Dealing with Temporal Change in Product Documentation for Manufacturing

Carsten Sinz* and **Wolfgang Küchlin**
Symbolic Computation Group, WSI for Computer Science
University of Tübingen, 72076 Tübingen, Germany
{sinz,kuechlin}@informatik.uni-tuebingen.de

Abstract

Product documentation for manufacturing is subject to temporal change: Exchange of parts by successor models, shift from in-house production to external procurement, or assembly line reconfiguration are just a few examples. In this paper we show how DaimlerChrysler AG manages configuration for manufacturing of their Mercedes lines. Furthermore, we identify typical situations of change, their representation on the product documentation level, and how to keep the documentation consistent over time. We then develop two verification methods for computing the induced change at the parts level. Finally, we show how our methods can be applied to handle model year change and production relocation.

1 Introduction

Product configuration plays a key role in markets for highly complex products such as, e.g., in the automotive or computer industry [McDermott, 1982; Günter and Kühn, 1999]. In these sectors a high degree of personalization with immense configuration possibilities comes together with large numbers of produced units. Electronic product data management (EPDM) systems are therefore employed, which accompany a customer's order from its reception up to the final manufacturing of the ordered product.

The need for configuration usually occurs at several stages in the supply chain, like sales, engineering or manufacturing. The requirements on the EPDM system may differ greatly from one stage to the other [Wright *et al.*, 1993; Timmermans, 1999]. However, the majority of commercially available configuration tools concentrate on the sales aspect as can be seen, e.g., in a recent survey [Sabin and Weigel, 1998].

Our paper deals with the role of configuration for manufacturing. Here, configuration requirements are similar to the engineering stage, in that the product has to be considered not merely in functional (sales-)categories, but down to the level of parts assembly. Especially in the automotive industry, where—as in our case—an individual vehicle can consist of up to 15,000 parts, this rules out the use of conventional sales-configurators. In [Haag, 1998] the notions of high-level and low-level configuration are introduced, where the low-level is characterized by non-interactive, procedural processing. In this sense we address low-level configuration here.

Product documentation on the manufacturing stage is characterized by frequent temporal change: Parts may be available or unavailable at certain points of time or may be exchanged by successor models, subassemblies may shift from in-house production to external procurement, assembly lines may be reconfigured. Additionally, changes on the engineering level usually have a direct impact on the manufacturing documentation. To name just a few, we can think of start-up or runout of supplementary equipment or whole model lines, or sharpened or relaxed constraints between parts or subassemblies due to further product development.

The continuous change of the product documentation aggravates the maintenance of a consistent and error-free data base, which should contain, for example, no unnecessary parts or equipment codes, and no constructibility contradictions. Thus, verification methodologies are highly desirable.

We present two methods, the $\pm\delta$-method and the 3-point approach, to compute the changes induced by high-level product changes on the parts level. These methods generate propositional formulae, which are then checked with a state-of-the-art SAT-checker. Furthermore, we show how our methods can be applied to handle two typical situations of change: model year change and production relocation.

We have prototypically implemented our methods within a specialized version of the BIS system [Sinz *et al.*, 2001]. This implementation is currently being evaluated by the production department of the Sindelfingen (Germany) plant of DaimlerChrysler AG.

2 Product Documentation for DaimlerChrysler's Mercedes Lines

In the following we will describe the EPDM system DIALOG that is used in its two variants DIALOG/E and DIALOG/P in the engineering resp. production departments of DaimlerChrysler for configuration of their Mercedes lines.

*Supported by DaimlerChrysler AG and debis Systemhaus Industry GmbH.

2.1 Documentation at the Engineering Stage

Using the terminology of [Sabin and Weigel, 1998], DIALOG can be classified as a rule-based reasoning system for batch configuration. It consists of a function-oriented and of a parts-oriented level. The former is characterized by equipment codes (sales options) and control codes, as well as rules for on the one hand describing constraints between these codes and on the other hand completion of partially specified orders. This level thus constitutes a description of the set of manufacturable products from an engineering point of view, which we will also call the *product overview* in the following. The parts-oriented level is characterized by a modularized hierarchical parts list, where alternatives are selected based on rules. These rules contain the function-oriented sales and control codes and therefore provide the mapping from the high-level functional to the low-level aggregational view. More information on the documentation method and a synopsis of the requirements from different departments can be found in [Kaiser and Küchlin, 2001].

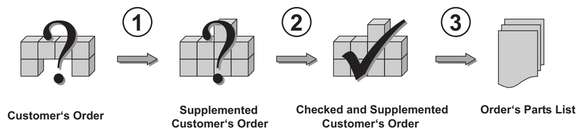

Figure 1: Processing a customer's order.

A customer's order within DIALOG/E consists of a model line selection together with a set of further equipment codes which describe additional features. Each equipment code is represented as a Boolean variable, which is set to *true* exactly when the piece of equipment is chosen. Thus, an order is a fixed assignment to the propositional variables of the product documentation. Orders are processed in three major steps, as depicted in Figure 1: (1) order completion, (2) constructibility check and (3) parts list generation. All of these steps are controlled by rules. Rules can be of three different types, all of which contain a propositional logic formula that is built out of the usual connectives `AND`, `OR` and `NOT`, and the above-mentioned codes serving as propositional variables. No restrictions are placed upon the structure of the rules' formulae, so there is, in particular, no restriction to Horn formulae. The whole order processing is controlled exclusively by evaluating the rule's formulae under the variable assignment induced by the customer's order, and executing suitable actions based on whether the formula evaluates to *true* or *false*.

The three types of rules that occur in the product documentation reflect the three stages of the order processing. There are supplementing rules $S(x)$ which control the order completion process, constructibility rules $C(x)$ to check manufacturability of an order, and part selection rules $R(p)$ to generate the bill of materials. All rules are associated with either a code or a part they refer to. This code (in case of supplementing or constructibility rules) or part (for part selection rules) also serves as a unique index for look-up of the rules' formulae $S(x)$, $C(x)$ or $R(p)$[1]. Rules are interpreted differently, depending on their function. The different interpretation results in different actions taken when the rule's formula evaluates to *true* or *false*. We will now describe in detail the individual order processing steps and the actions taken.

Order completion. The order completion or supplementing process adds implied codes to an order. The supplementing formula $S(x)$ specifies the condition under which code x is added to order O. When $S(x)$ evaluates to *true* under the variable assignment induced by O, i.e. when O is a (logical) model of $S(x)$, then code x is added to that order. The order completion process is repeated until no further changes result.

Constructibility check. Constructibility of a customer's order is checked according to the following scheme: For each code x there is a constructibility condition $C(x)$. This formula interrelates x with other codes by encoding, e.g., requirements or exclusion conditions for using code x. A code is called constructible or valid within a given order O if $C(x)$ evaluates to *true* under O. All codes of a possibly supplemented order must be valid in a constructible order, non-constructible orders are rejected.

Parts list generation. The parts list is hierarchically structured using modules, positions and variants. Parts are grouped into modules depending on functional and geometrical aspects, positions contain mutually exclusive alternative parts, called variants, for each installation point. A part p is selected based on the part selection formula $R(p)$ associated with it: if $R(p)$ evaluates to *true* under the checked and possibly extended order O, then part p is included into the bill of materials for O.

Supplementing rules:

$$\mathrm{S}[\![x]\!] \;:=\; S(x) \Rightarrow x$$

Constructibility rules:

$$\mathrm{C}[\![x]\!] \;:=\; x \Rightarrow C(x)$$

Product overview:

$$\mathrm{PO} \;:=\; \bigwedge_{x \in \mathcal{C}} \Big(\mathrm{S}[\![x]\!] \wedge \mathrm{C}[\![x]\!] \Big)$$

Part selection rules:

$$\mathrm{R}[\![p]\!] \;:=\; R(p)$$

Order validity for order O:

$$O \models \mathrm{PO}$$

Selection of part p for order O:

$$O \models \mathrm{R}[\![p]\!]$$

Figure 2: Verification Semantics of Rules.

We will now introduce a notation for rule semantics which aims at describing the whole set of constructible orders, i.e.

[1]Note, that in our notation, we make no distinction between the formula of the rule and the rule itself.

the product overview, in one propositional formula. This semantics is a logical formalization of the DIALOG/E behavior. It is not employed within DIALOG/E, e.g. to check individual orders, but is very helpful to express consistency assertions about the rule base as a whole as well as for the handling of temporal change. In this context the semantics of a rule is a propositional formula, and is denoted by $[\![\cdot]\!]$. So, e.g., $\mathrm{C}[\![x]\!]$ denotes the semantics of constructibility rule $C(x)$. Figure 2 shows the formal definitions of the rule semantics, as well as some derived formulae describing important properties: Formula PO describes the product overview, i.e. the set of all constructible, fully supplemented orders. This set is characterized by the property that for each code x out of the set $\mathcal{C}$ of all available codes a satisfied supplementing formula $S(x)$ includes the supplemented code x, and if a code x is part of an order, then its constructibility condition $C(x)$ holds. Therefore, an order O is constructible and fully supplemented exactly when it is a logical model of PO, i.e. when $O \models \mathrm{PO}$. Similarly, part p is included in the bill of materials for order O if $O \models \mathrm{R}[\![p]\!]$.

Note that with this encoding we are already able to detect parts which are not used by any possible constructible order: If the formula $\mathrm{PO} \wedge \mathrm{R}[\![p]\!]$ has no logical models, i.e. is unsatisfiable, then part p is never used.

We will now turn to documentation at the manufacturing stage and explain the extensions relative to the engineering documentation just presented.

2.2 Documentation at the Manufacturing Stage

The main difference between engineering and manufacturing documentation is the inclusion of time dependencies into the latter. Within DIALOG/P this is accomplished by adding a validity time interval and timing control codes to each rule of the DIALOG/E system.

In DIALOG/P, a rule $R(i)$ with index[2] i is therefore equipped with a validity time interval

$$I(i) = (t_\alpha(i), t_\omega(i))$$

with $t_\alpha(i) \leq t_\omega(i)$, indicating the soonest and latest time at and between which rule $R(i)$ is valid. An invalid rule is interpreted as switching off supplementation, constructibility or part selectability. To enable more complex temporal processes such as the phasing in and out of parts, each rule additionally owns a starting and a stopping control code CC_α resp. CC_ω, which allows to override the time interval limits. Intuitively, the meaning is as follows: Rule $R(i)$ is valid even before the start of the specified time interval, provided that the starting control code $CC_\alpha(i)$ is present in the order. Analogously, rule $R(i)$ is invalid even before the end of the time interval, as soon as the stopping control code $CC_\omega(i)$ occurs in the order. The exact formal time-dependent semantics of the rules—as above in the context of verification—is shown in Figure 3.

The general time-dependent semantics $[\![R, i, t]\!]$ of formula R at index i generates a formula R' representing the interpretation of rule $R(i)$ at time t. Before the starting time t_α of the rule, it is only valid if the starting control code CC_α is set and the stopping control code CC_ω is not set. Between t_α and t_ω the rule is valid as long as the stopping control code is not set, and after t_ω the rule is never valid. Note, that although an invalid rule's formula is always equivalent to $\perp$, interpretation of the whole rule can differ. So an invalid formula in supplementing rule $S(x)$ generates the rule semantics $\perp \Rightarrow x$ which is equivalent to $\top$, and thus switches off the supplementation of code x. On the other hand, an invalid formula in a constructibility rule $C(x)$ generates the rule semantics $x \Rightarrow \perp$ or, equivalently, $\neg x$, which excludes code x from any order, thus switching off constructibility of code x. Product overview, order validity and part selection are straightforward extensions of their untimed counterparts.

Timed rule semantics:

$$[\![R, i, t]\!] \;:=\; \begin{cases} R \wedge CC_\alpha(i) & \\ \quad \wedge \neg CC_\omega(i) & \text{if } t < t_\alpha(i), \\ R \wedge \neg CC_\omega(i) & \text{if } t_\alpha(i) \leq t < t_\omega(i), \\ \perp & \text{if } t \geq t_\omega(i). \end{cases}$$

Supplementing rules:

$$\mathrm{S}[\![x, t]\!] \;:=\; [\![S(x), S.x, t]\!] \Rightarrow x$$

Constructibility rules:

$$\mathrm{C}[\![x, t]\!] \;:=\; x \Rightarrow [\![C(x), C.x, t]\!]$$

Product overview:

$$\mathrm{PO}[\![t]\!] \;:=\; \bigwedge_{x \in \mathcal{C}} \Big(\mathrm{S}[\![x, t]\!] \wedge \mathrm{C}[\![x, t]\!]\Big)$$

Part selection rules:

$$\mathrm{R}[\![p, t]\!] \;:=\; [\![R(p), p, t]\!]$$

Order validity for order O at time t:

$$O \models \mathrm{PO}[\![t]\!]$$

Selection of part p for order O at time t:

$$O \models \mathrm{R}[\![p, t]\!]$$

Figure 3: Verification Semantics of Timed Rules.

There are three final remarks: First, the range of the starting and stopping times t_α and t_β is extended by the pseudo-values $+\infty$ and $-\infty$ in order to model unbounded time intervals. Second, if the control codes are not set, they are initialized to their default value $\perp$. So in the case of unspecified control codes, we get a simplified timed rule semantics:

$$[\![R, i, t]\!] \;=\; \begin{cases} R & \text{if } t_\alpha(i) \leq t < t_\omega(i), \\ \perp & \text{otherwise.} \end{cases}$$

And third, by using the extended indices $S.x$ and $C.x$ in the semantics of the supplementing and constructibility rules, we allow different timing behavior for the two kinds of rule.

How these timed rules are used to specify typical situations of change will now be shown.

[2] The index is either the code, prefixed by $C.$ or $S.$ to indicate reference to constructibility or supplementation, or the part number.

3 Typical Scenarios of Change

Many years can pass between the first prototype of a new product and the last time an instance of it is manufactured. It is not surprising that during this period of time the product itself as well as the production environment may change considerably. All this has to be reflected in the product documentation for manufacturing. Amongst the many possible changes a product and its production process can undergo, we exemplarily pick out three situations that make up a huge part of the changes in our case of the automotive industry. These are part exchange, equipment code start-up and runout, and assembly line reconfiguration. These scenarios cover changes of both the product and the production environment, as well as modifications of the product overview and the parts list.

3.1 Part Exchange

The reasons that make the exchange of parts necessary can be manifold, e.g. technical progress, change between in-house production and external procurement, or change of the supplier. The way in which the exchange is performed may also vary. There might be a cut-off date at which part p_1 is replaced immediately by part p_2 as is depicted in Figure 4a). Or the exchange has to take place over a period of time during which both variants with either part p_1 or part p_2 have to be manufactured, and for each product instance it is exactly determined by control codes which of the two parts has to be used, as is shown in Figure 4b). A third possibility, which we will not consider here, is that the new part p_2 has to be used as soon as part p_1 runs out of stock.

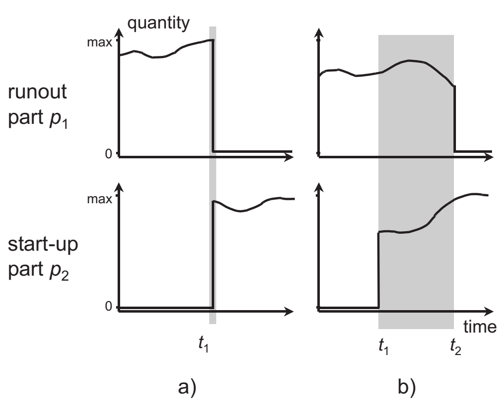

Figure 4: Part Exchange: a) Fixed Time b) Overlapping.

Fixed-time as well as overlapping part exchange can be modeled easily with the control code and time interval additions of DIALOG/P.

In the fixed-time case we get the following conditions for the selection rules of parts p_1 and p_2 to model a part exchange at time t_1:

$$t_\omega(p_1) = t_1 \qquad t_\alpha(p_2) = t_1 \quad .$$

The other time values $t_\alpha(p_1)$ and $t_\omega(p_2)$ may be set to sensible values arbitrarily, the control codes are left unspecified.

To model an overlapping part exchange we need support from the control codes. Leaving the start time of the overlap interval open, and assuming the end of the overlap at time t_2 we get:

$$t_\omega(p_1) = t_2 \qquad t_\alpha(p_2) = t_2$$
$$CC_\omega(p_1) = x_c \qquad CC_\alpha(p_2) = x_c,$$

where x_c is the control code of the overlap, i.e. all orders containing x_c use part p_2, orders not containing x_c use part p_1. Again, the remaining time values may be set to any suitable value, the control codes not mentioned are left unspecified and thus default to $\perp$. If the interval start time is to be fixed, this has to be controlled using the constructibility rule of control code x_c. Adding $t_\alpha(C.x_c) = t_1$ we get the behavior depicted in Figure 4b).

3.2 Equipment Code Start-up and Runout

We now turn to the next scenario in our presentation of common events of change: the start-up and runout of equipment or control codes. New equipment codes may show up as part of the continuous development of products. Other equipment codes may run out because they are not requested by customers any more or they have been integrated into standard packages. Most of these changes are triggered by the engineering or even the sales department. This is in contrast to the case of timing control codes, which are set by the production department, mainly to handle model year change. Model year change is an important issue and requires a lot of re-documentation, as usually quite substantial parts of the product change from one year to another. Most of the overlapping part exchanges mentioned above stem from this modification.

What makes code start-up and runout a non-trivial documentation task is that the high-level changes of the product overview influence the low-level part structure via the selection rules. In case of starting and stopping control codes the direct influence is clearly visible, but this may not be the case for other codes, or if a timing control code is used inside a rule.

Such induced, dependent changes are often very hard to detect, as can be seen from the following example: Assume a part p with an unrestricted validity time interval $I(p) = (-\infty, +\infty)$ and no timing control codes, and a selection rule $R(p) = x \wedge y$. Furthermore, let the constructibility rule of code x be $C(x) = z$ and assume an intended code runout for code z at time t_1, i.e. $t_\omega(C.z) = t_1$. Then after t_1, p cannot be part of a valid order, since runout of z induces invalidity of code x, which forces the selection rule of p to *false*.

What makes these induced runout parts hard to detect for the documentation personnel is that the codes planned for runout need not occur in the part selection rule. Besides, for complex products, different persons may be involved in the documentation of change. Automatic support by an EPDM system to find such induced runout parts is therefore highly desirable. We will present our approach to solve this problem below.

3.3 Assembly Line Reconfiguration

The last scenario of change we have a look at is to its greatest part caused by modifications of the production environment. For instance, assembly lines are reconfigured from time to time to adapt them to the actual production load. Not so frequent, but entailing considerable changes of the documentation, is the movement of whole model lines or parts of them from one assembly line to another, or even between plants.

The challenges for the documentation personnel are similar to the case of equipment code change, but they often go even beyond. The main problem is to determine the influence of the change to the parts level, with the same obstacles emerging as mentioned above.

Moreover, at least in our case, some changes are not—or not early enough—documented or even cannot be documented at all within the EPDM system. This poses the problem of handling undocumented change. For the purpose of verification, we thus need an external formalism to specify certain documentation changes that cannot be handled by the EPDM system itself.

4 Two Methods to Detect Induced Changes

We now turn to the question how the abovementioned induced changes on the part level can be determined. Our approach is as follows: Using the timed semantics developed in Section 2.2 we can convert the product overview into a time dependent propositional formula representing all constructible, fully supplemented orders at time t. This formula is the basis for formulating many criteria concerning, e.g., consistency of the product documentation or unnecessary parts. Details concerning maintenance of the product overview's consistency can be found elsewhere [Küchlin and Sinz, 2000]. Criteria generated in this way are propositional formulae, which we then test for validity or satisfiability with conventional propositional provers or SAT-checkers.

We will now present two methods to identify changes on the parts level. The first is suitable for documented changes at a fixed point of time, whereas the latter can also handle undocumented modifications of the product overview and cope with time intervals.

4.1 The $\pm\delta$-Method

With the $\pm\delta$-method we can determine which parts become superfluous resp. are additionally needed after a critical change that is already known to occur at a fixed time t_c in the future, and where the change is already documented. The procedure works in three steps:

Step 1: Determine the set P_1 of needed parts just before t_c:

$$P_1 = \{p \in \mathcal{P} \mid \mathrm{PO}[\![t_c - \delta]\!] \wedge \mathrm{R}[\![p, t_c - \delta]\!] \in \mathrm{SAT}^3\}$$

Step 2: Determine the set P_2 of needed parts just after t_c:

$$P_2 = \{p \in \mathcal{P} \mid \mathrm{PO}[\![t_c + \delta]\!] \wedge \mathrm{R}[\![p, t_c + \delta]\!] \in \mathrm{SAT}\}$$

Step 3: Compute the differences $S = P_1 \setminus P_2$ and $A = P_2 \setminus P_1$.

The resulting sets S resp. A give the sets of parts that are superfluous resp. are additionally needed after the change. The parameter δ has to be chosen such that only the critical change falls into the time interval $(t_c - \delta, t_c + \delta)$. Note, that this is—at least theoretically—a limiting factor of the $\pm\delta$-method, as it may be impossible to separate the critical change from other changes. In practice, this effect occurs rarely as the primary interest is in the situation after accumulating all changes at the critical time t_c.

[3] SAT denotes the set of all satisfiable propositional formulae.

4.2 The 3-Point Method

In contrast to the $\pm\delta$-method, the 3-point method is capable of handling documented as well as yet undocumented change. This is accomplished by providing an external (with respect to the EPDM system) formalism for specifying changes. The changes that are expressible within this formalism include:

- Equipment or control codes becoming valid or invalid.
- Arbitrary code combinations becoming invalid.

In our formalism, changes are specified as modifications of the product overview. This obviously implies a modification of the product overview's semantics. We denote the changed semantics by $\mathrm{PO}^*[\![\mathcal{C}_V, A, t]\!]$, where $\mathcal{C}_V$ is the set of codes, for which the constructibility and supplementing rules are ignored, A is the additional side condition formula. The changed semantics is defined by

$$\mathrm{PO}^*[\![\mathcal{C}_V, A, t]\!] \;:=\; A \wedge \bigwedge_{x \in \mathcal{C} \setminus \mathcal{C}_V} \Big(\mathrm{S}[\![x, t]\!] \wedge \mathrm{C}[\![x, t]\!]\Big) \ .$$

Validation of an invalid code x, i.e. a code with constructibility rule $C(x) = \bot$, can be achieved by including code x into the set of validated codes $\mathcal{C}_V$. If it should be necessary, a new constructibility or supplementing rule can be specified as conjunctive part of formula A. Invalidation of codes, as well as additional side conditions, are specified by conjunctively adding formulae to A, e.g. $\neg x$ to indicate code x getting invalid.

Having now briefly introduced our external specification formalism, we now turn to describing the 3-point method.

For the 3-point method, two points in time, t_0 and t_1, have to be fixed between which the undocumented changes should occur. Moreover a modified product overview semantics $\mathrm{PO}^*[\![\mathcal{C}_V, A, t]\!]$ with a fixed set $\mathcal{C}_V$ and a side-condition formula A, as introduced above, is employed to reflect/indicate undocumented changes. The 3-point method is composed of four steps:

Step 1: Determine the set P_{t_0} of needed parts at time t_0, i.e. before the change:

$$P_{t_0} = \{p \in \mathcal{P} \mid \mathrm{PO}[\![t_0]\!] \wedge \mathrm{R}[\![p, t_0]\!] \in \mathrm{SAT}\}$$

Step 2: Determine the set P_{t_1} of needed parts at time t_1 without undocumented changes:

$$P_{t_1} = \{p \in \mathcal{P} \mid \mathrm{PO}[\![t_1]\!] \wedge \mathrm{R}[\![p, t_1]\!] \in \mathrm{SAT}\}$$

Step 3: Determine the set $P^*_{t_1}$ of needed parts at time t_1, including undocumented changes:

$$P^*_{t_1} = \{p \in \mathcal{P} \mid \mathrm{PO}^*[\![\mathcal{C}_V, A, t_1]\!] \wedge \mathrm{R}[\![p, t_1]\!] \in \mathrm{SAT}\}$$

Step 4: Compute the differences

$$\begin{aligned} A_{10} &= P_{t_1} \setminus P_{t_0} & \quad S_{10} &= P_{t_0} \setminus P_{t_1} \\ A_{*0} &= P^*_{t_1} \setminus P_{t_0} & \quad S_{*0} &= P_{t_0} \setminus P^*_{t_1} \\ A_{*1} &= P^*_{t_1} \setminus P_{t_1} & \quad S_{*1} &= P_{t_1} \setminus P^*_{t_1} \end{aligned}$$

Here, e.g., A_{10} indicates the additional parts needed at time t_1, ignoring undocumented changes, relative to the parts needed at time t_0. The relationship between the three sets of parts and the difference sets are graphically illustrated in Figure 5.

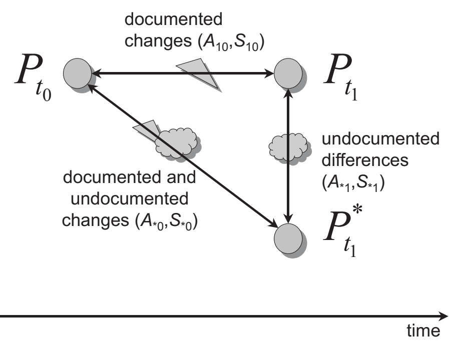

Figure 5: The 3-Point Approach.

So to find out what is the impact of an intended product overview change on the part usage, we have to take a look at the difference sets. The sets A_{*0}/S_{*0} indicate the overall change between t_0 and t_1 if the intended (undocumented) change really is performed, including all changes induced by already documented events. Using the difference sets A_{*1}/S_{*1} the changes induced at time t_1 by the undocumented modifications alone can be displayed. Moreover, and similar to the $\pm\delta$-method, the sets A_{10}/S_{10} only show the impact of already documented changes during the time interval (t_0, t_1).

4.3 Discussion of Both Methods

Comparing the two methods, the $\pm\delta$-approach offers the advantage of easy handling. To find out the impact of a change on the parts' world only the point of time of this change has to be specified. On the other hand, the intended modification already has to be documented, and the time of the change has to be fixed. Whereas this is usually the case for planned, regularly occurring events like code start-up and runout due to model year change, this may not be the case for other product modifications, e.g. by further product development. Here the 3-point method can play out its strength of handling even undocumented modification events, however, at the cost of increased complexity in usage. This shows up in the need to specify the modified product overview semantics $\mathrm{PO}^*[\![\mathcal{C}_V, A, t]\!]$. In most cases, though, the undocumented changes follow certain patterns, so that special cases of the modified semantics may be pre-encoded and offered as specialized tests.

Note that the 3-point method properly includes the $\pm\delta$-method. As we have $\mathrm{PO}^*[\![\emptyset, \top, t]\!] = \mathrm{PO}[\![t]\!]$, by setting $t_{0/1} = t_c \pm \delta$ in the 3-point method, we get a specialization equivalent to the $\pm\delta$-approach. In this case we have $P^*_{t_1} = P_{t_1}$ and only the difference sets A_{10} and S_{10} are of interest. Another weakness of the $\pm\delta$-method already mentioned in Section 4.1 is that separation of two events may be impossible. The 3-point method allows handling of such a case by re-modeling the relevant events externally.

4.4 Mapping of Typical Cases

We will now show how to map two important scenarios of change to our verification formalisms.

Our first case handles equipment code start-up and runout caused by model year change, for which we use the $\pm\delta$-method. Model year change usually is accompanied by lots of changes, mainly on the parts level, but also to a smaller fraction on the product overview level. During an overlapping interval both models from the old and the new model year have to be manufactured. Assume codes m_o and m_n are responsible for controlling model year change, i.e. orders for cars of the old model year are equipped with code m_o, for the new model year with code m_n. Assume further that the model year change is fixed to take place during the time interval (t_0, t_1). The interesting question is which parts are not needed any more after t_1. In the documentation the run-out of the old model year is reflected by code m_o becoming invalid, as well as code m_n becoming mandatory at t_1. Moreover, some parts may happen to have t_1 as a starting or stopping time. Summarized, the rules changing at time t_1 are:

$$t_\omega(C.m_o) = t_1,$$
$$t_\alpha(S.m_n) = t_1 \text{ with } S(m_n) = \top,$$

as well as part selection rules of parts p with either $t_\alpha(p) = t_1$ or $t_\omega(p) = t_1$. We thus set up the $\pm\delta$-method with $t_c = t_1$ and get resulting difference sets of A and S, indicating additionally needed and superfluous parts after the end t_1 of the model year change overlap interval. Obvious starting or stopping parts (i.e. parts with $t_\alpha(p) = t_1$ or $t_\omega(p) = t_1$) may additionally be filtered out to get a more concise result.

Let's now turn to production relocation, where we consider movements of parts of the production from one assembly line (or plant) to another. Of this two-sided problem of moving in and off, we concentrate on the move-off part. Such a kind of change cannot (easily) be handled within the DIALOG/E system, as not only individual codes, but arbitrary code combinations, representing the fraction of the production that is to be relocated, are getting invalid after the change. One big problem related to production move-off is to determine the induced parts shift.

To handle this case, we use the 3-point method to find out precisely the induced parts shift. We set up t_1 as the approximated time of the relocation event, and t_0 as the current time. The modified product overview semantics is set to $\mathrm{PO}^*[\![\emptyset, \neg F, t]\!]$ where F is a formula describing the fraction of the production to be moved off.

As an example, let us consider the situation where the production of cars containing the motor variants M1,M2 and M5 in cunjunction with automatic gears (A) is planned to be moved off, but not for the destination countries C1, C3 and C4. The formula

$$F \;=\; (\mathrm{M1} \vee \mathrm{M2} \vee \mathrm{M5}) \wedge A \wedge \neg(\mathrm{C1} \vee \mathrm{C3} \vee \mathrm{C4})$$

describes this production shift.

The results delivered by the 3-point method are manifold. Perhaps the most important parts shift sets are A_{*1}/S_{*1}. They indicate the additional and superfluous parts after the relocation at t_1 relative to the situation at the same time without the relocation having taken place. If the overall change on the parts level between the current situation (at t_0) and the projected situation after the relocation at t_1, also including already documented product changes, is of interest, then the difference sets A_{*0}/S_{*0} provide the appropriate information.

4.5 Verificational Aspects

To give an impression of the size of problems that has to be dealt with, Table 1 summarizes some characteristical values for three model lines we have investigated.

	Line 1	Line 2	Line 3
#Codes	567	567	567
#C-Rules	554	347	594
#S-Rules	133	79	139
#R-Rules	9427	4461	10779
$\|$PO$\|$	30826	15981	35342

Table 1: Characteristics of Three Model Lines.

#Codes denotes the number of codes occurring in the documentation of the model line. This number is equivalent to the number of different propositional variables in the formulae for the SAT-checker. The three lines *#C-Rules*, *#S-Rules* and *#R-Rules* indicate the sizes of the sets of constructibility rules, supplementing rules resp. part selection rules at a fixed point of time[4]. $|$PO$|$ shows the size (in logical symbols) of the product overview formula.

All of the above tests generate huge sets of propositional satisfiability (SAT) problems: two (in case of the $\pm\delta$-method) resp. three (for the 3-point approach) SAT-tests for each part selection rule. Fortunately, most of these problems are solved by today's SAT-checkers—as well as by the prover implementation that is part of BIS—in under a second [Küchlin and Sinz, 2000]. For the rare long-running exceptions, a parallelization approach can dramatically speed up average waiting times for individual proof results [Blochinger *et al.*, 2001].

5 Conclusion

We presented two methods for computing the influence of high-level product changes on the parts level, which is frequently needed to keep product documentation for manufacturing in a consistent state.

Although we described our ideas from the viewpoint of the automotive industry, we expect our methods to be transferable to other industries as well, especially when lots of configuration possibilities fall together with large production numbers.

Our prototypical implementation, which is part of the BIS[5] project, is currently evaluated at DaimlerChrysler's Sindelfingen production plant for the Mercedes lines. We expect valuable feedback from this pilot project.

[4]Missing constructibility rules and supplementing rules are interpreted as $\perp$.

[5]See also http://www-sr.informatik.uni-tuebingen.de/pdm.

References

[Blochinger *et al.*, 2001] W. Blochinger, C. Sinz, and W. Küchlin. Parallel consistency checking of automotive product data. 2001. Submitted to ParCo'01.

[Günter and Kühn, 1999] A. Günter and C. Kühn. Knowledge-based configuration: Survey and future directions. In *XPS 1999*, number 1570 in LNCS, pages 47–66. Springer-Verlag, 1999.

[Haag, 1998] A. Haag. Sales configuration in business processes. *IEEE Intelligent Systems*, 13(4):78–85, July/August 1998.

[Kaiser and Küchlin, 2001] A. Kaiser and W. Küchlin. Automotive product documentation. In *Proceedings of the 14th International IEA/AIE Conference*, LNCS. Springer-Verlag, 2001. To appear.

[Küchlin and Sinz, 2000] W. Küchlin and C. Sinz. Proving consistency assertions for automotive product data management. *J. Automated Reasoning*, 24(1–2):145–163, February 2000.

[McDermott, 1982] J. McDermott. A rule-based configurer of computer systems. *Artificial Intelligence*, 19(1):39–88, 1982.

[Sabin and Weigel, 1998] D. Sabin and R. Weigel. Product configuration frameworks – a survey. *IEEE Intelligent Systems*, 13(4):42–49, July/August 1998.

[Sinz *et al.*, 2001] C. Sinz, A. Kaiser, and W. Küchlin. Detection of inconsistencies in complex product model data using extended propositional SAT-checking. In *Proceedings of the 14th International FLAIRS Conference*. AAAI Press, 2001. To appear.

[Timmermans, 1999] P. Timmermans. The business challenge of configuration. In B. Faltings, E. Freuder, G. Friedrich, and A. Felfernig, editors, *Configuration*, number WS-99-05 in Workshop Technical Reports, pages 119–122. AAAI Press, 1999.

[Wright *et al.*, 1993] J. R. Wright, E. Weixelbaum, G. T. Vesonder, K. E. Brown, S. R. Palmer, J. I. Berman, and H. H. Moore. A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T Network Systems. *AI Magazin*, 14(3):69–80, 1993.