ISSN 22773061

Labeled UML model fragments composition by the SPL strategy

Rim Bouhaouel , Naoufel Kraïem , Zuhoor Al-Khanjari
RIADI Lab, ENSI, Compus of Manouba, Tunisia
bouhaouel.rim@gmail.com
RIADI Lab, ENSI, Compus of Manouba, Tunisia
naoufel.kraiem@ensi.rnu.tn
Department of CS, Sultan Qaboos University, Oman.
zuhoor@squ.edu.om

ABSTRACT

The software community intends to make use of a standard approach for the software development to not build software product from scratch. This approach ensures a high quality of software with a controllable cost. It affects the whole process of the software development, especially in the early phases e.g. analysis and design. One of the most widespread language to modulate and document those two stages is UML (Unified Modeling Language), but the reuse of the UML model is used in ad-hoc method so why do not build a systematic method for reusing some fragments of UML? To realize it, we need to adopt a reuse approach, so we choose the software product line (SPL), based in collecting variability of the domain (in our case is UML). In fact, UML and SPL have a common issue is the model driven engineering, since both of them based on the modeling approach. This paper overviews the different areas of UML and explains the process of software Product line with its born issues to wit: MDE (Model Driven Engineering) and MDA (Model Driven Architect).

Keywords

UML, Software Product Line Engineering, Model Driven Engineering, Model Driven Architect.

Academic Discipline And Sub-Disciplines

Computer Science

SUBJECT CLASSIFICATION

Software Engineering

TYPE (METHOD/APPROACH)

Research Paper

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol 10, No. 10
editor@cirworld.com
www.cirworld.com, member.cirworld.com

INTRODUCTION

The major concern of the engineering regardless of the domain is to manage the compromise quality over cost. To ensure this equation, especially the budget constraints software engineering adopts approach and methods that increase the reuse among those approaches is the product line engineering. It optimizes the construction of products by leveraging their common characteristics and managing their differences in a systematic way [Clements and Norhethrop, 2001].

A product in fact is configured and derived with a specific method from the same product line which presents the variability and the constraints. In the industry to construct and develop any good needs to analyze requirements and elaborate the solution, those two phases must be documented and understood regardless of their user. Hence, it is recommended to use a standard representation. One of the most widespread models is the presented by Unified Modeling Language (UML). According to [Sliwa, 2004] the UML/MDA approach is gaining increasingly wider adoption by application architects, UML usage includes no more than 15% of developers. IBM Group defines it as “The UML is the standard language for visualizing, specifying, constructing, and documenting the artifacts of a software intensive system”. In fact, UML is the issue of the adoption of model driven architecture (MDA) introduced by Object Management Group (OMG) as “an approach to system-specification and interoperability based on the use of formal models”. The reuse of the UML model is usually ad-hoc, not based on a strategy and not automated. Our solution is to develop a framework that generate systematically a labeled UML model fragments using the SPL strategy, so we automate the reuse in earlier stage although line product is intended principally for the industry. It is used in advanced stage of the production processes to generate different configuration of a final product like mobile phone, car industry. Indeed SPL is not the only strategy based on reuse, but to generate an UML diagram with criteria specified by the user needs a management of differences between each element of UML diagram, named in SPL engineering variability. In fact, we must determine all a for the systematic transformation of the users requirements to a labeled UML model. The approach is based on three major concepts, the model driven engineering (MDE), Software product line (SPL) and UML (Unified modeling language). The rest of the paper is organized as follows. Section 1 provides a brief representation of the model driven engineering and introduces its instances MDA in Section 2. UML and its issues are defined in section3. Section 4 provides an explanation of the different phases in process construction of line product. Section 5 compares the model driven engineering with the line product engineering. Section 6 gives a short overview of the related works.

1. MODEL DRIVEN ENGINEERING (MDE)

Quite a long time people use the design and the abstraction to explain and simplify a concept, a method or an idea. In the same context, the software engineering community needs to schematize and abstract the solution, a part of solution, the process to define solution etc. We named this approach as model driven engineering. The abstraction appears with the complexity of the programming languages, such as Assembly and Fortran [1]. To understand MDE, let’s first define a model. According to [2] "A model is a simplification of a system built with an intended goal in mind. The model should be able to answer questions in place of the actual system." So, a model actually is a translation of goals or ideas in palpable way. Some researchers disagree with this definition and insist on that the model must describe not an idea in general but a solution. In [20], Warmer and his colleagues define a model as: “A model is a description of a (part of) systems written in a well-defined language. A well-defined language is a language with a well-defined form (syntax), and meaning (semantics), which is suitable for automated interpretation by a computer”. Model can be used, developed and modified. It is following the change of requirements, the point of view of the author etc. The adoption of the MDE approach increases because the models can be reused and kept as an historical idea achieved. But model can’t be understood if we don’t understand the notation and its signification. This means a model without its meta-model doesn’t make a sense, and loses its value. This explains why the quality of model is in reality the quality of the meta-model.

Model driven engineering adopts this approach of modeling to reduce the complexity and the ambiguity of languages, and to facilitate the comprehension for some concepts. In SPL, we have a set of variability, and we need to elaborate the relationships between them. So, to facilitate the comprehension line product must be modeled, in section 3, we will represent the different types of model of line product.

MDE can be used to express mainly two phases [1]: Domain-specific modeling languages (DSML): In our solution, we have two types of diagrams classified as a DSMLs. In our solution, we have two types of diagrams classified as a DSMLs: the line product model and the fragment of UML. Both represent an abstraction of the requirements of a domain. The second phase is Transformation engines and generators: Analyze the model to extract some aspects or to transform it to another code or even another model. This transformation respects relationships and constraints specified in the DSMLs.

The biggest issue of the model driven engineering is the model driven architecture (MDA).

2. MODEL DRIVEN ARCHITECTURE (MDA)

The model driven architecture (MDA) is based on the idea to separate the abstraction from the specification. When we talk about specification we mean the technological details of the system and the abstraction is the general concept of the system independently of the technology used to realize the system. MDA is a concept for software development of IS based on creation of models and transformations between them, defined by a standardization body in software engineering [4]. This approach provides many advantages

Reusability: the system is represented in many MDA levels: CIM, PIM, PSM and each level can be refined and reused specially for the anterior levels. “Companies that adopt the MDA gain the ultimate level in flexibility: the ability to derive code from a stable model as the underlying infrastructure shifts over time. ROI flows from the reuse of application and domain models across the software lifespan.” [Richard Soley, OMG Chairman and CEO].

Transferability: the model is independent from a specific platform. We can use the same model for many platforms.

Interoperability: it is closely related to standard development and reusability that is the result of the previous two advantages [4]. It is called model driven because it is based on models. We represent the system, a part of system, design, operation, deployment etc, with a specific notation instanced from a meta-model.

2.1 Computation Independent Model (CIM)

CIM is a view of a system from the computation independent viewpoint. It is not the expression of system on view of user without details. It sometimes describes the functionality of the system in a familiar language. The domain practitioner expresses the first view of the system without any indication on how the functionalities can be realized.

The user enters his requirements for the diagram UML with a familiar language. The framework will interpret it and generate a labeled UML model. As an example of user requirements to generate a label of UML class diagram, we could consider reservation of a hotel room. The reservation of one or many rooms of one hotel is done by a client and supervised by a commercial entity.

2.2 Platform Independent Model (PIM)

PIM is a view of a system from the platform independent viewpoint. A PIM exhibits specified degree of platform independence, which could be suitable for use with a number of different platforms of similar types.

The framework will generate a labeled UML model without any technical specification.

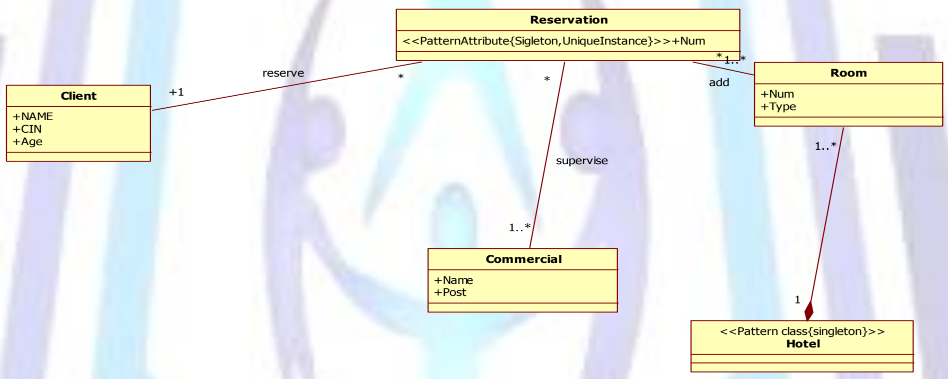


Fig1r: Example of a labeled UML class diagram in PIM laye

2.3 Platform Specific Model (PSM)

PSM is a view of a system from the platform specific viewpoint. We integrate the details of technology used in the PSM model.

The core standards of the MDA are UML, MOF, XMI, CWM but UML and MOF were regrouped on another standard named OMG. In this section, we will define it and explain its architecture to derive the MOF concept and an instance of OMG idea based on modeling UML.

In the example in Figure 2 we choose a 3 layer architecture. This technical information will be modulated in this PSM model .

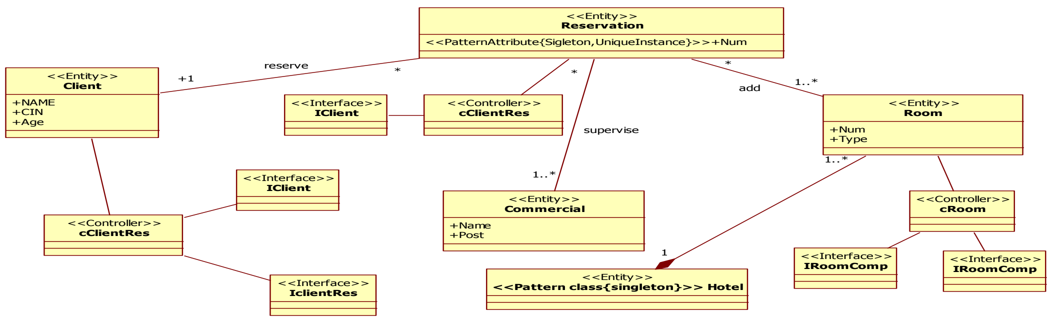


Fig2: Example of labeled UML class diagram in PSM layer

3. UNIFIED LANGUAGE MODELING (UML)

Unified Modeling Language (UML) is a graphical language. It is based on a series of symbols and connectors that can be used to create process diagrams. UML is often used to model computer programs and workflows and expresses both the static and dynamic aspects of the system. UML was proposed by the Object Management Group (OMG). The Object Management Group (OMG) has proposed a modeling language called UML (Unified Modeling Language) for describing all kinds of object-oriented software artifacts [8]. UML [7] is the most referred modeling language. Its last version, UML2, is widely adopted for modeling not only the application structure, but also its behavior, and architecture [6]. In [11], author considered MDA as the central vision of OMG and the most successful issue of MDA was realized by OMG.

UML expresses the static and dynamic aspects. In static, UML represents the permanent elements of the system and the relation between them. The goal of the static aspect is to define characteristics and the function of these elements. Mainly, UML is expressed by the class diagram, object diagram and use case diagram. The dynamic aspect aims to describe the behavior of these elements specified in static aspect and the interaction between them in a particular time and with a different level of scope.

As we explain in the introduction, our solution aims to generate a UML diagram, but the notation from a diagram to another is different. MDA is based on model, which is an instance of a meta-model. The meta-model must explain all the concepts and define the possible relationships between concepts. But if this information should be modeled, we need to have a meta-meta-model. In this case we need also the meta-meta-meta model that can connect us to indeterminate boucle. To resolve this problem, OMG (object management group) proposes a standard to define a static meta-meta-model to construct an oriented object meta-model. Is characterized by a high level of abstraction and some authors call it “ontology” [12]. The aims of MOF is to construct elements and relationship between them, every meta-model should respect it and instanced it from this meta-meta-model.

The MOF must provide:

- A generic elements instantiated from any meta-model.
- A set of rules to define relationships between elements.
- A set of rules to define the instantiation constraints

In this solution the final product is a fragment of UML diagram, which will be instantiated from the meta_model, so the use of MOF will be necessary. The generation of UML diagram will be ensured by the SPL strategy.

4. Software Product Line Engineering Process

A software product line (SPL) is a set of software product that shares a common characteristics and have a different named variability. In SPL, variability is managed to satisfy the requirement of user, eg: type of the UML diagram, domain business…etc.

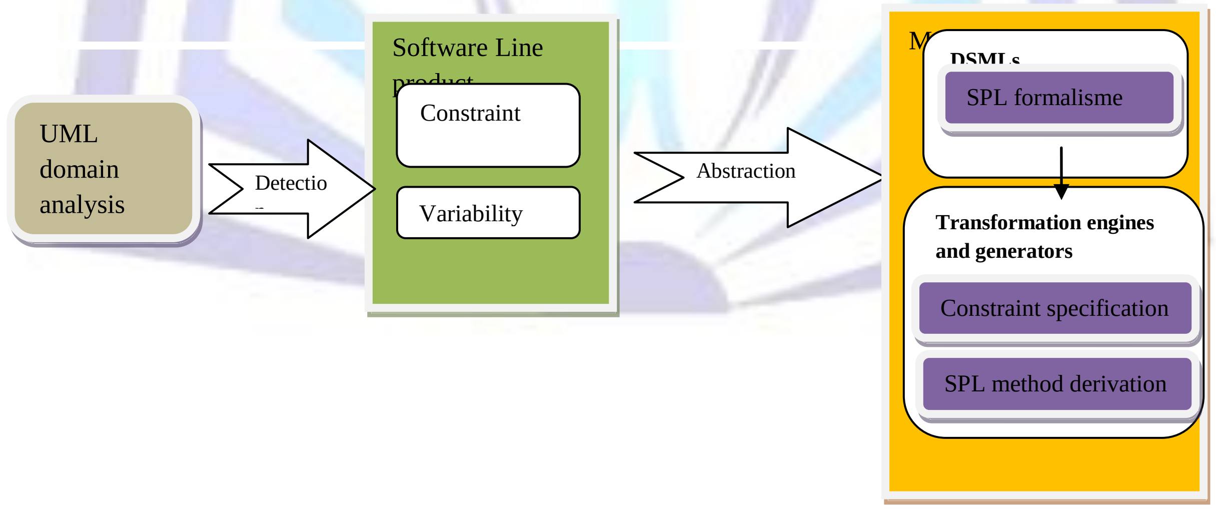


Fig3: Process of the Product Line Construction

In Figure3, we represent the different phases to elaborate our framework using the SPL strategy. The first phase analyzes the UML domain to determine variability and constraints. Then those will be abstracted and modulated in the second phase. The relationships between the different features must be expressed to finally develop a method to generate the UML fragments.

4.1 Analyze The Domain Engineering

The domain engineering presents the business domain of the product line determined by the contribution of experts of domain. This phase aims to collect and analyze the whole domain and detect variability and the relationship between members of the product. Then They will be introduced in a line product model. The domain engineering is the UML diagrams and to ensure an accurate analysis, we need data warehouse of UML diagram and the real challenge will be to choose the appropriate diagrams to construct. The output of this phase is the detection of the variability and the relationship between it.

4.2 Software Line Product

Software product line engineering optimizes the development of individual systems by leveraging their common

characteristics and managing their differences in a systematic way. These differences are called variabilities.

4.2.1 Variability

The variability expression includes all the assumptions of how the members of the product line products differ.

Variability multi-dimension: The product line model contains the variability information of the domain, often expressed in terms of features that the products contain. A feature represents a product characteristic that customers view as important in describing and distinguishing members of a software product. The first step is to capture variability and represent it. But to be able to do it we need a large repository of UML diagrams. We will represent the variability with a defined goal or we represent the variability independently of the final goal? The multi-dimension variability represents variability for many goals to increase the reuse of the product line model.

A variability dimension is a kind of variability that is important for a stakeholder. In current research, the variability dimensions of an SPL are sometimes separated from each other and are sometimes mixed in a single feature model.

The word SPL was faced with a challenge, which is to model the components of the product line as they may be thousands and in some cases to represent the relationships between the elements of the ldp and constraints. To solve this problem several modeling formalisms have been proposed.

4.2.2 Constraint

A constraint is simply a logical relation among several unknowns (or variables), each taking a value in a given domain[8]

In the Line product, the relation between elements must be well defined and precise. This is because it will be used in the derivation of products (fragment of UML diagram). Hence, constraints must be expressed by an interpretable language. There are many specification languages. Some of them will be presented below. So a selection of a language depends on some criteria like the type of model and the type of the derivation

4.3 Model Driven Software Line Product

The model driven engineering aims to explore a complicate domain, using the abstraction and the software product engineering analyzes and combines thousands of constraints and variability. So to decrease the complexity, SPL bases on MDE. MDE represents a design approach that enables description of the essential characteristics problem in a manner that is decoupled from the details of a specific solution space [15].

As we indicate in section 1 that the model driven software line product is divided in two phases: Domain-specific modeling languages (DSML) and Transformation engines and generators.

4.3.1 Domain-Specific Modeling Languages (DSML)

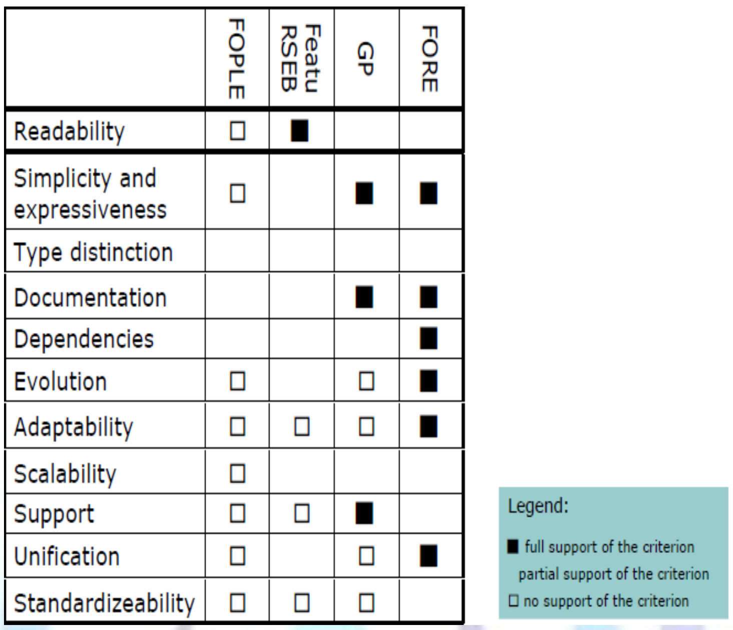
In MDE the domain-specific modeling languages (DSML) presents the requirements of a particular domain. For each DSML we must have a meta-model, which does not only describe the concepts of domain but the relationship between them. Generally, this relationship can be specified with some constraints.

In SPL the DSML is a modelisation of the product line domain. There are many formalisms to adopt in product line and each formalism was developed to represent a line product model. But despite all their differences they must be able to modulate the line product domain composed as shown in Figure 2 on variability and constraints. That's why formalisms in SPL has two common features as provided below:

i) are oriented variability: all these models must be able to represent the variability between common features (commonality) and the different characteristics (variability).
ii) specify constraints: identify constraints, which are compatible and not compatible with each product line (inclusion, and exclusion) characteristics.

ISSN 22773061

Table1 : Comparative study of SPL formalisme[16]



	FOPLE	FeatuRSEB	GP	FORE
Readability	□	■		
Simplicity and expressiveness	□		■	■
Type distinction				
Documentation			■	■
Dependencies				■
Evolution	□		□	■
Adaptability	□	□	□	■
Scalability	□			
Support	□	□	■	
Unification	□		□	■
Standardizeability	□	□	□	

Legend:

■ full support of the criterion

partial support of the criterion

□ no support of the criterion

In this comparative study presented in Figure 4 we define some criteria to satisfy. So to know with which formalism we should work depends on what criteria we need.

4.3.2 Transformation Engines And Generators

Transformation engines and generators: Analyze the model to extract some aspects or to transform it to another code, another model. This transformation respects relationships and constraints specified in the DSMLs. In SPL the constraints must be expressed in a specified constraint language and then generate with a specific method fragments of UML diagram.

4.3.2.1 Specification Language: Constraints Expression

i. Object Constraint Language (OCL)

OCL is related to modeling. Indeed, with OCL we describe type of information dependently of the relationship between the elements of model. OCL is related to UML, it describes defect of the notation. OCL is based on the mathematical representation and algorithm.But OCL is not a programmable language, we can't build an executable code with OCL but the OCL expressions are evaluated and a fixed value is returned.

Where to use OCL [10]:

- To specify type invariants for stereotypes
- To specify invariants on classes and types in the class model
- To describe pre- and post-conditions on operations and methods
- To describe guards
- As a navigation language
- To specify constraints on operations

OCL is used to specify the well-formedness rules of the UML meta-model.

ii. Constraint Programming (CP)

The constraint programming is a new software technology to describe problem with high degree of difficulty especially combinatorial problem and especially in areas of planning and scheduling [8]. For this potential to resolve a lot of problems in different areas of CP has recently been recognized by the ACM (Association for Computing Machinery) as one of the strategies in computer research.The dependencies is formalized in physical worlds and their mathematical abstractions are represented naturally and transparently [8]. But the uses of the mathematical operation make the execution and manipulation more expensive.

iii. Constraint Satisfaction

Constraint satisfaction is used in a specific and a finite domain. That means each concept of this domain is defined. This type of constraint is widespread in the industry area but it concerns a specific domain so we can't reuse the constraint satisfaction in another domain.

iv. Constraint Solving

Constraint solving presents the problem as an ensemble of constraints and it is solved using mathematic techniques as Newton method, Taylor series. Constraints are defined in an infinite domain or more complex domain then in the constraint satisfaction.

3.3.2.2 Derivation of a product: Configuration of product line

Configuration management (CM) is the control of the evolution of systems. It represents the tools and techniques tend to focus on managing individual products. In our case product is fragment of UML diagram. A custom asset contains a set of specific components application. It's not designed for reuse, but produced for a specific application. The quality requirement (for example, reusability) of custom assets is not as high as core assets and effort spent on maintenance of custom assets is less than core assets.

5. MODEL DRIVEN ENGINEERING Vs SOFTWARE LINE PRODUCT ENGINEERING

The difference between MDE and SPLE is how they need to ensure reuse. In MDE we adopt the abstraction strategy. That means make a model with less detail without losing information. In the line product engineering, we are based on variability between elements of the line product. More the variabilities are expressed more the reuse is ensured.

The second difference is in the construction process. The MDE use the transformation of models to other models and the refinement. This process is contained more in the MDA (instance of MDE), we found 4 levels (CIM, PIM, PSM, Code). But in the line product engineering there are many models with much formalism but we adopt just one of them. We use the incremental construction to build a line product [9].

6. RELATED WORK

In this section, we will present some frameworks that generate UML diagram to wit: F_UML and IMLAUT.

6.1 F_UML

F_UML is a framework to generate four types of diagrams according to defined rules and using some specific notation:

- Use case diagram: it determines the functionalities of the system and the users. It represents the domain and its limits.
- Class diagram: it represents the static aspect of the solution and its architecture.
- Pattern diagram: it injects in class diagram the design patterns and the metapatterns. "It is obtained from the class diagram by filtering out the details inside the classes and by matching the resulting structure to a set of design patterns" [13] .
- Sequence diagram**:** it describes interaction between objects (instance of classes defined in pattern diagram and class diagram).
- It is based on FDBM (Framework-Based Design Method) notation. FUML generates an UML profile for three diagrams. The first diagram is the use case and it should be specified directly by the user. The limits of F_UML is that to generate an UML profile by adding some specific notations [13]

6.2 UMLAUT

Is an extensible UML transformation framework based on the transformation of UML models. It generates code, injects design pattern and even simulates a functional and non-functional behavior using a formal language as the algebraic notation.

Is a free framework to manipulate UML diagram and the issue of the UML diagram as code. Users can import or export different types of diagrams in different format such as CDIF and XMI. The implementation of Umlaut needs a set of meta-model diagrams and the transformation and mapping between diagrams, which are needed for an approach. UMLAUT adopts an applicative approach [14].

CONCUSION AND FUTUR WORK

Reuse is the purpose and the objective of both types of engineering presented in this paper, MDE and LPE. They are widely used in the industry domain to improve quality and to minimize the time of production. This approach will be used to derive fragments of UML diagram, to ensure reuse in earlier phase production but to determine the line product model we need to analyze a set of UML diagram. This paper details the different axes involved to realize this approach (MDE, UML and SPL) and explains the software product line engineering process. The first phase is the UML domain analysis to detect variability and constraints. But before launch it, we must be precise if it will elaborate a line product model which concerns all UML diagrams or for each UML diagram we need to choose one of those two and that will be the purpose of the next works.

REFERENCES

[1] Douglas, C. Schmidt.2006. “Model-Driven Engineering”. Vanderbilt University.

[2] J. Bézivin, O. Gerbé. 2001. "Towards a Precise Definition of the OMG/MDA Framework". ASE’01

[3] A. Kleppe, S. Warmer, W. Bast. 2003. "MDA Explained. The Model Driven Architecture: Practice and Promise". Addison-Wesley

[4] Martin Kardoš, Matilda Drozdová . 2010. ”Analytical Method of CIM to PIM Transformation in Model Driven Architecture (MDA)”.

[5] Miller, Jishnu Mukerji. 2003. ”MDA Guide” Version 1.0.1,OMG.

[6] S. Kherraf, E. Lefebvre, W. Suryn. “Transformation From CIM to PIM Using Patterns and Archetypes”.

[7] OMG.2007. UML 2.1.1 Superstructure Specification.

[8] Roman Barták. “Constraint Programming: In Pursuit of the Holy Grail

[9] Azanza Sesé, “Model Driven Product Line Engineering: Core Asset and Process Implications Dissertation

[10] Object Constraint Language. 2008

[11] Emerson, Sztipanovits, Bapty. 2004. “A MOF-Based Metamodeling EnvironmentJournal of Universal Computer Science”.

[12] John D. Poole. 2001. “Model-Driven Architecture: Vision, Standards And Emerging Technologies”. ECOOP .

[13] N.Bouassida, H.Ben-Abdallah, A. Ben Hamadou. “F-UMLTool for the formal design of frameworks”.

[14] Wai Ming Ho , Jean-Marc Jézéquel , Alain Le Guennec , François Pennaneac’h . “ UMLAUT: an Extendible UML Transformation Framework”

[15] G.Deng,, Douglas C. Schmidt, Aniruddha Gokhal. “Evolution in Model-Driven Software Product-line Architectures”. Engineering Department Siemens Corporate Research

[16] Djebi, Salinesi. “vriteria for Comparing Requirements Variability Modeling Notations forProduct Line”.

[17] Clements, P. & Northrop L. 2001. “Software Product-lines: Practices and Patterns, Addison-Wesley“.

[18] Ouali, S., Kraiem, N., Ben ghezala. 2014. “H.: From Intentions to Software Design using an Intentional Software Product Line Meta-Model”. International Conference on Innovations In Information Technology, in Al Ain, UAE.

[19] Clauß, M.2001. “ Generic modeling using Uml extensions for variability”. IWorkshop on Domain Specific Visual Languages at OOPSLA.

[20] Marko Rosenm¨uller, Norbert Siegmund. “ Modeling Dependent Software Product Lines”.

[21] Clements, P, Northrop.L . 2001. “ L.,Software Product Lines: Practices and Patterns”. Addison-Wesley, Boston, MA.

[22] Bosch. 2000. “Design & Use of Software Architectures, Adopting and Evolving a product-line approach”. Addison-Wesley.