International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:10, No:1, 2016

A Generic Approach to Reuse Unified Modeling Language Components Following an Agile Process

Rim Bouhaouel, Naoufel Kraïem, Zuhoor Al Khanjari

***Abstract*—**Unified Modeling Language (UML) is considered as one of the widespread modeling language standardized by the Object Management Group (OMG). Therefore, the model driving engineering (MDE) community attempts to provide reuse of UML diagrams, and do not construct it from scratch. The UML model appears according to a specific software development process. The existing method generation models focused on the different techniques of transformation without considering the development process. Our work aims to construct an UML component from fragments of UML diagram basing on an agile method. We define UML fragment as a portion of a UML diagram, which express a business target. To guide the generation of fragments of UML models using an agile process, we need a flexible approach, which adapts to the agile changes and covers all its activities. We use the software product line (SPL) to derive a fragment of process agile method. This paper explains our approach, named RECUP, to generate UML fragments following an agile process, and overviews the different aspects. In this paper, we present the approach and we define the different phases and artifacts.

***Keywords*—**UML, component, fragment, agile, SPL.

I. Introduction

ONE of problems in software engineering is the complexity of the software development process. This complexity of process affects the quality and the cost of the software. Therefore, the software engineering world attempts to manage this complexity by adopting reuse approaches. It is recommended to manage the reuse in the earlier phases in the development process. MDE aims to abstract and express those phases using models. One of the most widespread models is the UML. Therefore, we need to reuse the elaboration of UML models. In software engineering, we detect two families of development process: The classical family and the agile family. In our works, we are interested on the agile process development that provides more flexibility on development activities. As we mentioned, we need to provide reuse. The software product line is a reuse approach. It optimizes the construction of products by leveraging their common characteristics and managing their differences in a systematic way [3]. Based on these observations, the objective of our research is to define an approach that generates UML components based on agile methods process. The components contain UML fragments constructed for a specific context of use. We define UML component as a set of fragments UML which serves a business objective. The construction of the UML component is not ad hoc; it respects the activities of an agile process fragment. In our work, we have chosen to model agile variations with an SPL from which we derive the process fragments. The construction of the components must be directed and oriented by agility. In fact, we must determine transform of the user's requirements to a labeled UML model (UML fragment). The approach is based on three major domain of engineering, MDE, SPL and the agile methodology.

A. Motivation and Contribution

Various methods have been proposed to support the model generation. Therefore, we have established in [1] a framework comparison for the model generation methods based on four facets: Usage world, nature world, development world, and system world. An empirical study was presented in previous work [1], to outline limits of existing methods generation methods. We have applied the comparison framework among the most referenced model generation methods (UMLAUT, FUML, RSM, FUJABA) according to facets. This comparison revealed particularly three limits [1]

1) The existing methods automate the generation of models but do not decompose them as fragments. Generally, to reuse a model, we do not need the whole model but a specific part named fragment. This deficiency will be one of our contributions.
2) The four methods presented are restricted in a vertical transformation (from a type of UML diagram to another). We cover by the approach presented in this paper the vertical and the horizontal transformation (refinement of an UML diagram).
3) The method generation model is not guided by a development process. We aim to guide the generation of UML fragments by following an agile process method.

Considering all these issues, we present in this paper the meta-model of the approach that generates fragments of UML models. To ensure reuse, we construct a component of UML fragments used on specific situations.

The rest of the paper is organized in four sections as follows: Section II provides a brief literature review of the different approach of method engineering, then we represent the dimensions of evaluation of those approach and we define the fragment agile process. In Section III, we present the approach and explain the different elements and phases.

Rim Bouhaouel is with the RIADI Lab, National School of Computer Sciences of Manouba, Tunis, Tunisia (e-mail: bouhaouel.rim@gmail.com).
Naoufel Kraïem is with the Department Computer, Sultan Qaboos Uniuversity, Muscat, Oman and RIADI Lab, National School of Computer Sciences of Manouba, Tunisia (corresponding author, e-mail: naoufel@squ.edu.om).
Zuhoor Al Khanjari is with the Esprit School of Engineering, Tunis, Tunisia (e-mail: zuhoor@squ.edu.om).

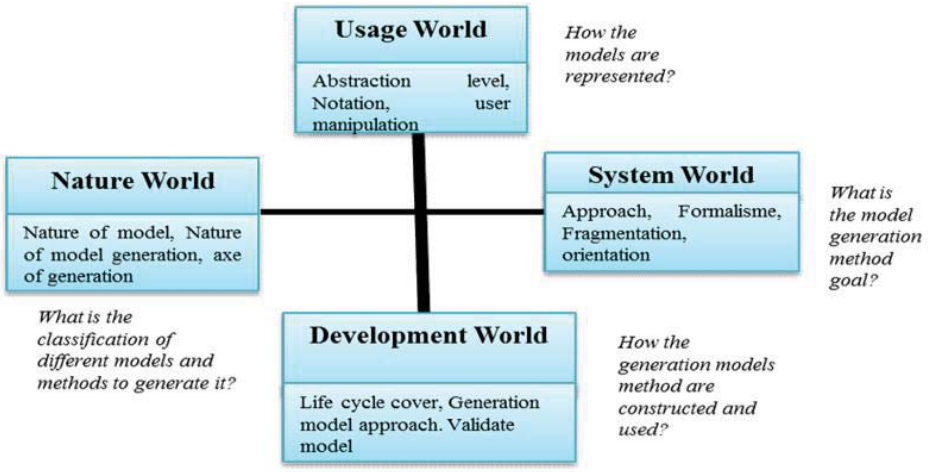

Fig. 1 The framework comparison for the model generation methods [1]

II. Literature Review

Our work aims to construct method to generate UML fragments based on agile process. Therefore, we decide to elaborate a literature review of the different existing approaches of engineering construction method.

A. Model Driven Engineering (MDE)

The purpose of MDE is to accelerate and facilitate the construction of complex software using objects named models. MDE uses models to represent the elements of software system and expresses solutions and phases in the process of software construction (such as requirements, designs, data structures, scenarios, and code). According to [8], MDE is composed from two principal aspects: The Domain Specific Language (DSL) and the transformation engines and generators [7].

B. Model

The use of model is not related to the recent science or to software engineering world. Regardless of the domain or the age of use, the goal stays the same. It facilitates the representation of thoughts or concepts. Usually, when we explain an idea, we find words insufficient and the exhaustive description with all details is onerous.

In literature, there are many definitions of the model, it can be defined as a "simplification of a system built with an intended goal in mind" [6]. The model should be able to answer questions in place of the actual system. The model driven architecture (MDA) defines a model of a system as [9] "a description or specification of that system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. The text may be in a modeling language or in a natural language". So, a model is built in an area of representation that captures a set of common elements or concepts and establishes relationships between them.

Although there is no consensus that defines model, all the definitions agree that a model represents a semantic concept represented with a set of symbols or elements; each one explains a part of the concepts and notations.

C. Approach of Method Engineering Construction

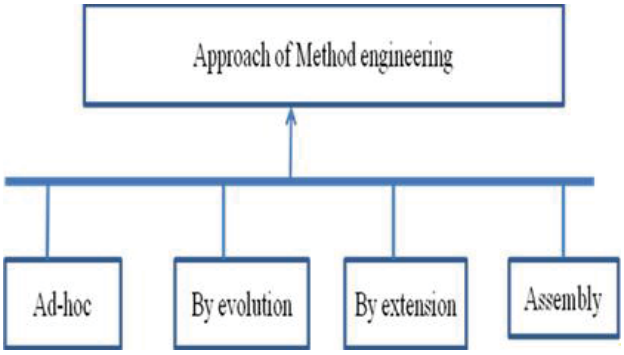

Fig. 2 Typology of construction methods approaches

For the construction of methods, we detect 4 approaches in the literature as shown in Fig. 2. The first is the composition using an ad hoc approach, non-guided and without user's experience. It is adequate if we need to start the composition of methods from scratch. The second is the assembly approach; it is based on the selection of fragments of different methods corresponding to the current situation. The third approach is by extension, it is used to enrich a method through a new concept, or a new property. The fourth is an approach by evolution. This type of construction method requires a basic pattern.

D. Approach Evaluation

Based on literature, there are four dimensions to evaluate composition methods: Perspective, abstraction, and granularity [2].

1. Perspective

The perspective dimension handles methods from the product and the process point of view. Some approaches separate the product perspective and the process perspective to offer two types of components: Those of the product and the process. Our approach constructs an UML component composed of UML fragments. This assembly will be guided with an agile development process. According to this definition

our work is localized on the fragmentation of component and the fragmentation of process. It adopts the both of perspectives.

2. Abstraction

Reference [5] defines two levels of abstraction: Conceptual and technical. The conceptual level consists to fragment the design methods. The technical level offers fragments that represent the operational specifications of methods, i.e. Tools. Our UML fragment ensure the modeling both of the technical aspects and design concepts. We consider that we cover the two dimensions of abstraction concept.

3. Granularity

The granularity dimension covering engineering methods is based on levels of decomposition. We select four useful levels that can be applied to our fragmentation method: Step, Model, Diagram, and Concept. The Step levels for segments in the life cycle of an information system. This level is specified by the agile process and life cycle as the escutcheon. But generally, the modeling will be in the stage of analysis and design. The model level takes into account the perspective of an information system. We consider the product as a model, since it represents a business regardless of its presentation. A fragment of type Diagram corresponds to a possible representation of a component of model type. The representation of the product concept is made by the product modeling. That's why we cover the model level. Finally, the concept level caters to concepts and associations that exist between them in the level diagram.

III. Overview of the Approach

A. Principle

To guide the generation of component of UML fragments, UML methods using an agile process, we need a flexible approach which adapts the agile changes and which covers all its activities. The process that will guide the generation of UML diagrams is not scalable. The components are constructed in accordance with a fragment of a well-defined process [4], and to serve two goals: refinement or transformation from one component to another. We can define the transformation when we have the same situation and the same intentions but different fragment modeling product. This approach will ensure coherence between the components constructed and the agile process. In fact, the fragment is depending on the needs of the user to guide the generation of a reusable component expressed by the situational factors. Methods based on the Assembly descriptor determine the label of the reuse of this fragment. As mentioned in Fig. 3, in our work, we derive a process fragment using a product line for the agile process strategy which we present the different variabilities of the agile process and which we guide the fragmentation of process.

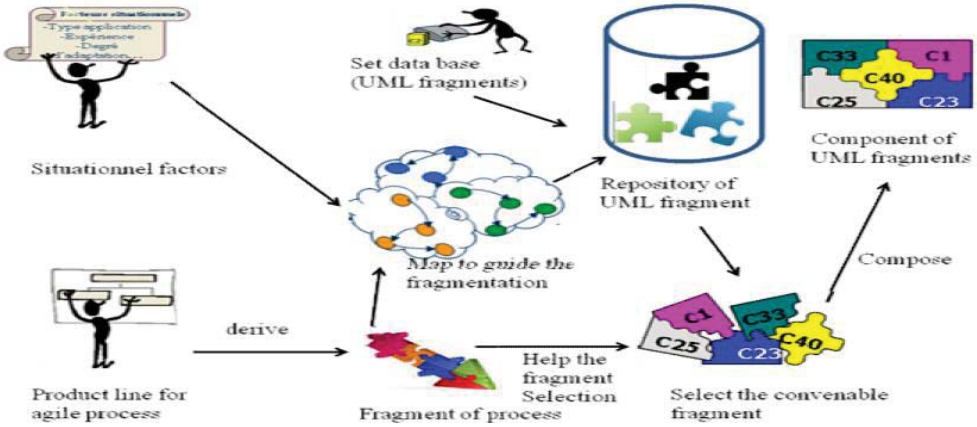

Fig. 3 RECUP representation

The user introduces the situational factors and using the process fragment we attempt to guide the generation of an UML fragment. Then, basing on the fragment of process we select from the repository of UML fragment a set of fragments which we will combine it to construct an UML component. As we explain, the inputs of our approach are the situational factors and the fragment of process. The output is a component of UML diagram. In the next paragraphs we define those artifacts: Situational factor, component, UML fragment and the MAP.

1. Situational Factor

At the beginning of the fragmentation of UML diagram, the user is invited to specify the current situation of the UML component and the intentions of the fragments under this component to modulate through the situational factors. Then, he specifies a set of criteria to derive a fragment of agile process.

B. Product Line for Agile Process

We adopt in our work an approach to generate UML fragments based on agile process. We need to identify the variations existing in agile models and presented in a product line model.

Fragments of the agile process are derived using the software line product configured based on user input. We explore the various agile methods and the differences in UML modeling among their activities. Regardless of the agile method, the process is always based on iterations composed by activities. By analyzing the agile methods, we infer that modeling is generally involved early in this process by refinement the existing diagrams (like SCRUM and KANBAN), or enrichment in case the context of the system is unclear in the first iterations (like XP). We should have an initial fragment which we iterate to refine the fragment more or to enrich it with new product concepts or product modeling following an activity of the agile process fragment.

Variability of the agile process will guide the Assembly of fragments UML to build a reusable component. The agile process is incremental. We consider the assembling component as an increment.

1. UML Fragment

UML fragment is a single Product concept expressed by an UML Product Modeling. In engineering of method, different nomenclature to express fragment exists. Our definition of fragment is the expression of a product model with UML formalism to attempts a specified intention and includes an agile process. The result is an UML representation that must respect the OMG standard. OMG does not use the term 'product' but 'model' to express an information system [5]. But as we have already explained our goal is to express parts of the information in the form of UML diagram system, we call them "UML fragments". It's description of the concepts in the formalism of UML to meet a goal (intention). However, we make a difference between the concept model, named as "product concept" and the representation target designated by our work by 'product model '.

2. Component

To provide the reuse of the component, we use its context expressed by a reuse interface. The directive is composed by signatures of input type "intentions" and "situation" [10]. The last one express the context of use, e.g. the context may be "buying online", while the intention is electronic payment of purchase.

3. MAP

The Map is a formalism of representation of process model. It's based on different categories: oriented product, oriented decision, oriented activity, oriented context and oriented strategy. We choose to adopt the Map oriented strategy which we can guide the selection of UML fragments by introducing the situational factors. The Map guides the user on the selection of UML fragments to be assembled in the UML component.

IV. Conclusion and Future Works

This paper presents an approach to generate a component of UML fragments based on an agile process. We have focused to present the approach which we define the different artifacts involved.

The generation of UML fragments requires a strategy that covers the modeling of several product concepts. We need to collect concepts by assembling them from a product model and expresses knowledge in a particular business domain. The collection of the concept must be guided with a contextual strategy. The reuse of component is based on Assembly strategy. The composition of component must be guided by the user's experience. Therefore, in our future work we will focus in the assembly strategy of agile process fragment and UML fragment to construct the component. Then we will develop a prototype that implements the meta-model approach presented in this paper.

References

[1] R. Bouhaouel, N. Kraïem, C. Salinesi, Z. Al-khanjari, S. Ouali, Framework to compare the model generation methods. Published in Asian Journal of Scientific Research, 2015 (AJSR -scopus).
[2] N. Kraiem et al. (2008). 'Methods Engineering for new trends of development a computer applications ". CPU. ISBN:1 - 964832-25-7 (2008).
[3] Clements, P. & Northrop L. 2001. Software Product-lines: Practices and Patterns, Addison-Wesley.
[4] S. Brinkkemper, M. Saeki, F. Harmsen, Assembly Techniques for Method Engineering, Proceedings of the 10th Conference on Advanced Information Systems Engineering, CAiSE'98. PisaItaly, 8-12 June, 1998.
[5] Meta-Object Facility (MOF) Specification, Knowledge Discovery Metamodel (KDM)version 1.4.
[6] J. Bezivin, 2005. The Unification Power of Models. In Software and Systems Modeling.
[7] Andreas Metzger,2005. A Systematic Look at Mode lTransformations.
[8] Douglas, C. Schmidt, 2006. Model-Driven Engineering. Vanderbilt University.
[9] MDA Guide version 1.0.1, 2003. OMG Document: omg/2003-06-01,
[10] J. Ralyté, Ingénierie des méthodes à base de composants, PhD thesis, University of Paris 1-Sorbonne, 2001.