UTS: A new modeling methodology with UML and Test case generation, applied in conjunction with Scrum framework

Bouchaib Falah
School of Science and Engineering
Al Akhawayn University
Ifrane, Morocco
b.falah@aui.ma

Soufiane Karroumi
School of Science and Engineering
Al Akhawayn University
Ifrane, Morocco
s.karroumi@aui.ma

Omar Lahkim
School of Science and Engineering
Al Akhawayn University
Ifrane, Morocco
o.lahkim@aui.ma

***Abstract*—The software engineering discipline involves numerous chained processes, the bare minimum being requirements identification, analysis and design, implementation, testing and deployment. These processes can be tackled following different traditional project management methods and approaches, namely the Waterfall, V-Model…. The problems of these linear approaches with the listed processes are the impossibility to generate test cases at an early stage of the lifecycle, with the delayed testing process at later stages, changes become very costly both in terms of budget and human resources.**

This paper introduces a standalone modeling methodology which encompasses the most used UML diagrams, a model-based test case generation technique all applied Scrum Framework, The most used and popular modern project management framework

Keywords—UML, Test Case Generation, Agile, Scrum, Project Management

I. Introduction

Even with UML being the industry standard for software modeling and design, not all the UML diagrams in its metamodel specification are being used, often times than not, researchers and engineers find themselves using just a limited set of diagrams out of all 14 diagrams in the Unified Modeling Language set [1]. On his blog I. Jacobson states "For 80% of all software only 20% of UML is needed". The large UML set of diagrams might confuse beginners in the modeling process and more time will be spent on deciding which diagrams to use for what problems, which is not practical especially knowing how tight project schedules can be.

Another issue with UML is support of testing, this latter is a critical phase in the lifecycle of any software system. Every software needs to go through the testing phase in order to evaluate the level of conformance of the actual software output with the expected output which was initially described in the requirements. One way of doing so is by using Test Cases, which can also be called Test Suites, some of the various test case generation techniques are:

- Model-based Testing
- Goal-oriented approach
- Specification-based approach
- Source-code-based approach

Even with a proper, filtered set of UML diagrams and a good model-based test cases generation technique, if we are still in the context of traditional software management, the outcome will not be as fruitful. Traditional software development methods (V-Model…) are no longer suitable, they are heavy-weight processes focusing more on the industrialization and standardization. To deal with this outdated methods, Agile development methodologies have emerged and have drawn increasingly more attention because of the way they handle the processes of software projects.

For all the previously mentioned problems, this paper presents a standalone modeling approach featuring:

- **A selection of the most used UML diagrams -or what could be referred to as "essential UML"-** containing only a small set of the diagrams and not all 14 diagrams. The choice of which diagrams are the most used and which ones are the less used is based on an investigation of the following sources: The books about the UML, IT UML University courses, Tutorials, Tools for UML models production, conducting a personal opinion survey [2]: asking different roles (academics and researchers, IT practitioners) about their usage, which parts of UML they know and which they have never used.

- **Test case generation from UML diagrams:** a model-diagram-based testing technique, it will rely on the previously selected UML diagrams (most used ones) to generate test cases which are also called test suites. This way, UML will be also used for a purpose different than its origin and it will start playing a significant role in testing phase too, helping project team determine the conformance of the test output with the described requirements, the test case generation process will be shifted to earlier steps of the software engineering process.

- **A combination of Scrum -Agile framework- with the chosen UML diagrams and their corresponding test case generation techniques:** this is an attempt to map the UML modeling technologies with the practices of the most popular

and used Agile framework, that is Scrum. This combination is called the Scrum-UML modeling approach and it is aimed to provide a modern project management environment for the UML modeling process; focus on rapid and frequent deliverables or what can be called "A Working Piece of Software".

This way, the modeling task will be done in an iterative and an incremental manner alongside test case generations: Software modelers will have "Essential UML" instead of the entire UML diagrams collection, they will perform the modeling process in an incremental manner as opposed to traditional project management, and they will continuously generate test cases for the slight improvements committed in the multiple iterations.

II. Related Work

The importance of modeling and design in the successful deployment of software projects is undeniable; hence there has always been attempts to improve the modeling tools and languages, especially the Unified Modeling Language. Norman W. et al. introduced an extension of the UML called The Unified Modeling Language for Interactive Applications; it preserves the semantics of UML but adds supports for UI design. Birgit et al. also proposed an extension of the UML 2 Activity diagrams with Business Process Goals and Performance Measures. However, both these attempts focus on just one aspect of the UML diagram and tries to improve it without dealing with UML as a whole and trying to sort out which diagrams should the propositions focus on the most.

Additionally, Gianna et al. from the University of Genova conducted a study -on which this paper is based- to determine which UML diagrams are the most used, investigating both academic and IT industries as well. These investigations covered books, courses, tutorials and tools about UML, and the outcome of this research is just the Usage Levels of UML diagrams, the results of this study were not used anywhere else to introduce a new modeling methodology whatsoever.

Model-Based testing on the other hand has undergone significant work, Bouchaib Falah et al. [7] has contributed significantly by inspecting and evaluating some test case generation techniques which led to TCG from Class, Use Case, Activity and Sequence diagrams. Still, this value proposition is thought of in the context of Traditional project Management with all its already discussed drawbacks.

All in all, numerous attempts have been conducted in UML, Testing and Project management, however, none of the attempts has tried to investigate collectively all the work done in those different disciplines in order to introduce a whole new, standalone approach. This paper aims to collect and combine separate work achieved in modeling, testing and project management and as a result, it will introduce a new modeling methodology based on UML, supporting test case generation, and applied in the context of Modern Project Management (Agile).

III. Methodology

This section is divided as follows, first we cover the conducted study and surveys about what UML diagrams are used and which ones are the less known. in the second part, a model-based test case generation technique based on those most used UML diagrams is introduced. The third and final part of the methodology is dedicated to the conjunction of the first two contributions (diagrams, testing) with the Agile Framework Scrum. The final approach will be called Scrum-UML-TCG.

A. *A Selection of the most used UML diagrams*

1) Population Identification

The target population of the conducted study consists of the following UML sources: books, tools, course, and tutorials as well as a personal opinion survey of both academics and industrial practitioners in order to figure out which parts of UML they know, which parts they use and which parts they have never heard of or used.

Data Inclusion and Exclusion Criteria: For all the chosen sources, only sources concerning UML 2.0 or newer versions are included.

Books: when faced with multiple editions of the same book, only the last one is opted for. Moreover, books without ISBN are excluded from the study.

Tools: Only modeling tools specific to UML are included, both commercial and non-commercial ones, the rest is all excluded

Courses: university courses concerning IT studies in different languages are included (English, French, Italian and Spanish)

Tutorials: the study considered tutorials available online as written documents and also videos (how to videos). Screen recording-based tutorials and interactive tutorials are excluded altogether

2) Process description

The study followed different data collection processes for the different sources we have:

Books: Amazon website and its search form was used to look for UML related books under the category of "Computers & Technology". Experiments with several search combinations of strings were tried to find out the search string with the highest number of items in the results, that string turned out to be "UML 2" with a result of 2726 books.

Search results were then filtered based on the inclusion exclusion criteria explained above. Finally, 30 books were collected and analyzed, the list of the selected books is shown in Fig. 1[3]

	Title	Edition	Author(s)	Year	Publisher
UML Notation Guides	UML 2.0 in a Nutshell	1st	Pilone, Pitman	2005	O'Reilly Media Inc.
	The Elements of UML 2.0 Style	1st	Ambler	2005	Cambridge University Press
	Sams Teach Yourself UML in 24 Hours	3rd	Schmuller	2004	Sams Publishing
	UML 2 Certification Guide: Fundamental & Intermediate Exams	1st	Welkiens, Oestereich	2006	Morgan Kaufmann Publishers
	UML Distilled: A Brief Guide to the Standard Object Modeling Language	3rd	Fowler	2003	Addison-Wesley
	Learning UML 2.0	1st	Miles, Hamilton	2006	O'Reilly Media Inc.
	UML 2 for Dummies	1st	Chonoles, Schardt	2003	Wiley Publishing Inc.
	UML 2 Toolkit	2nd	Eriksson, Penker, Lyons, Fado	2004	Wiley Publishing Inc.
	UML 2.0 in Action	1st	Grassle, Baumann, Baumann	2005	Packt Publishing Ltd
	UML Bible	1st	Pender	2003	Wiley Publishing Inc.
	UML Demystified	1st	Kimmel	2005	McGraw-Hill
	UML for the IT Business Analyst	1st	Podeswa	2005	Muska & Lipman Pub
	Verification and Validation for Quality of UML 2.0 Models	1st	Unhelkar	2005	John Wiley & Sons
	The Unified Modeling Language Reference Manual	2nd	Rumbaugh, Jacobson, Booch	2005	Addison-Wesley
	The Unified Modeling Language User Guide	2nd	Booch, Rumbaugh, Jacobson	2005	Addison-Wesley
Software Engineering books based on UML	Object-Oriented Software Engineering Using UML, Patterns and Java	3rd	Bruegge, Dutoit	2010	Prentice Hall
	System Analysis & Design with UML version 2.0: An Object-Oriented Approach	3rd	Dennis, Wixom, Tegarden	2009	John Wiley & Sons
	UML 2 and the Unified Process: Practical Object-Oriented Analysis & Design	2nd	Arlow, Neustadt	2005	Addison-Wesley
	UML 2 Semantics and Applications	1st	Lano	2009	John Wiley & Sons
	Object-Oriented Analysis & Design: Understanding System Development with UML 2.0	1st	O'Docherty	2005	John Wiley & Sons
	Using UML: Software Engineering with Objects and Components	2nd	Stevens, Pooley	2006	Addison-Wesley
	UML 2 Pour les bases de donnees	1st	Soutou	2007	Éditions Eyrolles
	Fast Track UML 2.0	1st	Scott	2004	Apress Media LLC
	Model-Driven Development with Executable UML	1st	Milicev	2009	Wiley Publishing Inc.
	Professional Application Lifecycle Management with Visual Studio 2010	1st	Gousset, Keller, Krishnamoorthy, Woodward	2010	Wiley Publishing Inc.
	Software Modeling and Design	1st	Gomaa	2011	Cambridge University Press
	Systems Engineering with SysML UML: Modeling, Analysis, Design	1st	Weilkiens	2006	Morgan Kaufmann Publishers
	Use Case Driven Object Modeling with UML: Theory and Practice	1st	Rosenberg, Stephens	2007	Apress Media LLC
	Management of The Object-Oriented Development Process	1st	Liu, Roussev	2006	Idea Group Inc.
	Real-Time Object Uniform Design Methodology with UML	1st	Duc	2007	Springer

Fig. 1. UML Books Considered

Tools: Wikipedia page "List of Unified Modeling Langue Tools" contains 49 UML tools. Moreover, just like with Amazon search form, Internet search was also carried out using Google and with a combination of strings ("UML tools list", "UML tools"…). Each tool in the resulting list of tools was then compared against the inclusion criteria and then downloaded and installed from its official website. At the end, 20 different tools were collected and analyzed. Fig.2 [3] shows this complete list.

Name	Release	Year	Licence	Web Site
Altova Umodel		2012	Commercial (Enterprise – Trial)	www.altova.com/umodel.html
Artisan Studio	7.4	2012	Commercial (Trial)	www.atego.com/products/artisan-studio/
Astah	6.6	2012	Commercial (Community Edition)	astah.net/
Borland Together	12.0	2012	Commercial (Trial)	www.borland.com/products/together/
BOUML	6.4.3	2013	Commercial (Viewer - Limited)	www.bouml.fr/
Enterprise Architect	10	2013	Commercial (Trial 30 days)	www.sparxsystems.eu/enterprisearchitect/
IBM Rational Rhapsody Modeler	7.5	2009	Free	www-01.ibm.com/software/awdtools/modeler/
IBM Rational SW Architect	8.5.1	2012	Commercial (Trial 30 days)	www-01.ibm.com/software/awdtools/swarchitect/
MagicDraw	17.0.3	2012	Commercial (Enterprise – Trial)	https://www.magicdraw.com/
Metamill	6.1	2012	Commercial (Trial)	www.metamill.com/
Modelio	2.2.1	2012	Free	sourceforge.net/projects/modeliouml/
Open Modelsphere	3.2	2012	Free	www.modelsphere.org/
Papyrus	0.9.1	2012	Free (Eclipse Plug in)	www.eclipse.org/papyrus/
Poseidon for UML	8	2009	Commercial (Community Edition)	www.gentleware.com/
Power Designer	16.1	2012	Commercial (Trial)	www.sybase.com/products/
RedKoda	3.0.7	2012	Commercial (Community Edition)	www.redkoda.com/
Software Ideas Modeler	5.82	2013	Free	www.softwareideas.net/
StarUML	5.0.2.1570	2006	Free	staruml.sourceforge.net/
Violet	0.21.1	2007	Free	sourceforge.net/projects/violet/
Visual Paradigm	10.1	2013	Commercial (Community Edition)	www.visual-paradigm.com/product/vpuml/

Fig. 2. UML Tools Considered

Courses: Usage of the following online search criteria "UML course", "UML lecture" and "UML university course" resulted in several university courses satisfying the inclusion criteria.
However, it was almost impossible to cover all the slides of the lectures and sometimes the material was not available to public, only the table of content of the lessons was uploaded online. Finally, 22 different university courses were collected and analyzed as shown in Fig.3 [3]

Lecturer	Country	Title	Year
Afsarmanesh	Netherlands	Project Analysis	2012
De Angelis	Italy	Lab. Ingegneria del SW	2012/13
Ciancarini, Iorio	Italy	Lab. Ingegneria del SW	2012/13
Vincent	Australia	System Analysis and Modeling	2012/13
Casalicchio	Italy	Progettazione SW	2009/10
Gérard	France	UML	
Prié	France	Systèmes d'information méthodes avancées	2011/12
Felici	UK	Software Engineering with Objects and Components	2011/12
Siebers	UK	Object Oriented Systems	2012/13
Varrò	Hungary	Modellalapú szoftvertervezés	2012
Lehre	Germany	Softwaretechnik	2012/13
Rumpe	Germany	Modellbasierte Softwareentwicklung	2011/12
Correo, Rossi	Argentina	Uml Basico	
Brambilla	Italy	Ingegneria del SW	2012/13
Alkan	Turkey	Object Oriented Software Engineering	2012/13
Farrow	UK	Software Engineering	2012/13
Easterbrook	Canada	Engineering Large SW Systems	2012
Negre	France	Ingéniérie des Systèmes d'Information	2012/13
Sellares	Spain	Enginyeria del Software	2008/09
Jezequel	France	Approche objet pour le développement de logiciels par objets avec UML	
Turgut	US	Software Engineering I	2009
Cheng	US	Advanced Software Engineering	2013

Fig. 3. UML Courses Considered

Tutorials: At first, three websites were selected to analyze tutorials from, later on, this data was integrated with tutorials resulting from the Google strings "UML Tutorials" and "UML guide".
Finally, 18 tutorials were collected and analyzed. Fig.4 [3] shows the complete tutorials list

Author / Source	Title	Web Site
Allen Holub	Allen Holub's UML Quick Reference	www.holub.com/goodies/uml/index.html
Analisi-disegno	Introduzione a UML	www.analisi-disegno.com/uml/uml.htm
Crag Systems	A UML Tutorial Introduction	www.cragsystems.co.uk/uml_tutorial/index.htm
devmentor	UML Guide v2.1	devmentor.org/references/uml/uml.php
Dumke	UML Tutorial	www-ivs.cs.uni-magdeburg.de/~dumke/UML/index.htm
Embarcadero	Practical UML: A Hands-On Introduction for Developers	edn.embarcadero.com/article/31863
HTML.it	Guida UML	www.html.it/guide/guida-uml/
John Deacon	Developer's Guide to UML 2: A UML Tutorial	www.johndeacon.net/UML/UML_Appendix/Generated/UML_Appendix.asp
lemiffe	Reference Guide for UML 2.0	www.lemiffe.com/wp-content/uploads/2008/12/uml2.pdf
New Think Tank	Video Tutorials	www.newthinktank.com/2012/11/
Online Teach	UML Training	www.online-teach.com/u-m-l.php
Parlezuml	UML Tutorial	www.codemanship.co.uk/parlezuml/
Richard Botting	A Beginners Guide to The Unified Modeling Language (UML)	www.csci.csusb.edu/dick/cs201/uml.html
SmartDraw	What is UML?	www.smartdraw.com/resources/tutorials/uml-diagrams/
Sparx Systems	UML 2 Tutorial	www.sparxsystems.com.au/resources/uml2_tutorial/index.html
Storrle & Knapp	Unified Modeling Language 2.0	www.pst.ifi.lmu.de/veroeffentlichungen/UML-2.0-Tutorial.pdf
Uml.free	UML, le langage de modélisation objet unifié	uml.free.fr/index-cours.html
uml-diagrams	UML 2.5 Diagrams Overview	www.uml-diagrams.org/uml-25-diagrams.html

Fig. 4. UML Tutorials Considered

3) Results of study

After data collection, which was detailed in process description, the analysis was finally performed with the following interpretations in mind:

- A diagram is "widely used" is it present in 90% or more of the sources
- A diagram is "scarcely used" if it is present in the 40% or less of the sources
- Presence of some non-defined cases (grey zones)

Section 3.1 describes in detail the results of the findings regarding the levels of usage of the UML diagrams:

a) Levels of Usage of the UML diagrams

Fig. 5. [3] summarizes the levels of usage of the UML diagrams in the various sources: books, courses, tutorials, tools, and the totality of use disregarding the kind

The widely used diagrams are (listed according to their usage level): class (present in 100% of the sources), activity,

sequence, use cases and state machine diagrams. Class diagrams are considered as the main building block of the UML and therefore ranking first in the list shouldn't be surprising, also note that all the widely used diagrams were present in the early versions of UML (1.x versions).

The scarcely used diagrams on the other hand are, timing, interaction overview and profile all of which were not present in the early versions of UML (1.x). Moreover, the profile diagram appeared only in version 2.2. Profile and timing diagrams; low usage is due to both their late appearance and their limited scope. As for interaction overview, it is quite complex and can be replaced by sequence and activity diagrams which are widely used and relatively simple

UML Diagram	Book Guide	Book Spec	Book Tot	Tool	Course	Tutorial	All Sources
Class	100%	100%	100%	100%	100%	100%	**100%**
Activity	100%	93%	97%	100%	95%	100%	**98%**
Sequence	100%	93%	97%	100%	100%	89%	**97%**
Use Case	100%	93%	97%	100%	95%	89%	**96%**
State Machine	100%	93%	97%	100%	95%	89%	**96%**
Communication	100%	80%	90%	90%	59%	89%	**82%**
Component	93%	80%	87%	85%	59%	89%	**80%**
Deployment	93%	80%	87%	90%	55%	89%	**80%**
Object	93%	80%	87%	70%	55%	67%	**71%**
Package	100%	79%	89%	65%	52%	67%	**70%**
Composite Structure	87%	60%	73%	80%	14%	33%	**52%**
Timing	87%	53%	70%	40%	5%	33%	**40%**
Interaction Overview	80%	53%	67%	45%	5%	28%	**39%**
Profile	7%	13%	10%	30%	0%	6%	**11%**

Book Guide = UML Notation Guides, Book Spec = Software Engineering books based on UML, Book Tot = All books Dashed lines represent the 40% and 60% thresholds

Fig. 5. Usage levels of UML diagrams

This study made it possible to determine the level of usage of all the UML diagrams in its latest specification, thus, we now have a solid proof from both academics and IT industries of which diagrams to include in the modeling approach and which ones to exclude.

The chosen diagrams to be integrated in our proposed approach are the five most used diagrams: **Class**, **Activity**, **Sequence**, **Use Case**, **State Machine.**

B. UML Test Case Generation

Now that we have a set of "widely used" UML diagrams, those will be used in a process other than modeling, which is contribution into the testing phase, and more specifically, Test Case Generation (TCG).

Test Cases are a set of conditions used to ensure that whatever software being implemented satisfies all the stakeholders' expectations and requirements. Test cases work by trying every combination of an action and every possible execution path of a program. TCs differ based on the specified and agreed upon criteria, the latter are a set of rules and guidelines testers are supposed to stick to and measure code satisfaction levels against them.

There are various test generation techniques as mentioned in the introduction. In this paper, we will adopt a UML-Based Test Generation Technique, and the following explanations concern Class, Activity, Sequence, Use Case and State Machine diagrams

1) TCG from Class Diagram

Class diagram's specifications contain the following constructs: Associations, Multiplicities, Inheritance, Other Relationships, Class Attributes

From these constructs, three testing criteria can be deduced: Association End Multiplicity (AEM), Generalization (GN), Class Attributes (CA).

These deduced criteria will be formulated as a set of representative values, we then apply a cartesian product against each value set so that all the possible test combinations are generated. TCs are then expressed accordingly and based on the modeled Class diagram (assuming it is correct) the valid and the invalid tests will be identified. The last step is to test the program against the implemented test suites

2) TCG from Activity Diagram

This is by far, the most used model when generating test cases, it consists in turning the Activity Diagram into a graph called The Activity Diagram Graph (ADG) from which Test cases will be generated. Moreover, there are different coverage models with different coverage levels when trying to generate TCs from an Activity diagram:

Node Coverage: The simplest model, it works by considering only the nodes in the diagram. If, for example we're generating a TC from a diagram with two final nodes, then at least two different test cases will be generated

Edge Coverage: As opposed to considering nodes, this model traverse the diagram and examine only the edges of the graph in order to generate the possible test cases

Specified-Path Coverage: also called Edge-Pair coverage, it covers the testing requirements fully which makes this model the most complete one. And It generates all the potential outputs/outcomes of the software system to be met in the most appropriate way.

Moreover, another way of generating cases is the combination of the Activity diagrams with the sequence diagrams, this will be detailed later in this paper.

3) TCG from Sequence Diagram

A sequence diagram contains and allows extraction of multiple testing criteria, out of which we list is the All-Message Paths (AMP) criterion. AMP tests all possible combinations of every message passing figuring in a specific sequence diagram [6]. The workflow can be expressed in this fashion: creation of test sets for every sequence diagram, and then verification that all possible message paths are being executed correctly and in the right order.

Fig.1 [4] shows a sequence diagram for an online voting system and Fig.2 [4] shows the corresponding Sequence Diagram Graph (SDG).

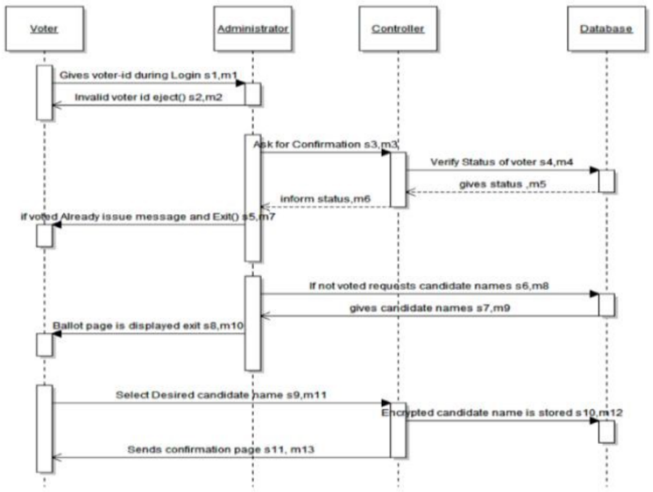

Fig 1: Sequence Diagram for Online Voting System

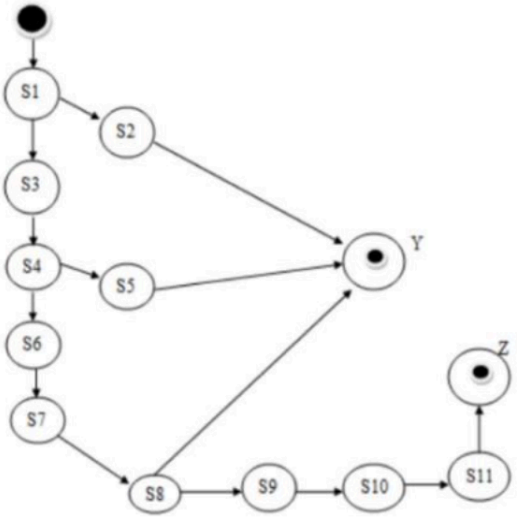

Fig. 2: Corresponding Sequence Diagram Graph

Back to the conjunction of Activity and Sequence. First, both diagrams must be converted to their corresponding testing Graphs: Activity Diagram Graph (ADG) and Sequence Diagram Graph (SDG) respectively.
Then a combination of the two generated Graphs to form a System Testing Graph is done in order to generate TCs.

4) TCG from Use Case Diagram

For the fourth most used UML diagram, one way of generating test cases is by performing a transformation of the diagram into a graph called Use Case Diagram Graph (UDG). This graph allows identification of the single path use case scenario which covers the different actions represented in the corresponding use case entirely.

Regardless of the usefulness of the Use Case Diagram Graph, it remains insufficient because it doesn't cover testing entirely due to the lack of information. Therefore, Activity diagrams and use case diagrams better be combined for a more efficient and complete test case generation.

5) TCG from State Machine Diagram

Test case generation from this diagram ensure the internal behaviors of the objects are being tested, this is thanks to the nature of State machine diagrams in that they model all possible scenarios for all the present objects in the analyzed system. Just like Sequence diagrams and Activity diagrams, State machine diagrams can also be converted to their corresponding State Chart Diagram Graph (SMDG).

Still in the Online Voting System, Fig.3 [4] shows a State Machine diagram and Fig.4 [4] shows the corresponding State Machine Diagram Graph (SMDG).

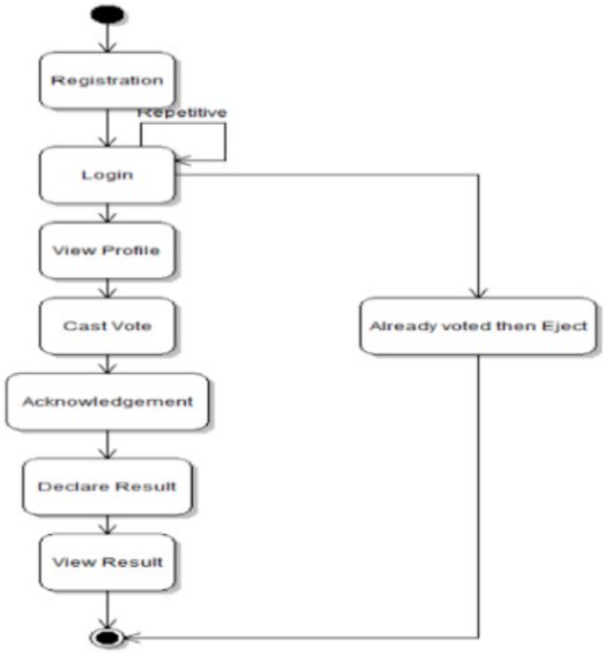

Fig 3: State Machine Diagram for Online Voting System

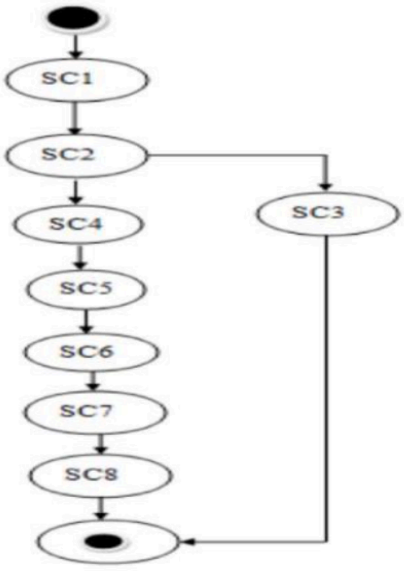

Fig 4: Corresponding State Machine Diagram Graph for Online Voting System

C. *SCRUM-UML*

1) Overview of the Scum-UML Modeling Approach

This is the final step in our proposed methodology, following the selection of UML diagrams, the generation of test cases from those diagrams, we are now going to apply all of this in the context of modern project management, precisely with Agile and under the most popular and used framework which is Agile. This part of the methodology follows the ideology of scrum and UML. And it maps the UML modeling technologies (in our case only the 5 selected diagrams with their test cases) onto the scrum practices.

The development process of the Scrum-UML approach is expressed as follows [5]:

Process = lifecycle + activity + artifact + role

The lifecycle mentioned is divided into two phases:

- Requirement Analysis
- Sprint with UML Modeling

Table 1 [5] shows how UML modeling technologies and scrum practices are being mapped into these two phases.
In Agile in general and in Scrum particularly, the product backlog is the first thing to be established, the backlog is an unordered list containing features and functionalities which need to be prioritized by the product owner and shifted to the "to-do" section in the board. The thing is that there is no method in Scrum which dictates how to conduct analysis of the product backlog and establish it. Fortunately, analysis of software requirements can be the key to successfully start Agile software development.

Scrum-UML introduces use case modeling technology into the requirement analysis phase, and therefore its name becomes "Agile requirement analysis". For sprints, when these are being implemented, the project organization and management framework based on scrum is followed.

Moreover, UML modeling's static and dynamic modeling technologies chosen earlier in 3.1 are applied. This phase in the process Lifecyle is called Sprint with UML modeling.

Fig.1 [5] shows the activities and artifacts of both Agile requirement Analysis and Sprint with UML Modeling phases. In the first phase, there are three activities not shown in the figure, but they will be discussed later (Establishing use case model, converting use case to user story, Making product backlog). For the artifacts of phase 1, we have product backlog and use case models.
Sprint with UML modeling is implemented via rounds of iterations which can be called sprints. In every sprint, the team is required to deliver a "working piece of software" or "potentially shippable product increment". Each sprint begins with a planning meeting where the work to be done is selected by the team, the sprint concludes with a sprint retrospective where the work done in the sprint duration is evaluated and learned from it to better carry-on upcoming sprints, in-between, focus is shifted towards working on UML modeling technology.

Scrum-UML is inspired by Scrum as well as UML modeling and object-oriented analysis, thus, it's both use case and user story driven, architecture-centric, iterative, and incremental.

The approach is use case and user story driven, since use case driven development has proven its efficiency in offering excellent solutions to inaccurate, incomplete, and inconsistent requirements. For user stories, they are essential in Agile practices.
Thanks to the mentioned advantages of adopting a use case driven approach, use cases will be converted to user stories (detailed later)

TABLE1 MAPPING FROM UML MODELING AND SCRUM INTO SCRUM-UML MODELING APPROACH

Scrum-UML Modeling Approach	Scrum methodology	UML modeling
Agile Requirement Analysis	Nothing about how to implement but the product backlog	Use case modeling
Sprint with UML Modeling	Organization and management framework of the project	Static and dynamic modeling including all of the modeling technologies

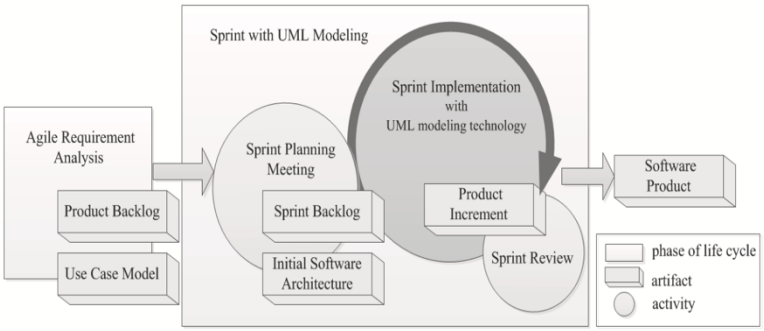

Figure 1. Process of the Scrum-UML Modeling Approach

2) Process of the Scrum-UML Modeling Approach

This section explains in detail the development process of the Scrum-UML modeling approach including roles, activities, artifacts, and implementations

a) Agile Requirements Analysis

Englobes both, use case modeling of UML into the Agile requirement analysis phases. Usually the product owner, team and the rest of the stakeholders contribute collectively to this phase. The outcome of this phase is a product backlog and Use Case Models. As stated earlier, this phase involves three activities described as follows:

1. Establishing use case model

We use UML modeling as described in its specification and come up with a use case model as an output. Establishing use case model is a process of analyzing software requirements from the end user's perspective. The boundary and range of the system is the first thing which must be determined, then comes recognizing the actors in the determined system, actors can be human or other actors (servers, API…). The last two things are the analysis and identification of the use cases corresponding to the previously determined actors, and the mapping between the actors and their use cases into what's called "relations".

2. Converting use case to user story

User stories, -which are short descriptions of what users might want the system to do and they are expressed using this template "As a …. I want to ……. In order to ……."- are the predominant way of expressing features on the product backlog for an Agile-based team. During use case modeling, the completed use case diagrams should be converted to user stories. So, at the end of this phase, we will have use case models containing UC diagrams, Activity diagrams for detailed specifications of the use cases. And many user stories.

Use cases and User stories are essential in requirements analysis. User stories describe what's desired from the system and it's expressed from end users' perspectives. As for use cases, they are descriptions of functionalities the system will certainly provide, and it's also analyzed from the end user's perspective.

3. Making product backlog

It is the product owner's responsibility to communicate to the rest of the team which features should the end product have. And it's his responsibility to prioritize product backlog according to what he sees most fit (dependencies, risks, priorities…). His job is very demanding and requires availability, business savvy and communication skills. In Scrum-UML, the generated use case model and the generation process itself help reduce the complexity of the product owner's job.

More specifically, a user story is a feature in the product backlog and it's also a use case outcome, converted from use case diagrams. All that's left for the product owner to do is to prioritize these user stories whether from the dependencies between user stories or the relations between use cases.

Product backlogs now will become different from the traditional ones present in Scrum alone. Table. 2 shows product backlog and how to describe user stories in it.

Even though Scrum-UML embraces changes, requirements can be changed only outside the sprints, once the team starts a sprint, it keeps focusing entirely on achieving sprint's goals. Therefore, use cases models and product backlog can't be changed at all at that time.

b) Sprint with UML Modeling

Scrum-UML process's modeling approach is iterative and incremental, projects progress via a series of predetermined sprints with a duration varying from 1 to 4 works (ideally 2 weeks).

TABLE 2 SEVERAL ITEMS FROM THE PRODUCT BACKLOG OF THE GUARANTEE MANAGEMENT SYSTEM

ID	Name	Imp	Est	How to demo	Notes
1.1	guaranteed loan application	110	4	Customer logins into the system; Open the guaranteed loan application screen, Fill in personal information, including borrower, guarantee, mortgage, loan information etc.; Submit information and waiting.	Classifying these information reasonably using paging techniques
1.2	customer evaluation	70	8	Evaluation managementer logins into the system; Choose cases need to be evaluated, and examine the detail information; Choose function of adding evaluation report to open the corresponding screen; Compile the report and submit.	

When implementing a sprint, features of the user stories from the product backlog are being coded, test case generations are also being generated from the five UML diagrams and tests are conducted to the written code.

At the end of each sprint, tested and agreed upon increments are integrated into the evolving product or system. Sprint with UML modeling contains three activities described as follows:

1. Sprint planning meeting

Supervised by the product owner, inputs of this activity are Use case models and product backlog established earlier, During the meeting, the product owner picks the most important features to be implemented. The rest of the team starts asking as many questions as possible to better understand the features from the product owner, then the entire team starts doing estimation of how much time these high-level user stories will take. Estimation can be absolute (estimating each user story independently) or relative (story points) and there are dedicated techniques like Poker Planning, Card Sorting….

This discussion is supported by use case models such as user stories, use case diagrams and activity diagrams. Use case models make the planning meeting smoother and more efficient

Place, time of demonstration, and daily standup meetings should be fixed in this meeting.

2. Sprint implementation

During the sprint, there is a collectively done critical step which is Sprint breakdown. in this phase, the team does what's called "Tasking out" the sprints, that is dividing every user story into very small chunks of tasks. It also estimates how many stories points each task is supposed to take in order to complete. In Agile, teams are self-organized and self-managed, therefore, every team member contributes in whatever way they can to complete the sprint's goals and they independently decide to implement a set of tasks and assign them to themselves. Fig. 6 shows an example of Sprint Planning.

US ID	USER STORY	TASKS	HOURS ESTIMATE	ASSIGNED TO
10	As a HS student, I want to write my questions and get answers so that I can get an explanation for my special case	Create the questions table in the database (with necessary associations)	3	Ghita
		Populate the database with Q&As	1	Soufiane
		Create UI for FAQs page	3	Ghita
		Create form page for questions	3	Omar
		Create chatbot	2	Omar
		Configure chatbot	8	Soufiane
		Create APIs	4	Ghita
		Link backend to frontend	3	Ghita
		Write tests	4	Omar
		Run tests	2	Omar
11	As a staff member, I want to answer some FAQs once and for all so that I don't have to re-answer duplicate questions	Create staff portal (with login)	3	Soufiane
		Create staff table	2	Omar
		Create ML tool to clean questions data (remove duplicates) to group them and display number of occurences	9	Ghita
		Create questions management page	4	Omar
		Create answer box and add to the FAQ button	2	Omar
		Create FAQ management page (Re-order/add/edit...)	1	Omar
		Write tests	4	Soufiane
		Run tests	2	Soufiane
1	As an international student, I want to be aware of the visa process to know what I need to prepare.	Create Internationals page	3	Soufiane
		Create form to input international students info (Select country button...)	2	Ghita
		Web scraping to get visa requirements from embassy websites	6	Omar
		Populate database with requirements info (from web scraping)	2	Soufiane
		Display corresponding visa requirements	2	Omar
		Write tests	4	Ghita
		Run tests	1	Soufiane
			80	

UML technologies are applied when sprint implementing. Both static and dynamic views of the system are analyzed and then coding can be accomplished correspondingly. For the static view of the system, Class diagrams are the ones used while Sequence, Activity and State Machine diagrams are used to describe the dynamic view. In order to prevent Agile and Programming from taking way more time than necessary, UML diagrams are not drawn using formal modeling tools, instead, they are drawn on white papers or even papers.

3. Sprint retrospective

Conducted at the end of each sprint, in the Sprint retrospective, the new functionalities are demonstrated primarily to the product owner and the rest of the stakeholders whose feedbacks are welcomed. The more participation of stakeholders in this retrospective, the more feedback will be returned. The sprint's current deliverable is compared against what was agreed upon in the sprint planning meeting, this activity results in revising or adding more items to the product backlog.

The team is encouraged to reflect on how the sprint had been for them in terms of roadblocks, satisfaction, and suggestions to improve upcoming iterations.

IV. Future Work

The outcomes of this research are meant to contribute in both academia as well as IT industry. Since what we are proposing is the first of its kind, there are no software tools which are built to support exactly what we came up with; modeling, testing, modern project management all under one umbrella, instead there are independent tools focusing on just one area; Either they are built for modeling (PowerAMC Designer, Draw.io…) Testing, or management (Atlassians Jira, Trello…). What we want to do in work is beyond just research. We will develop a software tool (either web-based or a desktop application) encompassing all the features and functionalities described in our methodology, which will significantly serve and optimize the work of IT industry personnel, because instead of mastering multiple tools and switching from them continuously, the software tool will include everything needed in just one place.

V. Conclusion

This paper presented a new modeling approach based on already existing solutions. The methodology is divided into three major pillars; First, an investigation of four kinds of sources: books, tools, tutorials, and courses which led to a certain determination of the level of usage of the UML diagrams. Based on this study, 5 diagrams: Class, Activity, Sequence, Use Case and State Machine were judged to be the most popular out of the 14 UML diagrams and therefore these diagrams were chosen to be part of our modeling methodology. Secondly, we presented model-based test case generation techniques from the 5 Unified Modeling Language diagrams that were chosen earlier in order to find out all the execution paths to be tested. Finally, a conjunction of the Scrum with the UML and Test Case Generation was established. This has enabled mapping of modeling and testing with Modern Project Management practices (Continuous Delivery, Agility, Sprints, Change Control) instead of the heavy-weight traditional project management environments.

VI. References

[1] UML Revision Task Force. OMG Unified Modeling Language (OMG UML), Superstructure, V2.4.1, 2011.

[2] G. Reggio, M. Leotta, and F. Ricca. Who knows/uses what of the UML: A personal opinion survey. Submitted to *17th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2014)*, 2014.

[3] Gianna Reggio, Maurizio Leotta, Filippo Ricca, Diego Clerissi, "What Are the Used UML Diagram Constructs? A Document and Tool Analysis Study covering Activity and Use Case Diagrams"

[4] Khurana, Namita, and R.S. Chillar, "Test Case Generation and Optimization Using UML Models and Genetic Algorithm", ScienceDirect, 1 Jan. 2015.

[5] Quan Wei, Guo Danwei, Xue Yaohong, Fan Jingtao, Han Cheng, Jiang Zhengang, "Research on Software Development Process Conjunction of Scrum and UML Modeling.

[6] Wang, Y., & Zheng, M. (n,d.). "Test Case Generation from UML Models Retrieved from mzheng@uwlax.edu,"

[7] Bouchaib Falah, Ghita El AlaouiTalibi, Zineb Bouayad "Test Case Generation From Unified Modeling Language Diagrams"