Towards a Smart Feature Model Evolution

Olfa Ferchichi[1]
[1] Laboratoire de Recherche en GénieLogiciel, Applications distribuées, Systèmes décisionnels et Imagerie intelligentes (RIADI), Université de Manouba
Tunisie
Email:olfaferchi@yahoo.fr

Raoudha Beltaifa[2]
[1] Laboratoire de Recherche en Génie Logiciel,Applications distribuées, Systèmes décisionnels et Imagerie intelligentes (RIADI), Université de Manouba
Tunisie
Email :raoudha.beltaifa@ensi.rnu.tn

Lamia Labed Jilani[3]
[1] Laboratoire de Recherche en Génie Logiciel, Applications distribuées, Systèmes décisionnels et Imagerie intelligentes (RIADI), Université de Manouba
Tunisie
Email : lamia.labed@isg.rnu.tn

Raúl Mazo[4]
Lab-STICC,
ENSTA Bretagne,
Brest, France.
GIDITIC, Universidad Eafit, Medellin - Colombia.
Email: raul.mazo@ensta-bretagne.fr

***Abstract*— With the proliferation of new technology platforms, new operational requirements, different contexts and so on, agility remains more and more solicitated for software evolution. For software evolution of Software Product Line Engineering (SPLE), the Feature Model (FM) is the basic instrument that supports the evolution of SPL at the variability level. We would like to improve FM diagrams to make them understandable during the evolution of the corresponding product lines. More precisely, FM evolution can become more systematic and more intelligent. In our work, we aim to evolve FMs by means of smart techniques. Hence, we represent feature models by an ontology. This latter will permit, among others, the inference of knowledge about the evolution of the FMs. By obtaining different versions of the FMs, these can be used as a learning base of a learning algorithm. So, for a given FM, a new version can be predicted as being an evolution version of the FM. In this paper, we present the FM metamodel extension necessary to represent the semantics of the evolution rules. Thus, with a model driven approach, FMs are transformed into FM ontologies. A running example about an Electric Brake Parking System extracted from the SPLOT repository is presented.**

***Keywords-** Software Product Line Engineering; Variability Modeling; Feature Models (FM); Feature Oriented Domain Analysis (FODA); non-functional features.*

I. Introduction

“A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way” [19]. In the SPLE approach, variability is seen as a key concept in its processes and artifacts, and it is usually defined in terms of features, variants, variation points (Variation Point (VP) and the relationships among them. According to Bosh, *“a feature is a logical unit of a behavior defined by a set of functional and non-functional requirements*” [17] while variants (VA) represent the different possibilities that exist to satisfy a PV [5] and [23]. Kang et al. [2] define a (VP) “*as being identification at one or more locations at which variation may occur”.* A FM is a tree with the root representing a concept, and its descendent nodes are features, see Figure 1 as an example. A FM is a compact representation of all possible products of an SPL. In the Feature Oriented Domain Analysis (FODA) [2], features can be mandatory or optional, and be related through choice (alternative or multiple), requires and excludes relationships. Feature models are feature diagrams plus additional information such as feature descriptions, binding times, priorities, stakeholders, and so forth. The purpose of using FM is to express the existing relationships between the different features of the product line. A FM is a tree– like structure and consists of: i) relations between a parent feature and its child features. ii) cross–tree constraints that are typically inclusion or exclusion statements of the form “if feature F is included, then feature X must also be included (or excluded)” [12].

Software product lines are long-lived systems that undergo significant evolution throughout their lifespan. This latter concerns domain engineering (development for reuse) and application engineering (development with reuse) processes. This evolution allows companies to align their products with new technological platforms, the evolution of commercial strategies, the emergence of new customers operational needs and new technological challenges in general. Therefore, a product line development process must make evolve the whole product line taking into account the changes at the Domain Engineering level. The evolution of domain assets, as for example the feature models, has received a great attention from researchers as it represents a key success aspect of SPLs. It does not only consist in adding, modifying or deleting features in the FM, but also adding semantics about the features' characteristics. Given a feature, it can represent a quality feature, a software feature, a structural feature, a hardware feature, etc. and can be in constantly evolution. Some studies propose to improve FM models [11] [20] [22] and [27] but despite these various attempts, the semantics extension of FMs remains limited and no promising approach has been proposed to develop FMs as part of a common evolution approach.

In our work, we aim to evolve FMs by means of smart techniques. In this paper, we present the FM metamodel extension necessary for representing semantics important for the evolution rules. Thus, with a model driven approach, FMs are transformed into FM ontologies. This is a first attempt to define a smart FM evolution approach in a knowledge-based framework. Thus, ontology of a FM will permit among others, the inferring of knowledge about FM evolution. Our running example of feature model is about an Electric Braking Parking system shown in Figure 1.

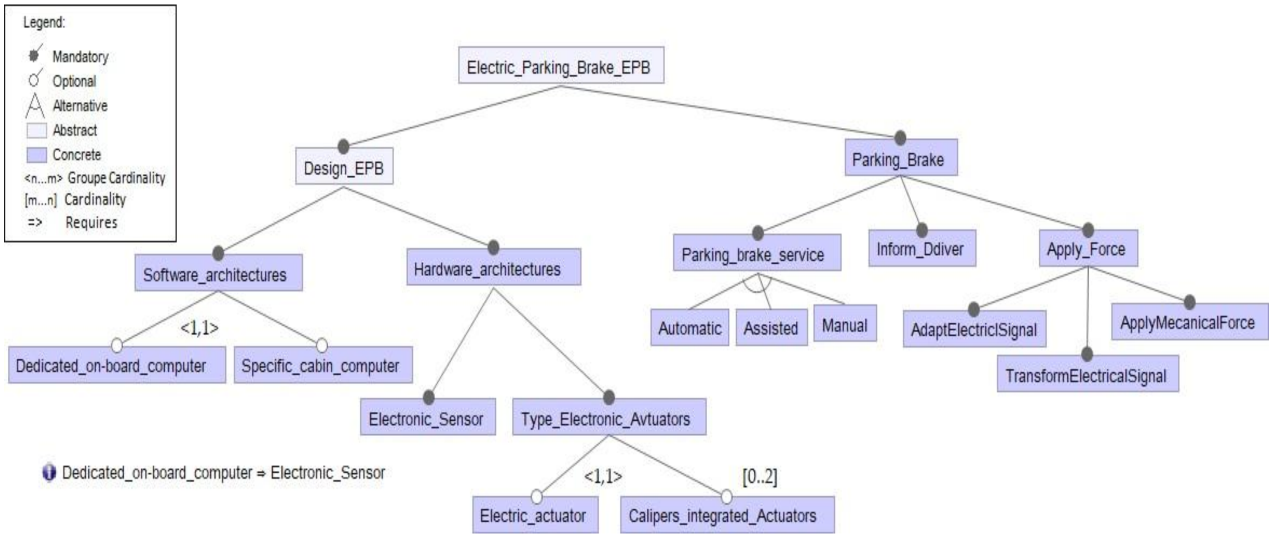

Figure 1: Electric Brake System FM [16]

The remainder of this paper is structured as follows: section 2 presents the related work of modeling variability with FM. Section highlights the issues treated in this paper. In section 4, the metamodel of an extended feature model (EVO-FM metamodel) is presented in the context of a model driven approach. It enriches the FM semantics in order to better handle evolution concerns. In the same section we present the transformation rules between the EVO-FM metamodel and the ontology metamodel to enable reasoning on the enriched models. Section 5 is devoted to present the implementation of the proposed approach under the Eclipse Framework. This operational aspect serves as a proof of concept. A conclusion summarizes the work and presents future workas perspectives in section 6.

II. Related works and issues

Concerning the literature review on modeling feature models, several works have been done for making improvements and extensions to FM. The variability in the product family is represented by feature cardinality [1][2][5][14][16] and [25], a cardinality group of features [7][14][16][17][23] and [25], cardinality-Based feature models with constraints and feature attributes [17]. In our case, we also make extensions to FM for enriching its semantic. This latter is essential for evolution rules. Bhushan and al. [27] present the managing of Software Product Line using an Ontological Rule-Based Framework. Nieke and al [28] provide an ontology to check FM evolution. This latter is defined by feature models supporting temporal concepts. Rincón, Giraldo et al [29] propose an ontological rule-based approach to analyze dead and false optional features in FM as well as identifying certain causes of these defects, and explaining these causes in natural language. Our approach is also based on an ontology but it provides more semantic to FM features and relationships. In our work, we have temporal evolution rules. Feature modeling is the most popular technique to represent domain requirements variability in SPLs.
However, FMs have several limitations related to the lack of means to represent explicitly the semantics of features and their relationships. In fact, it needs improvements to provide semantics to its components for dealing with agility.

In order to better understand the problem, we present in the sequel, some limitations of feature models [14] and [21].

– Lack of distinction between behavioral and structural features. In the running example presented in Figure 1, we need to precise that “electronic_Sensor” and “Type_Electronic_Actuators” are structural features, but “adaptElectricSignal”, “TransforElectricSignal” and “ApplyMecanicalForce” are behavioral features. Evolution rules can need the feature semantics in order to decide how to make changes in the FM.

– Evolutions of features in time and in space are not expressed. Variants of a feature may represent its evolution in time or space. For instance, in Figure 1, “Manual”, “Assisted” and “Automatic” features represent the evolution of the “Parking_Brake_Service” feature.

- Features such as quality attributes are rarely specified in feature models and their variability is neglected. QoS feature can have different attributes such as response time, availability, reliability, throughput, etc. For a given product line, quality attributes can change during time so evolving from one kind to another.

– Distinguishing the nature of each feature; for specifying software concerns. This kind of information can be helpful for expressing evolution rules and/or inferring knowledge about evolution.

– Dependencies between a parent feature (variation point) and its children features (variants) should be more precise. For instance, a (VP) can represent an aggregate feature or a super-feature. To select features correctly, the semantics of these relationships have to be defined explicitly. In the running example, the “adaptElectricSignal”, “TransforElectricSignal” and “ApplyMecanicalForce” features are mandatory regarding their corresponding father (i.e., “apply_force”). It is clear that “apply_force” aggregates these three variants; however, this information is not explicitly represented in the feature model and we consider that this information is useful to represent evolution rules.

III. MODEL DRIVEN APPROACH FOR EVO-FM CONSTRUCTION

EVO-FM is a feature model that represents knowledge and information for its evolution. Thus, EVO-FM is a feature model enriched with some concepts to support its evolution. In order to construct the EVO-FM, we adopt a model driven approach as shown in Figure 3. Hence, EVO-FM metamodel be transformed into an ontology metamodel supporting semantics information and allowing intelligent reasoning. In the sequel, we first recall some aspects of the model driven approach. Then, we present the EVO-FM metamodel conforms to ecore meta metamodel. This is done in the syntactic domain level and then as we will see in the next section, the Eclipse environment offers an ATL transformation facility to defined the transformation rules between EVO-FM metamodel and the ontology metamodel (semantic domain) [15].

A. *Modeling and metamodeling*

A well-defined language is a language with well-defined form (syntax), and meaning (semantics), which is suitable for automated interpretation by a computer. For a model to be useful, OMG [9] and [13], recommends that: "*A model needs to be expressed in a way that communicates information about a system among involved stakeholders that can be correctly interpreted by the stakeholders and supporting technologies. This requires the model to be expressed in a language understood by these stakeholders and their supporting technologies*."[8].

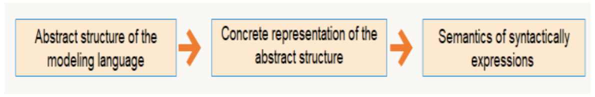
Abstract structure of the modeling language
Concrete representation of the abstract structure
Semantics of syntactically expressions

Figure 2: Meta model, model and concrete model

As illustrated in Figure 2, a modeling language is defined by an abstract structure, a concrete model and a model representing a real case enriched with additional semantics, which is in fact an instance of the concrete model [16].

B. Model transformation

Another key activity of model driven engineering [10] is the concept of model transformation, which is the automatic process to transform a source model into a target model. According to Bézivin et al [9] transformation is done by a collection of transformation rules that are "a description of how one or more constructs in the source language can be transformed into one or more constructs in the target language". More precisely, there are different levels of transformations as illustrated in Figure 3. Once the FM metamodel is created, a transformation to the FM ontology metamodel [15] is done. The extended metamodel (level 2) is consistent to the corresponding meta metamodel (level 3) and the FM model (level 1) is consistent to its corresponding metamodel (level 2). The transformations can be done horizontally in each level. The final real case ontology (level 0) can be obtained by instantiation of the meta ontology in OWL language and SWLR for the reasoning rules to be added manually [15]. The meta-model specification uses the TOPCASED ECORE editor. We checked the correctness of that transformation by verifying that each item in the source model has its corresponding item(s) in the target model. These are some transformation rules :

```
-- @path MM=/ATL_FM/Papier.ecore
--@path MM1=file:/C:/Users/HP/Desktop/OCTA_2019
/onto_FM/Ontology_FM.owl
module FM;
create OUT : MM1 from IN : MM;
rule Concept_Feature { from f : Feature!Feature
to out : Feature!Feature (
name <- f.name,
FeatureID <- f.FeatureID()
)}
rule Concept_Hardware_Feature{ from b : Feature!Feature
to out : Hardware_Feature!Hardware_Feature ()}
```

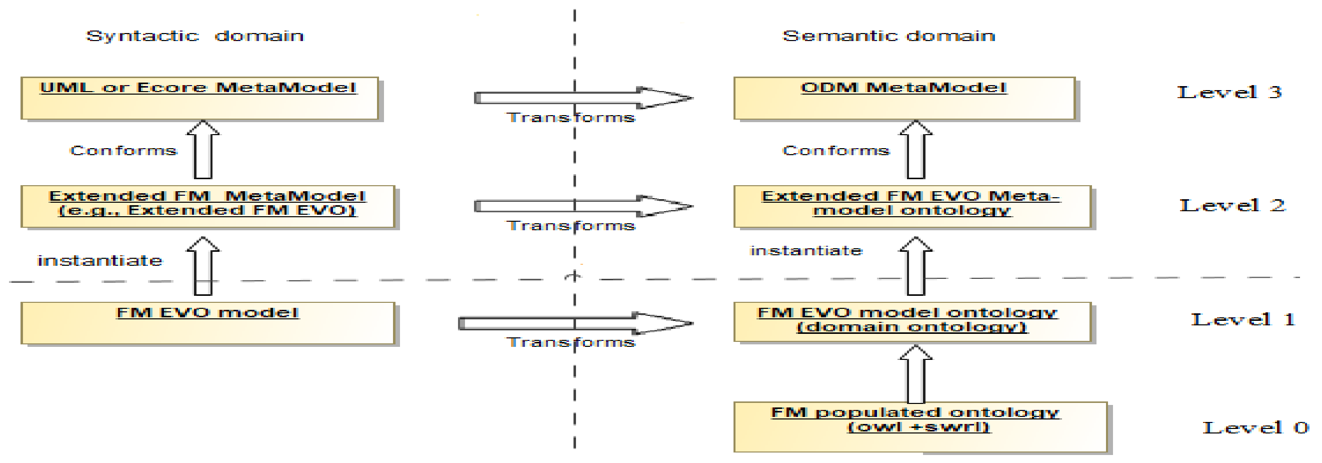

Figure 3 : Model-driven approach for EVO-FM construction

C. EVO-FM: Extended Feature Model Meta-Model

To help evolving feature models, we extend the FM metamodel using the UML notation [10] and [13]. In particular, we improved FM with three aspects: 1) the semantics of the features so that each feature of a FM can be characterized in one of the following four categories: software, hardware, structure and behavior, 2) the possibility to represent non-functional requirements (i.e., security, performance and accuracy) in the features of each FM and 3) the semantics of three new relationships: “compose”, “aggregate” and “is_a”. The concepts we integrate into the FM language are represented by the colored boxes in Figure 5. A non-functional feature represents the quality that the product family must have in order to meet the requested needs. We use the notion of stereotype [1] to specify the new concepts to the extended FODA language. Creating a non- functional feature should add the «stereotype_name» to the feature. For example, a feature that represents security will be stereotyped «Security» and similarly for the «Structural», «Behavioral», «Hardware» and «Software» features. The sample model shown in Figure 4 shows how to use the new relationships and quality requirements in our Electric Brake Parking System example.

IV. IMPLEMENTATION FRAMEWORK

In order to be able to construct EVO-FM, we implement ourmodel driven approach with the Eclipse family of integrated development environments, FeatureIDE and Xtext that we present in the sequel..The Eclipse community, withsupport from the Eclipse Foundation, provides integrated development environments (IDEs) targeting different developer profiles. It is a framework and code generation facility for building Java applications based on simple model definitions. Among the development tools provided by the IDE we used three complementary Eclipse frameworks: EMF, Xtext and FeatureIDE..

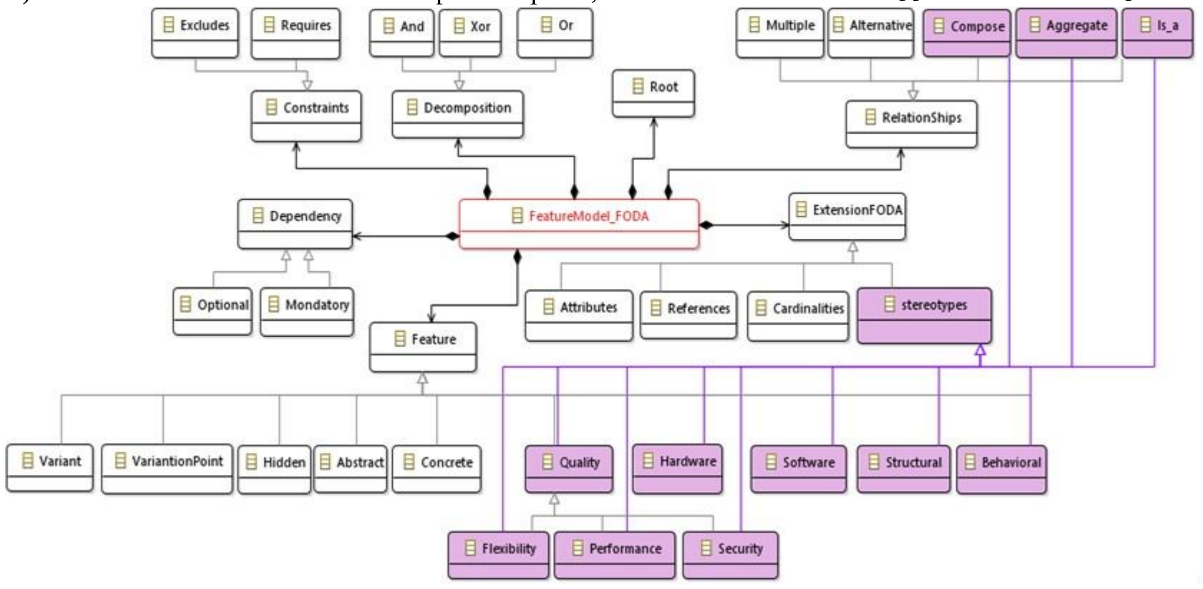

Figure 4: EVO-FM extended Feature model

The Eclipse Modeling Framework (EMF) [6] provides a modeling and code generation framework for Eclipse applications based on structured data models. Although EMF supports the key MDA concept of using models as input to development and integration tools, it does not use however any one of the MOF compliance points previously described. Instead, EMF uses ECore, a not fully alignedvariant of OMG's EMOF. Essentially, among other elements, an ecore meta-model allows to define an EClass, an EAttribute, an EReference. To overcome this issue, EMF supplies model transformation and grammar capabilities. Model transformation can be model-to text (M2T) by the Textual Modeling Framework (TMF) [26]. It is an EMP's project aiming to support the development of textual concrete syntax. TMF is based on a meta-model and syntax specification, offering several functionalities that include a parser that reads the textual representation of the model and instantiates the corresponding EMF model, an eclipse text editor that supports syntax highlighting, code completion, navigation, and other features.

Xtext is a framework for development of programming languages and domain- specific languages. With Xtext you define your language using a powerful grammar language. With Xtext, we define grammars that implement stereotyped non-functional features.

FeatureIDE [24] and [25] is an Eclipse-based IDE that supports all phases of feature-oriented software development for the development of SPLs: domain analysis, domain design, domain implementation, requirements analysis, softwaregeneration, and quality assurance. Different SPL *implementation* techniques are integrated such as feature- oriented programming (FOP), aspect-oriented programming (AOP), preprocessors, and plug-ins. Full infrastructure, including parser, linker, type checker, compiler as ell as editing support for Eclipse, any editor that supports in [15] shows the ontology that we obtained for our running example.

V. CONCLUSION

In this paper, we have proposed EVO-FM, an extension to the FODA metamodel, which enriches it with knowledge and information to support the evolution phase of the models created with this engineering language. In particular, we enrich the feature models with quality and semantic features. Hence, we also add support for new types of feature relationships and extensions with stereotypes. We have adopted a model driven approach for constructing EVO-FM with also the possibility to transform them to ontologies that enforce the feature model semantics and intelligence by inferring new information. The obtained ontologies are not just enriched with SWRL rules for checking the EVO-FM consistency but also with the mechanisms to run the evolution rules [15]. To implement our approach, we advocate the adoption of metamodeling tools such as Eclipse modeling Framework and Xtext. Thus, the main contributions of our approach are:

- Add semantic features in the form of stereotypes such as «software», «hardware», «structural» and «Quality»
- Add quality features in the form of stereotypes such as «security», «Performance», «Flexibility».
- Import a textual specification into grammars using the Xtext framework to process it and transform it into XML;
- Import the EVO-FM in XMI, XML and Java format and soenhance reuse of it.

The main perspectives of our work are: 1) apply our model driven approach to different system families and validate the evolution rules that are behind the EVO-FM ontology 2) enable a smarter evolution of feature models by using different versions of EVO-FM feature model, these can be used as a learning base of a learning algorithm. So, for a given EVO-FM, a new version can be predicted as being a new FM evolution version. More specifically, suppose that we have the trace (history) of

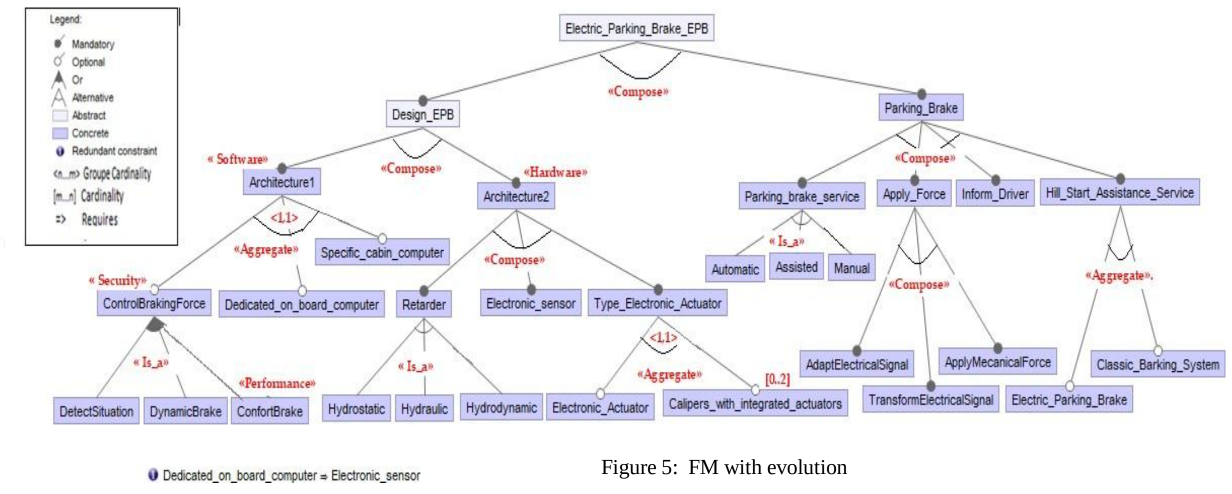

Figure 5: FM with evolution

the 50th previous versions of the EBP where the feature "Type-electronic-actuator" feature was "electric-actuator" in 10th first versions and after the 11th version becomes always "calipers_integrated_actuator" so the Type-electronic-actuator feature will evolve to become mandatory "calipers_integrated_actuator" instead of having two optional features. Also, we can have an aggregation of features that evolve to a composition of features because the learning algorithm find that after a certain number of FM versions, this happens. A quality feature can also evolve from performance to agility because this was learned from previous versions. We would like to investigate these concerns in the near future.

REFERENCES

[1] H. Papajewskiand D. Beuche and W. Schroder-Preikschat. Variability management with feature models. Science of Computer Programming, December 2004, vol. 53, no 3, pages333{352, 2004..

[2] K-C. Kang, S-G. Cohen, J-A. Hess, W-E. Novak, and A-S. Peterson,"Feature Oriented Domain Analysis FODA Feasibility Study", Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh,PA, USA, November 1990.

[3] J. Christophe, " The Reuse and Variability software product lines ", Periodic publication smals,Junary 2009

[4] N. Noda, and T. Kishi,"Aspect oriented Modeling for Variability Management", 12th International Software Product Line Conference, September 2008.

[5] B. Frank, E.Raymond, S.David G.Timothy M.Ed .Eclipse modeling framework: a developer's guide. Addison-Wesley Professional. December 2004

[6] A. Classen, P. Heymans, and P. Schobbens,"What's in a Feature: A Requirements Engineering Perspective", In J. Fiadeiro and P. Inverardi (Eds.): FASE 2008, LNCS 4961 .Proceedings of the 11th International Conference on Fundamental Approaches to Software Engineering, held as part of ETAPS, Budapest, Hungary, pp.4-- 30,April, 2008.

[7] L. Moigne, Jean-Louis. Modeling of complex systems. Paris: Bordas,1990

[8] J Bézivin, O Gerbé. Towards a precise definition of the OMG/MDA framework. Proceedings 16th Annual International Conference on Automated Software Engineering,pp 273—280, IEEE , November 2001.

[9] OMG, "Unified Modeling Language TM (OMG UML)",Superstructure ,2011.

[10] M-A. Laguna, B. González-Baixauli, J-M. Marqués, and R. Fernándezet,"Feature Patterns and Multi Paradigm Variability Models",GIRO Technical Report 2008/01, v0.91, 2008.

[11] E-A.Oliveira, I-M. Gimenes, E. Hatsue, and M. Huzita, "A Variability Management Process for Software Product Lines",In Proceedings conference of the Centre for Advanced Studies on Collaborative research , IBM Press, pp.225-241,October 2005

[12] OMG Unified Modeling Language (OMG UML, OMG Document Number:formal/2007-11-04Standard document:URL:http://www.omg.org/spec/UML/2.1.2/Infra structure V2.1.2/PDF, Associated Schema Files.

[13] R. Mazo, Advantages and limitations of feature models in modeling variability requirements., Software engineering NO 11, Paris France, January 2015

[14] O. Ferchichi. R.beltaifa , L.Elabed. , *An ontological Rule-Based Approach for Software Product Lines Evolution.* International Multi-Conference on: "Organization of Knowledge anddvanced Technologies" (OCTA), IEEE, Tunisie. September 2020

[15] http://www.splot-research.org/Decenber 2010,

[16] J. Bosch, G. Florijn, D. Greefhorst, and J. Kuusela, "Variability issues in software product lines", In Software Product-Family Engineering (pp. 13-21). Springer Berlin Heidelberg,pp. 13-21, April 2001.

[17] D. Beuche, H. Papajewskiand, and W. Schroder-Preikschat, "Variability management with feature models", Science of Computer Programming, vol. 53, no 3, pp.333—352, 2004.

[18] P. Clements and L. Northrop. Software Product Lines: Practices and Patterns. SEI series in software engineering. Addison- esley,2002.

[19] D. Benavides, P.Trinidad, and A. Ruiz-Cortes, "Using Constraint Programming to Reason on Feature Models", In The Seventeenth International Conference on Software Engineering and Knowledge Engineering (SEKE05), Juillet, 2005.

[20] J. Christophe, " Reuse and variability software product lines ", Publication périodique smals , March 2009.

[21] D. Benavides, S. Segura, P. Trinidad, and A. Ruiz-Cortes," Using Java CSP Solvers in the Automated Analyses of Feature Models", LNCS, to be asigned, TIC2003-02737-C02-01 (AgilWeb), January 2006.

[22] D.Benavides, "Automated Reasoning on Feature Models",In The 5th Conference on Advanced Information Systems Engineering (CAiSE05), LNCS, 3520:491503,Juin, 2005.

[23] T. Kastner, C.Benduhn, *et al.* FeatureIDE: An extensible framework for feature-oriented software development. *Science of Computer Programming*, vol. 79, p. 70-85. January 2014.

[24] K. Christian, T. Thomas, S.Gunter, *et al.* FeatureIDE: A tool framework for feature-oriented software development .

[25] W .Edward D aniel. "Re-en gineering eclipse MDT/OCL for xtext." *Electronic Communications of the EASST* 36 . March 2011.

[26] K. Czarnecki,"Generative Programming",Principals and Techniques of Software Engineering Based on Automated Configuration and Fragment Based Component Models, Tools, and Applications, Addison Wesley,ISBN 0201309777, Mai, 2000.

[27] M. Bhushan, S. Goel, A. Kumar and A. Negi. Managing Software Product Line using an Ontological Rule-Based Framework. International Conference on Infocom Technologies and Unmanned Systems (2017).

[28] M. Nieke, C. Seidl, T. Thum. ,Back to the future: avoiding paradoxes in feature-model evolution, SPLC (2), 2018.

[29] L. Rincón., G. Giraldo, R .Mazo and C. Salinesi. An ontological rule-based approach for analyzing dead and false optional features in feature models. Electronic notes in theoretical computer science, 302(0): 111 – 132. 2013.