ELSEVIER

Data & Knowledge Engineering 36 (2001) 55–78

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

Multiple abstraction levels in modelling product structures

Tomi Männistö *, Hannu Peltonen, Timo Soininen, Reijo Sulonen

Product Data Management Group, TAI Research Centre and Laboratory of Information Processing Science, Helsinki University of Technology, P.O. Box 9555, 02015 Hut, Finland

Received 20 September 1999; received in revised form 14 March 2000; accepted 20 June 2000

Abstract

The need for product customisation is driving industrial companies towards a very large product variety, which affects many functions of a company, including the after-sales. Systematic maintenance records of very different product individuals cannot be kept without an abstract view to the population of delivered products. However, the older the product individual, the less systematically recorded information there usually is about it. We defined a novel mechanism based on generic models of product individuals organised into a specialisation hierarchy to support multiple abstraction levels. For creating such hierarchies, we defined a set of transformation operations on models. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Conceptual modelling; Generic product structure

1. Introduction

Often, especially with rather expensive investment products, it is important for a manufacturing company to know exactly what has been delivered to a customer. That is, for each *product individual*, the *as-manufactured view* or, if the information of the installation environment is essential, the *as-installed view* may need to be recorded. The as-manufactured information serves at least partly as a basis for the *as-maintained* information, which is constantly updated information about the product individual, possibly including its maintenance history. The possibilities for recording such information vary considerably. In this section, we discuss some special aspects of the after-sales management of products that have a large number of variants.

First, data structures suitable for manufacturing are not the most appropriate for after-sales. After-sales requires its own concepts and its information must be presented in a view of its own. For example, the product breakdown structure for after-sales should reflect the relevance of the concepts in maintenance. Thus, components needing regular service, such as brakes, may deserve a more prominent position in the after-sales view than in the manufacturing view.

* Corresponding author. Tel.: +358-9-451-3373; fax: +358-9-451-4958.
E-mail address: Tomi.Mannisto@hut.fi (T. Männistö).

In addition, from the point of view of maintenance, it is not sufficient to know that a particular component is a part of a product; it is equally important to know which role or function the component serves. However, from a manufacturing *bill-of-materials* (BOM) it may be hard to identify, for example, the motors used in any delivered crane as a hoisting motor. That is because such function is not a unique property of a particular type of motor, which can be used as a hoisting motor in one crane and as a traversing motor in another crane. Therefore, the modelling approach must take a more functional view to the product than a BOM.

In addition to a role of a component, sometimes the separation of identical components in identical roles is relevant. For example, certain types of cranes have a pair of identical traversing motors, one at each end of the bridge. From the after-sales perspective they, however, may be quite different if the motor in one end wears out much faster than the motor in the other end. Therefore, as component types do not provide the required information, it must be modelled otherwise.

Furthermore, instead of buying service hours after something breaks down, customers want to buy 'up-time', that is, guaranteed operation without unexpected service breaks. This requires accurate knowledge on the wear of different components and problems encountered earlier. It is difficult to gather such statistical information about product individuals that are made to order and are thus quite different from each other. To enable this, standardised means for modelling product individuals are needed, i.e., a mechanism is needed for stating what information is relevant, so that the same information is collected from all individuals. That is, for statistical analyses, it is more important to have some systematic information on all individuals than detailed information on some individuals.

In addition, the wide variation in the age of product individuals means that one cannot speak of all product individuals at the same level of accuracy – what is known of an old product individual can be very inaccurate. The problem is increased when a service organisation also maintains product individuals made by other manufacturers. Therefore, it must be possible to model product individuals at varying levels of abstraction.

An approach for solving these problems is presented in this paper. The approach is a data modelling methodology that supports the description of product individuals for after-sales purposes. The essential issue is the possibility to identify similar functional components in different product individuals so that after-sales operations, such as the change of a particular kind of motor, can be recorded for them. Such information can then be utilised in various analyses, for example, to improve preventive maintenance.

2. Background and related work

In this section, some common terms are defined as used in this paper and related work is briefly reviewed. Not all terms have a universally accepted meaning and therefore they may be used differently in other contexts.

A *product* is composed of *components*, which in turn are composed of further components, etc. This hierarchical breakdown of a product is called its *product structure*, or also BOM. Sometimes a distinction is made between a *master BOM* and an *order BOM*. The former serves as a recipe for manufacturing many products, for example, in mass-production, while the latter specifically describes the structure of the *product individual* of a single customer order, for example, in make-to-order or assemble-to-order production. In the following, the term ‘BOM’ refers to an ‘order BOM’. When the distinction is of importance, the term *product type* is used to refer to a type of product, for example, ‘Car *Y* model *LXi*’. When the distinction is less important or clear from the context the term ‘product’ is used. The terms ‘component’, *component type* and *component individual* are used similarly. The term *part* refers to the relation between components. A component can thus have other components as parts.

In illustrations, such as Fig. 1, we present components as rounded boxes, which contain an identification of the component type, e.g., an item code, and possibly a serial number of the component individual. The whole product and some critical components are typically identified with a serial number but for basic components such as screws and bolts it is not necessary to identify them individually. In figures, we show the parts of a component individual below the component individual. In Fig. 1, for example, ‘crane #699861’ has ‘t012’ as a part. Note that in Fig. 1 the names in the boxes are names for components; they are not names for the part relation, i.e., *part names.* A part name ‘hoist’ could for example, be attached to the relation between ‘crane #699861’ and ‘t012’. In an ordinary BOM, part names are typically just position numbers of the parts. The issue of not having part names will be discussed in Section 7 at the end of this paper.

This paper concerns products that have a large number of variants. One special class of such products is *configurable products.* They are *configured* to the needs of each customer in a

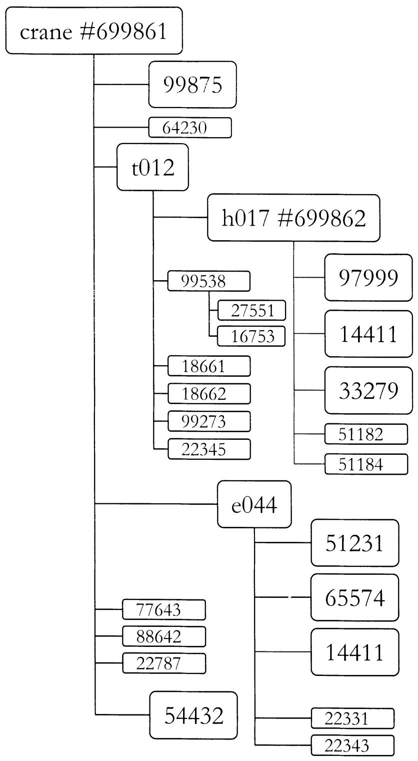


Fig. 1. Sample BOM with components as parts of components.

configuration process according to a pre-defined *configuration model* [14,15]. A product with a large number of variants, however, need not be a configurable product. Therefore, a more generic term for a configuration model is a *generic product structure model* [10]. The term 'structure' indicates that the interest is mainly in the variation of product structure, i.e., the components of the product. The term 'generic' says that a single data model is used to represent multiple product variants. In a similar sense, the term *generic BOM* is used in data modelling approaches in which product variants are modelled by extending the traditional BOM concept [16]. For simplicity, we use in this paper the term 'configurable product' as a synonym for 'a product that has a large number of variants' although the usage is slightly inaccurate.

To efficiently support the after-sales, one needs to aggregate information from product individuals. For configurable products this may be particularly complicated as all product individuals are different [3]. The issue has also been addressed in STEP, which is a standard for product data modelling [5,11]. The development work within STEP includes Application Protocol 221 for process industries and in close relation to it, an initiative called STEPlib [6,7]. STEPlib aims at providing a standardised data model and a dictionary for describing individual process plants. There are, however, some fundamental problems in representing generic product structures in STEP [9]. For example, STEP makes it impossible for a company to build its generic products structures with the powerful is-a relation and inheritance mechanism of the EXPRESS language, which is only used for building the STEP data models. Nevertheless, also within STEP, it has become quite evident that no single data model for a product can serve all the needs of a company. For example, manufacturing and packaging see the structure of a product very differently. In addition, the location and functional information about the components and collections of them is sometimes important. These different aspects of a product are typically called *views* [3,17]. The breakdown structure is not the only feature of a product that may differ between views. A view may also need different concepts or different semantics for the concepts, such as, a is-a relation of its own [10]. An interesting issue is how the views relate to each other. In this paper, we concentrate on data models for the after-sales view of different product individuals and on the relation between this view and the manufacturing view.

3. Overview of the approach

The goal of this paper is to construct means for modelling product individuals of configurable products. The construction has three main concepts: *element structure, element model* and *element model hierarchy*. An overall picture of the goal is first given here with the help of Fig. 2; the rest of the paper then defines the concepts in detail.

The basic assumption is that each product individual has a product structure of its own, called an order BOM. Because of the very large variety, each product individual is potentially different. In each order BOM, some essential components, such as hoisting motor and end carriage of a crane, are identified. This is done by means of an *element structure*. An element structure consists of *elements*, e.g., 'aa' in the figure, and it provides another viewpoint to the product individual; a viewpoint relevant to after-sales.

The point, however, is not just to have another structure describing the product individual. The point is to describe the product individuals in a standard manner, so that their history can be consistently recorded and statistical analyses made on the similar properties of them. Therefore,

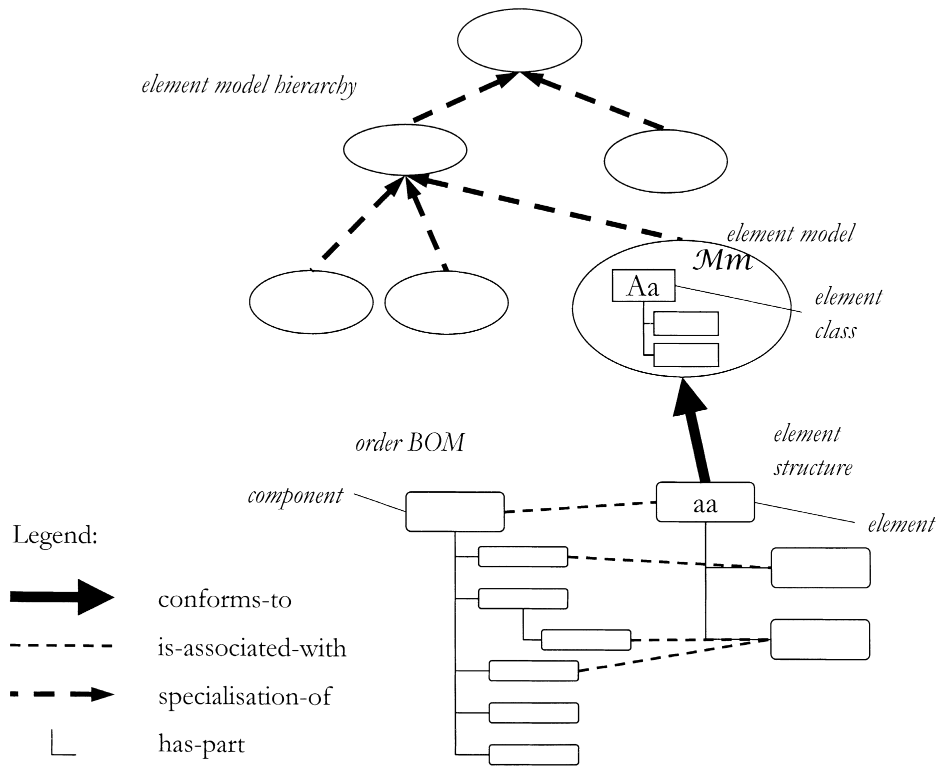


Fig. 2. Overall view to basic concepts defined in the paper.

the element structures are required to conform to an *element model*, e.g., '$\mathcal{M}m$' in the figure. An element structure describes by means of *element classes*, e.g., 'Aa' in the figure, what elements must be in an element structure and which are optional, for example. However, if there is only one element model for all products of a company, the element model needs to be rather general. Moreover, not all individuals are known at the same level of accuracy. For example, individuals manufactured by other companies are not known as accurately as those manufactured by the company itself and less strict element models are needed for them than for own products. Therefore, multiple element models are needed. The idea in this paper is to organise element models into a specialisation hierarchy called an *element model hierarchy*. From the element model hierarchy, the most appropriate element model is selected for the element structure of a particular product individual. Moreover, the element model hierarchy allows investigation of the individual through more general element models. For example, all individuals can be seen through the most general element model. In the following, the concepts will be dealt with in turn. Section 4 starts with element structures, Section 5 discusses element models and Section 6 element model hierarchy.

This works originates from the co-operation with a crane manufacturer, and therefore, as examples this paper uses cranes. However, it should be noted that the examples are mainly invented for illustrating the concepts and ideas of the paper and do not represent any actual products.

4. Element structure

In this section, we introduce the modelling basis of our approach for representing product individuals. The approach consists of *elements* that form an *element structure*. An element structure is an alternative view to a BOM of a product individual. While a BOM describes the breakdown of a product individual from the manufacturing point of view, an element structure describes the aspects of a product individual that are relevant for after-sales. We begin the introduction with the concept element.

4.1. Element

A BOM specifies the structure of a product very precisely, specifying the type of each component by an item code. Many functions, such as maintenance, however, need to identify the main components found in different kinds of products of a company. In this paper, such identification is realised by means of elements.

Components in a BOM can be *associated* with elements. When a particular component in a particular product is associated with a particular element, one can also say the component is in the *role* defined by the element. Different components can fulfil the role represented by an element.

As an example, consider a crane element, such as an end-carriage. Different cranes associate different components with this particular element (or with the pair of them). This makes it possible to take a crane and find the components that are used as end-carriages without knowing exactly which kind of end-carriages the particular crane happens to have. Elements can also record interesting data about cranes. For example, when the end-carriages of each delivered crane are associated with the corresponding elements, it becomes possible to attach maintenance data to the elements and analyse statistically the maintenance histories of end-carriages.

Note that elements are only defined for components that are "relevant" for some reason in maintenance. For example, no element corresponds to a particular bolt in a particular kind of product if the role or function of this particular bolt is not of a special interest.

4.2. Product individual and element structure

Consider a single delivered physical product, i.e., a product individual. A product individual has a BOM. In addition to the BOM, the product individual also has an *element structure*. Like the BOM, the element structure is a hierarchical breakdown of the product. The items in the structure, however, are elements. The term *part* is used with element structures in the same way as with BOMs. An element can thus have other elements as parts.

Elements in an element structure can be associated with components in a BOM. Compared to the BOM, the element structure is intended to provide a more abstract or "higher level" view of a product. Therefore, there are usually altogether much fewer different elements than different components.

As an example, consider Fig. 3. The left-hand side of the figure shows a BOM of a product and the right-hand side shows an element structure for the same product. The dashed lines denote the associations between elements and components.

Each element that does not have parts must be associated with some component of the corresponding BOM. That is, an element structure describes the after-sales structure of product

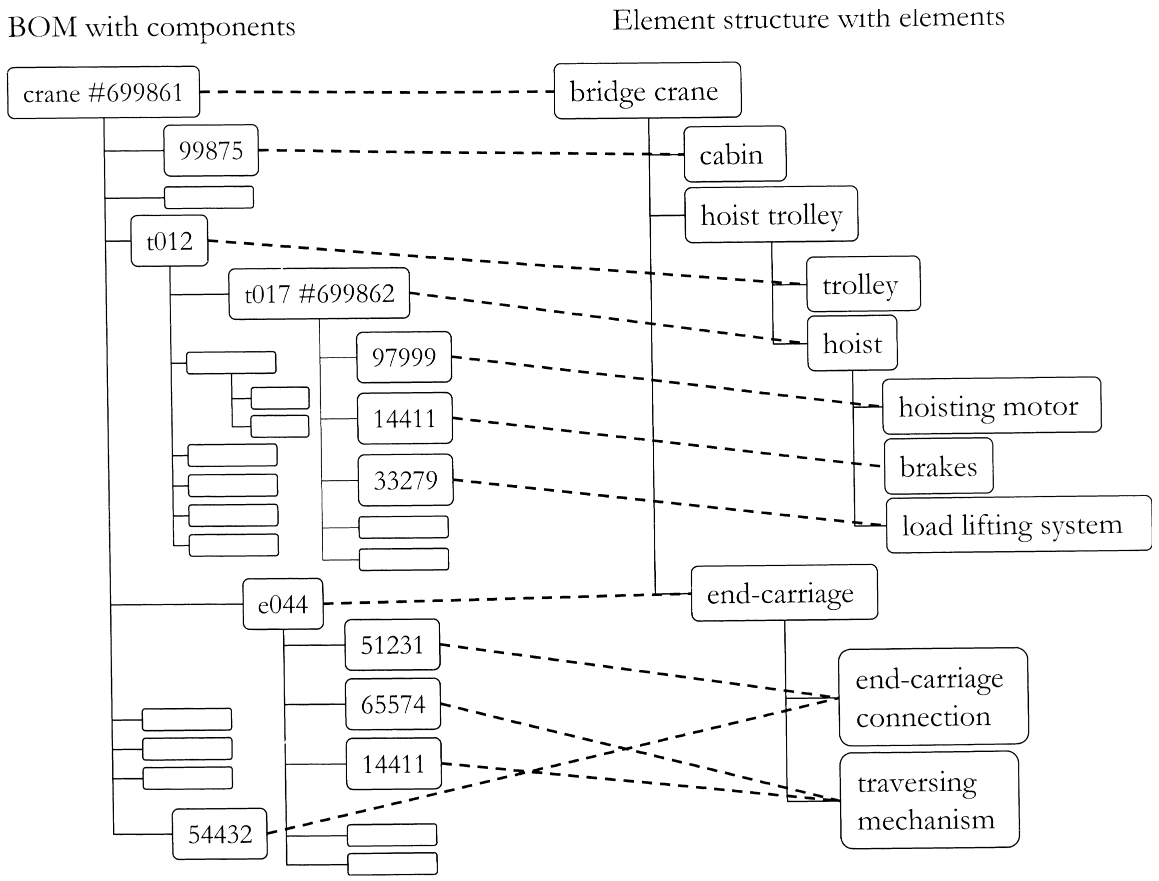


Fig. 3. Association of BOM and element structure.

individual and the components that take the roles specified by elements must be identified from the product individual. We say that an element is *validly associated* if it is associated with a component or it has parts and its immediate parts are all validly associated. For example in Fig. 3, element 'hoist trolley' is validly associated because all its parts are associated with components.

In Fig. 3, the product as a whole is represented by the component 'crane #699861', which is associated with the element 'bridge crane'. The element 'cabin' is an immediate part of the element 'bridge crane', and the associated components are related in the same way, i.e., the component '99875' associated with the element 'cabin' is an immediate part of the component 'crane #699861'. This condition need not be satisfied. For example, the element 'trolley' is not an immediate part of 'bridge crane' although component 't012' is an immediate part of 'crane #699861'. The idea of element structures is to provide a view to the existing set of product individuals. The view is organised according to the requirements of after-sales.

The association between a BOM and an element structure makes it possible to see the product from the point of view of the elements. The use of an element structure as a view to the product individual is based on the idea that an element structure alone is not a complete description for the product individual. It provides a point of reference and a navigational support to the components of the product individual, which are all included in the BOM of the product. Therefore, we see the different views to a product individual so that there is at least one primary view, for example, BOM. A primary view contains all the components and then there are separate views for accessing and complementing the information of the primary view.

After-sales is supposed to mainly operate with the element view; for example, to attach service operations, to analyse maintenance needs, and so on. Because of the association with BOMs, the information gathered or analysed in the element view is also linked to the product view, for example, to the actual components with high fault rates.

For recording the maintenance history, the creation and modification times of associations can be preserved. For example, when a new component is associated to element 'hoist', the time of the change is recorded. With such information, it is later possible to find the component associated to 'hoist' at a given time.

With respect to associating components with an element structure, we required only that all elements must be validly associated. This requirement is set to guarantee some standard use of the element structures. If all elements of an element structure cannot be validly associated, a different element structure must be used. We do not restrict the number of elements that can be associated with a single component or vice versa. It is possible, for example, that a role represented by an element is implemented by a set of components instead of a single component and thus multiple components are associated with the element. Examples for such associations are elements 'end carriage connection' and 'traversing mechanism' in Fig. 3. Similarly, it is possible to associate one component with multiple elements.

5. Element model

For after-sales, it is important to represent the product individual according to a standard vocabulary. Therefore, we define *element models*, which describe the valid element structures. In order to ensure standard descriptions, the element structure of a product individual must *conform* to an element model. Recall that an element structure describes the structure of a single product individual and that an element structure must be validly associated with the BOM. This, however, is not adequate if there is no control over what kind of element structures are possible. Therefore, to capture the wanted similarities in different products, the element structures must be constructed in a standard manner, which is achieved by describing the valid element structures using element models.

An element model is generic in the sense that it represents a number of structurally different element structures. Element models specify the legal variation of element structures by means of optional and alternative parts.

In general, we say that an element structure *conforms* to an element model if there is no contradiction between the structure and the model. For instance, the structure must include all elements that are compulsory in the model and it cannot include any elements that have not been mentioned in the model. Next, we will elaborate this in more detail; more precise definitions can be found in Appendix A.

5.1. Element classes

An element model is built of *element classes*. Each element in an element structure is related to an element class, and we say that the element is an *individual* of the element class. For simplicity, we do not discuss in this paper any typical properties, such as attributes, of element classes.

Element classes are organised into a *class hierarchy* in which an element class may have *subclasses*. For example, 'Hoist' can have subclasses 'Chain hoist' and 'Wire rope hoist', and 'Wire rope hoist' can further have subclasses 'Compact hoist' and 'Open winch hoist'. In figures, such as Fig. 4, element classes are graphically represented as boxes and so distinguished from elements, which have rounded corners. An element model is represented by an oval that contains element classes and the relationships between classes having other classes as parts. Fig. 4 uses abstract names for element classes because its purpose is to introduce in a concise form the various ways the element model represents the element structures. In practice, classes in an element model would most probably have similar names as elements in Fig. 3; element 'hoist' could be an instance of element class 'Hoist'. The point in Fig. 4 is to illustrate the means for representing the allowed variety for element structures. These means will be discussed next.

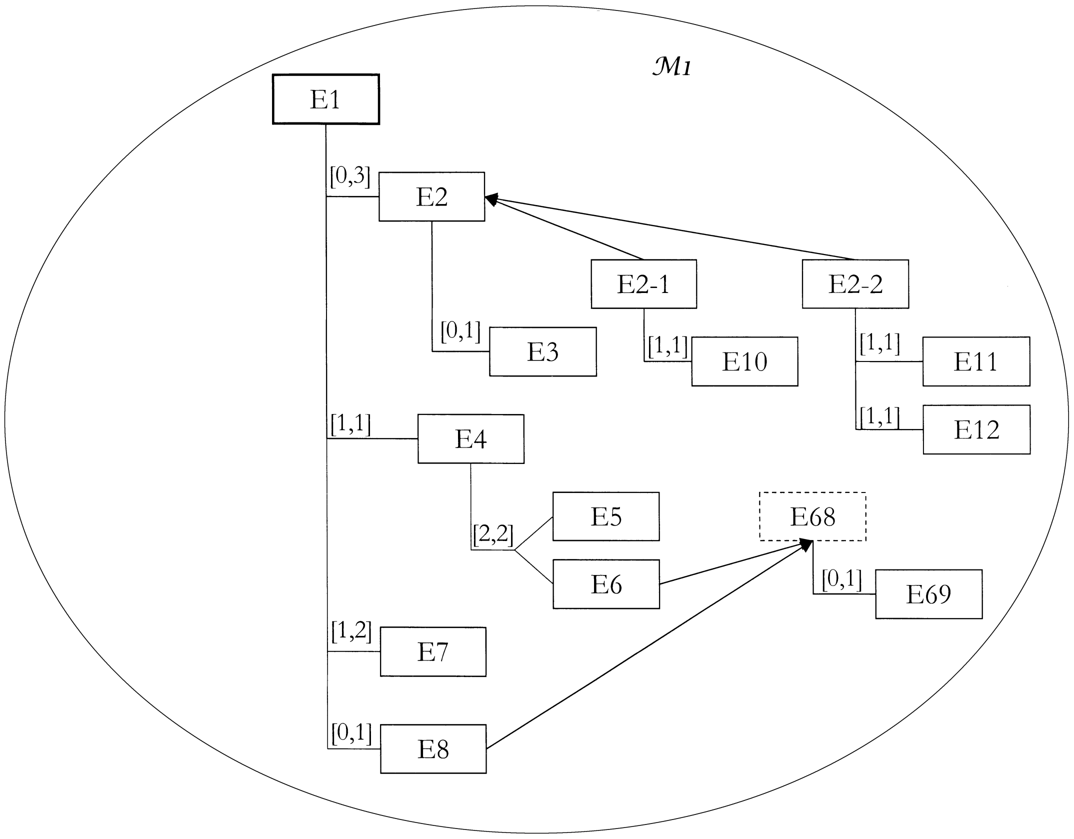


Fig. 4. Element model $\mathcal{M}1$.

We also use the term *superclass* for the relations between element classes in the class hierarchy. In figures, an arrow pointing from the subclass to the superclass represents *is-a relation* between the element classes. In Fig. 4, element classes E2-1 and E2-2 are subclasses of E2 and E2 is a superclass of E2-1.

The is-a relation is used together with parts. We say that an element class *inherits* the parts of its superclasses. For example, in Fig. 4 element class E2-1 has parts E10 and E3. This means, for example, that in an element structure an individual of E2-1 must have an individual of E10 as a part. Since we have ignored other properties of classes, parts are the only inherited properties in this paper.

If 'Hoist' has subclasses 'Wire rope hoist' and 'Chain hoist', there is the question whether elements of 'Hoist' are allowed in element structures or whether one is required to select one of its subclasses to describe the hoist element more precisely. As the answer seems to depend on the element class, each element class must be designated either as *abstract* or *concrete*. All elements in a valid element structure must be individuals of concrete element classes. For example, if 'Hoist' is an abstract element class in a model, a more specific hoist class must be selected in a valid element structure, e.g., 'Chain hoist'. In figures, we denote abstract element classes by dashed boxes. The terms have been used similarly in object-oriented modelling and in relation to product configuration modelling [8,13,15,18].

An abstract element class without subclasses does not "make sense" because there cannot be any individual of the class, and therefore, we require that a class without subclasses cannot be abstract. This invariant must be fulfilled by each element model. If a concrete element class has subclasses, typically the subclasses are concrete, too. This, however, is not required.

Concreteness of element classes is important for the definition of a model since it allows one to specify how strict an element model is. We will shortly return to this issue in the discussion of element model hierarchies.

5.2. Generic has-part relation between element classes

In order to model multiple different element structures, an element model must include some mechanisms for generic aspects. Typical concepts of generic has-part relations are *optional part*, *alternative parts* and *cardinality*, i.e., a specified range for the valid number of parts [15]. For example, 'Lighting' could be an optional part of a crane. 'Thyristor control' and 'Inverter control' could be alternative parts. A cardinality specification could state that there must be exactly two 'End-carriages' in a particular type of crane.

Fig. 4 shows an example of the generic has-part concepts we consider in this paper. The cardinality is denoted as a pair of numbers in square brackets next to the has-part relation, e.g., as [1,2] between E1 and E7. An optional part can be represented as a special case of cardinality in which the lower limit for the number of parts is zero. In the figure, E3 is an optional part of E2.

For alternative parts, we use graphical notation of a branching line, for example, E5 and E6 are alternative parts of E4. It could be possible to attach a cardinality before the branch or to the branches (or to both). A cardinality for a single branch would define the number of parts from that particular element class whereas the cardinality before the branch restrict the total number of parts from the alternative element classes. For simplicity, we only allow a cardinality to be attached before the branch as in Fig. 4. The element model specifies that an E4 has as its parts two E5s, two E6s or one of each.

Next, we briefly discuss some detailed properties of element models that are needed for consistency. Our intention is to support element models such as in Fig. 4, and therefore, some invariants are set for element models and element structures.

- *No multiple inheritance*. We do not find the expressive power of multiple inheritance necessary for the models presented in this paper, and since it always induces further problems, we opt not to include it into the supported concepts. Therefore, it is required that each element class has at most one immediate superclass.
- *No circular class definitions*. It is required that no element class is directly or transitively a subclass of itself. This is simply an anomaly we want to rule out.
- *No element class is part of itself*. It is also required that no element class is a part (directly or transitively) of itself. For example, X1 must not be both a part and a subclass of class X because then X1 would have itself as a part via inheritance. (This invariant is actually not necessary, but we decided to keep the model simpler by requiring it.)
- *Element structures are trees*. For element structures, the requirement that an element cannot be a part (directly or transitively) of itself is quite natural. That and having an element as part of two elements is prohibited by requiring that an element structure must form a tree. This requirement also prohibits element structures that are not connected.
- *Existence of root element class*. It is also required that exactly one element class is declared as the *root element class*. The root element class must not have superclasses, and neither the root element class itself nor any of its subclasses must be a part of any element class. The root of the element structure must be an individual of the root element class or its subclass. For example, in Fig. 4, the box of E1 is drawn with a thick line to denote it is the root element class.

6. Element model hierarchy

It is impossible to describe all the product individuals of a company with a single element model-unless the model is extremely generic and therefore rather useless. This means that a company needs multiple element models, for example, one for each different basic construction. To make it easier to use and manage element models, we organise the models in an *element model hierarchy*. With these concepts, a large number of different product individuals can be represented with a vocabulary that allows an effective analysis of them.

As discussed before, we want to define element models at different abstraction levels. The structure that is common to all product individuals having a similar product architecture and manufactured or maintained by the company is specified by the most general element model, such as a general crane model. The product can be further divided into different types. For each product type, there can be a separate, more specific element model describing the structure of the particular product type. Furthermore, these models can have more specific element models with more detailed descriptions. The result is a tree-like hierarchy of element models as illustrated in Fig. 5. The general element model is at the top of the hierarchy, and the element models for different product types are located under the general model. For example, jib crane is a particular type of crane with its own element model. Jib cranes can further be specialised into pillar jib cranes, wall jib cranes and fixed jib cranes, each of which in turn has its own element model under the model describing jib cranes.

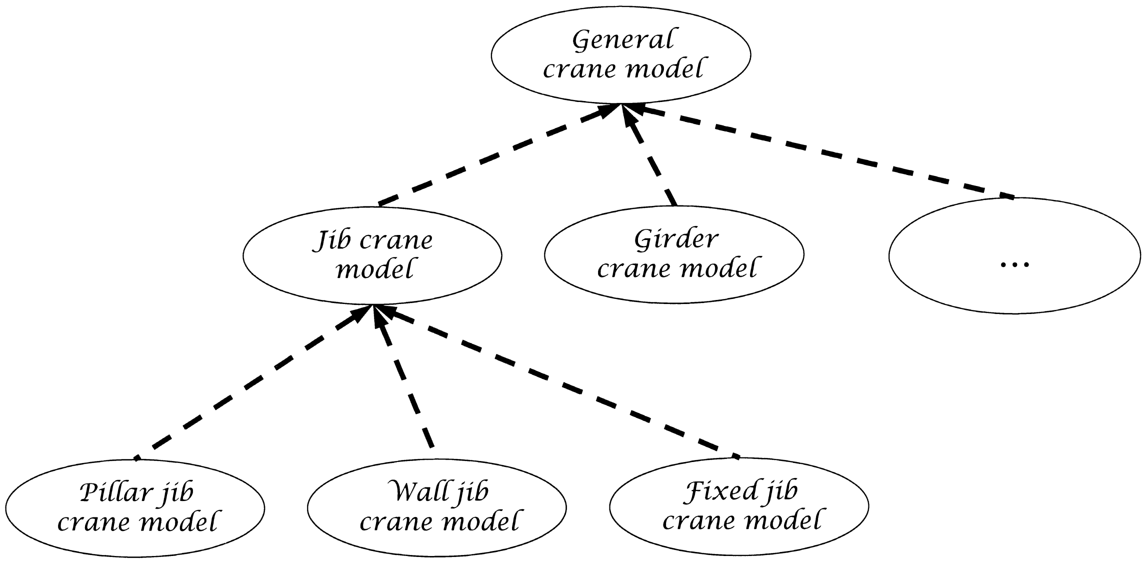


Fig. 5. Example of an element model hierarchy.

Each model in the hierarchy, e.g., each oval in Fig. 5, contains an element model as presented earlier, e.g., $\mathscr{M}$**1** of Fig. 4. The point is that one chooses as detailed an element model as possible to represent an element structure of a particular product individual. An element model can be used for analysing all element structures conforming to it or any of the models "below" it. Consequently, all element structures can be analysed through the most generic element model, i.e., 'general crane model' in Fig. 5. We will next elaborate on the relation of models in the hierarchy.

6.1. Specialisation of element model

Two element models can be related through *specialisation*. We say that element model $\mathscr{S}$ is a specialisation of element model $\mathscr{M}$ if any element structure that conforms to element model $\mathscr{S}$ also conforms to element model $\mathscr{M}$. In other words, if element model $\mathscr{S}$ is a specialisation of element model $\mathscr{M}$ and we have an element structure that conforms to model $\mathscr{S}$, we know that the structure also conforms to model $\mathscr{M}$.

The specialisation relation is transitive. In other words, if element model $\mathscr{A}$ is a specialisation of element model $\mathscr{S}$, and model $\mathscr{S}$ is a specialisation of model $\mathscr{M}$, then model $\mathscr{A}$ is also a specialisation of model $\mathscr{M}$.

In this form, specialisation is a nice property, since it allows one to look at different product individuals through the same concepts. For example, in Fig. 5, the concepts defined in the model for jib cranes apply to all models below it. The definition, however, does not provide any instructions for creating specialisations. Therefore, we define a set of transformation operations, called *specialisation operations*, that produce a specialisation for a given element model. The idea is that a "higher level" element model declares all the parts that are used in its specialisations, which represent more specific product types. In the higher level model, many parts are defined as optional whereas the specific models either forbid or allow them. This is an example of one specialisation operation, namely replacement of cardinality range. With the specialisation operations, one can construct a useful element model hierarchy that has the specialisation property.

Definition. Specialisation operations:

- *A class is changed from concrete to abstract*. This operation makes the model stricter by requiring a more detailed specification of element classes in element structures.
- *An alternative part of an element class is removed*, e.g., E6 is removed as a part of E4 in Fig. 4. This simply removes one choice, thus making the model stricter.
- *A cardinality range is replaced by a subrange*. This includes refinement of an optional part as compulsory (e.g., $[0,1] \rightarrow [1,1]$) and removal of an optional part (e.g., $[0,1] \rightarrow [0,0]$). Again, the possibilities are more limited after the operation.
- *Part refinement* means, for example, that in the most generic crane model the element class 'Crane' may have 'Hoist' as a part with cardinality $[1,2]$. In a specialised element model for jib cranes, the element class 'Crane' may have refined the part to 'Jib hoist' with cardinality $[1,1]$.

  The definition of the operation is rather complex. Assume a part definition for element class E that specifies the part must be from the set of element classes $\mathbf{D}$. This part definition is refined by moving a subrange of its cardinality to another part definition for E. The domain for the other part definition must be a set $\mathbf{D}'$ such that each member of $\mathbf{D}'$ is either a member of $\mathbf{D}$ or a subclass of a member of $\mathbf{D}$. In addition, the sum of new cardinalities must be contained by the sum of original cardinalities (for E has-part $\mathbf{D}$ and E has-part $\mathbf{D}'$).

  The operation is defined more formally in Appendix A and the idea is illustrated by the following example. The has-part between E1 and E2 with cardinality $[0,3]$ is refined into two relations: E1 has-part E2 with $[0,1]$ and E1 has-part E2-1 with $[1,2]$. The sum of original cardinalities of E1 has-part E2 and E1 has-part E2-1 was $[0,3] + [0,0] = [0,3]$ and the new cardinalities are $[0,1] + [1,2] = [1,3]$, which is contained by $[0,3]$.
- *A tree that is unconnected with the root is removed*. This clean-up operation does not actually affect the extension of the model since an element structure cannot have elements not connected with the root.

Theorem. *If an element model is modified by applying specialisation operations, the resulting model is a specialisation of the original.*

The proof is presented in Appendix A.

The specialisation operations are selected to cover the most useful and intuitive ways to specialise an element model. There are specialisations of a model that satisfy the definition of specialisation as given above but cannot be obtained by the defined specialisation operations. In Fig. 6, element models at the bottom row are examples of specialisations of their counterparts at the top. The examples are presented here merely to show that the specialisation operations are not complete. That is, not all specialisations can be derived using them.

6.2. Example of specialisation

Fig. 7 shows element model $\mathscr{M}\mathbf{2}$, which is a specialisation of element model $\mathscr{M}\mathbf{1}$ presented in Fig. 4. The specialisation of $\mathscr{M}\mathbf{1}$ to $\mathscr{M}\mathbf{2}$ is made with the following operations:

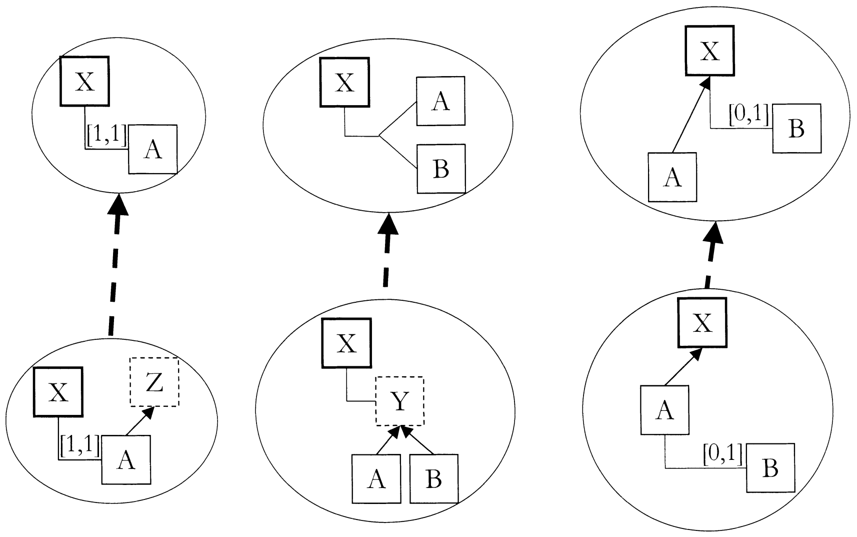


Fig. 6. Examples of specialisations that cannot be derived by the specialisation operations.

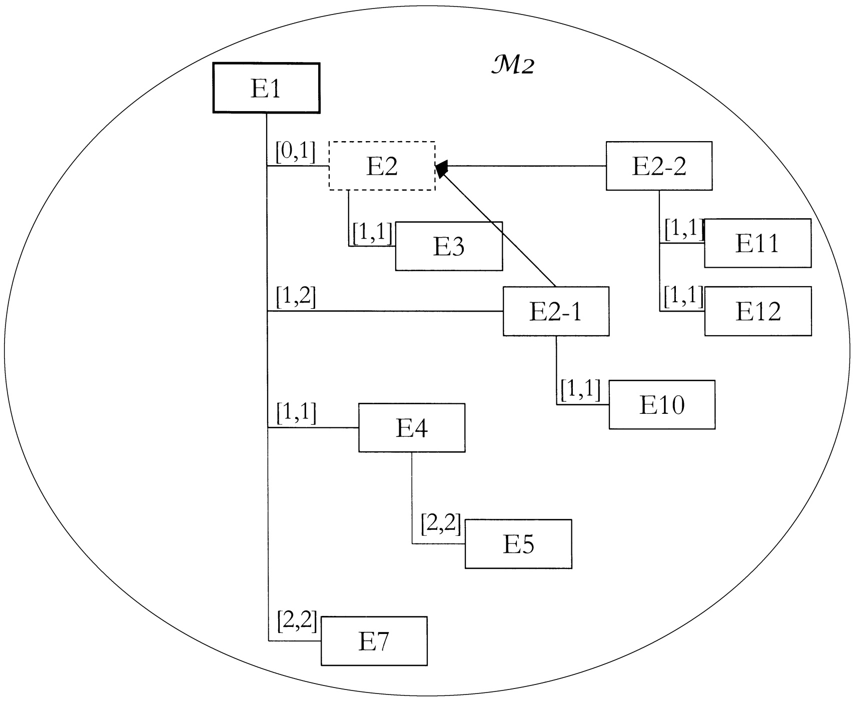


Fig. 7. Element model $\mathscr{M}\mathbf{2}$, which is a specialisation of element model $\mathscr{M}\mathbf{1}$ of Fig. 4.

- Element class E2, which was originally concrete, is changed to abstract.
- The optional has-part between E2 and E3 is changed to compulsory, i.e., its cardinality is changed from $[0, 1]$ to $[1, 1]$.

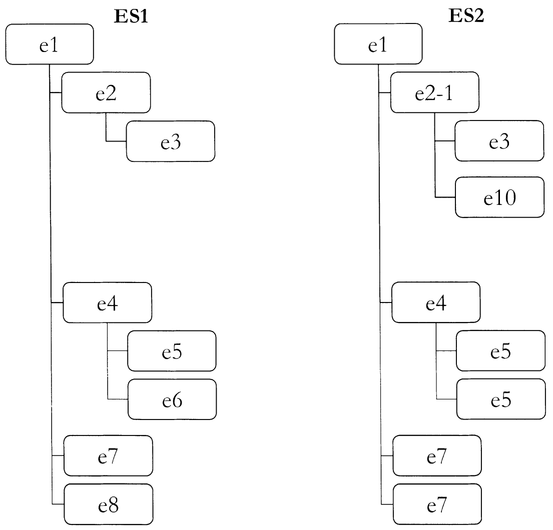


Fig. 8. Two element structures.

- The has-part between E1 and E2 with cardinality $[0, 3]$ is refined as E1 has-part E2 with $[0, 1]$ and E1 has-part E2-1 with $[1, 2]$.
- E6 is removed from the alternative parts of E4.
- The optional part E8 of E1 is removed, i.e., the cardinality is changed from $[0, 1]$ to $[0, 0]$.
- The unconnected tree of E68 (i.e., E68, E69, E6 and E8) is removed.
- The cardinality of E7 as a part of E1 is limited to two, i.e., to $[2, 2]$.

Fig. 8 shows two element structures, ES1 and ES2. The labels in the boxes refer to the element classes of the elements. In practice, the elements would be given unique identifications and the relation to the element classes is recorded separately. ES1 and ES2 both conform to the element model $\mathscr{M}\mathbf{1}$ of Fig. 4, but only ES2 conforms to the element model $\mathscr{M}\mathbf{2}$, which is a specialisation of $\mathscr{M}\mathbf{1}$. Since E2 is not abstract in $\mathscr{M}\mathbf{1}$, it can be used as a definition for valid elements as has been done in ES1. Element model $\mathscr{M}\mathbf{2}$ on the other hand has specified E2 as abstract, which necessitates the selection of the appropriate subclass of it as the element class for the corresponding element, e.g., E2-1.

7. Discussion and conclusions

We have addressed the after-sales support of products with a large number of variants. The large variety means, for example, that the aggregation of meaningful data from the product individuals is difficult with traditional BOM information. Therefore, we proposed a data modelling methodology for the after-sales support of such products. The proposed approach provides a more abstract view to product individuals than an order BOM. It captures the information of a product individual by means of elements that corresponds to the roles of components relevant to after-sales. The systematic collection of information is ensured by conformance to element models. A specialisation hierarchy of element models provides different levels of abstraction in collecting the information so that the more special cases still conform to general models. This means that all product individuals can be investigated according to the most general model. The approach was developed in a co-operation with a crane manufacturer. They defined a number of element classes, a general crane model and models for certain specific cranes in a prototype system. The prototype proved the concept but additional work is still needed before the approach can be taken in real use.

Each product individual is represented by an element structure. A set of potential element structures is modelled by an element model. Element models are organised into a specialisation hierarchy called element model hierarchy. An element model higher in the hierarchy describes a larger number of element structures, although at a higher level of abstraction; that is, with less detailed information. Each product individual that is maintained by the company is described by an element structure that conforms to the element model that most appropriately corresponds to the product individual. In addition to an element structure, a product individual also has a BOM description that tells the components of the individual. The components in the BOM are associated with the elements in the element structure. As the element structures are less detailed than BOMs, products with different BOMs may have identical element structures. The idea is that a wide range of different product individuals can be represented by element structures of one element model, which then provides a uniform representation for the product individuals in a manner relevant for after-sales.

The presented modelling methodology is a simplification of the concepts needed in configuration models [13,15] with the important distinction that parts (i.e., relations between components or elements) are not named in this approach. In this respect, our element models closely resemble to the conceptual hierarchies of PLAKON configurator [1]. The main difference to PLAKON and other configuration models is that the goal of this paper was to construct a specialisation hierarchy of models to support after-sales rather than to construct a mechanism for searching a product individual as a solution to a configuration task. There are also many other approaches to configuration modelling [2,4], but most of them do not give product structures as important role as emphasised in this paper. Some approaches based on description logics, however, have quite similar concepts as presented in this paper [12,18]. One distinction lies in the treatment of the part relation. With subsumption, description logics provide a specialisation mechanism, which is based on role names corresponding to part names. However, that does not directly allow the specialisation of a single part relation to multiple ones, e.g., by splitting a set of alternative parts into two, which is included in our approach. This view to part refinement may be applicable to other modelling in which the relations has-part and is-a are combined.

Not having part names has its own problems as it is impossible to distinguish between parts that are identical but need to be separately identified. The end-carriages with identical brakes that wear out differently were already mentioned as an example. This can be circumvented by defining separate element classes for the right and the left end-carriage as subclasses of end-carriage. If the subclasses have exactly the same properties in their different roles, it would be conceptually more appropriate to make the distinction with part names than by defining new subclasses. The combination of specialisation operations of this paper and part names increases the conceptual complexity of the methodology and is left here as future work.

The main difference to the basic description logics and previous product configuration modelling is the introduction of the specialisation for models. In principle, the specialisation of a model can be seen as a specialisation of its root element class. There are, however, some differences, which are best explained by an example in Fig. 9. On the left-hand side the model $\mathscr{M}$ is specialised in $\mathscr{M}'$ so that all the optional parts become compulsory. If the same is modelled by deriving a subclass for A, B also needs a "dummy" subclass B', which specifies C as a compulsory part. Our approach also more explicitly distinguishes between $\mathscr{M}$ and its specialisation $\mathscr{M}'$. The modelling methods proposed in this paper and their formalisation in Appendix A were constructed from the after-sales viewpoint but they are applicable to representing structural variety of products, especially with specialisations of models.

The hierarchy of element models in this paper is constructed so that the root model has all the element classes that can occur in the models. The root model in a way introduces all the element classes, which has its advantages. It defines the basic terminology for the element structures, that is, the names of the element classes as well as their position in the element structure. It would be possible to allow the addition of new subclasses in more specific models. Then, however, an element class could be in different positions in different element models, for example, as a subclass of totally different element classes. This can be avoided by requiring that all element classes are introduced in the root model. This has also disadvantages, as adding a new element class in any model requires modification of the root model. These changes, however, are not needed each time a product is changed since the element models provide an abstract view to the product more on an architectural level, which does not change that often. That is, adding a new type of a hoist into some product line does not affect element models unless it requires architectural changes to the cranes that use the particular type of hoist.

Another disadvantage in having all element models in the root model is that the root model may become hard to understand and manage. This can be helped to some extent by hiding

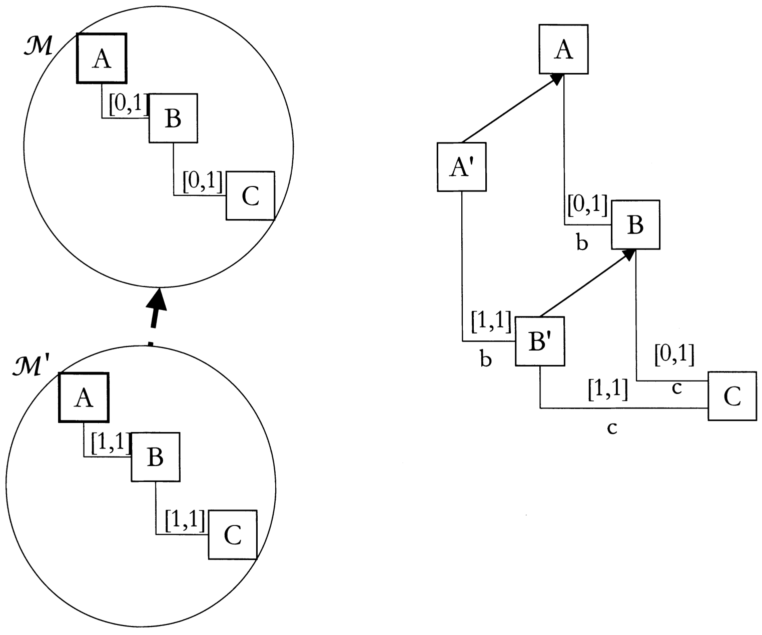


Fig. 9. Example of specialisation between and within models.

unnecessary classes in tools for manipulating the element models. For example, subclasses of a concrete class can be hidden since valid element structures can be defined without them.

There is always the dilemma between the full and free description of individuals and the ability to aggregate statistical information about them. On one hand, the more freely the individuals are described the more difficult it is to aggregate common information about them. On the other hand, the more strictly their structure is dictated the easier it is to analyse them statistically but the more difficult it is to describe all their nuances. The methodology described in this paper aims specifically to support statistical analysis of varying product individuals and has therefore a rather strict nature.

For new designs, the constructed model can be populated with relatively accurate data but old product individuals or ones manufactured by another company may have less accurate information. These differences are supported by different models that describe the product individuals at different levels of abstraction. The models are organised into an element model hierarchy, which enables aggregation of data from product individuals even though they are all different, and moreover, described with different accuracy.

The methodology provides means for supporting the evolution of product individuals. The evolution of models themselves is more complicated. New models can be added to the hierarchy quite freely as long as the specialisation property of the model remains. Therefore, if the more abstract models remain rather stable when the product is further developed, the evolution of the models can be supported, at least partly. Radical changes to the basic architecture of the product cannot be supported by such means. It is, however, a totally different question whether and how the common aspects of product individuals of entirely different technological generations should be recorded. In practice, the most feasible solution may be to create a separate model hierarchy for a product generation that significantly differs from the old products.

Acknowledgements

We gratefully acknowledge the financial support from Technology Development Center of Finland (TEKES) and Helsinki Graduate School of Computer Science and Engineering (HeCSE). We also thank Asko Martio for his valuable comments on the paper.

Appendix A

Definition 1 (*Element model*). An element model $\mathscr{M} = \langle \mathbf{S}_E, \mathbf{S}_\mathbf{C}, \boldsymbol{P}_\mathbf{D}, \boldsymbol{T} \rangle$, where $\mathbf{S}_\mathbf{E}$ is a non-empty finite set of element classes, $\mathbf{S}_\mathbf{C}$ is a non-empty set of concrete element classes, i.e., $\mathbf{S}_\mathbf{C} \subseteq \mathbf{S}_\mathbf{E}$. $\boldsymbol{P}_\mathbf{D}$ is a set of part definitions and $\boldsymbol{T}$ is the 'is-a' relation between element classes.

Definition 2 (*Part definition*). A part definition in $\boldsymbol{P}_\mathbf{D}$ for an element class $E \in \mathbf{S}_\mathbf{E}$ with domain $\mathbf{D} \subseteq \mathbf{S}_\mathbf{E}$ of $\mathscr{M}$ is given as $\langle \mathscr{M}, E, \mathbf{D}, r_1, r_2 \rangle$, where $\mathbf{D}$ is a non-empty set of element classes (i.e., the alternative parts or just one class if there are no alternatives). The cardinality is given by r_1 and r_2 s.t. $0 \leqslant r_1 \leqslant r_2$ and $r_2 > 0$ and denoted as $[r_1, r_2]$. (For clarity, the model is repeated in part definitions, although it is redundant.)

Definition 3 (*Is-a relation*). $\boldsymbol{T} \subseteq \mathbf{S_E} \times \mathbf{S_E}$ is a relation such that the transitive closure of $\boldsymbol{T}$ is irreflexive and $(E,X) \in \boldsymbol{T} \wedge (E,Y) \in \boldsymbol{T} \Rightarrow X = Y$. That is, with $\mathbf{S_E}, \boldsymbol{T}$ forms trees. In addition, each abstract class must have at least one subclass, i.e., $\forall E \in (\mathbf{S_E} - \mathbf{S_C})\ \exists E' \in \mathbf{S_E}$ such that $(E',E) \in \boldsymbol{T}$.

Definition 4 (*Subclasses*). $E' \leqslant_{\mathscr{M}} E$ denotes that E' is E or a subclass of E in $\mathscr{M}$. In other words, relation $\leqslant_{\mathscr{M}}$ is the reflexive transitive closure of $\boldsymbol{T}$. In addition, we define, $\mathbf{C_s}(\mathscr{M},\mathbf{S}) = \{E | E \leqslant_{\mathscr{M}} E_i \text{ and } 1 \leqslant i \leqslant k\}$, where $\mathbf{S} = \{E_1, E_2, \ldots, E_k\}$. That is, $\mathbf{C_s}(\mathscr{M},\mathbf{S})$ denotes the union of $\mathbf{S}$ and the subclasses of its members.

Definition 5 (*Concrete subclasses*). $\mathbf{C_C}(\mathscr{M},\mathbf{S})$ denotes the subset of concrete classes in $\mathbf{C_S}(\mathscr{M},\mathbf{S})$.

Definition 6 (*Root element class*). One of the element classes denoted as $E_R \in \mathbf{S_E}$ of $\mathscr{M}$ is designated as the *root element class*. It is required that E_R has no superclasses, i.e., there exists no element class E such that $(E_R, E) \in \boldsymbol{T}$, and is not defined as a part, i.e., for no part definition $\langle \mathscr{M}, E, \mathbf{D}, r_1, r_2 \rangle, E_R \in \mathbf{C_s}(\mathscr{M}, \mathbf{D})$.

Note that the root element class represents the whole product, i.e., "the root of the part hierarchy"; it is also a root but not the only root of the taxonomy induced by is-a.

Definition 7. Basic arithmetic operation with cardinalities:
$[a,b] + [c,d] = [a+c, b+d]$,
$x \in [a,b]$ iff $a \leqslant x \leqslant b$ and
$[a,b] = [c,d]$ iff $c = a$ and $b = d$
$[a,b] \subset [c,d]$ iff $c \leqslant a$ and $b \leqslant d$ and $(c \neq a$ or $b \neq d)$
$[a,b] \subseteq [c,d]$ iff $c \leqslant a$ and $b \leqslant d$.

Definition 8 (*Inheritance of part definitions*). Subclasses of an element class inherit its part definitions. If element class E defines a part definition or inherits them with the domain $\mathbf{D}$, the effective part definition, denoted as $\boldsymbol{P}_{\mathbf{DE}} = \langle \mathscr{M}, E, \mathbf{D}, r_1, r_2 \rangle_E$, where $[r_1, r_2]$ is the sum of the cardinalities of all inherited part definitions and (possibly) the part definition defined in E.

As there are no part names, there is no part refinement within an element model; there is, however, part refinement between element models (defined below).

Definition 9 (*Is-individual-of*). $\mathrm{I}(e)$ denotes the element class element e is individual of.

Definition 10 (*Element structure*). An element structure is $\mathbf{E_S} = \langle r, \mathbf{S_e}, \boldsymbol{P}_\mathbf{e} \rangle$, where r is an element called *root element*, $\mathbf{S_e}$ is a set of elements, containing r, connected with *immediate parts relation* $\boldsymbol{P}_\mathbf{e} \subseteq \mathbf{S_e} \times \mathbf{S_e}$, so that there is no $e \in \mathbf{S_e}$, such that $(e,r) \in \boldsymbol{P}_\mathbf{e}$. (I.e., r is not a part of any element class in $\mathbf{S_e}$.)

Definition 11 (*Immediate parts of element*). $\mathbf{P}(e)$ denotes the set of elements that are the immediate parts of element e, i.e., $\{e' | (e,e') \in \boldsymbol{P}_\mathbf{e}\}$.

Definition 12 (*Validity of element structure*). An element structure $\mathbf{E}_S = \langle r, \mathbf{S_e}, \boldsymbol{P}_\mathbf{e} \rangle$ is valid if $\forall e \in \mathbf{S_e} : (r, e) \in \boldsymbol{P}^*_\mathbf{e}$ (where $\boldsymbol{P}^*_\mathbf{e}$, is the transitive closure of $\boldsymbol{P}_\mathbf{e}$), $\boldsymbol{P}^*_\mathbf{e}$ is irreflexive and $(x, e) \in \boldsymbol{P}_\mathbf{e} \wedge (y, e) \in \boldsymbol{P}_\mathbf{e} \Rightarrow x = y$. That is, $\boldsymbol{P}_\mathbf{e}$ with $\mathbf{S_e}$ induces a tree with r as the root.

Definition 13 (*Effective cardinality*). $\mathrm{Card}_{eff}(\mathscr{M}, E, \mathbf{D}, \mathbf{S})$ of part definition $\langle \mathscr{M}, E, \mathbf{D}, r_1, r_2 \rangle_E$ for a set of element classes $\mathbf{S}$ is:

$$\begin{cases} [r_1, r_2] & \text{if } \mathbf{C_C}(\mathscr{M}, \mathbf{D}) \subseteq \mathbf{C_C}(\mathscr{M}, \mathbf{S}), \\ [0, r_2] & \text{if } \mathbf{C_C}(\mathscr{M}, \mathbf{D}) \not\subseteq \mathbf{C_C}(\mathscr{M}, \mathbf{S}) \text{ and } \mathbf{C_C}(\mathscr{M}, \mathbf{D}) \cap \mathbf{C_C}(\mathscr{M}, \mathbf{S}) \neq \emptyset, \\ [0, 0] & \text{otherwise.} \end{cases}$$

Effective cardinality indicates how many elements as individuals of element classes in S or their subclasses the part definition for E with domain $\mathbf{D}$, i.e., $\langle \mathscr{M}, E, \mathbf{D}, r_1, r_2 \rangle_E$, requires and allows for an element structure.

Definition 14 (*Combined effective cardinality*). $\mathrm{Card}^*(\mathscr{M}, E, \mathbf{S})$ is the sum of effective cardinalities $\mathrm{Card}_{\mathrm{eff}}(\mathscr{M}, E, \mathbf{D_i}, \mathbf{S})$ over all domains $\mathbf{D_i}$ for which E defines or inherits a part definition in $\mathscr{M}$.

Card^* combines $\mathrm{Card}_{\mathrm{eff}}$'s for all part definitions for E. It provides, for example, the minimum and maximum values for the number of individuals of the element class E' (in case S contains only E') and its subclasses.

Definition 15 (*Conformance of an element*). An element e such that $\mathrm{I}(e) = E$ conforms to model $\mathscr{M} = \langle \mathbf{S_E}, \mathbf{S_C}, \boldsymbol{P}_\mathbf{D}, \boldsymbol{T} \rangle$, denoted as $e \models \mathscr{M}$, if
(i) $E \in \mathbf{S_C}$,
(ii) $\forall e_i \in \mathbf{P}(e) \exists \boldsymbol{P}_\mathbf{D} = \langle \mathscr{M}, E, \mathbf{D}, r_1, r_2 \rangle_E$ such that $\mathrm{I}(e_i) \in \mathbf{C_C}(\mathscr{M}, \mathbf{D})$,
(iii) $\forall \boldsymbol{P}_\mathbf{DE} = \langle \mathscr{M}, E, \mathbf{D}_i, r_1, r_2 \rangle_E$ holds that $| \{ e_j \mid e_j \in \mathbf{P}(e) \text{ and } I(e_j) \leqslant_{\mathscr{M}} E_i, \text{ where } E_i \in \mathbf{D}_i \} | \in \mathrm{Card}^*(\mathscr{M}, E, \mathbf{D_i})$.
That is, the element class $E = I(e)$ is concrete in $\mathscr{M}$ and all the immediate parts of e are individuals of concrete element classes (and subclasses of $\mathbf{D}$). In addition, for each part definition of E (including inherited ones), e has the correct number of parts.

Definition 16 (*Conformance of an element structure*). An element structure $\mathbf{E_S} = \langle r, \mathbf{S_e}, \boldsymbol{P}_\mathbf{e} \rangle$ conforms to model $\mathscr{M} = \langle \mathbf{S_E}, \mathbf{S_C}, \boldsymbol{P}_\mathbf{D}, \boldsymbol{T} \rangle$, denoted as $\mathbf{E_S} \models \mathscr{M}$, if $\mathbf{E_S}$ is valid and $\mathrm{I}(r) \in \mathbf{C_C}(\mathscr{M}, \{E_R\})$, where E_R is the root element class of $\mathscr{M}$, and $\{E \mid E = I(e) \text{ and } e \in \mathbf{S_e}\} \subseteq \mathbf{S_E}$ and each element of $\mathbf{E_S}$ conforms to $\mathscr{M}$, i.e., $\forall e \in \mathbf{S_e} \Rightarrow e \models \mathscr{M}$.

Definition 17 (*Specialisation*). A model $\mathscr{M}'$ is specialisation of model $\mathscr{M}$ if any element structure that conforms to $\mathscr{M}'$ also conforms to $\mathscr{M}$. That is, for any $\mathbf{E_S}$ it holds that $\mathbf{E_S} \models \mathscr{M}' \Rightarrow \mathbf{E_S} \models \mathscr{M}$.

Definition 18 (*Specialisation operations*). *Change class from concrete to abstract.* A concrete class that has subclasses is simply tagged as abstract.

Part refinement. A part refinement for a part definition $\langle \mathscr{M}, E, \mathbf{D_1}, r_1, r_2 \rangle$ specifies a domain $\mathbf{D}_2$ and a range $[r_1, r_2]$ such that $\mathbf{D}_2 \subseteq \mathbf{C_S}(\mathscr{M}, \mathbf{D_1})$, i.e., $\mathbf{D_2}$ contains only classes of $\mathbf{D}_1$ or their subclasses, and $[r_1, r_2] \subseteq [r_{11}, r_{12}]$. If E originally had part definition with domain $\mathbf{D_2} \neq \mathbf{D_1}$ assume its cardinality is given as $[r_{21}, r_{22}]$, i.e., the part definition was $\langle \mathscr{M}, E, \mathbf{D_2}, r_{21}, r_{22}\rangle$.Otherwise, i.e., $\mathbf{D_1} = \mathbf{D_2}$ or E has no part definition with domain $\mathbf{D_2}, r_{21} = r_{22} = 0$ After part refinement, the specialised model has:

$$\begin{cases} \langle \mathscr{M}, E, \mathbf{D_1}, r_{11}, r_{12}\rangle (\text{and no } \langle \mathscr{M}, E, \mathbf{D_2}, q_{21}, q_{22}\rangle) & \text{if } [r_1, r_2] = [r_{11}, r_{12}] \text{ and } \mathbf{D_1} = \mathbf{D_2}, \text{ or} \\ \langle \mathscr{M}, E, \mathbf{D_1}, q_{11}, q_{12}\rangle (\text{and no } \langle \mathscr{M}, E, \mathbf{D_2}, q_{21}, q_{22}\rangle) & \text{if } [r_1, r_2] \subset [r_{11}, r_{12}] \text{ and } \mathbf{D_1} = \mathbf{D_2}, \text{ or} \\ \langle \mathscr{M}, E, \mathbf{D_2}, q_{21}, q_{22}\rangle (\text{and no } \langle \mathscr{M}, E, \mathbf{D_1}, q_{11}, q_{12}\rangle) & \text{if } [r_1, r_2] = [r_{11}, r_{12}] \text{ and } \mathbf{D_1} \neq \mathbf{D_2}, \text{ or} \\ \langle \mathscr{M}, E, \mathbf{D_1}, q_{11}, q_{12}\rangle \text{ and } \langle \mathscr{M}, E, \mathbf{D_2}, q_{21}, q_{22}\rangle & \text{otherwise,} \end{cases}$$

where $[q_{11}, q_{12}] = [r_{11} - r_1, r_{12} - r_2]$ and $[q_{21}, q_{22}] = [r_1 + r_{21}, r_2 + r_{22}]$.

In other words, a subrange of the refined part definition is moved to a more specific part definition. Part refinement is a powerful operation. It has special cases of which some were introduced separately in the text because of their intuitive nature. For example,

- restriction of cardinality, i.e., $\mathbf{D_1} = \mathbf{D_2}$ and $[r_1, r_2] \subset [r_{11}, r_{12}]$,
- refinement of a part definition with a subclass of the original domain, i.e., $[r_1, r_2] = [r_{11}, r_{12}]$ and $\mathbf{D_2} \subset \mathbf{C_S}(\mathscr{M}, \mathbf{D_1})$,
- removal of an alternative part, e.g., $\mathbf{D_2} \subset \mathbf{D_1}$ and $[r_1, r_2] = [r_{11}, r_{12}]$,
- conversion of subclasses into an alternative part definition, e.g., $\mathbf{D_1} = \{A\}$, A has subclasses A_1 and A_2, $\mathbf{D_2} = \{A_1, A_2\}$ and $[r_1, r_2] = [r_{11}, r_{12}]$,
- splitting a set of alternative parts into two part definitions (with two operations).

Note that part refinement is allowed only for part definitions defined in E, not for inherited ones.

Removal of unconnected tree. Assume two sets of unconnected element classes $\mathbf{S_1}$ and $\mathbf{S_2}$. That is, for any pair $(E_1, E_2) \in \mathbf{S_1} \times \mathbf{S_2}$ it is not true in $\mathscr{M}$ that E_1 is subclass of E_2 or E_1 has E_2 as part (directly or via inheritance) or vice versa. Now, the one that does not contain the root element class is removed.

Theorem. *If element model $\mathscr{M}'$ is modified from element model $\mathscr{M}$ by specialisation operations, $\mathscr{M}'$ is a specialisation of $\mathscr{M}$.*

Proof. Before a specialisation operation $\mathscr{M} = \mathscr{M}'$, i.e., $\mathbf{S_E} = \mathbf{S'_E}, \mathbf{S_C} = \mathbf{S'_C}, \boldsymbol{P}_\mathbf{D} = \boldsymbol{P}_{\mathbf{D}'}$ and $\boldsymbol{T} = \boldsymbol{T}'$. Next, the result is shown for each specialisation operation.

(1) *Concrete class to abstract.* Let $\mathscr{M}'$ be modified from $\mathscr{M}$ by changing the element class E_1 from concrete to abstract, i.e., $\mathbf{S'_C} = \mathbf{S_C} - \{E_1\}$. Assume an e such that $\mathrm{I}(e) = E$ and $e \models \mathscr{M}' \Rightarrow e \not\models \mathscr{M}$ and any e' such that $e' \in \mathbf{P}(e)$:

1. there is $e' \in \mathbf{P}(e)$ such that $\mathrm{I}(e') = E_1$. Now, $e \not\models \mathscr{M}'$ because E_1 is abstract in $\mathscr{M}'$.
2. there is no element of E_1 in $\mathbf{P}(e)$. Thus, the part definitions for all element classes of elements of $\mathbf{P}(e)$ are equal in $\mathscr{M}'$ and $\mathscr{M}$, and therefore, $e \models \mathscr{M}' \Rightarrow e \models \mathscr{M}$.

No such e exists for which $e \models \mathscr{M}' \Rightarrow e \not\models \mathscr{M}$.

Assume element structure $\mathbf{E_S} = \langle r, \mathbf{S_e}, \boldsymbol{P}_\mathbf{e}\rangle \models \mathscr{M}'$. That is, $\mathbf{E_S}$ is valid, $\mathrm{I}(r) \in \mathbf{C_C}(\mathscr{M}', \{E_R\})$, $\{E \mid E = I(e'') \text{ and } e'' \in \mathbf{S_e}\} \subseteq \mathbf{S'_E}$ and $\forall e'' \in \mathbf{S_e} \Rightarrow e'' \models \mathscr{M}'$.

As $\mathbf{S_E} = \mathbf{S'_E}, \mathbf{S'_C} \subseteq \mathbf{S_C}$ and $\boldsymbol{T} = \boldsymbol{T}'$, it follows that $\mathrm{I}(r) \in \mathbf{C_C}(\mathscr{M}, \{E_R\}), \{E | E = \mathrm{I}(e'')$ and $e'' \in \mathbf{S_e}\} \subseteq \mathbf{S_E}$ and because $e \models \mathscr{M}' \Rightarrow e \models \mathscr{M}$ for all $e \in \mathbf{S_e}$, it holds that $\mathbf{E_S} \models \mathscr{M}' \Rightarrow \mathbf{E_S} \models \mathscr{M}$.

(2) *Part refinement.* Assume the part definition $\langle \mathscr{M}, E, \mathbf{D_1}, r_{11}, r_{12}\rangle$, domain $\mathbf{D_2} \subseteq \mathbf{C_S}(\mathscr{M}, \mathbf{D_1})$ and range $[r_1, r_2] \subseteq [r_{11}, r_{12}], r_2 > 0$ and the part definition $\langle \mathscr{M}, E, \mathbf{D_2}, r_{21}, r_{22}\rangle$ for which after the part

refinement there will (possibly) be $\langle \mathscr{M}', E, \mathbf{D_1}, q_{11}, q_{12} \rangle$ and $\langle \mathscr{M}'E, \mathbf{D_2}, q_{21}, q_{22} \rangle$ as defined in Definition 18.

Assume element e such that $\mathrm{I}(e) = E$ and $e \models \mathscr{M}' \Rightarrow e \not\models \mathscr{M}$. For such e to exists one of the requirements (i)–(iii) of Definition 15 must hold in $\mathscr{M}'$ but not in $\mathscr{M}$. These are proven in Cases A–C, respectively. (Note the specialisation relation is the same in $\mathscr{M}'$ and $\mathscr{M}$, i.e., $\boldsymbol{T} = \boldsymbol{T}'$.)

Case A. For (i) not to hold, $E \in \mathbf{S}'_\mathbf{C}$ but $E \notin \mathbf{S_C}$, which is a contradiction since $\mathbf{S}'_\mathbf{C} = \mathbf{S_C}$.

Case B. Assume that e has part $e' \in \mathbf{P}(e)$ and there is $\langle \mathscr{M}', E, \mathbf{D_k}, k_1, k_2 \rangle_E$ such that $E' = \mathrm{I}(e') \in \mathbf{C_C}(\mathscr{M}', \mathbf{D_k})$, but in $\mathscr{M}$ there is no part definition $\langle \mathscr{M}, E, \mathbf{D_m}, m_1, m_2 \rangle_E$ such that $E' \in \mathbf{C_C}(\mathscr{M}, \mathbf{D_m})$.

If $\mathbf{D_1} = \mathbf{D_2}$ then for E a part definition with $\mathbf{D_k}$ also exists in $\mathscr{M}$ as in part refinement with $\mathbf{D_1} = \mathbf{D_2}$ no part definition domain is changed (nor part definitions added).

Otherwise,

1. $\mathbf{D_k} \neq \mathbf{D_1}$ and $\mathbf{D_k} \neq \mathbf{D_2}$ then the part definition with $\mathbf{D_k}$ for E also exists in $\mathscr{M}$.
2. $\mathbf{D_k} = \mathbf{D_1}$, as was initially assumed, a part definition with $\mathbf{D_1}$ for E exists in $\mathscr{M}$.
3. $\mathbf{D_k} = \mathbf{D_2}$, now $\mathbf{D_2} \subseteq \mathbf{C_S}(\mathscr{M}, \mathbf{D_1}) \Rightarrow \mathbf{C_C}(\mathscr{M}, \mathbf{D_2}) \subseteq \mathbf{C_C}(\mathscr{M}, \mathbf{D_1})$ and thus $I(e') \in \mathbf{C_C}(\mathscr{M}, \mathbf{D_1})$.

Thus, the assumed e' cannot exist.

Case C. Now, for some domain $\mathbf{D_k}$ such that $\langle \mathscr{M}', E, \mathbf{D_k}, r_{11}, r_{12} \rangle_E$ exists, $|\ \{e_j \mid e_j \in \mathbf{P}(e)$ and $I(e_j) \leqslant_\mathscr{M} E_i$, where $E_i \in \mathbf{D_k}\}\ |$ is within $\mathrm{Card}^*(\mathscr{M}', E, \mathbf{D_k})$ but not $\mathrm{Card}^*(\mathscr{M}, E, \mathbf{D_k})$, i.e., $\mathrm{Card}^*(\mathscr{M}', E, \mathbf{D_k}) \not\subset \mathrm{Card}^*(\mathscr{M}, E, \mathbf{D_k})$.

If $\mathbf{D}_1 = \mathbf{D}_2$, exactly one cardinality is changed to a subrange, thus, $\mathrm{Card}^*(\mathscr{M}', E, \mathbf{D_k})$ must be contained in $\mathrm{Card}^*(\mathscr{M}, E, \mathbf{D_k})$.

For $\mathbf{D_1} \neq \mathbf{D_2}$, There are 16 cases how $\mathbf{C_C}(\mathscr{M}, \mathbf{D}_1)$ and $\mathbf{C_C}(\mathscr{M}, \mathbf{D_2})$ relate to $\mathbf{C_C}(\mathscr{M}, \mathbf{D_k})$.

In Table 1 the symbol in the column "$\mathbf{D_1}$" denotes the relation between $\mathbf{C_C}(\mathscr{M}, \mathbf{D}_1)$ and $\mathbf{C_C}(\mathscr{M}, \mathbf{D_k})$ so that $\cap$ means the sets intersect but neither contains the other and $\not\cap$ means the sets do not intersect, whereas $\supseteq$ and $\subseteq$ denote the set containment. Column "$\mathbf{D_2}$" has the same meaning for $\mathbf{D_2}$. Relation $\subseteq$ corresponds to the first case of Definition 13, $\cap$ and $\supseteq$ to the second case and $\not\cap$ to the last case.

Table 1

#	"$\mathbf{D_1}$"	"$\mathbf{D_2}$"	"$\mathrm{Card}^*(\mathscr{M}, E, \mathbf{D_k})$"	"$\mathrm{Card}^*(\mathscr{M}', E, \mathbf{D_k})$"
1	$\subseteq$	$\subseteq$	$[r_{11}, r_{12}] + [r_{21}, r_{22}]$	$[q_{11}, q_{12}] + [q_{21}, q_{22}] = [r_{11} + r_{21}, r_{12} + r_{22}]$
2	$\subseteq$	$\cap$	Not applicable since $\mathbf{C_C}(\mathscr{M}, \mathbf{D_1}) \subseteq \mathbf{C_C}(\mathscr{M}, \mathbf{D_k}) \Rightarrow \mathbf{C_C}(\mathscr{M}, \mathbf{D_2}) \subseteq \mathbf{C_C}(\mathscr{M}, \mathbf{D_k})$	
3	$\subseteq$	$\supseteq$	As 2	
4	$\subseteq$	$\not\cap$	As 2	
5	$\cap$	$\subseteq$	$[0, r_{12}] + [r_{21}, r_{22}]$	$[0, q_{12}] + [q_{21}, q_{22}] = [r_1 + r_{21}, r_{12} + r_{22}]$
6	$\cap$	$\cap$	$[0, r_{12}] + [0, r_{22}]$	$[0, q_{12}] + [0, q_{22}] = [0, r_{12} - r_2 + r_{22} + r_2] = [0, r_{12} + r_{22}]$
7	$\cap$	$\supseteq$	Not applicable since $\mathbf{C_C}(\mathscr{M}, \mathbf{D_k}) \subseteq \mathbf{C_C}(\mathscr{M}, \mathbf{D_2}) \Rightarrow \mathbf{C_C}(\mathscr{M}, \mathbf{D_k}) \subseteq \mathbf{C_C}(\mathscr{M}, \mathbf{D_1})$	
8	$\cap$	$\not\cap$	$[0, r_{12}] + [0, 0]$	$[0, q_{12}] + [0, 0] = [0, r_{12} - r_2]$
9	$\supseteq$	$\subseteq$	As 5	As 5
10	$\supseteq$	$\cap$	As 6	As 6
11	$\supseteq$	$\supseteq$	As 6	As 6
12	$\supseteq$	$\not\cap$	As 8	As 8
13	$\not\cap$	$\subseteq$	Not applicable since $\mathbf{C_C}(\mathscr{M}, \mathbf{D_2}) \cap \mathbf{C_C}(\mathscr{M}, \mathbf{D_k}) \neq \emptyset \Rightarrow \mathbf{C_C}(\mathscr{M}, \mathbf{D_1}) \cap \mathbf{C_C}(\mathscr{M}, \mathbf{D_k}) \neq \emptyset$	
14	$\not\cap$	$\cap$	As 13	
15	$\not\cap$	$\supseteq$	As 13	
16	$\not\cap$	$\not\cap$	$[0, 0] + [0, 0]$	$[0, 0] + [0, 0]$

$\text{Card}^*(\mathscr{M}, E, \mathbf{D_k}) = \text{Card}_{\text{eff}}(\mathscr{M}, E, \mathbf{D_1}, \mathbf{D_k}) + \text{Card}_{\text{eff}}(\mathscr{M}, E, \mathbf{D_2}, \mathbf{D_k}) + \cdots$ and
$\text{Card}^*(\mathscr{M}', E, \mathbf{D_k}) = \text{Card}_{\text{eff}}(\mathscr{M}', E, \mathbf{D_1}, \mathbf{D}_k) + \text{Card}_{\text{eff}}(\mathscr{M}', E, \mathbf{D_2}, \mathbf{D_k}) + \cdots$
The rest of the terms in the sums are equal between $\mathscr{M}'$ and $\mathscr{M}$. Table 1 shows the (possibly) changed terms for $\text{Card}^*(\mathscr{M}, E, \mathbf{D_k})$ and $\text{Card}^*(\mathscr{M}', E, \mathbf{D_k})$. That is, the terms resulting from the part definitions for $\mathbf{D_1}$ and $\mathbf{D_2}$ defined in E, as the inherited ones have not changed.

In all cases $\text{Card}^*(\mathscr{M}', E, \mathbf{D_k}) \subseteq \text{Card}^*(\mathscr{M}, E, \mathbf{D_k})$ thus (iii) holds.

Therefore, after a part refinement: $e \models \mathscr{M}' \Rightarrow e \models \mathscr{M}$. As $\mathbf{S_E} = \mathbf{S'_E}, \mathbf{S_C} = \mathbf{S'_C}$ and $\boldsymbol{T} = \boldsymbol{T}'$, the conclusion that $\mathbf{E_S} \models \mathscr{M}' \Rightarrow \mathbf{E_S} \models \mathscr{M}$ is similar as in (1).

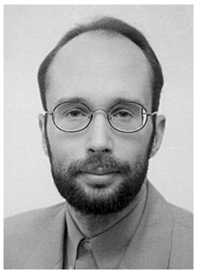
(3) *Removal of an unconnected tree*. No element class in the tree unconnected with the root element class belongs to the domain $\mathbf{D}$ of any part definition in an element class in the tree connected with root element class. Therefore, the unconnected tree does not affect the conformance of any element structure containing an element of the root class. Thus, $\mathbf{E_S} \models \mathscr{M}' \Rightarrow \mathbf{E_S} \models \mathscr{M}$.

(1)–(3) show that after any specialisation operation that modifies $\mathscr{M}$ to $\mathscr{M}'$, any element that conforms to $\mathscr{M}'$ also conforms to $\mathscr{M}$. Derivation of $\mathscr{M}'$ from $\mathscr{M}$ by series of specialisation operations therefore implies that $\mathscr{M}'$ is a specialisation of $\mathscr{M}$.

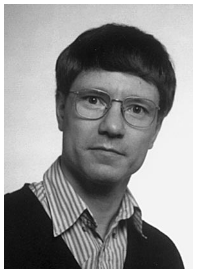
References

[1] R. Cunis, A. Günter, I. Syska, H. Peters, H. Bode, PLAKON – an approach to domain-independent construction, in: The Second International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems IEA/AIE-89, 1989, pp. 866–874.
[2] T. Darr, D. McGuinness, M. Klein (Eds.), AI EDAM (Special Issue on Configuration Design) 12(4) (1998).
[3] F. Erens, The synthesis of variety – developing product families, Ph.D. Thesis, Eindhoven University of Technology, 1996.
[4] B. Faltings, E.C. Freuder (Eds.), IEEE Intelligent Systems & Their Applications (Special Issue on Configuration) (1998) 29–85.
[5] ISO, ISO Standard 10303-41: Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 41: Integrated Generic Resources: Fundamentals of Product Description and Support, 1994.
[6] ISO, Guide on STEPlib, ISO TC184/SC4/W G3/N424, 1997.
[7] ISO, ISO Committee Draft 10303–221: Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 221: Application Protocol: Functional Data and Their Schematic Representation for Process Plant, 1997.
[8] B.M. Kramer, Knowledge-based configuration of computer systems using hierarchical partial choice, in: Proceedings of the Third International Conference on Tools for Artificial Intelligence TAI 91, IEEE, 1991, pp. 368–375.
[9] T. Männistö, H. Peltonen, A. Martio, R. Sulonen, Modelling generic product structures in STEP, Computer-Aided Design 30 (14) (1998) 1111–1118.
[10] T. Männistö, H. Peltonen, R. Sulonen, View to product configuration knowledge modelling and evolution, in: B. Faltings, E.C. Freuder (Eds.), Configuration – papers From the 1996 AAAI Fall Symposium, AAAI Tech. Rep. FS-96-03, AAAI Press, 1996, pp. 111–118.
[11] J. Owen, STEP – An Introduction, Information Geometers, 1997.
[12] B. Owsnicki-Klewe, Configuration as a consistency maintenance task, in: Hoeppner (Ed.), Proceedings of the Künstliche Intelligenz, Springer, Berlin, 1988, pp. 77–87.
[13] H. Peltonen, T. Männistö, T. Soininen, J. Tiihonen, A. Martio, R. Sulonen, Concepts for modelling configurable products, in: Proceedings of the Product Data Technology Days, Quality Marketing Services, 1998, pp. 189–196.
[14] D. Sabin, R. Weigel, Product configuration frameworks – a survey, IEEE Intelligent Systems & Their Applications 13 (4) (1998) 42–49.
[15] T. Soininen, J. Tiihonen, T. Männistö, R. Sulonen, Towards a general ontology of configuration, AI EDAM 12 (4) (1998).
[16] E.A. van Veen, Modelling product structures by generic bills-of-material, Ph.D. Thesis, Eindhoven University of Technology, 1991.
[17] F.R. Wagner, A.H.V. Lima, Design version management in the GARDEN framework, in: Proceedings of the 28th ACM/IEEE Design Automation Conference, 1991, pp. 704–710.
[18] R. Weida, Closed terminologies in description logics, in: B. Faltings, E.C. Freuder (Eds.). Configuration – papers From the 1996 AAAI Fall Symposium, AAAI Press, 1996, pp. 11–18.

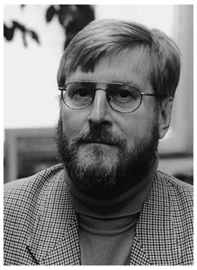
Tomi Männistö is a researcher at the Product Data Management Group of Helsinki University of Technology. His main interest lie in the conceptual modelling and management of product families. In particular, he is investigating the evolution of product family descriptions and the their individuals, on which subject he is currently finalizing his Ph.D. thesis at the Helsinki Graduate School in Computer Science and Engineering.



Timo Soininen works as a researcher at the Product Data Management Group of Helsinki University of Technology. His main research interests are conceptual and formal modeling and represenation of product knowledge and the related reasoning in product configurator systems, on which topic he is preparing his Ph.D. thesis at the Helsinki Graduate School in Computer Science and Engineering.



Hannu Peltonen has a Ph.D. degree in Computer Science from Helsinki University of Technology, where he works as a researcher in Product Data Management Group. He has investigated product data management in several joint projects with industry and is closely involved in the design and implementation of a document management system used by a large Finnish manufacturing company.



Reijo Sulonen is Professor of Computer Science at the Helsinki University of Technology and leader of the Product Data Management Group. His research interest include database system, product data management, process modelling, software engineering and electronic media.