Feature in product engineering with single and variant design approaches. A comparative review

Fernando F. Romero, Emilio Sanfilippo, Pedro Rosado, Stefano Borgo, Sergio Benavent

Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 41 (2019) 328–335

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

8th Manufacturing Engineering Society International Conference

Feature in product engineering with single and variant design approaches. A comparative review.

Fernando Romero[a], Emilio M. Sanfilippo[cd], Pedro Rosado[a]*, Stefano Borgo[b], Sergio Benavent[a]

[a]*Universitat Jaume I, Av. Vicent Sos Baynat, Castelló de la Plana 12007,Spain*
[b]*Laboratory for Applied Ontology ISTC-CNC, Trento 38123, Italy*
[c]*Le Studium Loire Valley Institute for Advanced Studies, Orléans, 45000, France*
[d]*CESR – Université de Tours, 59, rue Néricault-Destouches, 37020, Tours, France*

Abstract

The paper contributes to the study of feature concept by analyzing its use in techniques and models of both classical product engineering and product engineering with variants. In particular, the latter field broadens the classical modeling problem via the introduction of concepts like product family and product line.
Until today the literature dedicated to classical product engineering has focused on the study of the so-called e*ngineering features,* e.g. form or functional features, and has ignored the larger class of features used in techniques related to the analysis and design of products with variants. This work intends to overcome this deficit by providing a broader vision of the problem and by pointing out the need for a renewed analysis and classification of feature definitions based on their originating design context. The work is part of a research project which aims to provide a conceptual framework to unify multiple feature notions and develop feature-based methodologies and applications to support product design.

Keywords: product variant modelling, feature models, feature definitons, classical engineering, domain engineering

* Corresponding author. Tel.: +34-964728185; fax: +0-000-000-0000 .
E-mail address: rosado@uji.es

© 2019 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of the scientific committee of the 8th Manufacturing Engineering Society International Conference.

1. Introduction

The design and development of new products, independently of the product type and its scope, is a complex, interdisciplinary, and knowledge-intensive process. Experts need computer aided systems and tools to manipulate heterogeneous domain knowledge in a reliable and transparent way [1]. For this purpose, it is essential to gain a common and reliable understanding of the engineering concepts and informational artifacts that are commonly used. One of the most relevant concepts in design is undoubtedly that of *feature*. This concept is at the core of many techniques used in mechanical engineering [2], electromechanical systems [3], cyber-physical systems (CPS) [4] and software product line engineering (SPLE) [5]. Feature-based modeling approaches have been fundamental for the development of computational systems (like parametric feature-based CAD systems) and have greatly influenced the overall CAD industry [1].

From a practical perspective, features can be seen as modeling elements that have a role similar to objects in object-programming approaches. Once created, features can be reused and customized to facilitate product design tasks but also to model product variability, or to bridge different perspectives on the same product (e.g., the design and the manufacturing views). However, the literature shows that the semantics of the term feature is highly heterogeneous [6]. This heterogeneity can hinder design tasks, data exchange, or interoperability across engineering communities and applications if not properly handled.

In this paper we present an analysis of relevant definitions used to characterize the feature notions and their scope. This is a step in a broader research project whose aim is to develop a conceptual framework for comparing and possibly unifying different feature notions in engineering. The overall goal of our research is challenging, as several design approaches need to be carefully investigated and compared, their semantics to be clarified, and the inter-relationships made explicit. The paper begins with an initial comparison between the understanding of features in product family design and classical product engineering. The research questions we address in the paper are:

- RQ1: What are the most relevant ways to understand the notion of feature in classical product engineering and in product engineering with variants?
- RQ2: How are feature notions understood to model product variability?

To answer these research questions, we introduce in Section 2 the concept of Feature Modelling as a technique to manage the variability of product families and provide a brief overview on feature-based approaches for classical engineering. Section 3 presents a literature review on feature definitions and the basic framework for the analysis of feature definitions. Section 4 discusses an initial proposal towards the use of this framework, in particular relatively of feature-based models in knowledge-based computer systems. The paper ends with some clarifications of the results obtained up to this point and an indication of further steps.

2. Background

2.1. Variability modelling

The management of variability has seen great developments in the last two decades, first in the design of software systems, second in the construction of complex systems such as CPS. Building software-based products' variants for mass customization means to employ the concept of *managed variability* for modelling the Commonalities and Variabilities (C&V) of product's variants within the same product family [5].

Following [7], a product family is a set of similar products which are derived from a common platform[2] and possess specific features/functionalities to meet customers' requirements. Each individual product within a product family, i.e., a family's member, is called *product variant* or *instance*. While a product family targets a certain market segment, each product variant is developed to address a specific subset of the identified customers' needs.

---

[2] A product platform is "a set of subsystems and interfaces developed to form a common structure from which a stream of derivative products can be efficiently developed and produced" [7]. Another interesting definition, in this case from SPLE, states that reusable platform is the result of the Domain Engineering process and defines the C&V of the product line.

Different techniques for managing variability have been developed to assist engineers. Some of them were proposed in the context of SPLE, while others came from complex systems engineering. Feature Modeling (FM), Decision Modeling (DM) [8], Function-Means (F-M) and Configurable Components Modeling (CC) are some of these techniques. Given the scope of the paper and the space available, here we only consider the FM approach.

2.2. Feature Modeling for product family design

Feature modelling (FM) was initially developed in the context of SPLE [9] to describe the C&V between individual hardware/software systems. In models developed within the scope of FM, called *feature models*, the C&V are modeled in terms of products' features, which are understood as "stakeholder visible characteristics of products" [10]. According to [10], FM has been widely exploited in the software engineering community, because of the capability of *feature models* to effectively support communication among the diverse stakeholders of a product line.

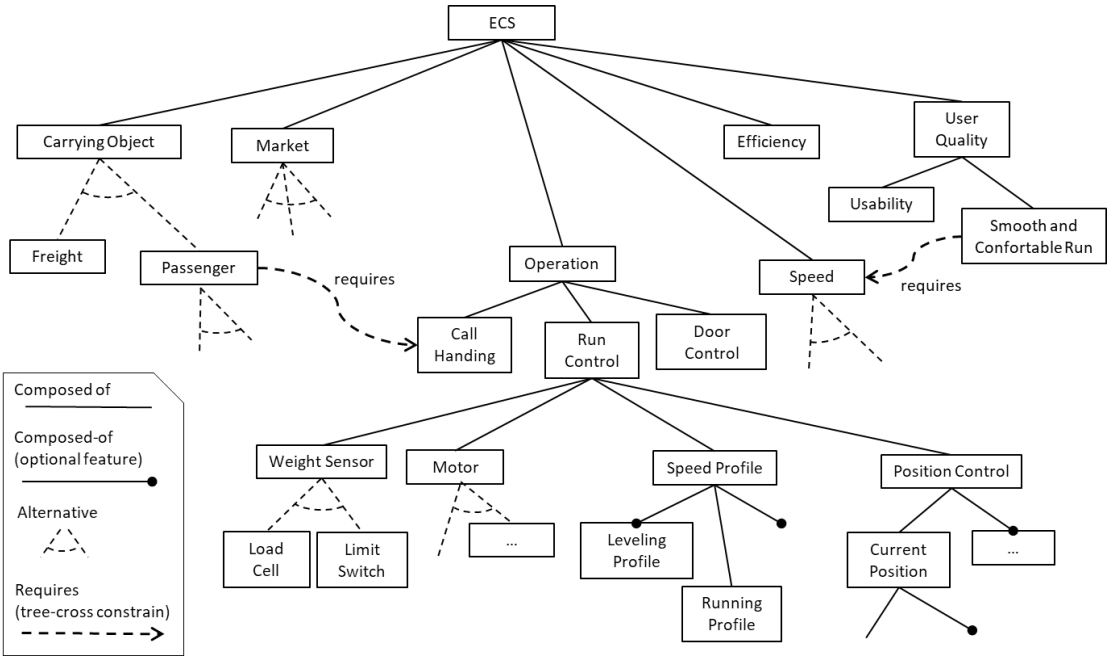

Fig 1. Feature diagram for Elevator Control System (ECS), based on [11].

Feature models consist of graphs of hierarchically arranged sets of features and are composed by: 1) the features themselves as nodes of the graph; 2) relationships between a parent (or compound) feature and its sub-features which can be labelled *mandatory*, *optional*, or *alternative* allowing in this way for *and*- and *or*-decomposition; and 3) cross-tree constraints, which are typically *inclusion* or *exclusion* statements. Feature models are often used to represent different stakeholders' areas of interest (concerns) in the same product line. For example, the features included in Fig. 1 identify different stakeholders' concerns about an Elevator Control Subsystem (ECS) associated to an Elevator System product line. The root node, named *ECS,* corresponds to an all-embracing abstract feature[3] that gathers all the characteristics for the achievement of the goal *Moving objects between different floors of a building vertically*. In order to concretize this feature, other abstract features are included in the model to represent the needed characteristics of the product. Note that different relations are used. Some relations show alternative features such as *Passengers* or *Freights*, which are alternative specializations of the *Carrying Object* feature. Other

[3] An abstract feature represents a variation point in a feature model that is not an eligible option (variant) for implementation.

relations express mandatory features that *must* be present in the product. For example, *User Quality* always involves the features *Usability*, and *Smooth and Comfortable Run*. Differently, optional features may not be present in the final product. For example, *Speed Profile* always involves a *Running Profile* to operate the elevator (mandatory feature) while the *Leveling Profile* is optional for enhancing the elevator stop position. The diagram also shows cross-tree constraints (labeled *requires* in Fig. 1) relating features across different branches. For example, whenever the *Smooth and Comfortable Run* feature is used, then a certain *Speed* feature has to be included in the model. *Require*-relations establish therefore dependencies between multiple features.

2.3. Feature modelling in classical product engineering.

In classical product engineering, mainly based on mechanical technologies and on single products defined without the support of variability models, the use of features has been fundamental to manage design or manufacturing alternatives. In a seminal paper on Feature technology [2], the notion of feature is defined as a prominent characteristic of an entity that enables persons or intelligent systems to distinguish between various similar elements (systems, artifacts, processes, etc.). Furthermore, in the classical product engineering scope, the variety of features is also significant.

Feature-based design has had more influence on the detailed design and process planning activities of product development processes. Currently, feature-based CAx systems are considered the state-of-art technology for product modeling [10, 12]. At first, features were used to easy the modeling of similar components by developing CAx libraries of pre-defined elements (*application feature*), but their potential to integrate multiple CAx applications was soon recognized. From this perspective, a feature, like a hole or a pocket, is an engineering meaningful set of related surfaces in a computer model associated to some parameters. Later, features were used to anchor other kinds of qualitative information useful for modeling tasks including non-geometric (product) properties. By developing further this view, today's features also include qualitative characteristics like colors.

Furthermore, feature technology can be useful for conceptual or pre-embodiment design activities, although this usage is not as mature as that for embodiment and detail design. Recently, some contributions have tried to link the most conceptual and declarative feature models with the current procedural CAD feature models. Among these proposals one finds the Functional Feature [13], which addresses the gaps between functional and geometrical representations of the design models.

3. Literature Review

The study presented in this section is part of a systematic review of the literature that takes into account different dimensions. The study started with the search of previous researches and publications works that constitute the state of art in the use and understanding of features, and that propose interpretations and classifications of this notion. Section 3.1 reports an extract of this review. The aim of this work is to identify different comparison criteria and to generate a comparative framework which we briefly describe in Section 3.2. This framework is used in Section 3.3 to analyze the collected feature definitions and to identify the problems that prevent the feature from being an integrating element.

3.1. Previous Researches

Several researches have gathered different definitions of feature and how it has been understood in engineering. Focusing on classical product engineering, the review on feature-based design in CAx tools presented in [12], collects numerous definitions for the 'feature' term; some definitions are limited to a specific domain, while others are very generic. The review also groups features into three groups: a) features used in the conceptual and configuration design phases; b) features related only to detail-design tasks (parametric design phase); and c) features linked to Design for X (DFx) tasks. In a more recent review of feature-based modeling approaches [6] the authors report a great number of feature definitions. In this case they introduce an analysis driven by theoretical insights and formal approaches in ontology engineering and propose the high-level distinction between *physical feature* and *information feature* to differentiate between the features of physical products and the feature specifications (e.g.

modeled as CAD features) that the former are meant to satisfy, respectively. Other research works have focused on the notion of functional features [14], which differently from other feature notions concern function, purpose or behavior.

In Domain Engineering (SPLE) several research works have analyzed different feature definitions as well. In [15], a work limited to Requirement Engineering (RE), the authors analyze a large set of definitions and conclude that some feature definitions mix to a varying degree elements belonging to different types of concerns. Another publication [16] focuses on feature-oriented software development, ordering the definitions from abstract to technical, where the last definitions describe features as design decisions and implementation-level concepts.

To assess the great diversity of feature types, the work presented in [4] is also relevant, as is focuses on complex systems and includes hardware and software constituents such as CPS. The authors identify two categories of system-level features, namely: a) the paradigmatic system features (PSF), that are generic and abstract and do not have explicit relations with engineering realization of CPS; and b) system manifestation features (SMF), which exist only in a particular implementation of a CPS and can be used to determine the overall composition or makeup of a system as well as the specific physical and visual traits of a system/component.

3.2. Comparative framework

Based on the works previously discussed, the comparative framework here proposed is based on two analysis criteria (cf. Fig. 2): 1) the information specified by the feature (Contained Descriptions) and 2) the granularity level. The first one is used in [15] to analyze feature definitions in SPLE according to the dimensions proposed in the Jackson-Zave framework [17] for RE, namely, Requirements (R), Domain knowledge (W) and Specifications (S). The Design (D) dimension is added by [15] to consider the role of features in software implementation stages. The second criterion considers the level of granularity in the hierarchical structure of the product implementation according to element containment relations.

The Jackson-Zave framework, which is widely accepted by the RE community with some small changes [18], covers all types of basic concerns that stakeholders communicate during RE processes. This is done by grouping them around three kinds of concepts (with the corresponding informational artifacts obtained in each process phase) involved in the communication act [19]: (i) statements about the domain describing properties that are true (assumptions taken about the environment) regardless of the presence or actions of the product at stake (W); (ii) statements about requirements, describing properties that the users want to be true of the world in the presence of the product (R); and (iii) statements about the specification describing what the product needs to do in order to achieve the requirements (S).

Once the distinction between R, W and S was formalized, Zave and Jackson established a statement that relates the three concepts and "suggests that the requirements problem amounts to finding the specification S that for given domain assumptions W satisfies the given requirements R". From this perspective, desires give "requirements" and beliefs "domain assumptions", intentions give "specifications".

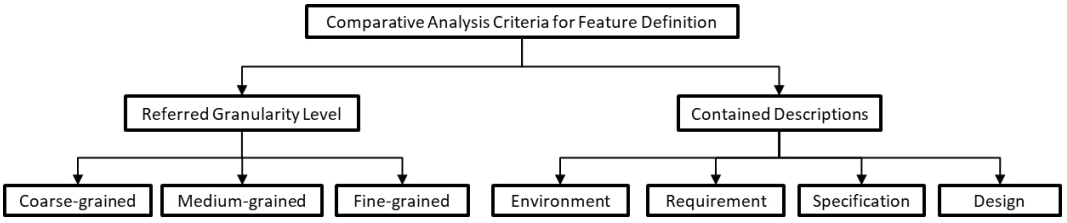

Fig 2. Feature definition comparative framework.

As indicated, the granularity level criterion is related to the structure of the product implementation (a program or a software system, an electromechanical assembly, an embedded system, etc.) to which a feature refers. This structure shows the hierarchical product decomposition from different points of view (logical, functional, physical, etc.) and three granularity levels are generally distinguished: a) coarse-grained; b) medium-grained; and c) fine-

grained. For example, in the software scope a feature can involve the addition of a new class, a new member to an existing class or the entire application. In a similar way, in the electromechanical scope, feature granularity varies from features affecting a system or system of systems to the more traditional features affecting single parts.

3.3. Feature definitions review and analysis.

Feature concepts can be arranged in two main groups: features focused on embodiment and detailed design in classical engineering (mechanical) and features focused on the software product line design. The first group covers the earlier works addressing the feature concept. In the second group, initially focused on software products, features are (mainly) used to model product variability to adapt products to customers' requirements.

Table 1. Feature definitions analysis (R-requirement, W-Domain knowledge, S-Specification, D-Design, C – coarse granularity, M – Medium granularity, F – Fine granularity, A – Any granularity)

Reference	Definition	Contained Descriptions				Gran. Level
		R	W	S	D	
	Single Product Engineering					
Shah and Mantyla [20]	a feature represents the engineering meaning or significance of the geometry of a part or assembly, which could serve as building blocks for product definition or geometric reasoning		X		X	M, F
Cheng and Ma, [13]	functional features are those features that generically integrate the functional design intent, engineering physics, and product geometric model in a consistent functional-physical modelling approach, enabling a functional-centric design		X	X	X	A
Shah [through 12]	recurring patterns of information related to a part description				X	F
Pourtalebi and Horvarth [through 6]	looking at features as complex properties of engineering systems		X	X	X	A
Bidarra [21]	a representation of shape aspects of a product that are mappable to a generic shape and functionally significant for some product life-cycle phase			X	X	F
Brown [14]	anything about the thing being designed that's from interest	X	X	X	X	A
Kün et al. [22]	As abstractions or groupings of requirements describing structural, behavioural or functional properties of a system that are relevant and understandable for different stakeholders	X		X		A
	Software Product Line Engineering					
Kang et al. [through 16]	a distinctively identifiable functional abstraction that must be implemented, tested, delivered, and maintained	X		X	X	A
Czarnecki and Eisenecker [through 16]	a distinguishable characteristic of a concept (e.g., system, component, and so on) that is relevant to some stakeholder of the concept	X	X	X	X	A
Chen et al. [through 16]	a product characteristic from user or customer views, which essentially consists of a cohesive set of individual requirements	X				C
Zave [through 16]	an optional or incremental unit of functionality			X	X	M, F
Apel et al., [23]	a characteristic or end-user-visible behavior of a software system. Features are used in product-line engineering to specify and communicate commonalities and differences of the products between stakeholders, and to guide structure, reuse, and variation across all phases of the software life cycle	X	X	X	X	A
Classen, [15]	a feature is a triplet (R, W, S), where a given (domain) assumptions W and specifications S guarantees that requirements R are met	X	X	X		A

Table 1 shows that in product design there are both abstract definitions, either generic or related to R, and definitions that have a more technical flavour related to S and/or D. However, the majority of definitions in the

SPLE scope are abstract, while the technical definitions, which are only linked to Feature-based Programming, are a minority. This difference is due to the fact that features in classical engineering have been developed with a finer granularity with respect to geometric specifications.

Therefore, it can be concluded that the tension between abstract and technical (or implementation) views and between different granularity levels is present in both scopes. The same thing happens in the engineering context R, W, S or D in which a feature is created. This is the subject of the following section because, differently from the SPLE scope, the analysis based on this criterion has not been done.

4. Discussion

The analysis of feature definitions presented above may need further development, but it can already be used in design methodologies and software applications for engineering. In our view, it is valuable to qualify features relatively to their design context (i.e., R, S, D).[4]

First of all, the design context of features gives further information – in the scope of engineering specifications – about *the intended semantic of features*, therefore it helps to disambiguate their meanings. For instance, consider a feature, call it *f*, about the red-color of a component forming the structure of the designed product. By simply looking at a design specification like a CAD model, it may not be clear *why* red was chosen as color attribute of the component; this information is indeed only in the designer's mind. However, knowing the context from which *f* originated, its intended semantics become clearer and can be shared with the other stakeholders. E.g., should *f* originate from customers' requirements, then it may carry an *aesthetic* meaning; should it originate from the development of design specifications matching R, it may carry an *interfacing* meaning (like when emergency buttons are red to be easily identifiable in emergency situations).

Second, design context of features serves *to support the integrated validation of features and the design models* that include them. This can be understood in at least two different ways. First, to validate the compliance of S-features with respect to R-features, and of D-features to S-features. The purpose is to guarantee that, once customers' requirements are collected, they are further considered in successive designing phases. For instance, by knowing that an S-feature is developed to comply with a R-feature, the compliance of the former with the latter has to be guaranteed and preserved along the entire design process. The same consideration is valid between S-features and the D-features that implement (embody) them. Also, since the former are meant to comply with certain R-features, it should be guaranteed that the D-features match with such R-features, too, since compliance with requirements should propagate from S-features to D-features. Clearly, design errors are possible, but a "good design" has to guarantee these sorts of compliance matches.

Third, the design context of features helps *to validate features with respect to the domain knowledge* (W). For instance, a R-feature may describe a product made of a certain material type, according to a customer's wishes. However, the R-feature may *partially* or even *entirely* conflict with domain knowledge, e.g., because the customer is not aware that the chosen material type is not suitable for the ordered product. A similar conflict may occur between domain knowledge and S-features or D-features. The consequences of this conflict can be much more serious than the R/W-feature conflict, since the functionality of the product and the safety of its users can be compromised (e.g., an elevator may crash because its designers did not properly consider the maximum load it is capable to support).

From an application perspective, the ideas just presented may be implemented in knowledge-based software applications to support design. For instance, semantic annotations may be introduced in (geometric) design models to enrich features with (non-geometric) information concerning both their originating contexts and their intended meanings. Also, if this information is formally specified in computational terms, e.g., by means of a *computational ontology*, then the dependencies and inter-relations between multiple features can be automatically checked to support the validation of features compliance. In addition, the use of an ontology can help to validate features and design models against experts' knowledge to avoid the proliferation of modeling mistakes. Similar ideas have been already explored in literature [24], even though current approaches do not rely on a principled classification of

[4] In this section we talk of S-features, R-features, and D-features as features stemming from the S, R, and D contexts, respectively.

features [6], nor on the inter-relationships between features across design contexts.

5. Conclusions

The paper makes a step toward a unified vision of feature notions in product engineering. It starts from an analysis of feature definitions proposed in SPLE and classical engineering. The analysis, further developed, can be: a) used to support the design of both product with variants and single products; b) linked to concepts of RE and the design and implementation of requirements; c) useful to design software application/systems, electromechanical systems or complex systems (embedded systems or CPS). Feature definitions have been here analyzed using the Zave-Jackson framework for RE, expanding it to make visible the presence of the feature concept in the design and implementation phases. Additionally, a second criterion has been considered to make visible the relationship between features and the different granularity levels at which they are developed.

This work is still preliminary but the future aim is to establish a paradigm for the Feature-oriented Development of Complex Systems, comparable to existing proposals in the software product scope [16]. Additional work on the analysis of feature classifications and robust cases studies is necessary, including the analysis of feature notions present in other variability modelling techniques such Function-Means and Configurable Component models.

References

[1] S. K. Chandrasegaran, K. Ramani, R. D. Sriram, I. Horváth, A. Bernard, R. F. Harik, W. Gao, The evolution, challenges, and future of knowledge representation in product, Computer-aided design 45 (2013) 204-228

[2] J. J. Shah, Designing with Parametric CAD: Classification and comparison of construction techniques, International Workshop on Geometric Modelling (1998) 53-68

[3] X. F. Zha, R. D. Sriram, Feature-based Component Model for Design of Embedded Systems, Intelligent Systems in Design and Manufacturing V 5605 (2004) 226-238.

[4] S. Pourtalebi, I. Horváth. Towards a methodology of system manifestation features-based pre-embodiment design. *Journal of Engineering Design* 27 (2016) 232-268

[5] K. Pohl, G. Böckle, F. van der Linden, Software product line engineering: foundations, principles and techniques, Springer Science & Business Media (2005)

[6] E. M. Sanfilippo, S. Borgo, What are features? An ontology-based review of the literature, Comput Aided Design, 80 (2016) 9-18.

[7] J. R. Jiao, T. W. Simpson, Z. Siddique, Product family design and platform-based product development: a state-of-the-art review, Journal of intelligent Manufacturing 18 (2007) 5-29

[8] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid, A. Wąsowski, Cool features and tough decisions: a comparison of variability modeling approaches, Proceedings of the sixth international workshop on variability modeling of software-intensive systems (2012) 173-182

[9] M. Riebisch, Towards a more precise definition of feature models. *Modelling Variability for Object-Oriented Product Lines*, (2003) 64-76.

[10] K. C. Kang, H. Lee, Variability Modeling, Systems and Software Variability Management, (2013) 25-42

[11] K. Lee, K. C. Kang, Usage context as key driver for feature selection. In International Conference on Software Product Lines (2010) 32-46.

[12] O.W. Salomons, F.J.A.M. van Houten, H.J.J. Kals, Review of research in Feature-based design, Journal of manufacturing systems 12 (1993) 113-132

[13] Z. Cheng, Y. Ma, Explicit function-based design modelling methodology with features, Journal of Engineering Design 28 (2017) 205-231.

[14] D. C. Brown, Functional, Behavioural and Structural Features, ASME 2003 International design engineering technical conferences and computers and information in engineering conference (2003) 895-900.

[15] A. Classen, P. Heymans, P.-Y. Schobbens. What's in a Feature: A Requirements Engineering Perspective,International Conference on Fundamental Approaches to Software Engineering, (2008) 16-30

[16] S. Apel, C. Kästner, An overview of feature-oriented software development, *Journal of Object Technology* 8 (2009) 49-84.

[17] C. A. Gunter, E. L. Gunter, M. Jackson, P. Zave, A reference model for requirements and specifications. *IEEE Software*, *17* (2000) 37-43

[18] I. J. Jureta, J. Mylopoulos, S. Faulkner, A core ontology for requirements. *Applied Ontology*, *4* (2009) 169-244.

[19] K. Pohl, G. Böckle, F. J. van Der Linden, F. J.*Software product line engineering: foundations, principles and techniques*. (2005) Springer Science & Business Media.

[20] J.J. Shah, M. Mäntylä, *Parametric and Feature-based CAD/CAM*, John Wiley & Sons: Toronto, Canada, (1995), ISBN 0-471-00214-3

[21] R. Bidarra, W. F. Bronswoort, Semantic feature modelling, Computer-Aided Design 32 (2002) 201-225.

[22] A. Kühn, C. Bremer, R. Dumitrescu, J. Gausemeier, Feature models supporting trade-off decisions in early mechatronic systems design, DS81 Proceeding of NordDesign, (2014)

[23] S. Apel, D. Batory, C. Kästner, G. Saake, *Feature-oriented software product lines*. (2016) Springer-Verlag Berlin An.

[24] M. Imran, B. Young, The application of common logic based formal ontologies to assembly knowledge sharing. Journal of intelligent manufacturing, 26 (2015) 139-158.