Towards A UML profile for modeling variability

Benselim Mohamed Salah
LabSTIC Laboratory
University of "08 Mai 1945"
Guelma, Algeria
benselim.mohamed@univ-guelma.dz;

msbenselim@yahoo.fr;

***Abstract*— UML is a general modeling language that offers standardized modeling notations. Some specific domains require to be modeled by specific notations other that proposed in standard UML. Variability managing is one of these specific domains that necessitate specific modeling notations. In this paper we propose a UML profile composed by a set of UML extended notations and destined for modeling all aspects of variability in software development process. The proposed profile is defined by a package of specific profiles that extend the UML standard notations of five UML diagrams (class diagram, use case diagram, sequence diagram, activity diagram and state diagram). In order to model the variability requirements and specifications by adequate graphic representation we propose some extension mechanisms such as stereotypes, constraints and tagged values that extend and personalize the existing standard UML notations.**

Keywords—UML, diagrams, profile, extension, modeling, variability

I. INTRODUCTION

The software development is a hard process because of modeling specific requirements and characteristics of a system. Some of these systems or research domains may have unstable characteristics such as changeable properties in space and time, appearance or disappearance of properties, context of use of a situation, changeable state of an entity, variable behaviour of an entity,...etc. These specific characteristics have a common ownership of changing and variability. To model such characteristics developers are still looking to use well-adapted and adequate modeling notations. The best and well known standard modeling language is UML (Unified Modeling Language) [1]. This universal language is considered as too general because it does not take into account some specific domains and specific technologies that require specific modeling notations; but it covers this gap or insufficiency by providing some extensibility mechanisms that allow extending the existed UML elements to be able to represent adequately any specific characteristic or requirement. In this paper we seek to complete our previous study [2] by proposing a set of UML extended notations especially destined for modeling specific characteristic and requirements in the domain of variability management. The proposed notations are grouped in a package as an UML profile and are defined by stereotypes, constraints and tagged values. Each of these new extensions is created by extending UML model elements such as: classes, attributes, operation, association, activity, actor...etc; and concern one of the five UML diagrams studied in this work (class diagram, use case diagram, sequence diagram, activity diagram and state). A stereotype allows giving a new signification of an existing UML element in order to create a new modeling notation that can represent a specific characteristic. Constraints permits to specify the limits and restrictions of semantics for the created elements. Tagged values represent the new attributes of the created stereotypes. All the proposed notations are defined and implemented by using the StarUML software modeling extensible platform that supports excellently UML language and provides many opportunities like extensibility, customizability and flexibility [3]. A UML profile which contains all the proposed notations is created and experimented with five UML diagrams (class, use case, sequence and activity). The reminder of this document is organized as follows: Section 2 overviews some previous works that are related to our research topics. Section 3 justifies the motivations and objectives of this paper. In section 4 we give details of the proposed profile. Section 5 shows how the proposed notations are implemented and experimented. Finally in section 6 we recapitulate the proposal and we present future works.

II. OUTLINE OF PREVIOUS WORKS

Variability modeling is a process that permits representing all changeable aspects of a system with adequate notations and graphics. Especially in software-intensive systems we need more specific notations to be able to manage the concept of variability. Given that UML is considered as a general purpose modeling language, several works have used the extensibility capacity of this language (as a profile) to adapt it for representing the specific requirements of a system and for modeling variability. Several extended versions (as profile) of this language have been developed in order to take into account and to model all the specific systems requirements [4][5][6][7][8]. Other works have treated the topic of variability in different way such as: [9][10][11][12][13][14]. A variety of UML extensibility mechanisms have been introduced in previous works in order to create profiles that support the concept of variability. Indeed the UML profile V-ICSOLAP [15] supports variety (in terms of complex data types) and variability (in terms of both extensibility and type/name variability) at the conceptual level. Another UML profile is provided for managing the variability models [16]. It shows the dependency from the UML profile to activity diagrams to make the relationship between variability models and process models visible. A set of extensibility mechanisms as a UML profile are proposed for the domain of context-aware applications development [17]. This profile takes into account the context of use of a situation including all properties that are continuously variable in time or space. The work presented in [18] shows how the concept of variability is used in combining two types of engineering: Software product line engineering and Model Driven Engineering.

By studying and analyzing these previous works we were able to recapitulate and synthesize the basic principles of the variability modeling domain especially by using the extensibility mechanisms of UML language. Through our proposal we were able also to provide a set of extended UML notations that are easily useable and directly

exploitable for modeling the specific requirements (commonalities and variabilities) of a system. The proposed notations are grouped in a package as UML profile and concern five types of UML diagrams (class, use case, activity, state and sequence).

III. MOTIVATIONS

The concept of variability aims to capture and to represent all the requirements of a system including those that are permanent and common (commonalities) and those that are optional and variable (variabilities). Modeling variability requirements is a hard and fastidious task because there is no standardisation of specific modeling notations or a specific modeling tool that allows representing these specific requirements in an adequate form. Even the UML language does not offer the necessary notations to remedy this insufficiency, despite the fact that this language is considered as one of the most widely used standards for specifying, modeling and documenting information systems. To motivate our study we must answer the three basic research questions: What to do? Why to do it? How to do it?

Firstly, as a result we aim to create a UML profile which contains specific notations for the variability domain and which can help developers to better analyze and represent all the requirements related to variability. Secondly, about the purpose of creating an UML profile for variability modeling that is to fill the deficiency noted in the area of modeling notations destined for representing variability specifications. Other reason is that the standard UML considered as the well known universal modeling language does not propose (in explicit way) modeling tools for the variability management domain. The proposed profile can facilitate the job of software developers when dealing with variability and can widely optimize the development process in term of time, costs and efforts. Thirdly, the way or the manner to create an UML profile is by using the extensibility mechanisms offered by UML language. The mechanism of extension permits to create and to add new modeling notations that are more adapted with a specific domain. The created new notations can be in the form of stereotypes, constraints or tagged values that are extended from UML elements.

IV. THE PROPOSED UML PROFILE

Variability modeling is a specific domain that requires specific modeling notations. To model all aspects of variability we need to use appropriate notations that are not available in standard version of UML language. Thus we have to extend UML notations by creating new elements that can represent variability characteristics in adequate form. Each of these characteristics must be represented by an adequate notation of UML language. For this, we provide an extension of UML as a profile "UML variability profile" that contains stereotypes, tagged values and constraints. A stereotype permits to define a new meaning of an existing UML metamodel element. Tagged values are always attached to a stereotype and their role is to indicate attributes of the created stereotype. Constraints define the restrictions of semantics for each added new element. Our proposed profile is a structure diagram which describes lightweight extension mechanism to the UML language by defining custom stereotypes, tagged values and constraints. This profile is a package of other profiles that extend UML metamodel (figure 1).

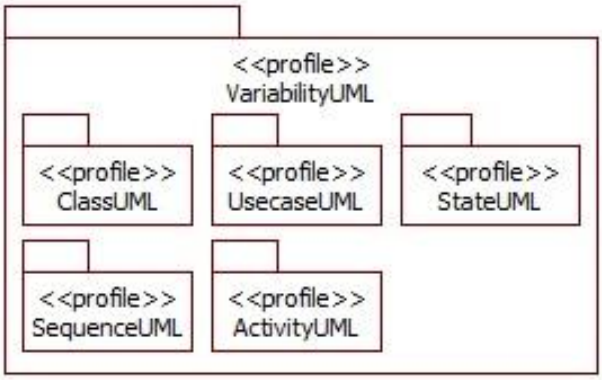

Fig. 1. Package of the proposed UML profile

We chose five UML diagrams (class, use case, sequence and activity) that represent the main tools of an application development (analysis, design, and implementation). New UML diagrams notations are obtained by extending the existing elements of UML metamodel. The UML variability profile package is composed of five major profile packages: ClassUML, UsecaseUML, SequenceUML, stateUML and ActivityUML. Each package is inherited from the common UML metaclass 'package' and will comprise extended notations of the correspondent diagram (Table 1).

TABLE I. COMPOSITION OF THE PROPOSED UML VARIABILITY PROFILE PACKAGE (CLASS PROFILE)

Profile name	**Reference Metamodel**	**Metamodel element (metaclass)**	**Description**
ClassUML profile	UML metamodel	'Package'	Extends UML class diagram notations to support variability modeling

Generally, all UML diagram concepts (class, association, attribute, operation, actor, use case, object, lifeline, message, activity, transition, etc.) can be stereotyped. In Table 2, we show an example of the main UML metamodel elements that will be extended and studied in our work.

TABLE II. UML DIAGRAMS AND CORRESPONDING EXTENSION MECHANISMS (CLASS DIAGRAM).

UML Diagram name	**Stereotypes**	**Metamodel element (metaclass)**
Class diagram	VariableClass	Class
	VariableAssociation	Association
	VariableAttribute	Attribute
	VariableOperation	Operation

Proposed UML extension enables an appropriate use of the domain specific notation in place of the standard UML concepts. System entities which have specific properties and constraints that cannot be represented with standard UML notations have to be modelled with the proposed stereotypes. The specific characteristics (limits, conditions, means, semantics...) will be taken in charge in stereotypes by attached constraints and tagged values. Below, we are going

to describe the five profile packages and we expose the main proposed concepts and notations to respectively five UML diagram (class, use case, activity, state and sequence). It is necessary to note that, in MDA architecture, defining a UML profile means that standard UML will be extended at the metamodel level M2. Thus, proposed extensions are done at metamodel level M2 and then diagram notations (at model level M1) are built to be conform to these metamodel concepts. In class diagram, existing UML notations will be extended in order to obtain new notations that will be more adapted to model specific features and entities of variability requirements. All UML concepts can be stereotyped to be adequate with particular domain or platform. Standard UML diagram concepts have to be extended in order to create specific notations (as stereotypes) for variability modeling. Figure 2 illustrates a package of UML proposed profiles and shows how UML diagram concepts are stereotyped. The stereotypes <<VariableClass>> and <<VariableAssociation>> are respectively obtained from the standard UML metaclass "Class" and metaclass "Association" by extending and customizing their syntax and semantics. In the same way we can create the stereotypes <<VariableAttribute>> and <<VariableOperation>> by extending the two metaclasses "Attribute" and "Operation" respectively. Similarly, we can define the packages of use case diagram profile, activity diagram profile, sequence diagram profile and state diagram profile. Indeed in use case diagram, we extend the main metaclasses of standard UML use case diagram to create new modeling concepts as stereotypes. This extension permits to specify the field of use of use case diagram concepts to be used in modeling specific users and scenarios of different variability situations. Thus, metaclasses "Actor", "UseCase" and "Dependency" are respectively extended by the stereotypes <<VariableActor>>, <<VariableUseCase>> and <<VariableDependency>>. To each created stereotype we attach some constraints and tagged values. Constraints are used to specify limits and restrictions according to particular specifications of the variability situations. Tagged values show and determine the new properties (attributes) of the created stereotypes.

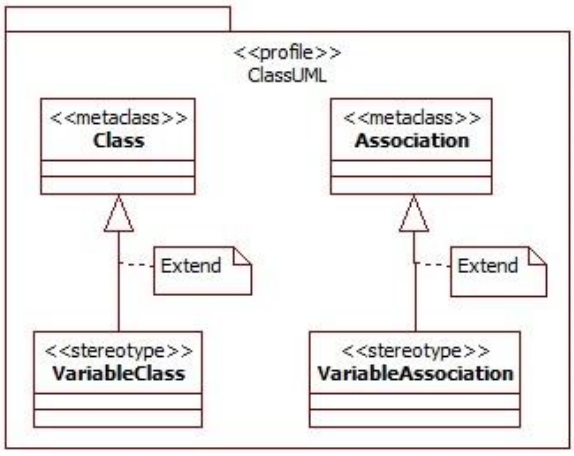

Fig. 2. Example of package of class diagram profile

In UML activity diagram, two main metaclasses ("Activity" and "Transition") are extended by two stereotypes (<<VariableActivity>> and <<VariableTransition>>). In variability modeling domain, tasks (or activities) perform and execute variable actions that are influenced by a changing context of use (mobility of users, used devices, heterogeneity of information sources and system distribution). Such activities cannot be represented by standard UML notation because each of them can have several variable forms according to incurred variations of the current situation. Thus, we have to use the proposed stereotypes to model all tasks and activities in variability management. UML sequence diagram allows representing the successive interactions of a use case (or a scenario) in chronological order. It illustrates objects interactions by showing sent and received messages between these objects. Because objects are instances of classes and because we operate in variability modeling domain (for which we have proposed the use of variable classes such as <<VariableClass>> stereotypes), thus we are obliged to work with <<VariableObject>> concepts as instances of <<VariableClass>> stereotypes. UML metaclasses ("Object", "LifeLine" and "Message") are extended by new stereotypes (respectively <<VariableObject>>, <<VariableLifeLine>> and <<VariableMessage>>). In UML state diagram, three main metaclasses ("InitialState", ("FinalState" and "Transition") are extended by three stereotypes (<<VariableInitialState>>, <<VariableFinalState>> and <<VariableTransition>>). The created stereotypes provide new meaning and well defined semantics to the existing UML metaclasses by specifying the exact goal to which they will be used. New stereotypes are able to model any object that varies with the context of use of a changing situation. Object varying characteristics are taken in charge by constraints and tagged values that are attached to the corresponding object stereotype <<VariableObject>>. This will permit customization of the use of UML standard sequence diagram notation in the specific variability domain. Also, it will permit, precise and explicit modeling way of variable features and factors that are continuously changing..

A. Proposed stereotypes

The main standard UML metaclasses ("Class", "Association", "Actor", "UseCase", "Transition", "Activity", "Object", etc.) are extended in order to create stereotypes that model the different variable features of a system. These stereotypes define how existing metaclasses may be extended as a part of the proposed profile. A short description of proposed stereotypes is shown in Table 3. Indeed, in use case diagram, the <<VariableActor>> stereotype will be used to model entities (such as users, systems, hardware, software, etc.) that present variable constraints and that have changing properties. The stereotype <<VariableUseCase>> represents all needed functions that are composed by variable actions and varying tasks in time and space of a system. The stereotype <<VariableDependency>> shows a specific relationship that relates two <<VariableUseCase>> stereotypes. When the stereotype is applied to a model element, an instance of this stereotype is associated to an instance of the corresponding metaclass. This stereotype is specialized in several instances according to attached tagged values.

TABLE III. DESCRIPTION OF THE PROPOSED UML STEREOTYPES (CLASS DIAGRAM).

UML diagram name	Stereotype	UML construct (Metaclass)	Description
Class diagram	VariableClass	Class	Is an abstract illustration that represents an entity of a variability feature Can be associated with one or more other VariableClass.
	VariableAssociation	Association	Is a relationship that associates two VariableClass stereotypes Can be a simple association, a generalization (or specialization) or a composition relationship between two VariableClass stereotypes.
	VariableAttribute	Attribute	Permits to represent the new properties of a VariableClass stereotype and to define all values that can be attached to instances of this stereotype.
	VariableOperation	Operation	Permits to represent the new operations of a VariableClass stereotype and to define all conditions and restrictions that can be attached to instances of this stereotype.

B. Proposed constraints

UML diagram constraints are defined by clear conditions and restrictions that are specified to limit and to determine how proposed stereotypes (<<VariableActor>>, <<VariableUseCase>>, <<VariableObject>>, <<VariableActivity>>, <<VariableTransition>>, etc.) are used in modeling process. These constraints show the main differences between proposed stereotypes and existing concepts of standard UML. Generally, they can be expressed or written by several ways and tools such as: Natural languages (English, French, etc.), Programming languages (Java, etc.), Mathematical Notations (AND, OR, +, =, etc.), Graphical diagrams (UML diagrams, etc.), OCL language (Object Constraint Language). Below we describe the proposed constraints with three representation kinds (by natural language or by UML class diagram). Firstly, these constraints are described in Table 4 by using natural language (English). Secondly, we can use UML class diagram to express these proposed constraints such as illustrated in figure 3 that concerns some examples of UML diagrams.

TABLE IV. DESCRIPTION OF THE PROPOSED UML CONSTRAINTS WITH NATURAL LANGUAGE (CLASS DIAGRAM)..

UML diagram name	Stereotype	Description of attached constraints
Class diagram	VariableClass	It must be extended from the UML metaclass "Class" It must be related to another VariableClass by VariableAssociation relationship
	VariableAssociation	It must be extended from the UML metaclass "Association" It relates two VariableClass It shows the kind of existing dependence (Association, Aggregation, Composition, Generalization) between related VariableClass
	VariableAttribute	Two attributes must not have the same name
	VariableOperation	Two operations must not have the same name

C. Proposed tagged values

UML diagrams tagged values define the properties (or attributes) of the proposed stereotypes for the corresponding diagram. Tagged values are attached to stereotypes and they are considered as meta-attributes of metaclasses.

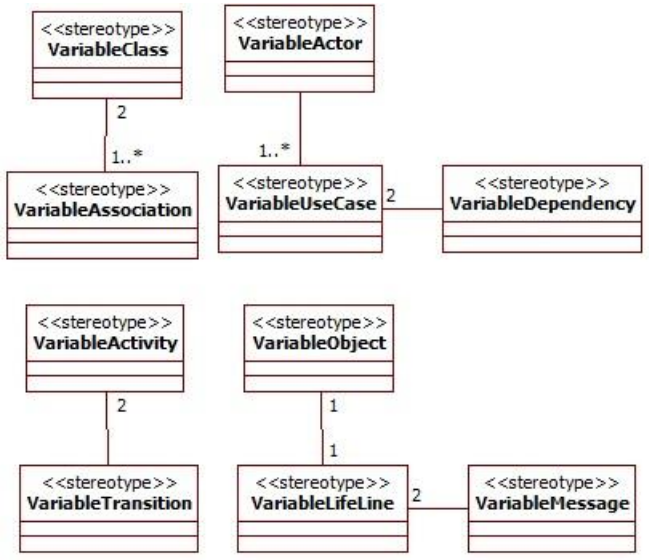

Fig. 3. Example of package of class diagram profile

They permit to differentiate the proposed stereotypes (<<VariableActor>>, <<VariableUseCase>>, <<VariableLifeLine>>, <<VariableActivity>>, <<VariableTransition>>, <<VariableObject>>, etc.) from the existing UML metaclasses (respectively “Actor”, “UseCase”, “LifeLine”, “Activity”, “Transition”, “Object”, etc.).}. In Table 5, we specify some examples of tagged values that are applied to the proposed stereotypes. Each tagged value is defined by a property definition (attribute name) and its possible values. Proposed tagged values will provide predefined values of specific and changing characteristics (features or properties) that cannot be modelled with standard UML notations. For example, when we apply the tagged value “State_of_user” to the stereotype <<VariableActor>>, we include predefined values such as “walking”, “sitting”, “standing” and “traveling” that can be used to model many changing situations of the user’s state while executing an application. In activity diagram, applying the tagged value “Guard_Of_Transition” on <<VariableTransition>> stereotype means that the guard (or condition) of the proposed transition will take into account the context of use of any transition and will not have the same effect as in standard UML because this tagged value will have many results according to its predefined values (“day”, “night”, “indoor” or “outdoor”). As an example, we suppose that we have a transition between two activities “act01” and “act02” and this transition has a guard (act02.IsOpen) that verifies if the entity act02 is open or not. Here, we see that obtained results with UML standard notations will be general and imprecise because it do not take into account that the opening time (availability) of act02 vary from day to night. But, this problem will find a solution by

using the new notations of the proposed UML profile because it distinguishes between day and night (as tagged values) before returning the needed results.

TABLE V. DESCRIPTION OF THE PROPOSED UML PROFILE TAGGED VALUES (CLASS DIAGRAM)..

UML diagram	Applied to (stereotype)	Property definition	Possible Values
Class diagram	VariableClass	PriorityOfClass	{1, 2, 3, 4, 5}
		Is_Inherited	"True", "False"
	VariableAssociation	TypeOfAssociation	"Simple", "Aggregation", "Composition", "Generalization"
		NatureOfAssociation	"Direct", "Indirect", "Alternative"
	VariableAttribute	PeriodicityOfAppearance	"Daily", "Weekly", "Monthly", "Annually"
	VariableOperation	LevelOfRelevance	"High", "Medium", "Low"

V. EXPERIMENTATION

Model UML has been the standard software modeling language that provides various concepts and notations for systems modeling. Many specific requirements of certain domains cannot be represented using UML standard notation. To solve this problem, UML provides some mechanisms for extending its notation to be adequate with any specific domain. In our study we have used this opportunity to model all variable requirements of a system. So, we have proposed new modeling elements such as stereotypes, constraints and tagged values that can be used to build a class diagram of a contextual model. To implement our proposal, we used StarUML software modeling platform because it is an extensible platform which supports UML language and provides excellent extensibility, customizability and flexibility. The implementation of the proposed concepts consists on developing extended notations of UML standard elements; and it needs creating the real file of UML profile (even for a part of diagrams) in order to be able to concretize the feasibility of our work. Below we present the major headlines of this implementation and we expose a short review of it by showing an overview of how the five packages (ClassUML profile, UsecaseUML profile, SequenceUML profile, StateUML profile and ActivityUML profile) are created. Also we give examples of implementing new concepts (creating profile files, creating stereotypes, creating constraints and creating tagged values) that compose these packages. The proposed UML Variability Profile will help and assist designers to model the changing requirements (or variable features) of a system with adequate notations and to develop variability-aware applications.

UML profile is a set of extensibility mechanisms (stereotypes, constraints and tagged values) that are required for a specific software domain or development platform. The first step of our implementation is to prepare the profile document file which will be defined in the XML (eXtended Markup Language) format. The list of proposed stereotypes includes: VariableClass, VariableAssociation, VariableAttribute, VariableOperation, VariableActor, VariableUseCase, VariableActivity, VariableTransition, VariableObject, VariableMessage. For each stereotype we indicate the name, a short description and the base UML class of this stereotype. Stereotypes are defined in the document file of "UML Variability Profile" by using XML format. For all created stereotypes we specify some constraints that can be restrictions, limitations or rules of use. As example a list of constraints is shown in Table 6.

TABLE VI. DESCRIPTION OF SPECIFIC CONSTRAINTS (CLASS DIAGRAM).

Stereotype	Attached constraints	Description
VariableClass	VariableClass.Constraint_1	It must be extended from the UML metaclass "Class"
	VariableClass.Constraint_2	Can be related to one or more VariableClass stereotypes
VariableAssociation	VariableAssociation.Constraint_1	It must be extended from the UML metaclass "Association"
	VariableAssociation.Constraint_2	Relates two VariableClass stereotypes
VariableAttribute	VariableAttribute.Constraint_1	It must be extended from the UML metaclass "Attribute"
	VariableAttribute.Constraint_2	Describes one or more specific properties of a feature
VariableOperation	VariableOperation.Constraint_1	It must be extended from the UML metaclass "Operation"
	VariableOperation.Constraint_2	Describes one or more specific operations of a feature

The constraints are introduced by using the "Constraint Editor" of StarUML menu System. For each constraint we indicate the constraint body by using the natural language or using OCL language. Tagged values are attached to stereotypes and they are considered as meta-attributes of metaclasses. They can be defined by a given attribute name and its possible values. They are used to represent and to specify attributes for the defined stereotypes. The list of proposed stereotypes includes: VariableClass, VariableAssociation, VariableAttribute, VariableOperation, VariableActor, VariableUseCase, VariableActivity, VariableTransition, VariableObject, VariableMessage. For example a short description of three proposed tagged values is presented in Table 7.

TABLE VII. TABLE VII: DESCRIPTION OF SPECIFIC TAGGED VALUES

Stereotype	Tagged value	Description
VariableOperation	LevelOfRelevance=high|medium|low	Degree of importance associated to each feature operation
VariableActor	TypeOfActor= human|hardware|software	Indicates the type of any entity that represents a feature
VariableActor	StateOfActor= sitting|standing|moving|in|out|on|off...	Indicates the state of any entity that represents a feature

The "Tagged Value Editor" of StarUML menu System can be personalized with needed features and properties. For this we have used an XML document in which we have specified all needed information for editing and creating the proposed tagged values such as: name, title, description, possible values and default value. Figure 4 shows the availability of our proposed UML variability profile to included and used in any UML project.

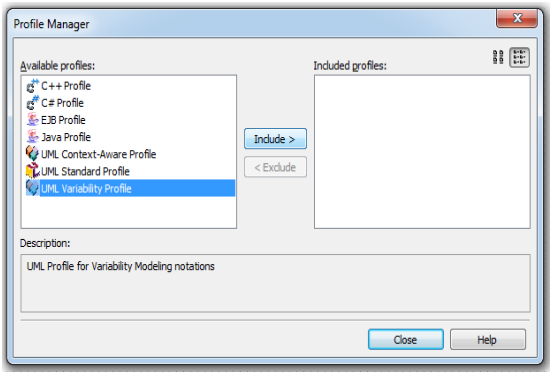

Fig. 4. StarUML profile manage before and after including available UML variability profile.

VI. CONCLUSION

In this paper we have proposed a UML profile specifically destined for modeling any particular characteristic in variability management domain. The proposed profile provides a convenient and easy modeling tool for practitioners in software development area. By using the concept of extensibility of UML language we created a set of stereotypes, constraints and tagged values to be used for modeling specific requirements of variability with an adequate representation. The proposed notations are created by extension of existing UML elements and concern five types of UML diagrams (Class, Use case, Activity, state and Sequence). All proposed notations of our UML variability profile have been implemented with the StarUML extensible platform and examples of stereotypes, constraints and tagged values have been experimented for creating diagrams. The objective of this experimentation is to demonstrate the feasibility of the proposed UML profile and to show how it is simple and easy to use the proposed extended UML notations for building specific UML diagrams that take into account the concept of variability. As future works we are working on finishing the proposed profile with other extended notations that cover other types of UML diagrams such as: CompositeStructure, Deployment, Component and Collaboration; and we hope providing a complete UML variability profile with full list of extended notations.

VII. REFERENCES

[1] Object Management Group. Unified Modeling Language, version 2.5.1. (2021). Retrieved from https://www.omg.org/spec/UML/2.5.1/PDF

[2] Mohamed Salah Benselim1 and Yacine Djebar1 "UML Extended Notations for Variability Modeling" 28th LISBON International Conference on Science, Engineering, Technology and Healthcare (SETH-24) Lisbon (Portugal), Dec. 16-18, 2024

[3] MKLab. (2020). StarUML. Retrieved from http://staruml.io/

[4] B. Sefid-dashti, J. Salimi, and H. Daghigh, "BitML: A UML Profile for Bitcoin Blockchain," International Journal of Web Research, vol. 6, issue 2, pp. 1-18, December 2023

[5] A. Kraas, "On the automation-supported derivation of domain-specific UML profiles considering static semantics," Software and Systems Modeling, vol. 21, no. 2, pp. 51-79, February 2022

[6] R. Pérez-Castillo and M. Piattini, "Design of classical-quantum systems with UML Computing," Archives for Informatics and Numerical Computation, vol. 104, no. 11, pp. 2375-2403, November 2022

[7] A. Amjad, S. Ul Haq, M. Abbas and M. H. Arif, "UML Profile for Business Process Modeling Notation," 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST), pp. 389-394, January 2021

[8] L. Addazi and F. Ciccozzi, "Blended graphical and textual modeling for UML profiles: A proof-of-concept implementation and experiment", Journal of Systems and Software, vol. 175,110912, May 2021

[9] C. Correa, J. Robin and R. Mazo, "Generating Constraint Programs for Variability Model Reasoning: A DSL and Solver Agnostic Approach,". ACM SIGPLAN Proceedings the 22nd International Conference on Generative Programming: Concepts and Experiences, pp. 138 – 152, October 2023

[10] A.P. Allian, L.F. Silva, E. OliveiraJr and E.Y Nakagawa, "VMTools-RA: a Reference Architecture for Software Variability Tools," Journal of Universal Computer Science, vol. 29, no. 7, pp. 649-690, July 2023

[11] S. Aleem, L.F. Capretz and F. Ahmed, "A Reference Framework for Variability Management of Software Product Lines," Computer and Information Science, vol. 16, no. 1, pp. 1-24, June 2023

[12] W. Mahmood, D. Struber, A. Anjorin and T. Berger, "Effects of variability in models: a family of experiments," Empirical Software Engineering, vol. 27, no. 72, March 2022

[13] O. Bisikaloa, O. Boivanb, O. Kovtunb and V. Kovtuna, "Research of the Influence of Phonation Variability on The Result of the Process of Recognition of Language Units," IntelITSIS'2022: 3rd International Workshop on Intelligent Information Technologies and Systems of Information Security, vol. 3156, no. 3, March 2022

[14] D. Struber, A. Anjorin, and T. Berger, "Variability Representations in Class Models: An Empirical Assessment," In ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, October, 2020

[15] H. Bazza, S. Bimonte , S. Rizzi, J. Laneurit and H. Badir, "A UML Profile for Variety and Variability Awareness in Multidimensional Design: An application to agricultural robots," CEUR Workshop Proceedings, vol. 3130, pp. 1-10, 2022

[16] B. Korherr and B. List, "A UML 2 Profile for Variability Models and their Dependency to Business Processes," 18th International Workshop on Database and Expert Systems Applications (DEXA 2007), pp. 829-834, 2007

[17] M.S. Benselim and H. Seridi-Bouchelaghem, "Towards A UML profile for context-awareness domain," The International Arab Journal of Information Technology, vol. 14, no. 2, pp. 195-207, March 2017

[18] M.S. Benselim and Y. Djebar, "Towards Combining Model Driven Engineering and Software Product Line Engineering Based on Variability," 2024 25th International Arab Conference on Information Technology (ACIT), Zarqa, Jordan, 2024, pp. 1-9, December 2024