Information model for configuration of modular products

Davor Pavlić

Keywords: configuration, product variants, product family, modular products, STEP

Abstract

The necessity for products adapted to the individual customer demands is more and more present in the current conditions of market globalization. Modular product architecture promises the advantages of high volume production while at the same time, being able to produce a high variety of products that are customized for individual customers. This paper presents the informational model for configuration product variants of modular architecture in relation with customer requirements and configuration knowledge. All variants of the product family are developed by the general product variant defined for a particular product family. General product variant could contain three classes of generic modules. Those are: working generic modules, auxiliary generic modules and secondary generic modules. The product variant is build up from the instances of the particular generic modules. The information models are developed according to the STEP standard 10303-214. The entities of the information models are representing using the EXPRESS-G notation.

1 Introduction

In the early part of the 20th century, Henry Ford said, "You can have any color you want as long as it's black". Since then, marketplace has dramatically changed and a product of abovementioned range could not possibly survive. Necessarily, products are being adapted to meet the individual demands of a customer enjoying the market globalization. It is variety in demand that makes companies offer a wider range of variant products in order to be competitive on the market. However, increased variety does not necessarily generate a larger profit because such a variety implicates higher fixed costs, (e.g. manufacture, installation, service, etc).

In order to increase the product variety and minimize costs, it is necessary to interrelate the modular architecture with customer's demands and configuration knowledge [Figure 1]. The moment the quantity of customer's demands and differences between them increase, the number of modules increases as well. Interrelation between demand and modules is realized through configuration knowledge, which defines the way demand is influenced in the selection of modules. Thus, higher demand increases the range of product variants, which in consequence decreases the design time. Design time decreases since the solution to a particular demand is found quicker with less chance of mistaking. Also, with a decrease in design time, there is a decrease in the both design and production costs, which again is a consequence of the production of modules. Eventually, the lower the design and production costs are, the larger the profit is.

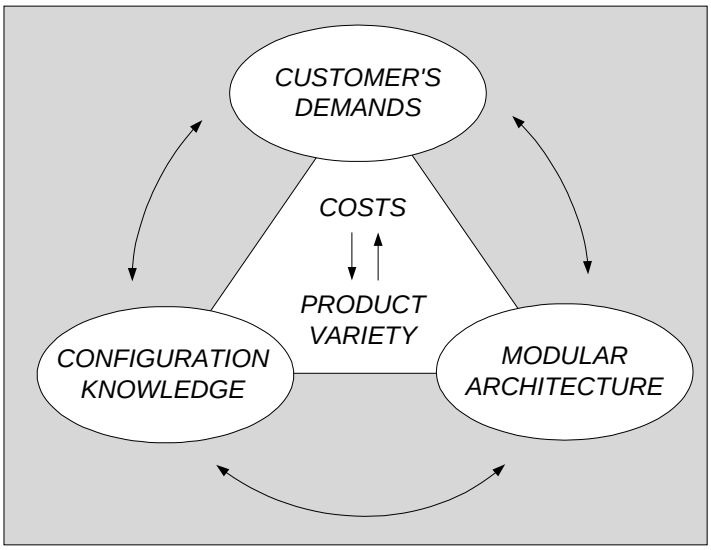

Figure 1. Aspects relevant for the managing of product variety

2 Fundamentals

Studying the product variants present on the market, one might notice they vary one from another with respect to the following [1]: size, market, performance, functionality and aesthetic. The differences between product variants along with variety in demand, force the designer to develop product variants adapted to the customer's demands in less time and with tight resources available. The configuration of product variants is seen as a method in developing product variants that should have a modular architecture. Therefore, they are called modular products. The advantage of modular architecture is high volume production combined with quite a variety of products customized for individual customers [2]. In the development of product variants, enterprises focus on the development of product family rather than an individual product. In this article, a product family is defined by a set of products originated from a common set of modules to obtain a range of product variants that would cover a certain market segment [3].

Each product variant is configured upon the customer's requirements previously specified by the requirement list for that particular product family. All variants of the product family are developed by the general product variant defined for that product family. The general product variant consists of generic modules classified by three module types: working, auxiliary and secondary module. Working generic modules are defined for each product variant of a particular product family. Auxiliary generic modules comprise of product variants listed by the customer's requirements. Secondary generic modules exist in the product variant only when auxiliary modules need some additional modules to fulfill the customer's requirements. A generic module can contain many instances. The product variants based on the general product variant comprises the instances of particular generic modules. Further diversification of generic modules into module instances enables defining each module instance by the customer's requirements.

The information model that is presented in this article, serves as a framework for configuration of product variants. The information model is developed in accordance with the product modeling standard of ISO 10303-214 [4]. The entities of the information models are representing using the EXPRESS-G notation. ISO 10303, or STEP (STandard for Exchange of Product model data), is an international standard that provides a representation

of product information along with the necessary mechanisms and definitions that enable the exchange of product data [5]. STEP consists of the following common parts: the Integrated Generic Resources (IGR), Integrated Application Resources (IAR) and application specific protocols (AP). AP214 is an application protocol developed as a standard for the exchange of information relevant to vehicle design. It comprises of the extensive constructs for representing structures, their hierarchies, views etc. EXPRESS is an object focused information modeling language with a graphical representation which is based on the entities and relations: EXPRESS-G.

3 Case study: Information model for configuration of modular products

The proposed information model for configuration of modular products consists of several modeling structures. Each model structure represents the entities that describe particular areas linked to the configuration of modular products. These entities correspond to the semantics of STEP.

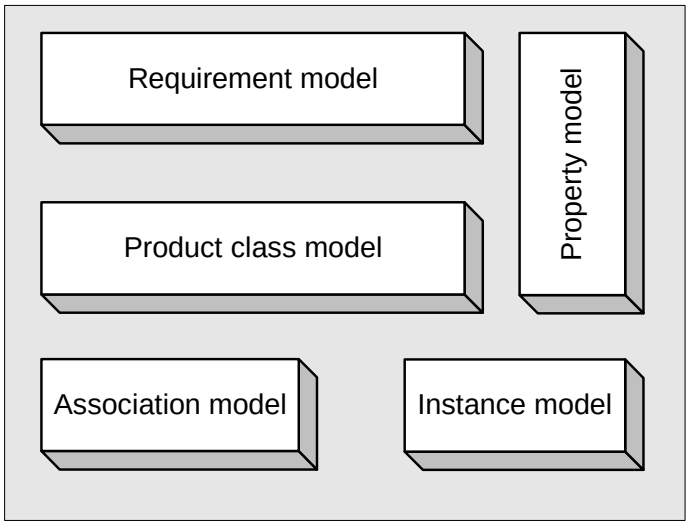

Figure 2. Modeling structures for the proposed information model

The following model structures are represented in Figure 2:

- *Requirement model* - defines the entities describing the customer's requirements and requirements list, both necessary to determine the product variant,
- *Product class model* - defines the entities describing product families and product variants,
- *Property model* - defines the entities describing the characteristics of requirements and modules,
- *Association model* - defines the entities describing the module classification,
- *Instance model* - defines the entities describing the module instances as well as those defining the rules and constraints for their utilization.

3.1 Requirement model

Entities for description of the customer's requirements and requirements list have not been defined in the application protocol AP214, ISO 10303 STEP standard. The configuration process starts with definition of the requirements and requirement list. The entities described

in this model as shown in Figure 3 represent the expansion of ISO 10303 standard and application protocol AP214.

The *requirement* entity describes requirements that are essential to product configuration. It is important to emphasize that *requirement* describes only the definition of requirements, without a value immanent to a particular requirement. The value of a particular requirement is described by *Property model* and is linked to the requirement by *item_property_association*. The following attributes describe the *requirement* entity:

- *id* - specifies the unique identifier of the requirement,
- *name* - specifies a word or group of words by which the requirement is referred to,
- *description* – specifies a textual definition of the requirement.

In the defining process, some requirements exclude others. Potentially, a particular requirement implies inclusion of an additional one. Therefore, *requirement_inclusion* entity specifies the influence between different requirements through the following attributes:

- *id* - specifies the unique identifier of the influences between different requirements,
- *description* - specifies additional information of requirement influences,
- *id_requirement_relating* - specifies the requirement which influences others,
- *id_requirement_related* - specifies the additional requirements influenced by the one specified in *id_requirement_relating* attribute,
- *id_requirement_expresion* - specifies the operator that determines the influence between different requirements, such as the following allowed ones:
    - *AND* - mutually inclusive requirements,
    - *NOT* - mutually exclusive requirements.

Requirement_list entity identifies the requirement list through the following attributes:

- *id* - specifies the unique identifier of the requirement list,
- *name* - specifies a word or group of words by which the requirement list is referred to,
- *description* - specifies additional information of requirement list,
- *level_type* - specifies the level of the requirement list. Requirement lists based on the template represent the lower level lists. The higher level list represents the templates of requirement lists,
- *version_id* - specifies the identification of a particular version of a requirement list.

A relationship between two requirement lists is defined by *requirement_list_relationship* and described through the following attributes:

- *description* - specifies additional information of the relationship between two requirement lists,
- *related* - specifies the second of the two requirement lists related by the entity *requirement_list_relationship*. The semantics of this attribute is defined by *relation_type*,
- *relating* - specifies the first of two requirement lists related by the entity *requirement_list_relationship*. The semantics of this attribute is defined by *relation_type*,
- *relation_type* - specifies the meaning of the relationship through the following values:
    - *derivation* - the *requirement_list_relationship* defines the relationship in which a related requirement list is derived from a relating requirement list (requirement lists are derived from the template requirement list),
    - *version_sequence* - the *requirement_list_relationship* defines the relationship in which a relating requirement list is a pre-revision of the requirement list and a related requirement list is a post-revision of the requirement list.

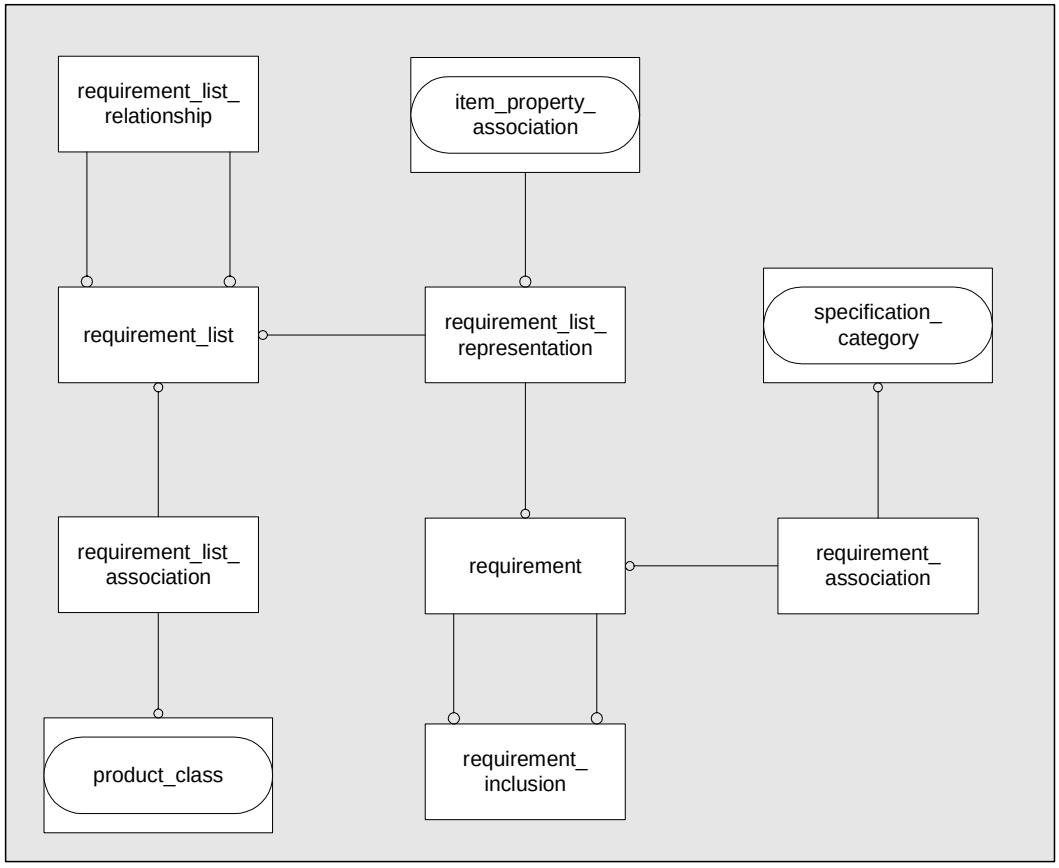

Figure 3. Entity diagram describing the customer's requirements and requirements list

Requirement_list_representation entity specifies an association between *requirement* and *requirement_list* entities. It specifies all requirements comprised in the requirement lists. The attributes describing *requirement_list_representation* are as follows:

- *description* - specifies additional information of an association,
- *id_specified_requirement* - specifies the requirement which is associated with the requirement list defined by *id_specified_requirement_list* attribute**,**
- *id_specified_requirement_list* - specifies the requirement list which consists of the requirements defined by *id_specified_requirement* attribute.

A separate requirement list is defined for a particular product family. The *requirement_list_association* entity specifies an association between a requirement list (*requirement_list)* and a particular product family (*product_class*). This *requirement_list_association* is describe through the following attributes:

- *description* - specifies additional information of an association,
- *describing_requirement_list* - specifies the requirement list (defined by the attribute *level_type* as a template requirement list) which is associated with the product family defined by the attribute *described_element*,
- *described_element* - specifies the product family for which requirement list is defined by the attribute *describing_requirement_list*.

Particular requirements influence the selection of the auxiliary generic modules. The *requirement_association* entity specifies the association between the requirement (*requirement)* and generic modules (*specification_category*). The following attributes describe the *requirement_association* entity:

- *description* - specifies additional information of an association,

- *describing_requirement* - specifies an requirement influencing the selection of auxiliary generic modules defined by *described_element*,
- *described_element* - specifies the auxiliary generic modules specified by the requirement and defined by *describing_requirement*.

3.2 Product class model

The product class model denotes the entities used in identification of both product family and product variants, as presented in the Figure 4 where *product_class* stands for the basic entity. As mentioned in the previous chapter, a requirement list is especially defined for a particular product family. The entity *product_class* can take a different meaning according to the level in which the *product_class* entity is considered. In an association between the requirement list and product family, the *product_class* entity is defined as a product family. In case the product family is discussed on the grounds of comprising product variants, *product_class* is defined as a product variant.

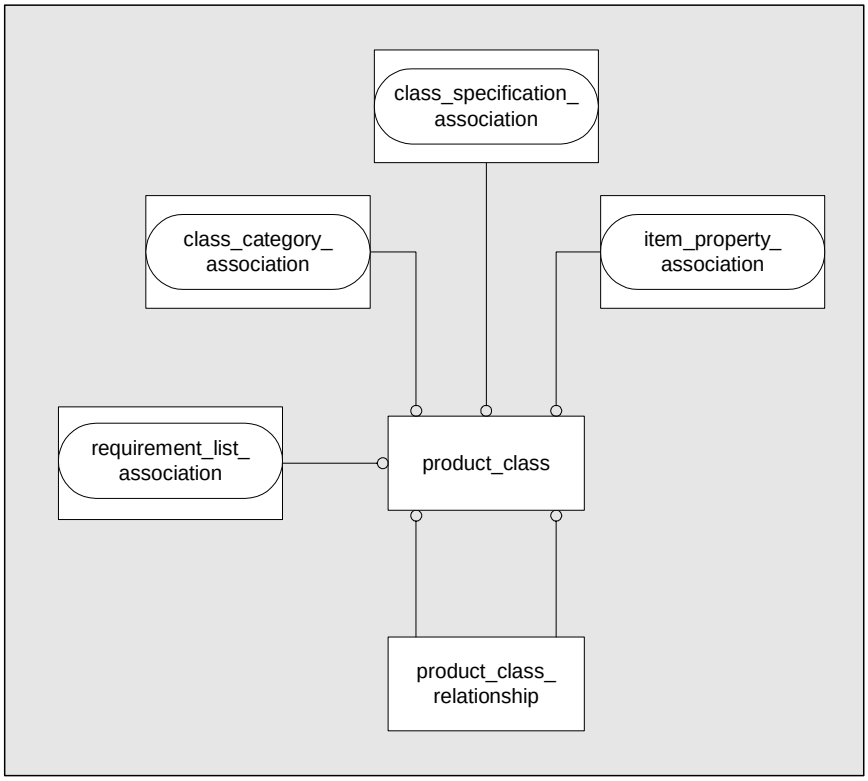

Figure 4. Entity diagram describing product family and product variants

Product_class entity is described through the following attributes:

- *id* - specifies the unique identifier of the product family or product variant,
- *name* - specifies a word or group of words by which the product family or product variant is referred to,
- *description* - specifies additional information of the product family or product variant,
- *level_type* - specifies the level which describes the meaning *product_class*. In this article, the *product_class* entity is being discussed on two levels: higher and lower. Where discussed on the higher level, *product_class* is considered as a product family. Where discussed on the lower level, *product_class* is considered as a product variant,
- *version_id* - specifies the identification of a particular version of the product family or product variant.

Due to different meanings the *product_class* entity takes, it is necessary to define the association between them, which is done by the *product_class_relationship* entity and described though the following attributes:

- *description* - specifies additional information of the association between different meanings,
- *related* - specifies the second level related by the entity *product_class*. The semantics of this attribute is defined by *relation_type*,
- *relating* - specifies the first level related by the *product_class* entity. The semantics of this attribute is defined by *relation_type*,
- *relation_type* - specifies the meaning of the association though the following values:
  - *hierarchy* – a general product variant defined by *related* is subordinate to the product family, defined by *relating*,
  - *derivation* – the product variants defined by *related* are derived from a general product variant, defined by *relating*.

3.3 Property model

The previous two chapters describe the entities of an information model which defined the terms like requirement, requirement list, product family and product variant. All entities used in that part of information model, only identified the terms and do not consider the values which are associated with them. Property model describes the entities for identification and association of the values with the other entities.

The *property_value* entity specifies either numerical or textual values defined by *value_with_unit* and *string_value*. Therefore, both *value_with_unit* and *string_value* are derived from *property_value*. The *value_with_unit* entity represents either single numerical measure or a range of ones with upper, lower or upper and lower bounds without further. It does not specifies them. The attributes describing the *value_with_unit* entity are as follows:

- *significant_digits* - specifies the number of decimal digits of numerical value,
- *unit_component* - specifies the unit in which the entity *value_with_unit* is expressed and it is defined by the entity *unit*.

Numerical values are further specified by *numerical_value*, *value_range* and *value_limit*, which are all derived from *value_with_unit*, inheriting its attributes. The *numerical_value* entity specifies the quantity of a numerical value by *value_component* attribute. The *value_range* entity specifies the pair of numerical values representing the range in which the value shall lie. The attributes of this entity are as follows:

- *lower_limit* - specifies the minimum acceptable value,
- *upper_limit* - specifies the maximum acceptable value.

The *value_limit* entity specifies a numerical value by either lower or the upper limit through the following attributes:

- *limit* - specifies the value of the limit,
- *limit_qualifier* - specifies the kind of limit. The following values shall be used: *maximum* (the specified limit is an upper limit) and *minimum* (the specified limit is a lower limit).

The *string_value* entity specifies the textual value comprising one or more alphanumeric characters defined by the *value_specification* attribute.

The *property* entity represents the physical characteristics of the product and customer's defined measurements of the product. The *property_value_representation* entity denotes *property* by a specified value defined by *property_value*. The attributes describing *property_value_representation* are as follows:

- *definition* - specifies the entity *property* which characterizes the entity *property_value_representation*,
- *qualifier* - specifies the kind of the value used, which is as follows:
  - *nominal* - the value is a nominal value,
  - *specified* - the value is a specified value,
  - *typical* - the value is a typical value.
- *specified_value* - specifies the value of *property* and is defined by *property_value*,
- *value_determination* - specifies information on how to interpret the value. The following values shall be used:
  - *calculated* - the value has been calculated,
  - *designed* - the value represents a value intended by the design,
  - *estimated* - the value has been estimated.

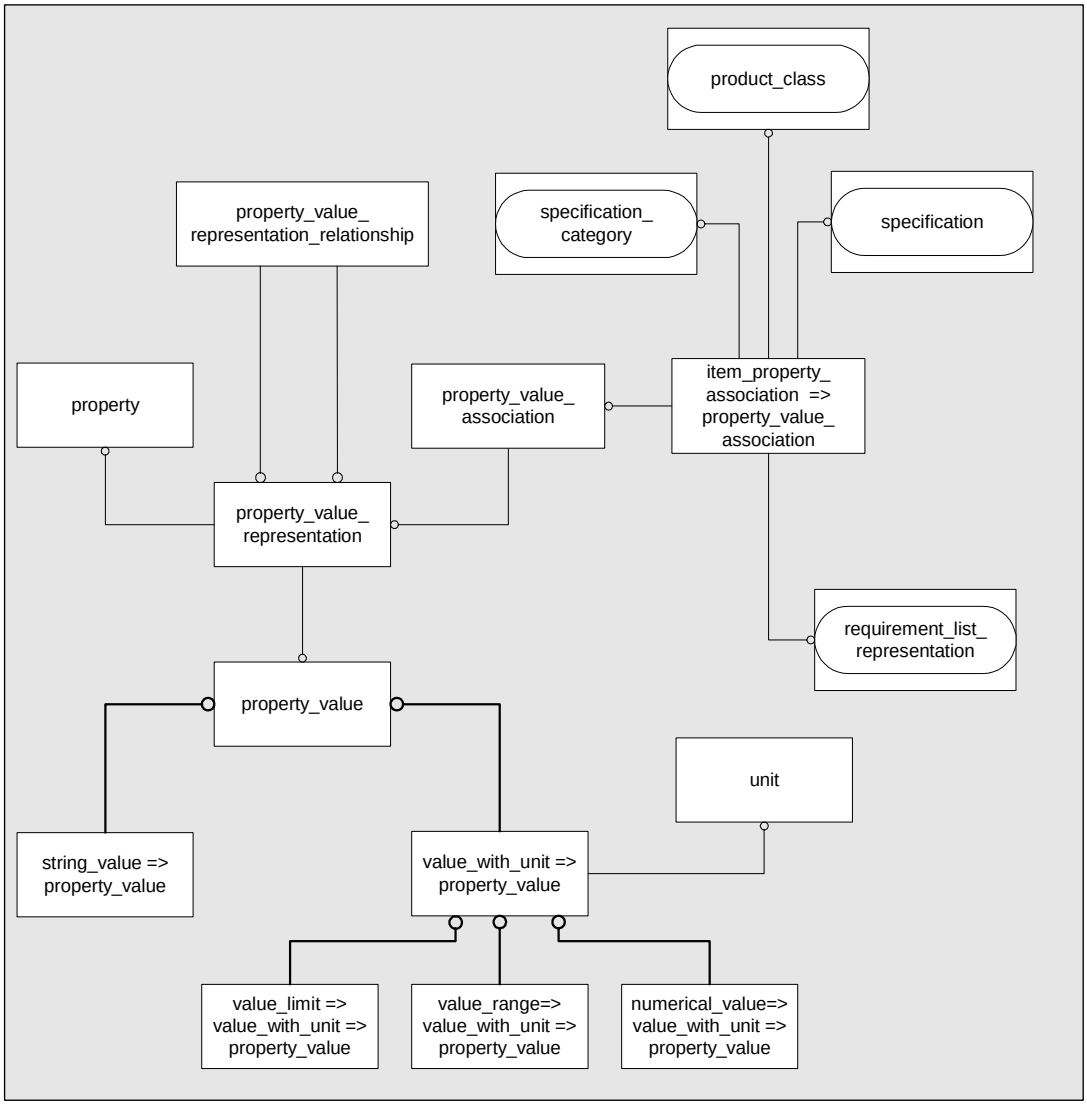

Figure 5. Entity diagram describing the property value

The *property_value_representation_relationship* entity represents the influence between the different values through the following attributes:

- *description* - specifies additional information on influence between different values,
- *related* - specifies the second of the two values which are mutually influenced. The semantics of this attribute is defined by *relation_type*,

- *relating* - specifies the first of the two values which are mutually influenced. The semantics of this attribute is defined by *relation_type*,
- *relation_type* - specifies the meaning of the association thought the following values:
  - *dependency* - value, defined by the attribute *related*, is influenced by the value, defined by the attribute *relating*,
  - *equivalence* - value, defined by the attribute *related*, represents the same value, defined by the attribute *relating*, differs from one another in different units and values.

The association of the values with other entities of the information model begins with the *property_value_association.* This entity specifies which value should be assign to the other entity through the following attributes:

- *description* - specifies additional information of the assigned value,
- *describing_property_value* - specifies the value, defined by the entity *property_value_representation*, that is being assigned.

Item_property_association specifies the entities by the attribute *described_element*, to which specified value is assigned. This entity is derived from the *property_value_association* entity.

3.4 Association model

General product variant (defined by entity *product_class*) consists of the several generic modules classified by three module types: working, auxiliary and secondary module. The entities described in the association model as shown in Figure 6

Figure 6, represent the identification and classification of modules. The entity *specification_category* that identifies the modules is described through the following attributes:

- *id* - specifies the unique identifier of the module,
- *description* - specifies additional information of the module,
- *implicit_exclusive_condition* - specifies whether the module instances are mutually exclusive in the particular product variant through the values of either *true* or *false*. The true value is used in this information model because general product variant consists of generic modules, which is realized by one instance only (defined by the entity *specification*) of each generic module.

All variants of the product family are developed by the general product variant defined for a particular product family. The entity *class_category_association* defines which generic modules are the working generic modules that are present in general product variant. This entity is described through the following attributes:

- *associated_category* - specifies the module of which the meaning in the product variant is specified by the attribute *mandatory*. The module is defined by the *specification_category* entity.
- *associated_product_class* - specifies the product variant, defined by the entity *product_class*, which consists of the module defined by the attribute *associated_category*,
- *mandatory* - specifies the working module type. Modules defined by the attribute *associated_category* are the working generic modules.

If requirement list consists of the requirements not realized by the working generic modules, then the requirements are realized by the auxiliary generic modules. The entity *class_specification_association* defines which generic modules are auxiliary or secondary generic modules that are present in the general product variant. This entity is described through the following attributes:

- *associated_product_class* - specifies the product variant, defined by the entity *product_class*, which consists of the module defined by the attribute *associated_specification*,
- *associated_specification* - specifies the module which type is defined by attribute *association_type*,
- *association_type* - specifies the module types through the following values:
    - *option* - auxiliary module type. The module defined by the attribute *associated_specification* exists in the product variant when specified by the requirement,
    - *addition* - secondary module type. The module is defined by the attribute *associated_specification* and present in the product variant only when auxiliary generic modules need some additional modules to meet the customer's requirements,

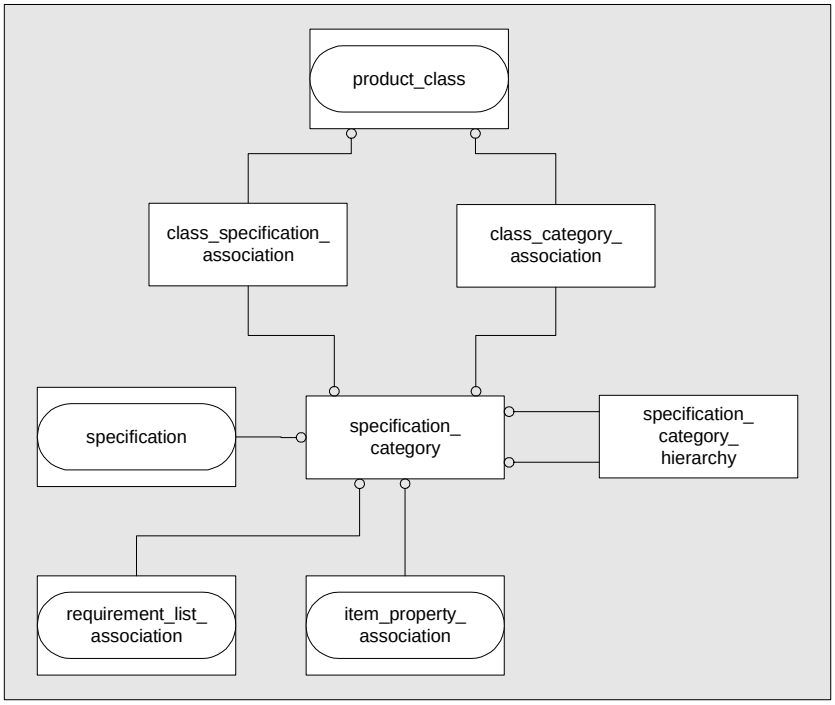

Figure 6. Entity diagram describing the classification of modules

The sequence of the determination of generic modules instances is described by the hierarchical structure of the generic modules, which is defined by the entity *specification_category_hierarchy* through the following attributes:

- *sub_category* - specifies the lower level in hierarchy of the generic modules in the product variant,
- *super_category* - specifies the higher level in hierarchy of generic modules in the product variant.

3.5 Instance model

A generic module can contain many instances. The instance of the module represents the generic module in the product variant adapted to the customer's requirements. The instance model denotes the entities used in description of module instances, as presented in the Figure7 where *specification* stands as an entity specifying the module instance. The entity *specification* is described through the following attributes:

- *id* - specifies the unique identifier of the module instance,
- *name* - specifies a word or group of words by which the module instance is referred to,
- *description* - specifies additional information of the module instance,
- *category* - specifies the module to which the instance belongs,
- *package* - specifies the necessity of existence the instances of secondary generic modules that enable us to meet the customer's requirements.

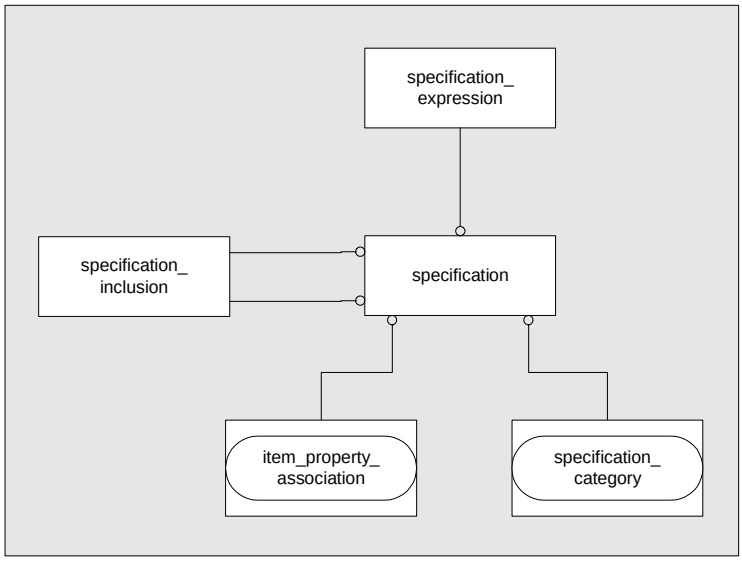

Figure 7. Entity diagram describing the module instances

In the process of determination of generic modules instances, some of them exclude others. Potentially, a particular instance implies inclusion of an additional one. Therefore, *specification_inclusion* entity specifies the influence between different instances through the following attributes:

- *id* - specifies the unique identifier of the influences between different instances,
- *description* - specifies additional information of instance influences,
- *if_condition* - specifies the instance which influences others,
- *included_specification* - specifies the additional instances influenced by the one specified in *if_condition* attribute,

The entity *specification_expression* specifies the kind of the influences between the instances. This entity is described through the following attributes:

- *id* - specifies the unique identifier of the kind of the influences,
- *description* - specifies additional information of the kind of the influences,
- *operand* - specifies the instance that the attribute *operation* is referred to,
- *operation* - specifies the operator that determines the influence between different instances, such as the following allowed ones:
  - *AND* - mutually inclusive influences,
  - *NOT* - mutually exclusive influences.

4. Conclusion and directions for further research

This research is focused on modules design, which involves selecting modules combination to best satisfy the given set of requirements. As it was mentioned before, a successful managing of product variety relies on a few aspects: modular product architecture, stronger interdependences between customer requirements and product architecture and configuration knowledge.

Managing product variety should be observed within the configuration of product variants, in which the abovementioned aspects are interrelated. In order to do so, an information model is defined to provide a framework for configuration of product variants. Proposed information model describes the configuration of general product variant adapted to the customer's requirements for a particular product family. General product variant consists of generic modules that are classified by three module types: working, auxiliary and secondary. Additional diversification of the generic modules into module instances provides a definition of each module instances by the customer's requirements. Dependences between the requirements, requirements and generic modules and between the instances of different generic modules are represented by the constraints immanent to configuration knowledge. Information model as presented here does not cover all aspects necessary for the description of the configuration of product variants. Thus, the presented information model does not cover configuration process and configuration knowledge managing the configuration process. Further research is focused on integration of presented information model with an information model of configuration process. Such integrated information models offer several important potential advantages: product variants are adapted according to the customer requirements; product variants are based on systematization knowledge; high complexity tasks are solving and shorten configuration time needed.

This study is a sequel to bigger-scheme effort to increase product variety without significant additional costs. Should we apt to have this research actually decrease the costs in industry, it is necessary to extend the study to the areas connected with product development. The aim of such extension is an integrated development of modular products. The extensions of the research are concern to the integration of the configuration model with the system for product data management (PDM) and CAD system.

References

[1] Liedholm, U., "Conceptual Design of Products and Products Families", Proceedings of the 4th WDK Workshop on Product Structuring, Tichem, M., Andreasen. M. M., Duffy, A. H. B., Delft University of Technology, Delft The Netherlands, 1998, pp. 91-112

[2] O'Grady, P.,Liang; W.Y., Tseng, T.L., Huang, C.C., Kusiak, A.: "Remote Collaborative Design With Modules"; Techical Report TR 97-03, University of Iowa, 1997.

[3] Andreasen, M.M., McAloone, T., Mortensen, N.H.; " Multi-Product Development - platforms and modularization"; Technical University of Denmark, ISBN: 87-90130-34-0, Lyngby, 2001.

[4] ISO 10303-214:2000, "Product data representation and exchange: Application protocol: Core data for automotive mechanical design processes", ISO, 2000.

[5] ISO 10303-1:1994, "Industrial automation systems and integration - Product data representation and exchange - Part 1: Overview and fundamental principles", ISO, 1994.

Davor Pavlić, B. Sc.
University of Zagreb
Faculty of Mechanical Engineering and Naval Architecture
Chair of Design Theory
I. Lučića 5, 10000 Zagreb, Croatia
Phone: +385-1-6168 117
e-mail: davor.pavlic@fsb.hr