UML Extended Notations for Variability Modelling

Mohamed Salah Benselim[1] and Yacine Djebar[1]

***Abstract*—Variability modelling is a specific domain that requires specific modelling notations. Unfortunately these notations are not explicitly provided in well known standards of modelling languages such as the Unified Modelling Language (UML). To address this limitation, we propose a set of UML extended notations as a new UML profile destined for modelling 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 four diagrams (class diagram, use case diagram, sequence diagram and activity diagram). To be able to model the variability requirements by adequate graphic representation we provide some extension mechanisms such as stereotypes, constraints and tagged values according to the chosen diagrams.**

***Keywords*— UML Profile, Extended Notation, Modelling, Variability.**

I. Introduction

The performance of a software-intensive system depends closely on the optimization of the development process of the basic component: the software. The development of this software is essentially based on the modelling of requirements and data. Some systems or research domains may have specific characteristics and requirements 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 modelling notations. The best and well known standard modelling language is UML (Unified Modelling 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 modelling 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 propose a set of UML extended notations especially destined for modelling 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 four UML diagrams studied in this work (class diagram, use case diagram, sequence diagram and activity diagram). A stereotype allows giving a new signification of an existing UML element in order to create a new modelling 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 modelling extensible platform that supports excellently UML language and provides many opportunities like extensibility, customizability and flexibility [2]. A UML profile which contains all the proposed notations is created and experimented with four UML diagrams (class, use case, sequence and activity). The proposed profile can be an effective support and a high quality modelling tool for developers wanting to work in the field of variability because it offers a set of adapted notations directly usable for the representation and modelling of variability elements. 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. Related Works

Variability modelling 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 modelling 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 modelling 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 [3][4][5][6][7][8]. Other works have treated the topic of variability in different way such as: [9][10][11][12][13][14][15]. 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 [16] 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 [17]. It shows the dependency from the UML profile to

[1]University of "08 Mai 1945", Guelma, Algeria

activity diagrams to make the relationship between variability models and process models visible. The paper [18] contributes to enhance the software product line modelling based on SysML by extending the SysML language. It aims to include various aspects of variability and for this a holistic variability model is proposed to define the SysML extensions by means of the UML profiling mechanism. An UML profile is proposed for real time design patterns representation in order to express the variability in patterns and to identify the pattern elements in its instance [19]. This proposed profile extends UML with concepts related to real-time databases and integrates Object Constraint Language to enforce the variation points consistency. These extensions allow distinguishing between the fundamental elements and the variable elements. . A set of extensibility mechanisms as a UML profile are proposed for the domain of context-aware applications development [20]. This profile takes into account the context of use of a situation including all properties that are continuously variable in time or space. For expressing the variability in a software product line design and for reinforcing its comprehension and for distinguishing the commonalities and variability, a profile containing UML extension is presented [21]. It offers extended diagrams enriched by the information extracted from the feature models which permits the differentiation between the common points and variabilities. Also, the concept of variability can be expressed by using the UML use case diagrams [22]. This work presents the different types of variability and examines the capability of use case diagrams to express these types of variability. An enhancement of the use case meta-model is introduced by integrating new modelling elements that allow expressing the identified types of variability. Feature models can be used as a representation of the common parts and variants contained in a system family [23] and so a feature diagram can be used as a basic representation of commonality, variability and dependencies. Here the use of UML extensibility mechanisms (stereotypes combined with tagged values) is recommended. Matthias Clauß introduces a UML extension to support feature diagrams and adds elements describing variability in the standard kinds of UML diagrams [24]. These extensions use the standardized extension mechanisms and are compatible with the standard and enable a broader usability. A variability modelling method with the extended UML is presented for variability modelling [25]. It supports variability constraint such as variant to variant, variant to variation point and variation to variation point. The study [26] supports variability modelling research with empirical data of the real-world use of its flagship concepts and provides requirements for concepts and mechanisms and challenges assumptions about size and complexity of variability models. Other works have dealt with the concept of variability by classifying, analyzing or comparing the existed techniques and approaches in variability modelling domain such as: Classifying variability modelling techniques [27], A Comparison of Variability Modelling Approaches [28], A Survey of Variability Modelling in Industrial Practice [29], Understanding and Modelling Variability: Practitioners' Perspectives [30], An Analysis of Variability Modelling Concepts: Expressiveness vs. Analyzability [31], Comparative analysis of variability modelling approaches in component models [32], Comparison of variability modelling techniques [33].

By studying and analyzing these previous works we were able to recapitulate and synthesize the basic principles of the variability modelling 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 modelling the specific requirements (commonalities and variabilities) of a system. The proposed notations are grouped in a package as UML profile and concern four types of UML diagrams (class, use case, activity and sequence).

III. Motivations and Objectives

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). Modelling variability requirements is a hard and fastidious task because there is no standardisation of specific modelling notations or a specific modelling 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, modelling 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 modelling that is to fill the deficiency noted in the area of modelling notations destined for representing variability specifications. Other reason is that the standard UML considered as the well known universal modelling language does not propose (in explicit way) modelling 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 modelling 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.

Our main objective is to provide a variability modelling tool that helps developers to find and to use easily the adapted and the adequate notation for modelling the specific requirements of variability. The proposed profile has all the opportunities to satisfy the growing demand for variability modelling notations by providing a set of stereotypes, constraints and tagged values

needed for modelling particular specifications in variability management.

IV. The Proposed UML Variability Profile

Variability modelling is a specific domain that requires specific modelling 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. We chose four 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 four major profile packages: ClassUML, UsecaseUML, SequenceUML 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

Profile name	Reference Metamodel	Metamodel element (metaclass)	Description
ClassUML profile	UML metamodel	'Package'	Extends UML class diagram notations to support variability modelling
Usecase UML profile	UML metamodel	'Package'	Extends UML use case diagram notations to support variability modelling
SequenceUML profile	UML metamodel	'Package'	Extends UML sequence diagram notations to support variability modelling
ActivityUML profile	UML metamodel	'Package'	Extends UML activity diagram notations to support variability modelling

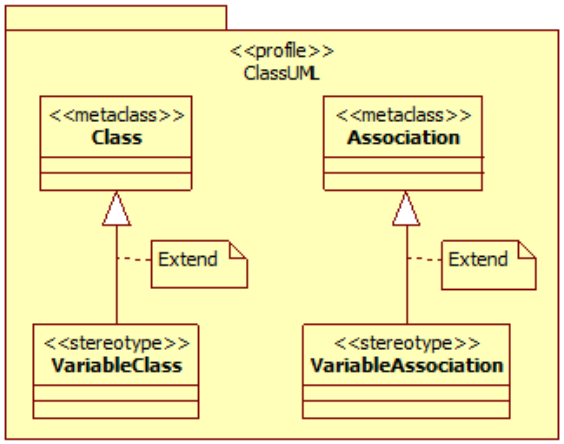
Generally, all UML diagram concepts (class, association, attribute, operation, actor, use case, object, lifeline, message, activity, transition, etc.) can be stereotyped. In Table 2, we list the main UML metamodel elements that will be extended and studied in our work.

TABLE II: UML DIAGRAMS AND CORRESPONDING EXTENSION MECHANISMS

UML Diagram name	Stereotypes	Metamodel element (metaclass)
Class diagram	VariableClass	Class
	VariableAssociation	Association
	VariableAttribute	Attribute
	VariableOperation	Operation
use case diagram	VariableActor	Actor
	VariableUseCase	Use case
	VariableDependency	Relationships (use, extend, include)
sequence diagram	VariableObject	Object
	VariableLifeLine	LifeLine
	VariableMessage	Message
activity diagram	VariableActivity	Activity
	VariableTransition	Transition

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 four profile packages and we expose the main proposed

concepts and notations to respectively four UML diagram (class, use case, activity 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 modelling. Figure 1 illustrates two packages 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 and sequence diagram profile. Indeed in use case diagram, we extend the main metaclasses of standard UML use case diagram to create new modelling concepts as stereotypes. This extension permits to specify the field of use of use case diagram concepts to be used in modelling 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.



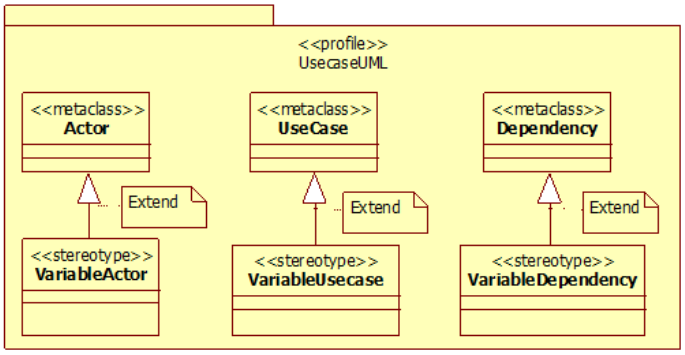


Fig. 1: Packages of class diagram profile and use case diagram profile.

In UML activity diagram, two main metaclasses ("Activity" and "Transition") are extended by two stereotypes (<<VariableActivity>> and <<VariableTransition>>). In variability modelling 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 modelling 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>>). 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 modelling way of variable features and factors that are continuously changing.

A. UML Variability Profile 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.

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.
Use case diagram	VariableActor	Actor	Represents roles that are played by main entities or other systems and subjects
	VariableUseCase	UseCase	Specifies a set of changing actions or tasks that are performed by a system
	VariableDependency	Dependency	Is a relationship which indicates that a model element needs another model element (VariableActor or VariableUseCase)
Activity diagram	VariableActivity	Activity	Is a set of changing actions (or methods) corresponding to operations of a VariableClass stereotype
	VariableTransition	Transition	Indicates the state switch from the end of a VariableActivity to the beginning of following VariableActivity
Sequence diagram	VariableObject	Object	Represents an individual participant (VariableObject or its instance) in the Interaction
	VariableLifeLine	LifeLine	Is associated to a VariableObject and represents the period of life (as a line) for which this VariableObject can participate in the interaction
	VariableMessage	Message	Defines a particular communication between two VariableLifelines corresponding to two interaction VariableObjects.

B. UML Variability Profile 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 modelling 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).

TABLE IV: DESCRIPTION OF THE PROPOSED UML CONSTRAINTS WITH NATURAL LANGUAGE.

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
Use case diagram	VariableActor	It must be extended from the UML metaclass "Actor" It must be related to VariableUseCase by VariableDependency relationship
	VariableUseCase	It must be extended from the UML metaclass "UseCase" It must be related to another VariableUseCase by VariableDependency relationship
	VariableDependency	It must be extended from the UML metaclass "Dependency" It relates two VariableUseCase It shows the kind of existing dependence (use, extend or include) between related VariableUseCase
Activity diagram	VariableActivity	It must be extended from the UML metaclass "Activity" It must be related to another VariableActivity by VariableTransition relationship
	VariableTransition	Is a relationship that relates two VariableActivity
Sequence diagram	VariableObject	It must be extended from the UML metaclass "Object" It can interact with another VariableObject by VariableMessage
	VariableLifeLine	It must be extended from the UML metaclass "LifeLine" It supports the points (where and when) of receiving and sending VariableMessage
	VariableMessage	It must be extended from the UML metaclass "Message" It must start from a VariableLifeLine (of a source VariableObject) to arrive to another VariableLifeLine (of a target VariableObject)

Secondly, we can use UML class diagram to express these proposed constraints such as illustrated in figure 2 that concerns four UML diagrams (class, use case, activity and sequence).

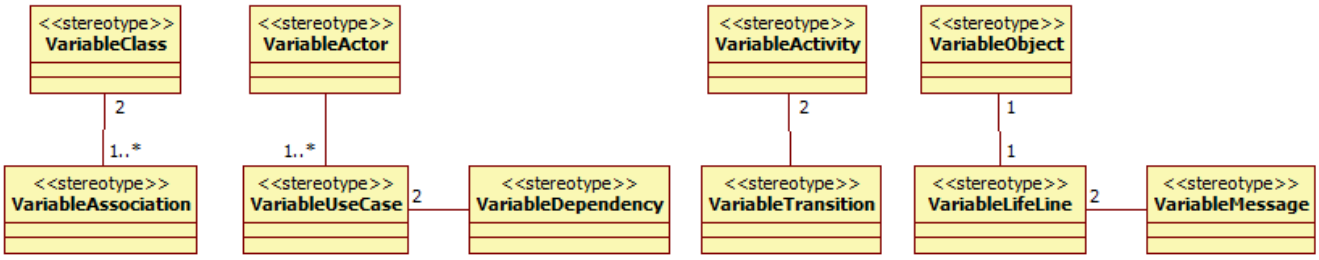


Fig. 2: Representation of use case diagram constraints

C. *UML Variability Profile 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. 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.

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"
Use case diagram	VariableActor	TypeOfActor	"Human", "Software", "Hardware"
		StateOfActor	"Sitting", "Standing", "Moving", "Out", "In", "On", "Off",
		Actual_Location	"Home", "Office", "Hotel"
		Spoken_Language	"English" , "Arabic", "French"
		Used_Money	"Dollar", "Euro", "Dinar"
		Used_Device	"PC", "Laptop", "Mobile", "PDA"
	VariableUseCase	Usecase_Type	"Local", "Distant"
		Usecase_ sharing	"Alone", "Common"
Activity diagram	VariableActivity	Activity_Visibility	"Public", "Private", "Protected"
		TypeOfActivity_Support	"Hardware", "Software", "Both"
	VariableTransition	GuardOfTransition	"Day", "Night", "Indoor", "Outdoor"
		Is_Periodic	"True", "False"
Sequence diagram	VariableObject	Object_Nature	"Internal", "External"
		Object_Type	"Permanent", "Temporary"
		Object_Role	"Creator", "Destructor"
	VariableMessage	Message_Quality	"High", "Medium", "Low"
		Is_Recursive	"True", "False"
		Is_Synchrone	"True", "False"
		Is_Asynchrone	"True", "False"

D. Implementation and Experimentation

UML has been the standard software modelling language that provides various concepts and notations for systems modelling. 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 modelling 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 modelling 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 four packages (ClassUML profile, UsecaseUML profile, SequenceUML profile and ActivityUML profiles) 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. The implementation of the proposed profile will be made on the following steps:

i. Preparation of UML profile file

ii. Creating stereotypes, constraints and tagged values

iii. Extending (or customizing) StarUML menu system

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 as follows (for example: VariableClass):

```
<STEREOTYPELIST>
<STEREOTYPE>
<NAME>VariableClass</NAME>
<DESCRIPTION> VariableClass </DESCRIPTION>
```

```
  <BASECLASSES>
    <BASECLASS>UMLClass</BASECLASS>
  </BASECLASSES>
</STEREOTYPE>
```

```
</STEREOTYPELIST>
```

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

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 (Figure 3) or using OCL language.

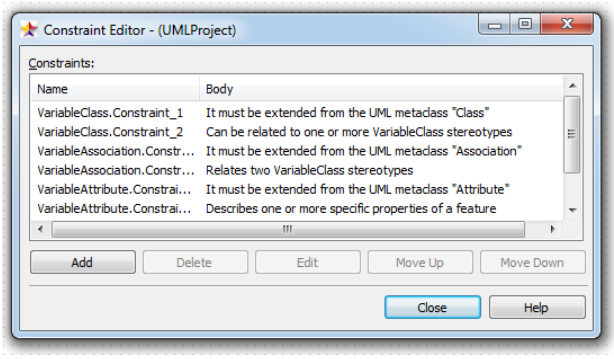


Fig. 3: Defining constraints with StarUML Constraint Editor

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: 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. A global view of the tagged value editor of experimented UML variability profile is shown by figure 4. .

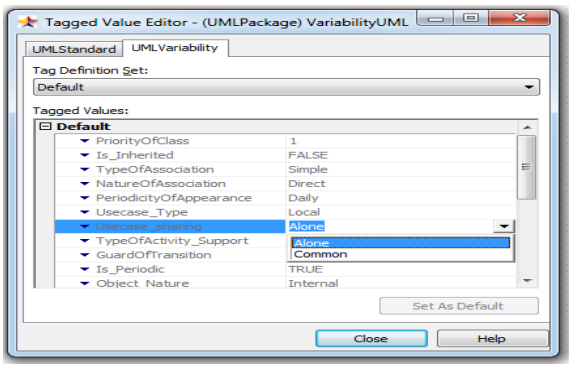


Fig. 4: A view of the tagged value editor of UML variability profile

Figure 5 shows the availability of our proposed UML variability profile to included and used in any UML project.

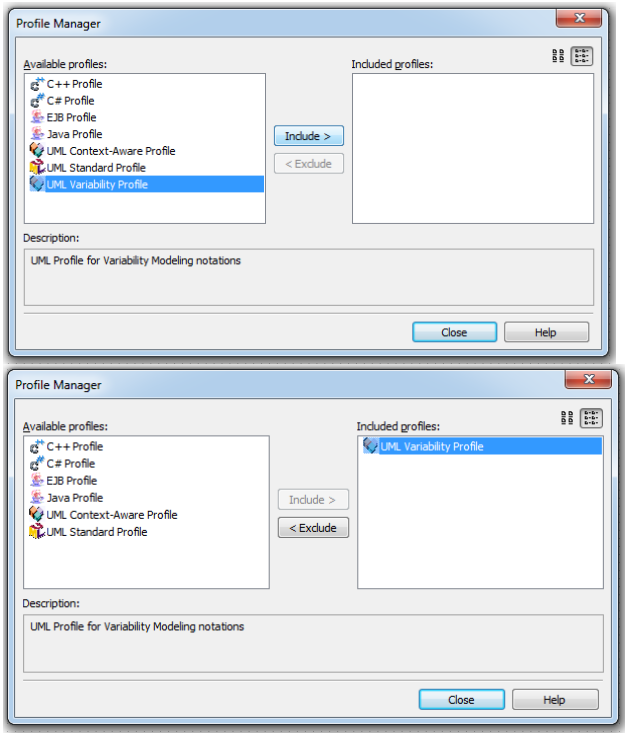


Fig. 5: StarUML profile manage before and after including available UML variability profile

V. Conclusion

In this paper we have proposed some UML extended notations as an UML profile specifically destined for modelling any particular characteristic in variability management domain. The proposed profile provides a convenient and easy modelling 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 modelling specific requirements of variability with an adequate representation. The proposed notations are created by extension of existing UML elements and concern four types of UML diagrams (Class, Use case, Activity 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. Actually 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 notationsas an UML profile specifically destined for modelling any particular characteristic in variability management domain. The proposed profile provides a convenient and easy modelling 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 modelling specific requirements of variability with an adequate representation. The proposed notations are created by extension of existing UML elements and concern four types of UML diagrams (Class, Use case, Activity 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. Actually 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.

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

[3] 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

[4] 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 https://doi.org/10.1007/s10270-021-00890-1

[5] 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 https://doi.org/10.1007/s00607-022-01091-4

[6] 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 https://doi.org/10.1109/IBCAST51254.2021.9393223

[7] L. Addazi and F. Ciccozzi, "Blended graphical and textual modelling for UML profiles: A proof-of-concept implementation and experiment", *Journal of Systems and Software*, vol. 175,110912, May 2021 https://doi.org/10.1016/j.jss.2021.110912

[8] R. Wei, A. Zolotas, H.H. Rodriguez, S. Gerasimou, D.S. Kolovos and R.F. Paige, "Automatic generation of UML profile graphical editors for Papyrus," *Software and Systems Modeling*, vol. 19, pp. 1083–1106, August 2020 https://doi.org/10.1007/s10270-020-00813-6

[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 https://doi.org/10.1145/3624007.3624060

[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 https://doi.org/10.3897/jucs.97113

[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 https://doi.org/10.5539/cis.v16n1p1

[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
https://doi.org/10.1007/s10664-021-10112-3

[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] M. Daniel, N. Lawrence and K. Michael, "Embedding Quality into Software Product Line Variability Artifacts," *International Journal of Software Engineering & Applications*, vol. 12, no. 2/3, pp. 11-25,May 2021
https://doi.org/10.5121/ijsea.2021.12302

[15] 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
https://doi.org/10.1145/3365438.3410935

[16] 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

[17] 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
https://doi.org/10.1109/DEXA.2007.96

[18] O. Ferchichi, R. Beltaifa, R. Mazo and L.L. Jilani, "A SysML-based Holistic Variability Modelling of Software and Systems Product Lines." *34th International Conference on Computer Applications in Industry and Engineering, CAINE 2021*, Virtual online, pp. 99-112, October 2021
https://doi.org/10.29007/fh6k

[19] H. Marouane, C. Duvallet, A. Makni, R. Bouaziz and B. Sadeg, "An UML profile for representing real-time design patterns," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, Issue 4, pp. 478-497, 2018
https://doi.org/10.1016/j.jksuci.2017.06.005

[20] 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

[21] J. Maazoun, N. Bouassida and H. Ben-Abdallah, "Variability modeling with a SPL-UML profile,"14th International Conference on Software Engineering Research, Management and Applications, pp. 201-207, June 2016
https://doi.org/10.1109/SERA.2016.7516147

[22] T. von der Maßen and H. Lichter, "Modeling Variability by UML Use Case Diagrams," *Proceedings REPL'02 , International Workshop on Requirements Engineering for Product Lines*, pp. 19-25, September 2002

[23] S. Robak, B. Franczyk and K. Politowicz, "Extending the UML for modelling variability for system families," International Journal of Applied Mathematics and Computer Science, vol. 12, issue 2, pp. 285-298, 2002

[24] C. Matthias, "Modeling variability with UML," 2001

[25] L. Daizhong and D. Shanhui, "Variability modeling for software product line," BioTechnology , An Indian Journal, vol. 10, issue 14, pp. 8118-8125, 2014

[26] T. Berger, S. She, R. Lotufo, A. Wasowski and K. Czarnecki, "Variability Modeling in the Systems Software Domain," GSDLAB TECHNICAL REPORT 2012–07–06 Version 2, March 2013

[27] M. Sinnema and S. Deelstra, "Classifying variability modeling techniques,"Information and Software Technology, vol. 49, pp. 717–739, 2007
https://doi.org/10.1016/j.infsof.2006.08.001

[28] K. Czarnecki, P. Grünbacher, R. Rabiser, K. Schmid and A. Wasowski, "Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches," Sixth International Workshop on Variability Modelling of Software-Intensive Systems, Proceedings VaMoS'12, January 2012
https://doi.org/10.1145/2110147.2110167

[29] T. Berger, R. Rublack, D. Nair, J.M. Atlee, M. Becker, K. Czarnecki and A. Wasowski, "A Survey of Variability Modeling in Industrial Practice," VaMoS'13, January 2013
https://doi.org/10.1145/2430502.2430513

[30] A. Phatak and G. Robinson, "Understanding and Modelling Variability: Practitioners' Perspectives," *International Statistical Institute*, 55th Session 2005

[31] E. Holger, C. Kröher and K. Schmid. "An Analysis of Variability Modeling Concepts: Expressiveness vs. Analyzability," *International Conference on Software Reuse*, vol. 7925, pp. 32-48, June 2013
https://doi.org/10.1007/978-3-642-38977-1_3

[32] S. Suloglu, M.C. Kaya, A. Karamanlıoğlu, S. Entekhabi, M.S. Nikoo, B. Tekinerdogan, and A.H. Doğru, "Comparative Analysis of Variability Modelling Approaches in Component Models*." IET Software*, pp. 437–445, 2018
https://doi.org/10.1049/iet-sen.2017.0202

[33] A. Asif and Q. Abbas, "Comparison of variability modeling techniques," M.S. thesis, Jönköping University, JTH, Computer and Electrical Engineering, June 2009