Computer-Aided Design, Vol. 30, No. 14, pp. 1111–1118, 1998

0010-4485/99/$ – see front matter

PII: S0010-4485(98)00067-0

Modelling generic product structures in STEP

Tomi Männistö*, Hannu Peltonen, Asko Martio and Reijo Sulonen

Industrial companies need powerful data modelling mechanisms, e.g. classification, for the description of their products. The companies that adapt their products to the needs of individual customers in a routine manner have perhaps the most urgent needs. They must efficiently describe large numbers of product variants. STEP Application Protocol 214 (AP 214) for the automotive industry also addresses the modelling of product variants, i.e. generic product structure modelling. In addition to AP 214, the same mechanisms are needed in other standardization efforts as well, e.g. STEPlib of AP 221 and PLIB. STEP, however, does not include a mechanism for using classification and inheritance for modelling products of an individual company. These facilities are included in EXPRESS, but EXPRESS can only be used for describing the data schema to be standardized. The fundamental structure of STEP, therefore, prohibits a company from modelling its products in an object-oriented manner. This is an issue that may seriously affect the future of STEP as a general product-modelling methodology. The problems and possibilities of extending STEP in this direction within its current structure are discussed and a new mechanism is outlined as an alternative solution.

Keywords: product models, standards

INTRODUCTION

An increasing number of companies try to increase their competitiveness by means of configurable products, which have a large number of variants and are routinely configured according to customer orders. This paper concentrates on the modelling of the part-of structures of configurable products, which we here refer to as generic product structures. By a generic product structure we mean a data description that represents multiple different but related product structures.

For example, consider a product with a machinery that has several choices for the motor. In principle, one can create distinct product structure descriptions, each corresponding to one choice of motor. This, however, may become impossible when the number of choices increases. For example, assume further that there are three types of machinery, a few choices of clutches, four different kinds of control mechanisms, each having further a model for normal and high accuracy control, etc. In such a situation, it is no longer feasible to list all the different product structures one by one. Therefore, one needs a mechanism for representing the valid combinations more efficiently. Such a mechanism is called a generic product structure. Other terms, synonymous with generic structure, include configuration model and generic Bill-of-Materials[1–3].

One of our aims is to understand what the STEP standard provides for modelling generic products structures. More generally, we investigate how the products of a company can be modelled within STEP using object-oriented methods, e.g. classification and inheritance. Such advanced methods are important for many kinds of complex industrial products, of which configurable products are an example.

Step standard

STEP is a large undertaking for the standardization of product data modelling. The international standard ISO-10303 was accepted in 1994. This is not the final word on STEP; the standard is continuously being extended, and there are a large number of drafts for new parts of STEP.

STEP has a long-range vision of product data management. Its intention is to define a uniform representation of product information and to provide mechanisms that enable the exchange of product data between different computer systems over the complete product life cycle[4,5].

The idea of STEP is to be a general platform for modelling products. STEP product modelling is based on Integrated Generic Resources. Integrated Generic Resources are refined according to the needs of different industrial areas in different Application Protocols (AP).

An application protocol is in principle first written independently of STEP using the terminology of the application area. The result is an Application Reference Model (ARM). This model is then implemented with the STEP concepts; i.e. using both the integrated resources and the extensions defined in the application protocol itself. The result is an Application Interpreted Model (AIM), which is the actual data model of the application protocol in STEP.

The data models of STEP are defined with the EXPRESS language, which itself is Part 11 of the standard[6]. It has both

*Corresponding author. E-mail: Tomi.Mannisto@hut.fi
Helsinki University of Technology, TAI Research Centre and Laboratory of Information Processing Science, Product Data Management Group, P.O. Box 9555, FIN-02015 HUT, Finland
Paper Received: 6 April 1998. Revised: 17 September 1998. Accepted: 17 November 1998

a textual and graphical notation; the latter is called EXPRESS-G.

The standard is very large, deals with highly abstract concepts and provides hardly any real examples of the modelled products. Therefore, the meaning of STEP concepts is not always clear, and we may have interpreted some concepts differently to what the authors of the standard had in mind.

Generic product structure modelling concepts

In addition to the basic concept of a component having other components as parts, a generic product structure typically supports the concept of classification. For example, the class automobile can have subclasses car and truck. Similarly, engine can have subclasses petrol engine and diesel engine. The interplay between the has-part relationship and classification is an essential question in all generic product structures.

One of the main advantages of classification is the sharing of information through inheritance. For example, suppose a generic product structure specifies that an automobile has an engine as a part. Since truck is a subclass of automobile, a truck also has an engine. The component subtyping property means that in all uses of a component class as a part, any of its subclasses can be used instead. Since petrol engine and diesel engine are subclasses of engine, either kind of engine can be used as part of an automobile. Often the combinations of different kinds of components must be restricted by means of part refinement. For example, one could specify that a truck must have a diesel engine instead of just any kind of engine. Generic product structures also need optional and alternative parts for stating the possible part-of structures of a product. For example, sunroof could be an optional part of car, whereas CD player and cassette player could be alternative parts of radio equipment.

A model that describes all the potential product structures is sometimes called an explicit product structure model[3]. *Figure 1* shows an example of an explicit product structure in the EXPRESS-G notation. For simplicity, the has-part relations are represented as reference attributes; there are many other ways to model them but that discussion is beyond the scope of this paper. The two subclasses of engine represent the engine models manufactured by a company.

An explicit product structure model defines a large number of component combinations. In practice, some of these combinations are usually invalid. A generic product structure thus includes an implicit product structure model, which gives additional conditions that must be satisfied by a valid product. For example, the implicit product structure could prevent the combination of heavy body and engine 1.8. This condition can be expressed as the following EXPRESS rule in car (assuming a function types_of_components that returns a set of types of recursive components of its argument):

```
NOT ('heavy_body'IN types_of_components(SELF)
AND'engine_1.8'IN types_of_ components(SELF))
```

A description of the actual product instance to be delivered to a customer is created on the basis of a generic product structure and a set of customer requirements in a configuration process. For example, 'nice car #127' with its parts 'engine 1.8 #883' and 'nice body #335' is a product instance description.

Generic product structures and EXPRESS

Although we have omitted many details, it seems that generic product structures could be modelled in the EXPRESS language. EXPRESS has many constructs that seem very suitable for this purpose, e.g. powerful mechanisms for classification and condition rules. Nevertheless, it is fundamental to understand the role of EXPRESS within STEP and to understand why EXPRESS cannot be used in this way. In STEP, EXPRESS is meant for describing the generic resources and application protocols, i.e. for defining a standardized data schema for the management of product data within companies and for the exchange of product data between companies. In other words, everything that is modelled in EXPRESS must

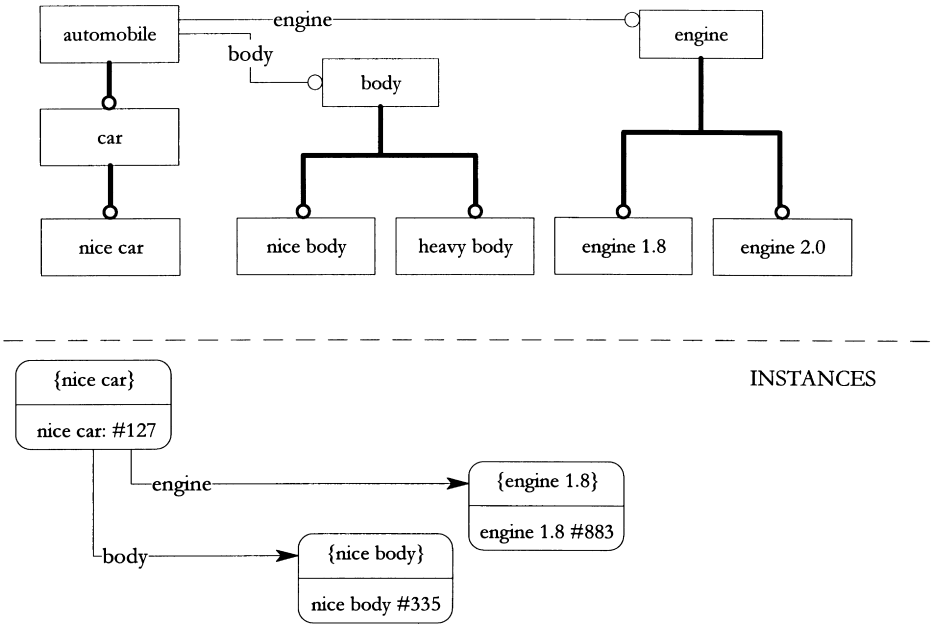


Figure 1 A generic product structure in EXPRESS

become part of the STEP standard. Within STEP it is impossible to write the model of *Figure 1* directly in EXPRESS, because EXPRESS would then be used for modelling data that is specific to a single company and accordingly cannot be standardized.

STEP must therefore define a schema that allows each company to represent its own product data as instances of the standardized schema. Instances of EXPRESS entity types are used for all company-specific data, e.g. component classes, their classification and has-part relationships, validity conditions, etc.

The next section briefly introduces STEP AP 214, and in particular its unit of functionality, called specification control, which addresses the description of automotive products with a large number of variants. Thereafter, we also briefly discuss other similar standardization efforts, e.g. PLIB and STEPlib. Finally, as a fundamentally different solution, we propose a way of extending STEP application protocols by company-specific extensions.

MODELLING PRODUCT VARIANTS IN AP 214

In this section, we introduce some concepts of STEP Part 214: Application Protocol: Core Data for Automotive Mechanical Design Process[7,8], which is still an evolving committee draft instead of an official part of the standard. The central concepts of AP 214 for product variant modelling are introduced with the help of an example shown in *Figure 2*.

In this paper, we only use the Application Reference Model (ARM) of AP 214 and ignore the Application Interpreted Model (AIM). The right-hand side of *Figure 2* shows the ARM terms, which are written in small caps (e.g. PRODUCT CLASS) in this paper. The objects at the same level as an ARM term are examples of this.

The goal of AP 214 is to represent all the data related to the mechanical design of automotive products. The description of product variants constitutes only a very small part of the whole application protocol.

Product class

PRODUCT CLASS objects are used for the classification of products offered by a company to the market. For example, car may have subclasses compact car and economy car.

The example in *Figure 2* considers an imaginary product nice car, which is represented by a PRODUCT CLASS object with the same name.

Specification and specification category

PRODUCT CLASSES can have SPECIFICATIONS attached to them. A SPECIFICATION is a characteristic or functionality of a product seen by the customers. In *Figure 2*, engine 1.8 and normal antenna are examples of SPECIFICATION objects.

SPECIFICATIONS are grouped into SPECIFICATION CATEGORIES. *Figure 2* shows two examples: radio equipment and engine. Engine is a 'mutually exclusive' category, which means that only one of the SPECIFICATIONS belonging to this category can be used in one product. Radio equipment

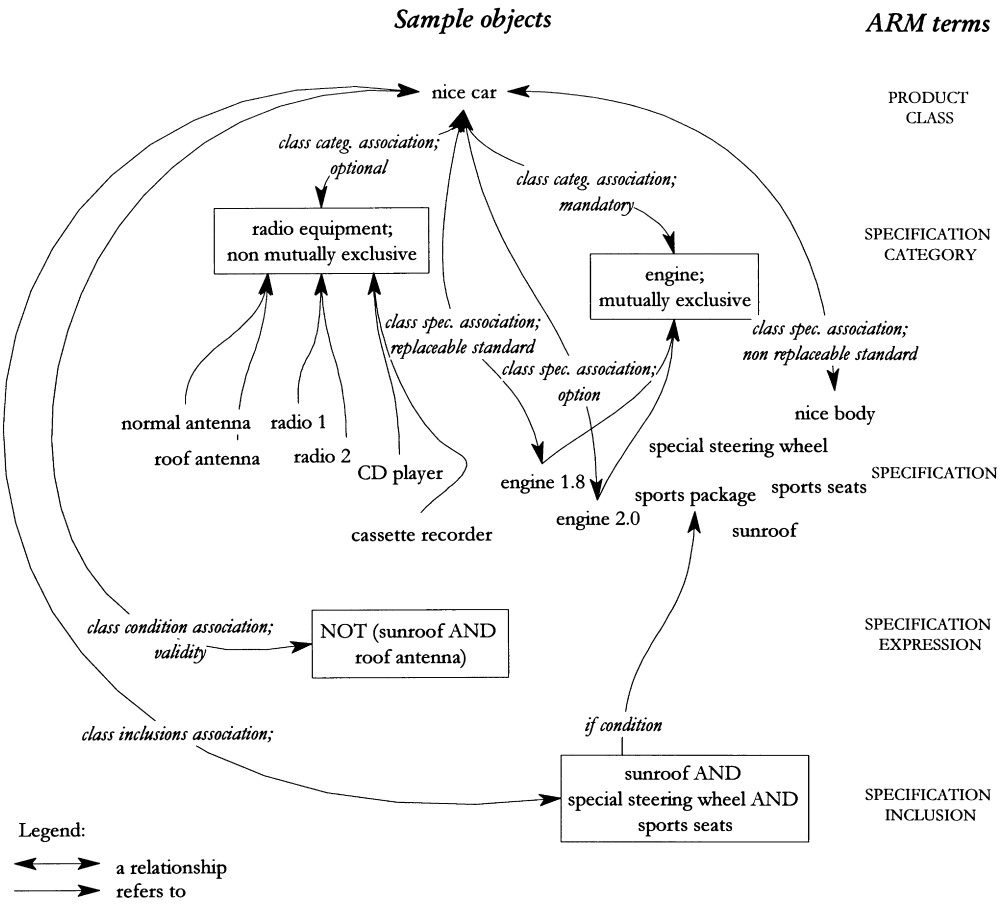


Figure 2 Example of AP 214 concepts

is a 'non mutually exclusive' category since multiple specifications can be selected from that category.

PRODUCT CLASSES and SPECIFICATION CATEGORIES are related by means of CLASS CATEGORY ASSOCIATIONS. Each association states whether a particular SPECIFICATION CATEGORY is mandatory or optional for a particular PRODUCT CLASS. For example, engine is a mandatory SPECIFICATION CATEGORY for the PRODUCT CLASS nice car, at least one SPECIFICATION from it must be selected. (Actually, exactly one engine will be selected because engine is a 'mutually exclusive' category.) Radio equipment, on the other hand, is optional as it is not compulsory to include it in a nice car.

If a PRODUCT CLASS is associated with a SPECIFICATION CATEGORY, the PRODUCT CLASS is also directly associated with all or some of the individual SPECIFICATIONS within the SPECIFICATION CATEGORY. Each such association is characterized by an association type. *Figure 2* shows examples of three association types. For simplicity, most of the associations have been omitted from the figure.

(1) The association between nice car and nice body is of the type 'non replaceable standard'. A nice body must be included in all nice cars. The SPECIFICATION CATEGORY of nice body is not shown in the figure.
(2) The association between nice car and engine 1.8 is of the type 'replaceable standard'. Engine 1.8 is the default selection from the SPECIFICATION CATEGORY engine for a nice car, but may be replaced by another selection from the same category.
(3) The association between nice car and engine 2.0 is of the type 'option'. A nice car can optionally include engine 2.0, which then replaces the replaceable standard, i.e. engine 1.8.

Specification expression

A SPECIFICATION EXPRESSION is a logical expression, which must be satisfied in a valid product instance. An expression consists of a set of operands related by an operator. An operand is either a SPECIFICATION or another SPECIFICATION EXPRESSION. The operator is either 'and', 'or', 'oneof' or 'not'.

For example, SPECIFICATIONS sunroof and antenna can be related through a condition stating that roof antenna cannot be used together with a sunroof. This condition can be modelled as a SPECIFICATION EXPRESSION: NOT ('sunroof' AND 'roof antenna'). A CLASS CONDITION ASSOCIATION then relates the condition with the PRODUCT CLASS nice car.

Specification inclusion

SPECIFICATION INCLUSION is a mechanism for including other SPECIFICATIONS into a product when a particular SPECIFICATION is included.

SPECIFICATION INCLUSION is only intended to automatically complete a set of SPECIFICATIONS after an initial selection of SPECIFICATIONS. A SPECIFICATION INCLUSION is defined by two entities: an 'if condition' and 'included specification'. They both can be either a SPECIFICATION or a SPECIFICATION EXPRESSION.

Figure 2 shows a SPECIFICATION INCLUSION, which defines a set of options called sports package. Selection of a sports package means that sunroof, special steering wheel and sports seats are also automatically included. Therefore, the 'if condition' is the sports package and the 'included specification' is ('sunroof' AND 'special steering wheel' AND 'sports seats'). The SPECIFICATION INCLUSION is associated with PRODUCT CLASS nice car by a CLASS INCLUSION ASSOCIATION.

Comparison with EXPRESS approach

Figures 1 and 2 are similar in the sense that they both represent variants of nice car. In *Figure 1*, engine and its specializations are represented as EXPRESS entities, while AP 214 uses specific concepts SPECIFICATION CATEGORY and SPECIFICATION for the same purpose.

The classification mechanisms provided by AP 214 for modelling products include a limited form of inheritance. AP 214 defines that instances of CLASS CONDITION ASSOCIATION and CLASS INCLUSION ASSOCIATION are inherited by subclasses represented as instances of PRODUCT CLASS. Other properties of a PRODUCT CLASS are not inherited. For example, if automobile, car and truck were modelled as PRODUCT CLASSES in AP 214, the SPECIFICATIONS related to automobile would not automatically be properties of truck. In the EXPRESS approach, i.e. in *Figure 1*, the properties of automobile would be inherited by its subtypes. Whether the part-of relations are inherited or can be refined in subtypes depends on how they are actually implemented. The details are not important here; it is enough to say that there are ways of modelling both the part inheritance and the part refinement in EXPRESS.

The important point is that AP 214 cannot propose 'real' classes, i.e. EXPRESS entity types, for modelling product classes of a company. Discussion of this is the topic of the next section.

DISCUSSION

Use of product classification within STEP

Classification is a powerful data-modelling mechanism that can be naturally applied to industrial products. Classification suits many purposes, even in product modelling. In the previous section, we explained how AP 214 proposes classification to be used for modelling product variants. Other important application areas are classification within component libraries and abstract dictionaries for describing product individuals.

Component data, in particular standard components that are purchased, must be organized for many reasons, e.g. efficient searching of components for new designs and various groupings for statistical analyses of component purchases, stock value, wear-out in use, etc. Classification of component types is a good tool for these purposes. PLIB standardization effort addresses classification of component libraries, especially taking into consideration parametric components[9,10].

For industries in which after-sales is of great importance, it is beneficial to refer to the parts of delivered products with a standard terminology. With such a dictionary on top of a database, one can, e.g. find all delivered products that use a particular kind of component. The required type of information of subsystems and components can be organized into a classification hierarchy or hierarchies. Classification of component types allows, e.g. the specification of the component type more precisely in the previous example. Within STEP, this kind of work has been carried out in

the development of AP 221 for process industries under the name STEPlib[11,12].

These approaches have slightly different goals, but what is common to them and important to this paper, is that they all propose a conceptually similar model. They all define an EXPRESS entity type (or entities types) that represents a class; e.g. the PRODUCT CLASS of AP 214. Consequently, the product classes, e.g. car, nice car, engine 1.8, are EXPRESS instances, and the relationships, including classification, between them are relations between instances. In other words, all these approaches represent both (product) classes and the descriptions of particular products as EXPRESS instance data. This is exemplified in *Figure 3*, which displays the same data as *Figure 1*.

There are natural consequences of representing classes as data. First, there is no inheritance. That is, nice car in *Figure 3* does not inherit anything from car. Second, there is no separation between classes and instances—they all are instances. That is, 'nice car#127' is not an instance of nice car, they are both EXPRESS instances.

However, in order to model classification with inheritance as data, the semantics must be described verbally. For example, it is necessary to describe which 'classes' inherit something, what they inherit and via which relationships. This inheritance must then be implemented by all applications manipulating the data. In addition, the applications should also understand the relation between 'classes' and their 'instances'.

From the STEP point of view, this seems slightly strange as EXPRESS has all the needed concepts for classification, and the tools that understand EXPRESS support inheritance between EXPRESS entity types. Now this must all also be implemented between instances of special entity types. Nevertheless, as explained earlier, there is no alternative because EXPRESS can only be used for the definition of standardized schemas.

This problem exists in all the cases described above. STEPlib needs only inheritance of simple attributes and possibly a few specific relations. It may, therefore, survive with a simple form of inheritance. PLIB has defined a powerful mechanism for describing component libraries. The mechanism includes classification with inheritance and description of class extensions by means of constraints. In the case of generic product structures, there is a need for an even more elaborate classification and inheritance mechanism. Extending the PLIB approach to these needs would implement much of EXPRESS by means of EXPRESS. That is achievable, but the meaningfulness of such an approach can be questioned.

As an alternative, we next outline an extension to STEP that would make it possible to use EXPRESS for modelling the products of a specific company.

Extending STEP

One possibility for representing generic product structures is to allow companies to specialize application protocols for their own needs. An application protocol would only define abstract, generic concepts. A company would then define an extension schema with subtypes of the concepts of the application protocol, e.g. the company-specific product classes nice car and nice body.

In *Figure 4*, the example of nice car is shown according to our proposal. All individual, physical products of a company

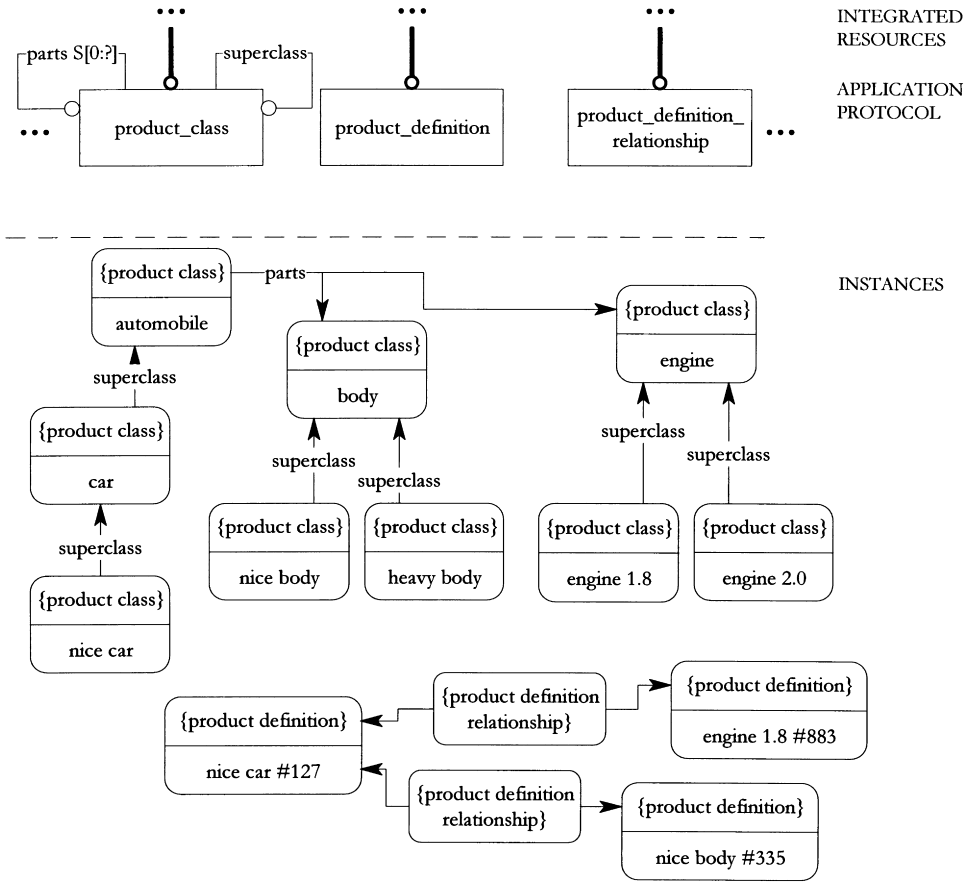


Figure 3 Example of product classes as data

are represented as instances of product_definition of Part 41[13]. Therefore, we have defined all product classes as subtypes of entity type product_definition_for_classification, which is a subtype of product_definition. We have also separated certain product classes, e.g. car and body, since it could be possible to define such product classes specific to an application area in an AP. The classes of a particular company are then specialized from them.

In this approach, a company would use EXPRESS to define its product classification. The full power of EXPRESS may be too much, and therefore, there should perhaps be a mechanism for restricting the allowed refinements. For example, definition of subtypes could be allowed only for product_definition_for_classification. In addition, the extension must be a proper refinement to the application protocol and not contradict it. However, we do not elaborate further on such details of the mechanism here.

The result would be an EXPRESS schema that consists of the necessary integrated resources, application protocol(s) and the company-specific extension. Descriptions of individual products of the company would then be instances of that schema. The company-specific part of the schema must naturally be communicated in the exchange of product data.

The extension schemas bring up some change management problems as they are bound to change more frequently than application protocols. One needs at least a mechanism for labelling the extension revisions. A more advanced model would also incorporate mechanisms for data conversion between revisions of schema extensions and perhaps a notation for the compatibility of schema revisions with respect to certain product classes.

This approach would also change the nature of schema from static to more dynamic and make it necessary to rethink many other fundamental aspects of the standard. The problems of representing generic product structures in a strict class-instance paradigm, e.g. STEP, have also been addressed by McKay et al.[14] in a three-layer model. The top layer contains a general data model for describing generic product structures in terms of parameters. The bottom layer contains a description of individual product instances. The middle layer contains a description of a generic product structure (i.e. a description of a product family) as a combination of instances and classes. A description of a generic product structure on the middle layer can be regarded at the same time as an instance of the schema on the top layer and a schema for the instances on the bottom layer. The middle layer can thus be interpreted as a layer between an application protocol and descriptions of individual product instances. It, therefore, corresponds to the extension schema proposed in this paper. However, the relation between the schema and instance views in the middle layer is rather vague and does not explicitly consider the classification of products.

Similarly to classification, logical expressions for the validity of product configurations can also be defined in an extension schema. Conditions stating the valid combinations of SPECIFICATIONS (including the subclasses of SPECIFICATION, e.g. sunroof) could be modelled as EXPRESS constraints defined in PRODUCT CLASSES (including subclasses of PRODUCT CLASS, e.g. car). For example, PRODUCT CLASS nice car would define a constraint that a set of SPECIFICATIONS related to an instance of nice car should not contain an

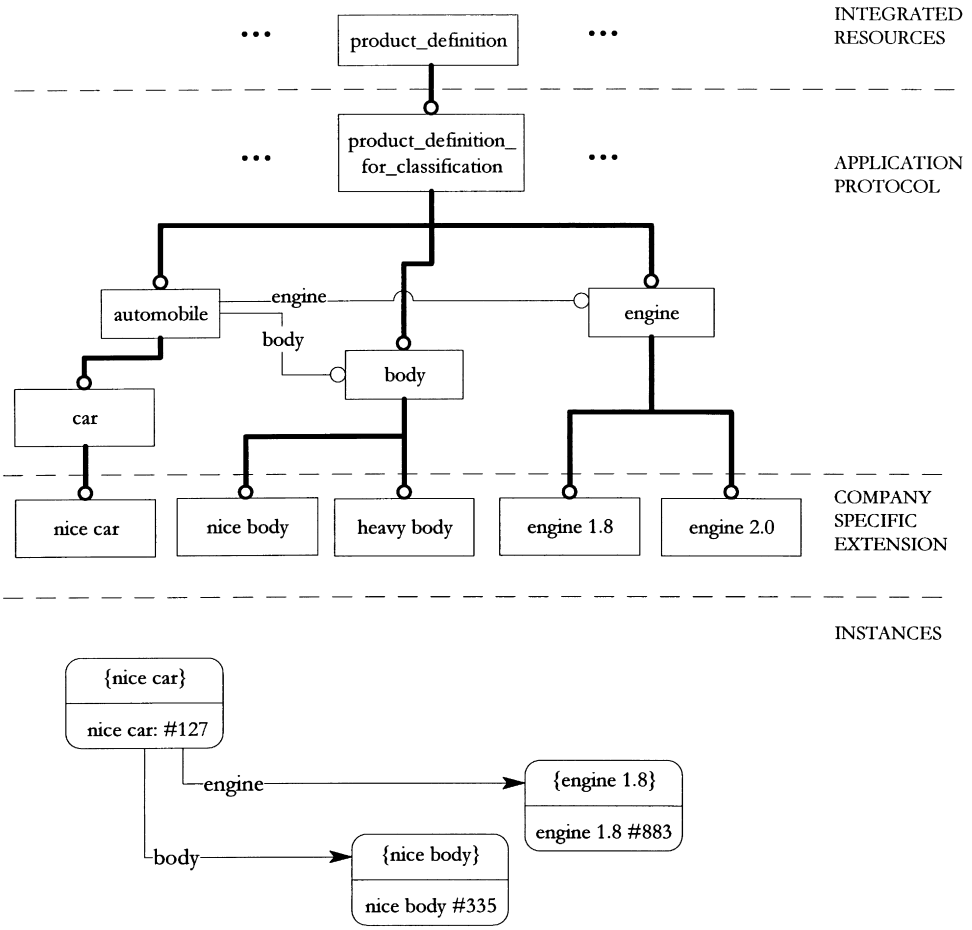


Figure 4 A generic product structure as a schema extension

instance of sunroof and an instance of roof antenna. Such use of EXPRESS constraints, however, contradicts the fundamental idea of STEP that a schema describes only valid instances, an issue also raised by Eastman and Fereshetian[15]. Therefore, it is necessary to allow invalid instances, perhaps in some limited sense, and to have a mechanism for checking the validity of a particular (sub)set of instances.

CONCLUSIONS

STEP is a product modelling approach that takes into account all the aspects of a product, including geometry and organizational data. This paper looked at one particular aspect of product modelling within STEP, i.e. the modelling of product structures and in particular generic product structures. Application protocol 214 for the automotive industry is probably the most advanced part of STEP in this area. AP 214 as well as PLIB and STEPlib provide a classification mechanism for product modelling.

The fundamental problem is that STEP is designed for modelling single products by means of a fixed standard data schema, in which products of a company are represented as instances of that schema. If an AP wants to provide a classification mechanism for products, the product classes of a company must also be represented as data. There are drawbacks in modelling classification and constraints as data; all concepts must be described as EXPRESS instances, eventually leading to the redefinition of much of EXPRESS.

We outlined an alternative, in which part of the schema, i.e. the company-specific specialization of the concepts of an application protocol, would be treated as data exchanged between systems. Although this approach would necessitate major changes in the principles of STEP, the changes are necessary if STEP is to be used for modelling generic product structures.

Much of what is said in this paper about modelling generic product structures is also applicable to other application areas of STEP. For example, classification and constraint mechanisms are by no means restricted to generic product structure modelling. The structure of STEP means that these mechanism must be described as data. STEP is fundamentally based on a fixed standardized product schema that cannot be extended for the purposes of a company. In our view, this seriously limits the potential of STEP when companies start utilizing more advanced product modelling concepts. This is going to happen when companies, instead of modelling their products one by one, extract common pieces of their product definitions so that they can be used in multiple variants or in different product families. Therefore, the strict static structure of STEP may even compromise future credibility of STEP, unless the basic structure of the standard is refined.

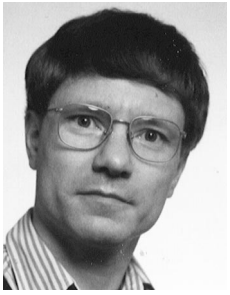
ACKNOWLEDGEMENTS

All authors belong to the Product Data Management Group of TAI Research Centre of Helsinki University of Technology; this work has benefited from the many discussions with the other members of the group. This research has been funded by the Finnish Technology Development Centre (TEKES) and Helsinki Graduate School of Computer and Engineering (HeCSE).

REFERENCES

1. van Veen, E.A., Modelling product structures by generic bills-of-material. PhD Thesis, Technische Universiteit Eindhoven, 1991.
2. Erens, F., McKay, A. and Bloor, S., Product modelling using multiple levels of abstraction—instances as types. *Computers in Industry*, 1994, **24**(1), 17–28.
3. Männistö, T., Peltonen, H., Sulonen, R., View to product configuration knowledge modelling and evolution. In: Faltings B, Freuder E, editors. Configuration—Papers from the 1996 AAAI Fall Symposium, Technical Report FS-96-03. The American Association for Artificial Intelligence, AAAI Press, 1996:111–118.
4. ISO International Standard 10303-1: Industrial automation systems and integration—Product data representation and exchange—Part 1: Overview and fundamental principles. 1994.
5. Owens, J., *STEP An Introduction*. 2nd edn. Information Geometers, UK, 1997.
6. ISO International Standard 10303-11: Industrial automation systems and integration—Product data representation and exchange—Part 11: Description methods: The EXPRESS language reference manual. 1994.
7. ISO Committee Draft 10303-214: Industrial automation systems and integration—Product data representation and exchange—Part 214: Application protocol: Core data for automotive mechanical design process. ISO TC184/SC4 N319, 1995.
8. ISO Second Committee Draft 10303-214: Industrial automation systems and integration—Product data representation and exchange—Part 214: Application protocol: Core data for automotive mechanical design process. ISO TC184/SC4 N577, 1997.
9. ISO Draft International Standard 13584-10: Industrial automation systems and integration—Parts library—Part 10: Conceptual model of parts library, 1995.
10. ISO Draft International Standard 13584-42: Industrial automation systems and integration—Parts library—Part 42: Methodology for structuring part families, 1996.
11. 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.
12. Guide on STEPlib. ISO TC184/SC4/WG3/N424, 1997.
13. ISO International 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.
14. McKay, A., Erens, F. and Bloor, S., Relating product definition and product variety. *Research in Engineering Design*, 1996, **8**(2), 63–80.
15. Eastman, C.M. and Fereshetian, N., Information models for use in product design: a comparison. *Computer-Aided Design*, 1994, **26**(7), 551–572.

Tomi Männistö works as a Research Scientist at the TAI Research Centre of the Helsinki University of Technology. His main interests lie in product configuration data modelling. In particular, he is investigating the evolution of configuration models and their interplay with the evolution of product instances, on which subject he is preparing his PhD thesis at the Helsinki Graduate School of Computer Science and Engineering.



Hannu Peltonen works as a Senior Assistant at the Laboratory of Information Processing Science at the Helsinki University of Technology. He is preparing his PhD thesis on Product Data Management. Hannu Peltonen has been designing and implementing document and software management systems in joint projects with industry.

Asko Martio works as a Research Director at the TAI Research Centre of the Helsinki University of Technology. Since completing his MSc degree in 1969, he has held different management positions at Stömberg Ltd (currently part of ABB Concern) and KONE Corporation. His research activities include product data management practices within the manufacturing industry.

Reijo Sulonen is Professor of Computer Science at the Helsinki University of Technology. His research interests include database systems, product data management, process modelling, software engineering and electronic media.