an enumeration approach is not possible. The German automotive industry presents a compelling example, where

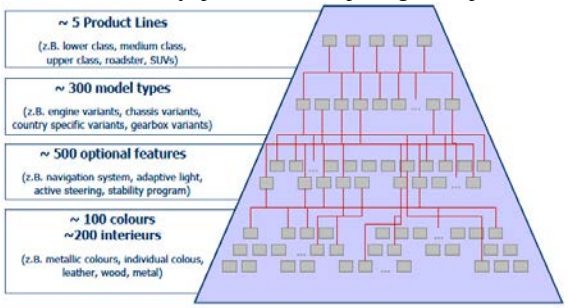

Fig. 1. Model of configuration variants in the automotive industry [7]

Peer-review under responsibility of the organizing committee of the 6th CIRP Conference on Assembly Technologies and Systems (CATS)
doi:10.1016/j.procir.2016.02.127

vehicle configuration is a composite of many smaller subsystem configuration problems, such as color of paint, engine size, trim and badge options, etc. Taken as a whole, up to 10^{32} unique offerings may exist for some vehicles [6]. Fig. 1 shows an example of configuration complexity in the automotive industry. This diagram depicts the vehicle configuration problem hierarchically according to the product family approach, with platform at the topmost level and more detailed configuration features below.

A configurator is a software alternative to catalogs that divides the product configuration process into stages. In the first stage the customer selects the platform, or base product. In successive stages the customer is queried over a set of options corresponding to one particular subsystem of the product. For example, a vehicle configurator may first query for the model (tagged name), then later query for engine and drivetrain options, interior cabin options, etc. Once the customer completes and submits an order, the configurator constructs a bill of materials (BOM) by translating configuration choices into a set of corresponding parts.

The manufacturer requires control over the choices rendered by the configurator at each stage, to ensure that final configured product conforms to technological or marketing specifications. To this end, a rule-based reasoning technique guides the configuration process. Rule-based reasoning encodes a set of production rules, or constraint relationship between options, usually expressed as conditions and consequences relating options (if “A” then “B”). The configurator consults these production rules between each stage, checks whether previous customer choices meet the conditions of each rule, then constructs the next configuration stage to only include those choices that satisfy all rules. Another set of rules manages the mapping of the configured product to part allocation. Examples of rule-based configurators are given in [8][9][10].

Technology and market conditions change through time, inducing changes in product offerings, parts usage, or both. In operational terms, this entails making changes to the rule sets that control the configurator customer interface, and the part allocation processes that creates the BOM. The configuration management task is to validate a candidate set of rule alterations, to ensure that the alterations map correctly to the intended change and don’t create unintended side effects, e.g. an incorrect BOM for some product configuration.

The purpose of this paper is to develop support methods for this validation process, based on needs identified in previously published research. Section 2 summarizes the particular motivating scenario. Section 3 provides the background for the resulting validation process, which is described in Section 4, with examples. Section 5 concludes the paper.

2. Motivation

A wide variety of configuration management techniques have been developed to assist in the implementation. Phelan et al [11] describe several methods as well as potential challenges, which directly motivates the solution proposed here. The remainder of this section summarizes the findings of Phelan et al [11] for the convenience of the reader.

The foundation for the manufacturer’s configuration management system is a rule database that contains the rules governing the possible options and packages for a specific vehicle, resulting in a rule-based configuration management system. Production rules can be described as a set of conditions and consequences (if “A” then “B”). Therefore, the condition relates to an existing component or state of the product which, if met, results in an execution of the consequence action. An example of this would be as follows: “If Part A is found in the configuration, then Part B cannot be used in this configuration”. The scope of the rule database (over one thousand rules per vehicle) makes it difficult to ensure the accuracy of all of the rules and to ensure that the rule database covers the complete set of feasible configurations for each vehicle. Additionally, maintaining the rule database, with either updates or changes, is equally challenging due to the amount of possible change propagation and ensuring that all necessary changes have been made.

The rule database is used for at least three separate functions in the company. First, it is used for the ordering of vehicles which are all specified external from the manufacturing site, either by a customer or a dealership. Each vehicle built results from a selection of the possible components or options that are available or feasible based on location and other specified options. The tool used for specifying the vehicles relies on the above rule database. Second, the rules are used for part-ordering. Once a vehicle has been ordered, a parts management system uses the specified options to identify the parts that are required (and therefore ordered from suppliers) for the vehicle. Third, the line balancing utilizes the rules to accurately predict the time utilized for each worker and station. Tasks that cannot occur on the same vehicle do not contribute to the takt time and are detected by “violations” of the rules in the database; the larger of these task times should be used as the time for the set of those tasks. As all of the systems rely on the rule database, it is imperative that all of the rules are accurate and complete.

The rule database is updated throughout time based on marketing or engineering changes. Phelan et al report that much of the verification process for rule change is based on individual employee experience. For example, one employee reported that his experience with different vehicle systems has taught him to examine some areas more than others. Such reports were typical in the case study. However, this type of human verification is not feasible due to the scope of the rule set. There are approximately 1,500 parts per vehicle, with nearly 10 variants per component. Additionally, there are a half dozen models with dozens of variants and scores of options in configuring these components. Ultimately, there are nearly 700 million possible configurations that must be checked for feasibility periodically.

Over the course of the case study, the researchers identified numerous opportunities for improvement, which highlights the need for the work reported here. These are classified as follows, a few with an example of a motivating scenario.

- “Rule conflict.” Is there a subset of two or more rules such that no possible configuration may satisfy them?
- “Object activation.” Can all options/parts/etc. that are declared as being available for selection actually be selected?