DOI: 10.15439/2018F358
ISSN 2300-5963 ACSIS, Vol. 15

Model Driven Architecture and Agile Methodologies: Reflexion and discussion of their combination

Imane ESSEBAA
Computer Science Laboratory of Mohammedia,
Faculty of Sciences and Technics Mohammedia,
Hassan II university of Casablanca, Mohammedia
Email: imane.essebaa@gmail.com

Salima CHANTIT
Computer Science Laboratory of Mohammedia,
Faculty of Sciences and Technics Mohammedia,
Hassan II university of Casablanca, Mohammedia
Email: salima.chantit@gmail.com

***Abstract*—Model Driven Architecture (MDA) and Agile Methods (AM) are two principal domains that are in the way of improvement and evolution in order to facilitate the development of IT projects. However, these areas evolve separately despite the great number of research that focuses on improving project' development techniques. Thus, our proposal aims to provide a method describing how can Agile Methodologies benefits from MDA, and how MDA can automate activities within AM. In this paper, we present a state of the art of existing works that combine a Model Driven Architecture approach and Agile Methodologies. Then we present our analysis of this combination to identify with which Agile methods, the MDA approach is more adequate. We also propose our vision about how to combine MDA with the selected Agile Methodologies and evaluate the strengths and weaknesses of each methodology during its combination with MDA and finally we present a case study of Rental Car Agency.**

I. Introduction

THE constant evolution of information system leads companies to search how to improve their productivity, their effectiveness and their profit margins in order to stay competitive. They are in a permanent quest for reliable tools that will cover the maximum of features.

In this context, two major domains have emerged in recent years and made an important place in companies' business: Model Driven Architecture and Agile Methodologies. On one hand, MDA has emerged as a new paradigm of software development that tends to use models as main artifacts in a higher level of abstraction, as well as the separation of the functional and technical specification. Indeed MDA has changed the view of software development: while classic methods concentrate on writing code, the MDA proposes a new method that focuses on analysis phases.

On the other hand, Agile Methods focus on the definition of best practices of information systems programming and their integration in the development process. It is an approach that defines a disciplined management of software development projects: Agility recommends an iterative and incremental method to develop software systems.

Agility puts the customer at the center of the development process of a project [1] and aims to develop software projects in the shortest possible time that satisfies all customer requirements and take into account requirements change, which is a fundamental principle of Agility.

Several works have been made on these two domains that help them to evolve and improve but separately. However, few of them focus on how to combine MDA and Agility.

The main idea in this paper is to present a state of the art of previous studies made in the context of combination of MDA and Agility.

We propose in this paper an analysis of Agile methodologies to define which are more appropriate to combine with MDA. We also give propositions on how to combine MDA and some Agile methodologies that we find appropriate to this combination.

This paper is organized as follows: after this introduction, the second section contains an overview of concepts in which this work is based, namely: MDA, Agile Methodology and RUP. Section 3 presents related works made in the context of combination of MDA and different agile methodologies. In section 4 we describes our analysis and proposed approach to combine MDA with V lifecycle in Scrum sprints. Section 5 is reserved to some perspectives of our future works, and finally we finish by a conclusion.

II. OVERVIEW OF CONCEPTS

A. *Model Driven Architecture*

The MDA (Model Driven Architecture) is an initiative of the OMG (Object Management Group) released in 2000 [2] The basic idea of the MDA approach is the separation of the functional system specifications and its implementation on a particular platform.

The MDA approach lies in the context of the Model Driven Engineering which involves the use of model and meta-models in the different phases of development lifecycle of an application[3], MDA defines three viewpoints:

- CIM (Computation Independent Model): the objective of this model is to represent the application in their environment independently of any computation information.
- PIM (Platform Independent Model): the role of the PIM is to give a static and dynamic vision of the application regardless of the technical conception of it.
- PSM (Platform Specific Model): This model depends on technical platforms, it represents a template of code that facilitates code generation.

B. Agile Methodologies

The ' Agile Manifesto ' published in in February 2001 [4] based on analysis of previous experiences that allow to propose good practices to developers, The agile principle introduced by the agile manifesto is related to time invested in analysis and design.[5]

Agility, a paradigm for a new vision of an organization, asserts itself as an alignment and coherence tool between internal forces and external challenges that give dynamism to an enterprise [6]. Agile methodology is a loom to project management, classically used in software development to manage IT projects development. It helps teams to respond to the changeability of building software through incremental, iterative work cadences, known as iterations.

III. STATE OF THE ART

The basic idea behind both Model Driven Engineering and Agile Methodologies is to create systems that can respond quickly to frequent changes, they propose different approaches resolve mentioned requirements; the agility focus on a methodological aspects that concerns an individual product, while Model Driven Engineering is more concerned by an architectural aspect defined by its specific variant MDA (Model Driven Architecture) that aims to separate system features from its implementation in technical platform.

Being aware of the importance of the agility and MDA in the development of software system, many works were focused on combining these areas, in this section we present some works previously made on this context.

In their paper [7] S.Hansson and al. collects different works made in practice on the context of the Model driven Agile Development and analyse the result of each approach, this approach consider that the basic idea behind the Model Agile Development approach is to benefit from practices proposed by agile methodologies, authors of this paper summarize the empirical literature made in the MAD context in order to extarct the lacks of this domain.

P.Cáceres and al. proposes in their paper [8] a case of study of an Agile Model Driven Development integrated in MIDAS framework which combines Model Driven Architecture approach and Agile practices based on eXtreme Progrming (XP). MIDAS is a model driven methodology for Web Information Systems (WIS) agile development, the architecture of MIDAS combines both MDA and n-tiers architecture, this combination is in order to propose a model of a WIS respecting the independence of the platform according to the MDA and in the same time take account of a middleware architecture of the Web services development platforms, while the process of the MIDAS methodology framework is based on Agile Modeling Driven Development practices that aims to write code progressively in agreement with the model. We mention that authors in this paper details the architecture of the MIDAS framework while it does not explain how the Agility is integrated in the process of MIDAS tool, moreover we note that the XP practice is dedicated specifically to the development phase during the software system development, which allows us to note that this approach does not implement all the aspect of the MDA approach.

In their paper [9] M.B.Nakicenovic presents an Agile Model Driven Development process developed in consideration of lean and agile practices, the paper aims to provide an approach that shows that MDD and agility can work together exploiting the benefits of each domain, the approach is applied on both forward and reverse engineering in order to respond to two issues; accelerating the re-engineering process of the MDD solution, how benefit from agility and lean while producing MDD solution within a short time frame. The paper describes an approach that combines MDD and agility based on lean, the implementation of the approach was made on the Market Server Capabilities (MSC) project proposed by SunGard company.

F.P.Basso and al. presents an approach that combines a Model Driven Architecture approach and Agility in the context of Rapid Application Prototyping (RAP) [10]; RAP allows the validation of software requirements before acceptance tests which in order to obtain a quick feedback from clients. This approach aims to take account of the agility principles in the context of MDA based on RAP methodology to generate front end and models based on MVC pattern, the implementation of this approach was applied on the generation of the Web Information System based on scrum methodology and MDE practices. The authors aims to ensure several benefits within this approach; better organization of source code, simplicity of changing the source code, simple and rapid design of models, and they still expecting to benefits from reuse of designed models. We note that this approach can't be generalized to all types of software systems, indeed it was dedicated to develop Web Information System, we also mention that in this approach authors do not detail how to integrate the MDA and scrum, i.e. they do not propose where to use each level of MDA in scrum methodology.

V.Kulkarni and al. discuss and argue in their paper [11] why agile methodology can't be used with Model Driven Engineering, then they propose a modification to make on agile methodologies in order to combine them with MDE. Indeed this paper describes a new Software Development process that combines Scrum and MDE, in this approach authors proposed the use of Meta-Sprints that run in parallel to Sprints in order to validate models, they suggest two to three months as timescales for meta-sprints where clients must provide feedback on models and prototyping , which is opposite to agility principles; indeed agility recommends that the feedback of clients must be in period less than what was proposed in this approach.

H.Alfraihi in its paper [12] analyses the challenge of combining Agility and Model Driven Development, the paper describes an approach that aims to increase the adaptability of these domains by proposing a framework that facilitate Agility and MDD, this approach proposes recommendations, guidelines, and procedure to can use Agile MDD in practice. We note that even if this approach proposes some practices to implement the Agile MDD it does not take account of the architecture of the MDD, Model Driven Architecture, and how to benefit from the different abstraction levels to produce sustainable software systems.

In the paper, H.Wegener [13]presents a study made on the context of the combination of agility and Model Driven Development, then to propose issues that show how this combination affect organizations, process and architecture, this paper presents a comparison of different approaches proposed to use Agility and Model Driven Development.

In their paper [14] V.Mahe and al. presents their first reflections about the fusion of the MDA and Agility in order to have a combination with improved properties than the additions of the two approaches, they propose a canvas based on processes and agile practices in both modeling and meta-modeling level.

V.Nikulsins and al. propose an implementation of the MDA into RUP as presented in figure bellow:

- CIM covers the Business Model and requirements as well as planning for the development of the PIM metamodels.
- Elaboration is the main phase impacted by the MDA project, which is covered by PIM.
- PIM also cover part of the construction.
- In the construction phase the transformation of PIM model to PSM model starts.
- PSM covers transition phase

Burden et al. [15] have conducted a systematic literature review by proposing two research questions with the goal to investigate the empirical evidence of the state of art of integration of agile and MDD approaches and what is lacking in that area. The study shows that Agile MDD is still in its early stages and there is a need for detailed experience reports.

In their paper [16], H.Alfraihi and al. presented a systematic literature review to complement the results of [15] where fifteen primary studies were reviewed. The main characteristics of Agile MDD approaches besides their benefits and problems are highlighted. Both systemic literature review studies provide broader coverage, but they are less in-depth than interview study.

Eliasson and Burden [17] have conducted an exploratory study at Volvo Car Corporation (VCC). At VCC, individual teams adopt Agile practices with MDD while the organisation at large still use plan-driven process. The aim of this exploratory study is to investigate how Agile practices can be extended to the organisation level. In specific, it aims to answer the following question: Which are the challenges and possibilities for a more Agile software development process on a system level?. They interviewed 17 engineers to identify the challenges of the current process at VCC and how to improve it. The results of the interviews revealed two main challenges: first, the developers have to wait long before getting feedback which forced them to make premature assumptions leading to unwanted side-effects and faults; second, the use of MDD tools force developers to employ a waterfall process. The main finding of this study is that there is a need for a more Agile way of working to obtain earlier and faster feedback. They conclude that Agile MDD can be useful in automotive development. However, their study is context specific to VCC to examine its case and limited to automotive domain which is difficult to be generalised.

IV. OUR PROPOSITIONS

To ensure the good combination of MDA and agility we have to answer four questions:

- What are the reasons that motivate integrating Agile practices and MDD processes?
- How are Agile practices and MDD integrated?
- What are the benefits of integrating Agile and MDD on development process?
- What are the challenges of integrating Agile and MDD?

In their paper R.Matinnejad [18] defined that the integration of Agile in MDD process can be developed by:

- MDD-based: introducing Agile method to a current MDD process which is called.
- Agile-based: applying MDD process to an agile method.
- Assembly-based: integrating some fragments from Agile and other from MDD to develop the process

Although, both Agile and MDD processes have been introduced for more than a decade, the basic principles on how to integrate them together are not well-known. In this regard, the use of agile and MDD do not follow a well-defined process or systematic guidelines to guide through development. As a consequence, development teams introduce practices of Agile and MDD in an ad-hoc manner or based on their personal experiences.

In following, we discuss our reflection and analysis about different software methodologies and also agile ones and the possibility of their combination with MDD depending on each methodology criteria. To this end, we choose the most popular and used methodologies:

- eXtreme Programing
- 2TUP
- RUP
- V Life Cycle
- Scrum

A. *eXtreme Programing*

eXtreme Programming is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.

The eXtreme Programming is a software development discipline that organize people to develop higher quality software system and be more productive. XP defines four activities that are performed with software development process: Coding, testing, listening and designing. [19]

Analysing the definition and principles of the eXtreme Programming we deduce that this process is more appropriate to development phase in the process of software development.

According to XP and MDA approaches principles we deduce that it is not possible to combine both of these domains taking into account the fact that XP is dedicated to manage the development phase while MDA approach is an architecture based on different level of conception using models to finally generate automatically the source code.

B. *2TUP: Two Track Unified Process*

2TUP is an instantiation of the UP (Unified Process) proposed by Valtech company, it takes into account constraints of rapid business changes of software system.

The fundamental principle of the 2TUP is to decompose and treat at the same time the business requirements, following two axis; functional one and technical one. At the end of the evolutions of the functional model and technical architecture, software development phase consists on combining the results of these two branches of the process as described in the figure bellow.

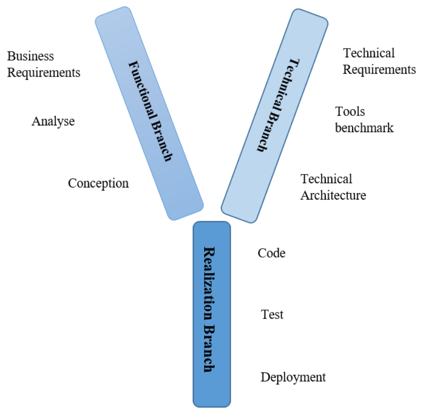


Fig. 1: 2TUP methodology

Analyzing 2TUP methodology we conclude that it can't be combined with MDA, indeed the MDA approach proposes to generate technical conception from functional one through model transformations which are the core of MDA approach. There are two types of model transformations; Model to Model (CIM to PIM and PIM to PSM) , and Model to Text (PIM to source code), while in 2TUP methodology the functional and technical conception work in parallel to finally combine their results in the development phase.

C. *RUP: Rational Unified Process*

The Rational Unified Process is a Software Engineering Process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule and budget. [20][21]

It is closely related to the Unified Modeling Language (UML); this model proposes to manage the IT developments using a model of activities divided in 4 categories described in table 1:

TABLE I: Description of RUP phases

Phase	Description
Inception	• Establish a business case of the system • Define different actors and use case
Elaboration	• Analyze the problems of the domain • Establish the system architecture • Define the project plan
Construction	• Develop, test and integrate the components of the project
Transition	• Forward the project to users

The general architecture of the Rational Unified Process is characterized by two dimensions:

- The horizontal axis represents time and shows the progress of the lifecycle of the process; this first dimension reflects the dynamic aspects of the process that is expressed in terms of cycles, phases, iterations and milestones.
- The vertical axis represents major patterns of activities which include activities according to their nature; this second dimension reports the static aspect of the process that is expressed in terms of components, processes, activities, artifacts and workers.

Analyzing the RUP method and its structural phases, we conclude that it is possible to combine it with MDA approach, in the following part of this section we propose our view of implementation of MDA into RUP that consists on:

- CIM level covers the inception phase, this level is modeled by Use Case diagram to represent a static and functional aspect, and Activity diagram to represent a dynamic aspect.
- The elaboration phase is covered by the PIM level obtained from a vertical transformation of CIM model, this level is presented by Class Diagram representing a static view while a dynamic one is represented by a Sequence Diagram.
- PSM level covers a construction phase after enriching a PIM models and transform them using vertical transformation into Design Class Diagram and Interaction Diagram.
- Transition phase is covered by a code obtained with Model to Text transformation.

We summarize our approach in the following table:

TABLE II: Description of RUP phases

RUP/MDA	Inception	Elaboration	Construction	Transition
CIM	• Use Case diagram • SBVR			
PIM		• "Business" Class Diagram • "System" Sequence Diagram		
PSM			• "Design" Class Diagram • "Detailed" Sequence Diagram	
Code				• Source Code

D. Scrum

The Scrum method is an agile method, founded in 2002, It relies on the Division of project into iterations still named "sprints". A sprint can have a duration which varies generally between two weeks and a month.

The estimation of task in time and complexity is made before each sprint using several methods, in order to plan the deliveries and also estimate the cost of each task for the customer.

Features called user stories are the subject of the sprint that constitute a "sprint backlog" that can be a deliverable product at the end of the sprint.

There is difference between the sprint backlog "product backlog", corresponding to all of the features expected for the product on all of the sprints.

The Scrum method is also characterized by a "melee" daily, called "morning" or "stand up", in which employees (project managers, developers and functional managers) in turn indicate tasks that they have performed the day before, the difficulties and finally this whereupon they will continue their work the next day. This allows to evaluate the progress of the project, resources where they are most needed, but also to provide assistance to workers facing difficulties when these have already been encountered previously by other members of the team.

To combine MDA and Scrum we can use in each sprint of the project the MDA principles, i.e. in each sprint we apply our approach of generating source code from business requirements, in scrum we have a sprint backlog that describes system' features, the combination can be described as follow:

- As first step we model the requirements in sprint backlog by the UseCase Diagram and OMG standard SBVR to represent the CIM level.
- After modeling the CIM level, these models are transformed automatically to PIM level which is modeled by "Business" Class Diagram and "System" Sequence Diagram.
- Models in PIM level are also transformed into "Design" Class Diagram and "Detailed" Sequence Diagram in the PSM level
- The last step is the automatic generation of Source Code from PSM level.

This combination of MDA in every sprint of Scrum methodology have many advantages:

- Reduce the duration of the sprint that influence also the total duration of the project.
- Facilitate the management of requirements' changes of the system view that models and code are generated automatically.

E. Combination of MDA and V lifecycle in Scrum

As known scrum methodology proposes best practices to develop a qualitative software system respecting Agility principles, most of time developers prefer to combine scrum with life cycles to ensure the good management of the project development; in each sprint of Scrum, V life Cycle is used to define steps of the development of system, in this part of this section we will propose a combination of MDA and Scrum+V life Cycle.

V model means Verification and Validation model. Just like the waterfall model, the V life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins; based on the requirement document that contains specifications of the system, developer team started working on the design and after completion on design start actual implementation and testing team starts working on test planning, test case writing, test scripting. Both activities are working parallel to each other.

Before presenting the combination of MDA and V lifecycle in scrum we present our MDA approach that consists on :

- Describing system requirements in CIM level by a structured English using SBVR.
- Transforming automatically Business Vocabulary and Business Rules of SBVR into Use Case Diagram (UCD) in CIM level [22]
- Applying transformation rules to generate Business Class Diagram (BCD) and System Sequence Diagram (SSD) from SBVR and UCD of CIM level to represent the PIM level [23].
- Generating PSM level of MVC architecture represented by Detailed Class Diagram (DCD) and Detailed Sequence Diagram (DSD) from PIM level using automatic transformation rules.
- Generating application source code from PSM level.

We mention that the automation of the two last steps of the approach will be implemented as an eclipse plugin which is the continuity of the previous one.

Choosing previous diagrams to model MDA levels is depending on different aspects that each level should cover according to OMG specifications:

- For CIM level, we define three aspects; Static and Dynamic that are covered by SBVR, while the Functional aspect is covered by UCD.
- For PIM level, we define two aspects; Structural aspect covered by BCD and Dy-namic one covered by SSD.
- For PSM level, we define in our approach 4 aspects; Static aspect covered by Model classes of DCD, Structural one covered by Class Diagram, Dynamic aspect covered by Controller classes and Behavioural one represented by DSD.

In this approach, we automate the two types of transformations Model-to-Model (M2M) and Model-to-Text (M2T). For M2M, we use QVT language while for M2T we use Acceleo transformation language. Transformation rules between the different levels of MDA are implemented as an Eclipse plugin to ensure automation and traceability of transformation rules. The figure below describes an overview of our approach:

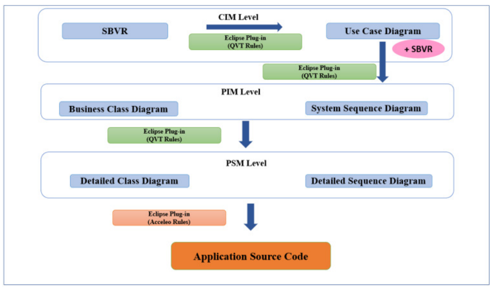


Fig. 2: Overview of our MDA transformation approach

Combining MDA and V life cycle in Scrum can be described as follows:

- Covering Requirements and functional specifications steps in V life cycle by CIM level of MDA which is represented in our approach by SBVR. The UCD and Business Rules generated at the CIM level are then used generate "Validation tests" to validate if the developed system responds to described requirements.
- Generating the High-level design represented in our approach by the PIM level which is generated automatically from CIM level (To generate PIM level from CIM one, we use our approach defined in our previous works (Essebaa and al, 2017). This step is represented by BCD and SSD for each use case element. We then generate "Integration tests" from these diagrams (BCD and SSD) to test the correct functioning between different elements of the system.
- Generating the low-level design represented by PSM level which is modelled by CD and DSD (The approach we propose to automate transformations between PIM and PSM levels will be discussed in our future works). We generate "Unit tests" from this level to test the generated code.

The figure 4 below describes the presented approach:

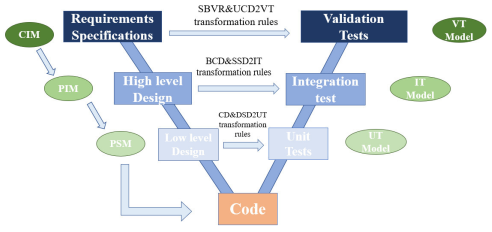

CIM
Requirements Specifications
SBVR&UCD2VT transformation rules
Validation Tests
VT Model
PIM
High level Design
BCD&SSD2IT transformation rules
Integration test
IT Model
PSM
Low level Design
CD&DSD2UT transformation rules
Unit Tests
UT Model
Code

Fig. 3: Combination of MDA and V life cycle

To generate tests from Models in our approach, we defined three main rules that are detailed in our previous work [24]; Rule 1: Generate Validation tests from CIM level: Validation tests are generated from SBVR and Use Case Diagram. Rule 2: Generate Integration tests from PIM level: Integration tests are generated from PIM level which is represented using Class Diagram and System Sequence Diagram. Rule 3: Generate Unit tests from PSM level: Unit tests in our approach are generated from PSM level using Detailed Sequence Diagram and Detailed Class Diagram. The table 3 below summarize these rules:

TABLE III: Test generation rules

Rule	Model	Target
SBVR&UCD2VT	Use Case Element	Requirement to validate
	Fact Type	Sub feature to test
	Business rules of a fact type	Validation tests
BCD&SSD2IT	Actor and DataObject lifecycle	Classes to test
	Relationship between classes	Integration tests
DCD&DSD2UT	Messages	Operation to test
	Operation in classes	Unit tests

In the previous part we present how we automate transformations in MDA and combine with V lifecycle to generate different type of tests, in this part we will present our approach of managing system' requirement using MDA inside a V lifecycle in scrum method. Our proposal is divided into 5 main steps:

- Step 1: Defining system requirement by Backlog Product.
- Step 2: Planning features in a RoadMap.
- Sprint to Begin:
  - Step 3: Apply our approach that combine V lifecycle, MDA and MBT presented in parts 3.1 and 3.2 of section 3.
  - Step 4: Adding code missing parts manually preceding them with "@added" annotation.
- Step 5: Validation and planning of following sprints. In this step we define two cases:
  - If there is no system evolution:
    - Restart from step 3 for the next sprint
  - If there is an evolution:
    - Restart from step 1 and keep the old code except the parts preceded by "@added" annotation of features that still exist in the system (added code of deleted features is deleted automatically after the new execution)

In the figure 5 below we describe an overview of the presented approach in this paper:

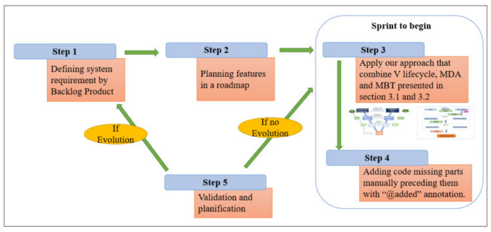

Sprint to begin
Step 1
Defining system requirement by Backlog Product
Step 2
Planning features in a roadmap
Step 3
Apply our approach that combine V lifecycle, MDA and MBT presented in section 3.1 and 3.2
Step 4
Adding code missing parts manually preceding them with "@added" annotation.
Step 5
Validation and planification
If Evolution
If no Evolution

Fig. 4: Overview of a combination of MDA, MBT, V lifecycle in scrum

V. Implementation of our approach

To automate defined transformation rules in our approach, we need tools that allow to create input elements (UML Diagrams), and tools that support the generation of output elements (UML diagrams for PSM level and Source code). After analysing and testing existing tools, we decided to use Eclipse platform with different needed plugins to implement our approach, for example we choose Papyrus Modelling plugin because it supports all UML diagrams elements. According to this we choose to implement our solution as an eclipse plugin that will be a continuity of the previous developed plugin presented in [23] that automate transformations from CIM to PIM. Implementing our approach as an eclipse plugin is in order to facilitate the use of our transformation approach by designers and developers, and also to benefit from existing plugin in Eclipse.

To implement our transformations, we have to use a transformation language; there exist many models of transformations language for both types of transformations M2M and M2T, in our case we chose to use QVT language for M2M transformations and Acceleo language for M2T transformations.

These transformation rules are automated in our Eclipse plugin that take in an input a CIM level and generate PIM, PSM and source code.

The figure 6 shows "Transforms" menu that contains different items that allow the automation of rules:

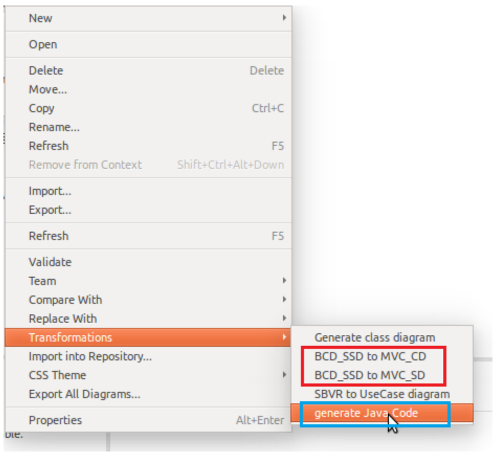


Fig. 5: Transforms menu of our plugin

VI. Case Study

To well illustrate our approach and transformation rules defined, we present in this section their application on a Rental Car Agency system. The case study must provide the following features:

- Visualization of available cars.
- Customers subscription.
- Cars booking.
- Visualization of reservations.
- Management of reservations (accept/decline) by a manager.
- Management of cars.
- Management of customers' accounts.
- Management of Managers' accounts.

The application has three users' profiles that have different privileges:

- Customer: A person who can view the cars available in the agency, rates and promotions and may subscribe. A client must register and authenticate in the system to search for available cars and book a car by indicating the reservation date and time.
- Manager: A Manager must also authenticate to view all cars, add, edit or remove cars. He can also view the bookings made by customers waiting for validation to decide to accept or refuse them.
- Administrator: Once authenticated into the system, the administrator has the privilege of modifying and deleting a customer account, as well as the management of managers account (add, change or delete)

We can also define some management rules as below:

- A customer can rent at least 1 car.
- A car can be rented by at least 1 customer.
- A manager can manage at least 1 car.
- A car is managed by at least 1 manager.
- An administrator can manage at least 1 customer account.
- An administrator can manage at least 1 manager account.

In the following part we present an application of our approach' steps on Rental Car Agency System example:

1) Defining a Backlog product by system requirements: After analyzing system requirements, the first step in our approach is to define the backlog product of the project then plan the RoadMap that describes different sprints of first project' requirement before any evolution, in this example we plan three sprints to develop the system, we define 3 sprints where each one takes 2 weeks. The figure 7 below describes the roadmap of Rental Car Agency system:

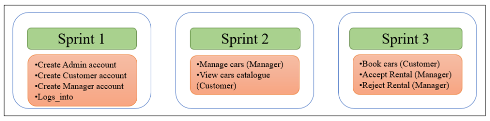
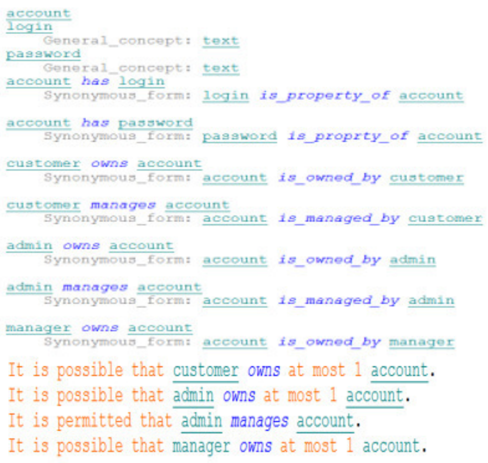


Fig. 6: Scrum RoadMap of Rental Car Agency System

2) Modelling user stories of the first sprint by SBVR and UCD to cover CIM level of MDA: The next step after dispatching features on sprints is to describe CIM level of first sprint by Business Vocabulary and Business Rules using SBVR standard as described in following figure 8.



```
account
login
    General_concept: text
password
    General_concept: text
account has login
    Synonymous_form: login is_property_of account

account has password
    Synonymous_form: password is_proprty_of account

customer owns account
    Synonymous_form: account is_owned_by customer

customer manages account
    Synonymous_form: account is_managed_by customer

admin owns account
    Synonymous_form: account is_owned_by admin

admin manages account
    Synonymous_form: account is_managed_by admin

manager owns account
    Synonymous_form: account is_owned_by manager
It is possible that customer owns at most 1 account.
It is possible that admin owns at most 1 account.
It is permitted that admin manages account.
It is possible that manager owns at most 1 account.
```

Fig. 7: Examples of SBVR of the first sprint of Rental Car agency

In the same level of MDA, we apply horizontal transformation rules, implemented as an eclipse plugin, to automatically generate UCD from SBVR. The figure 9 below represents the generated UCD of the first sprint for Rental Car Agency system.

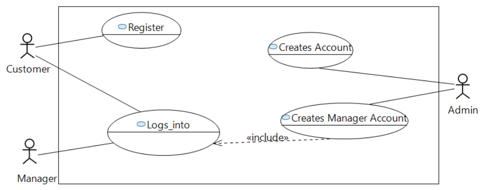


Fig. 8: UML Use Case Diagram of the first sprint of Rental Car Agency System

3) Generating Validation tests from CIM: As in our approach we use V lifecycle process combined with MDA, after defining CIM level, we can generate Validation tests from this level as described for "Logs-into" feature in table 4 below:

TABLE IV: Validation tests generation from PSM level

Source			Target		
Use Case element	Fact Type	Business rule	Requirement	sub feature	validation test
Logs_into	System requests user credential	It is obligatory that the system requests user credential if customer logs into system	The system must allow the customer to logs_into the system	Requests user credential	The system must request user credential if a customer try to logs into the system
	customer sends user credentials	It is necessary that customer sends user credentials if system requests user credential		sends use credentials	The system must allow customer to send user credential
	System verifies user credentials	It is obligatory that the system verifies user credentials if customer sends user credential		verifies user credential	The system must be able to check user credential
	System accepts user credentials	It is possible that system accepts user credential		accepts user credential	The system must be able to accept user credential

4) Applying transformation rules on CIM to generate BCD and SSD of PIM level: Generating PIM level is the first vertical Model-to-Model transformation that aims to automatically generate BCD and SSD from CIM level for Sprint 1 using our Eclipse plugin that implements transformation rules, the figure 10 below represents a BCD of PIM level of Rental Car Agency system:

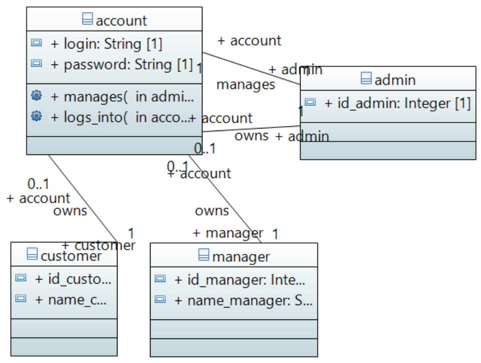


Fig. 9: Generated BCD of PIM level of the first sprint of Rental Car Agency System

The dynamic aspect of PIM level in our approach is represented also by different System Sequence Diagrams, the figure 11 describes System sequence diagram of "logs_into" feature in rental car agency.

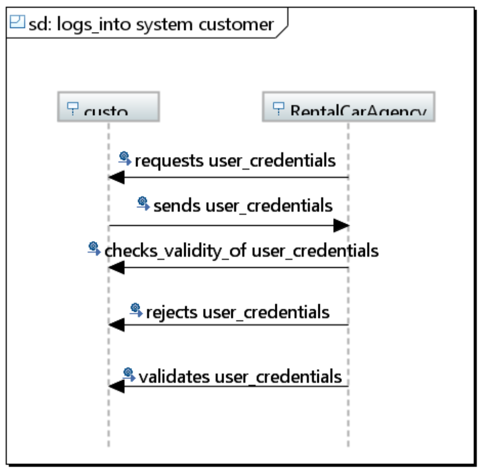

sd: logs_into system customer
custo
RentalCarAgency
requests user_credentials
sends user_credentials
checks_validity_of user_credentials
rejects user_credentials
validates user_credentials

Fig. 10: Generated SSD of PIM level of the first sprint of Rental Car Agency System

5) Generating Integration tests from PIM: According to V lifecycle used in our approach, Integration Tests are automatically generated from PIM level that covers high level design of V lifecycle, the table 5 below describes Integration test for "logs_into" feature in sprint 1:

TABLE V: Integration test generation form PIM level

Source		Target	
Requirements	SD Connection	Classes	Integration tests
Logs_into	The operation requires connection between "Customer" and "Account	Customer	Customer owns 1 account
		Account	Account belongs to 1 customer

6) Applying transformation rules to generate DCD and DSD of PSM level from PIM: The last level before code is PSM level, which is the result of M2M transformations applied on PIM level to automatically generate DCD and DSD, the figure 12 below defines DCD of PSM level of sprint 1:

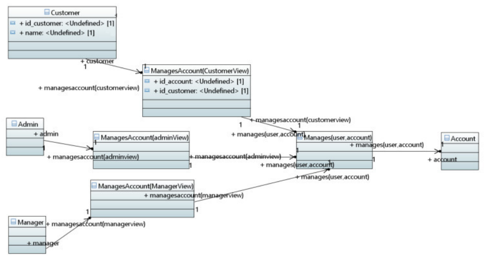

Customer
+ id_customer: <Undefined> [1]
+ name: <Undefined> [1]
ManagesAccount(CustomerView)
+ id_account: <Undefined> [1]
+ id_customer: <Undefined> [1]
Admin
ManagesAccount(adminView)
Manages(user.account)
Account
ManagesAccount(ManagerView)
Manager

Fig. 11: Generated DCD of PSM level of the first sprint of Rental Car Agency System

The dynamic aspect of PIM level in our approach is represented also by different System Sequence Diagrams, the figure 12 describes Detailed sequence diagram of "logs_into" feature in rental car agency.

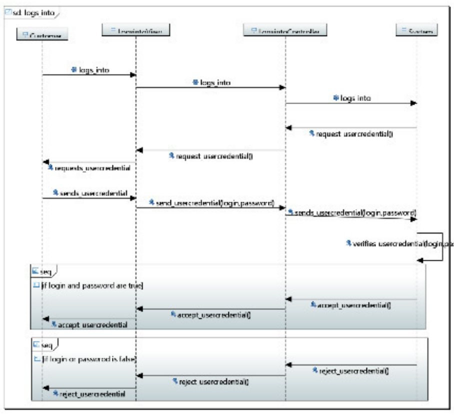


Fig. 12: Generated DSD of SIM level of the first sprint of Rental Car Agency System

7) Generating Unit tests from PSM: Unit tests in our approach are generated from low level design step of V lifecycle covered by PSM level, the table below describes example of Unit test of "logs_into" feature in sprint 1:

TABLE VI: Unit tests generation from PSM level

Source		Target	
Requirements	SD Messages	Operation to test	Unit tests
Logs_into	System requests User_credential	Request(User_credential)	Test "requests" operation
	Customer sends User_credential	Sends(User_credential)	Test "sends" operation
	System verifies User_credential	Verifies(User_credential)	Test "verifies" operation
	System accepts User_credential	Accepts(User_credential)	Test "accepts" operation

8) Generating application source code: The last transformation in our approach is automatic code generation which is the result of M2T transformations that takes as an input a DCD and DSD of PSM to generate as an output source code for MVC web application.

9) Adding missing parts of code manually: The last step in each sprint is to add manually the missing parts of code to complete the system' feature, this code must be preceded by "@adding" annotation.

Evolution of Rental Car Agency system' requirements

In this section we will make some evolutions to the system (addition, deletion and modification of features) in order to visualize the process of models' transformation and test generation in V lifecycle combined in scrum, the evolution will be as follow:

- *Modifications*: "View car catalogue" feature will be available for all users not only customers, this modification engender a new actor "User" that it will be a generalization of "Customer" actor, this modification requires changing the actor of "register" method too.
- *Addition*: In the new system, "Customer" will be able to validate its rental by "payment", The addition of a feature may engender some modifications to old ones, for example the verification of car availability will be made automatically by a system.
- *Deletion*: The addition of "payment" feature requires to delete "Manage rental" feature of "Manager" that allowed him to accept or reject the rental, in the new system the customer can validate its rental from the system, before proceeding to payment option the system must be able to check if the chosen car is available for date specified by the customer.

After studying system requirements' evolution, we have to make another feature dispatching on next sprints as presented in figure 14:

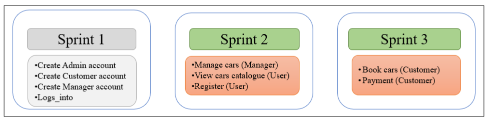


Fig. 13: Scrum RoadMap of Rental Car Agency system after requirements' changes

The next step is to develop the second sprint following same steps previously presented in an example for the first sprint even test cases that will be automatically generated.

VII. PERSPECTIVES AND FUTURE WORKS

This paper is an introduction to our future works on which we aim to combine MDA and Agility in order to propose an new method to develop software systems.

To combine MDA and Agility we aim in our future works to:

- Ensure the traceability between levels to manage requirements changes, as we know among agility principles is the adaptation to changes
- Propose an approach of Agile Model Driven Development appropriate to different types of Software systems (Web, Desktop,...)
- Propose a new Agile methodology that regroup all best practices proposed by other methodologies.

VIII. CONCLUSION

The basic idea in this paper is to combine the MDA approach and Agility in order to propose a new software development method, to well identify and situate the idea in the context, we describe in this paper a state of art of previous approach made in this context.

After analyzing previous works we conclude that proposed methods wasn't implemented to be applied to all types of software systems, we also present our analysis and reflexion on some existing agile methodologies in order to choose those more appropriate to be combined with MDA.

In this paper we propose to combine MDA and Scrum agile methodology in order to improve sprints of scrum and benefit from MDA principles, in this work we proposed to use V life cycle in each sprint of the project where we combine another variant of MDE, to generate automatically different tests applying MBT principles.

As previously mentioned this paper is an introduction to our future works on which we aim to propose a new methodology base on Agility and MDA.

REFERENCES

[1] A. Przybylek and M. Zakrzewski, "Adopting collaborative games into agile requirements engineering," in *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE,*, INSTICC. SciTePress, 2018, pp. 54–64.

[2] J. Miller and J. Mukerji, "Mda guide version 1.0.1." 2003.

[3] R.Soley, "Model driven architecture (mda)," in *http://www.omg.org/cgibin/doc?omg/00-11-05,*, 2000.

[4] K.Beck and al., "Agile manifesto," 2001–2015.

[5] T.Dyba and T.Dingsoyr, "What do we know about agile software development?" vol. 46, no. 5. Software, IEEE, 2009, pp. 6–9.

[6] J. P. Vickoff, "Agile why not?" in *www.entreprise-agile.com*, 2001.

[7] a. H. B. S. Hansson, Y. Zaho, "How mad are we? empirical evidence for model-driven agile development," in *Proceedings of XM 2014, 3rd Extreme Modeling Workshop*. CEUR, 2014.

[8] P. Cáceres, F. Díaz, and E. Marcos, "Integrating an agile process in a model driven architecture," in *INFORMATIK 2004 - Informatik verbindet, Band 1, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm, 20.-24. September 2004*, 2004, pp. 265–270.

[9] M. B. Nakićenović, "An agile driven architecture modernization to a model-driven development solution," *International Journal on Advances in Software*, vol. 5, no. 3,4, 2012.

[10] F. P. Basso, R. M. Pillat, F. Roos-Frantz, and R. Z. Frantz, "Combining mde and scrum on the rapid prototyping of web information systems," *Int. J. Web Engineering and Technology*, vol. 10, no. 3, 2015.

[11] V. Kulkarni, S. Barat, and U. Ramteerthkar, "Early experience with agile methodology in a model-driven approach," in *Model Driven Engineering Languages and Systems, 14th International Conference, MODELS 2011, Wellington, New Zealand, October 16-21, 2011. Proceedings*, 2011, pp. 578–590.

[12] H. Alfraihi, "Towards improving agility in model-driven development," in *Joint Proceedings of the Doctoral Symposium and Projects Showcase Held as Part of STAF 2016 co-located with Software Technologies: Applications and Foundations (STAF 2016), Vienna, Austria, July 4-7, 2016.*, 2016, pp. 2–10.

[13] H. Wegener, "Agility in model-driven software development? implications for organization, process, and architecture," in *OOPSLA 2002 Workshop on Generative Techniques in the Context of Model Driven Architecture,*, 2002.

[14] V. Mahe, B. Combemale, and J. Cadavid, "Crossing model driven engineering and agility – preliminary thoughts on benefits and challenges,," 2010.

[15] H. Burden, S. Hansson, and Y. Zhao, "How MAD are we? Empirical Evidence for Model-driven Agile Development," in *Proceedings of XM 2014, 3rd Extreme Modeling Workshop*, vol. 1239. Valencia, SPain: CEUR, September 2014, pp. 2–11.

[16] H. Alfraihi and K. Lano, "The integration of agile development and model driven development - a systematic literature review," in *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD,*, INSTICC. SciTePress, 2017, pp. 451–458.

[17] U. Eliasson and H. Burden, "Extending agile practices in automotive MDE," in *XM@MoDELS*, ser. CEUR Workshop Proceedings, vol. 1089. CEUR-WS.org, 2013, pp. 11–19.

[18] R. Matinnejad, "Agile model driven development: An intelligent compromise," in *Proceedings of the 2011 Ninth International Conference on Software Engineering Research, Management and Applications*, ser. SERA '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 197–202. [Online]. Available: https://doi.org/10.1109/SERA.2011.17

[19] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change (2Nd Edition)*. Addison-Wesley Professional, 2004.

[20] V. Jacobson, G. Booch, , and J. Rumbaugh, "Unified software development process." Addison-Wesley, 1992.

[21] P. Kurchten, "Rational unified process – an introduction." Addison-Wesley, 1999.

[22] I. Essebaa and S. Chantit, "Tool support to automate transformations from sbvr to uml use case diagram," in *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: MDI4SE,*, INSTICC. SciTePress, 2018, pp. 525–532.

[23] ——, "Tool support to automate transformations between cim and pim levels," in *Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: MDI4SE,*, INSTICC. SciTePress, 2017, pp. 367–378.

[24] ——, "A combination of v development life cycle and model-based testing to deal with software system evolution issues," in *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development - Volume 1: MODELSWARD,*, INSTICC. SciTePress, 2018, pp. 528–535.