Towards Using UML Scenarios and Reverse-Engineering For Developping Quranic-Recitation System

A. Jakimi[1,a], M. H. Aabidi[2,b], L. Hamidi[3,c], M. Elkoutbi[4,d]

[1,2]fste, R-O&I Team, Moulay Ismail University, Errachidia, Morocco
[3]fpe, Moulay Ismail University, Errachidia, Morocco
[4]ensias, Med V University, Souissi, Rabat, Morocco
[a]ajakimi@yahoo.fr, [b]myhafidaabidi@yahoo.fr, [c]lakbir1973@yahoo.fr, [d]elkoutb@ensias.ma

ABSTRACT
Identifying and resolving design problems in the early software development phases can help ensure software quality and save costs. In recent years, scenarios and Reverse-Engineering (RE) have become popular techniques for requirements elicitation, specification building, reading, code generation and producing design documents. The UML has been accepted by the industry as a de facto standard for object-oriented modeling and provides a suitable framework for scenario acquisition using use case and interaction diagrams (sequence or communication diagrams).
In this paper, we suggest a methodology for using scenarios and to apply RE for developing and design concept of the system (Quranic recitation system). This methodology supports the designer in designing, visualizing, documenting artefacts in object-oriented software systems and provides notations and guidance to model both the static structure of the program and the dynamic behaviour of the objects.
***Key* words: UML Scenario, Reverse Engineering, User interface, Quranic-recitation.**

1. Introduction

Scenarios have received significant attention and have been used for different purposes such as understanding requirements, human computer interaction analysis specification or prototype generation, and object-oriented (OO) analysis and design (Booch., 1994, Jacobson et al., 1992, Hsia et al., 1994, Elkoutbi et al., 2006). In software engineering, scenarios are one of techniques used to support requirement elicitation and specification building. Since scenarios represent partial description of system behavior, an approach for scenarios engineering (scenarios composition/merging/ integration or code generation) is needed to produce more complete specifications and skeleton code for the system.

Recently, three aspects have received a lot of attention in OO development: the emergence of the Unified Modeling Language (UML) (OMG, 2007) as a unified notation for OO analysis and design, a growing consensus on use case (or scenario) approaches to software development and the RE as a the main idea to maintain legacy systems throughout understanding the source code by analyzing code to reach the design. In fact, when the design achieved, any development process performed on these systems becomes easier and new requirements can be added through the design.

In our work, we use the UML that provides a suitable framework for scenario acquisition using use case diagrams for modeling static aspect and for behavioural specification using state diagrams and activity diagrams which are transformed in to different forms (automatas, mathematics formulas, Coloured Petri Net (CPN)…etc) and manipulate the RE that is the process of discovering the technological principles of a mechanical application through analysis of its structure, function and operation. It is the process of reading code and producing design documents. We propose a new approach for conception and development the system (Quranic recitation system) as a four activities process elaborating requirements acquisition, deriving formal specifications, user interface (UI) generation/code skeletons from UML scenarios and apply the RE.

Section 2 of this paper gives a brief overview of the UML Scenario, UML diagrams relevant for our work and RE. Section 3 describes the four activities of our process proposed. Finally, section 4 provides some concluding remarks and points out future work.

2. UML Scenario and reverse engineering

Scenarios have received significant attention; It has been used for different purposes and have been identified as a promising technique for requirements engineering. Also, UML covers a wide range of issues from use cases and scenarios to state behavior and operation declarations.

In this section, we discuss the scenarios aspects, UML diagrams that are relevant for our approach: use case diagrams (UsecaseD) and sequence diagrams (SequenceD) and the role of RE in the development of the applications.

2.1 Scenario Aspects

Scenarios have been evolved according to several aspects, and their interpretation seems to depend on the context of use and the way in which they were acquired or generated. In a survey, Rolland (Rolland et al., 1998) proposed a framework for the classification of scenarios according to four aspects: the form, the contents, the goal and the cycle of development (Fig.1).

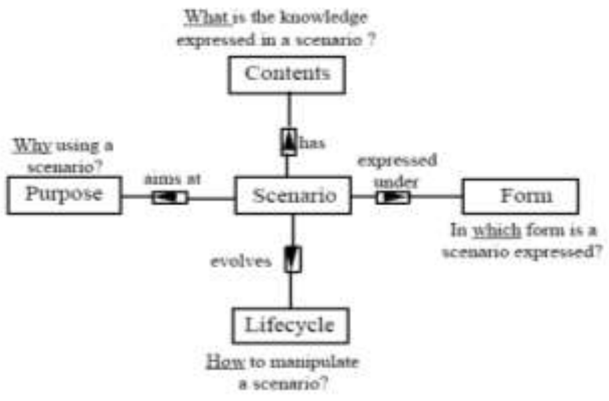

What is the knowledge expressed in a scenario ?
Contents
has
Why using a scenario?
Purpose
aims at
Scenario
expressed under
Form
In which form is a scenario expressed?
evolves
Lifecycle
How to manipulate a scenario?

Fig.1. Aspects of scenarios

The form view deals with the expression mode of a scenario. Are scenarios formally or informally described, in a static, animated or interactive form?

The contents view concerns the kind of knowledge which is expressed in a scenario. Scenarios can, for instance, focus on the description of the system functionality or they can describe a broader view in which the functionality is embedded into a larger business process with various stakeholders and resources bound to it.

The purpose view is used to capture the role that a scenario is aiming to play in the requirements engineering process. Describing the functionality of a system, exploring design alternatives or explaining drawbacks or inefficiencies of a system are examples of roles that can be assigned to a scenario.

The lifecycle view considers scenarios as artefacts existing and evolving in time through the execution of operations during the requirements engineering process. Creation, refinement or deletion are examples of such operations.

2.2 UML Scenarios

In our work, we have adopted the UML, which is a unified notation for OO analysis and design and offers a good framework for scenarios. Scenarios and use cases have been used interchangeably in several works meaning partial descriptions. UML distinguishes between these terms and gives them a more precise definitions. A use case is a generic description of an entire transaction involving several objects of the system. A UsecaseD is more concerned with the interaction between the system and actors (objects outside the system that interact directly with it). It presents a collection of use cases and their corresponding external actors. A scenario shows a particular series of interactions among objects in a single execution of a use case of a system (scenario is defined as an instance of a given use case). Scenarios can be viewed in two different ways through SequenceDs or communication diagrams (CommDs). Both types of diagrams rely on the same underlying semantics.

Use case diagrams. The UsecaseD in UML is concerned with the interaction between the system and external actors. One use case can call upon the services of another use case using some relations (include, extends, use). An example of the include relation is given in figure 2. This relation is represented by a directed dotted line and the label <<include>>, <<extend>> or <<use>>.

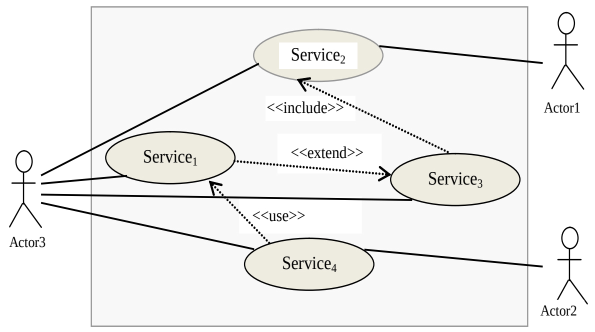


Fig.2.Use case diagram

Sequence diagrams. A SequenceD shows interactions among a set of objects in temporal order, which is good for understanding timing and interaction issues. It depicts the objects by their lifelines and shows the messages they exchange in time sequence. However, it does not capture the associations among the objects. A SequenceD has two dimensions: the vertical dimension represents time, and the horizontal dimension represents the objects (figurer 3). Messages are shown as horizontal solid arrows from the lifeline of the object sender to the lifeline of the object receiver. A message may be guarded by a condition, annotated by iteration or concurrency information, and/or constrained by an expression. Each message can be labeled by a sequence number representing the nested procedural calling sequence throughout the scenario and the message signature. Sequence numbers contain a list of sequence elements separated by dots. Each sequence element consists of a number of parts, such as: a compulsory number showing the sequential position of the message, a letter indicating a concurrent thread (see messages (m3, m4 and m5)), and an iteration indicator * (see message m2) indicating that several messages of the same form are sent sequentially to a single target or concurrently to a set of targets.

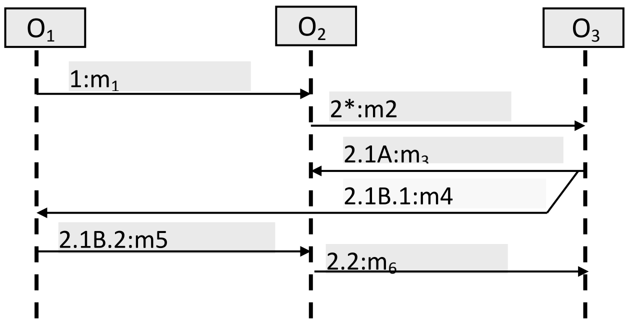


Fig.3. Example of a SequenceD

2.3 Reverse Engineering (RE)

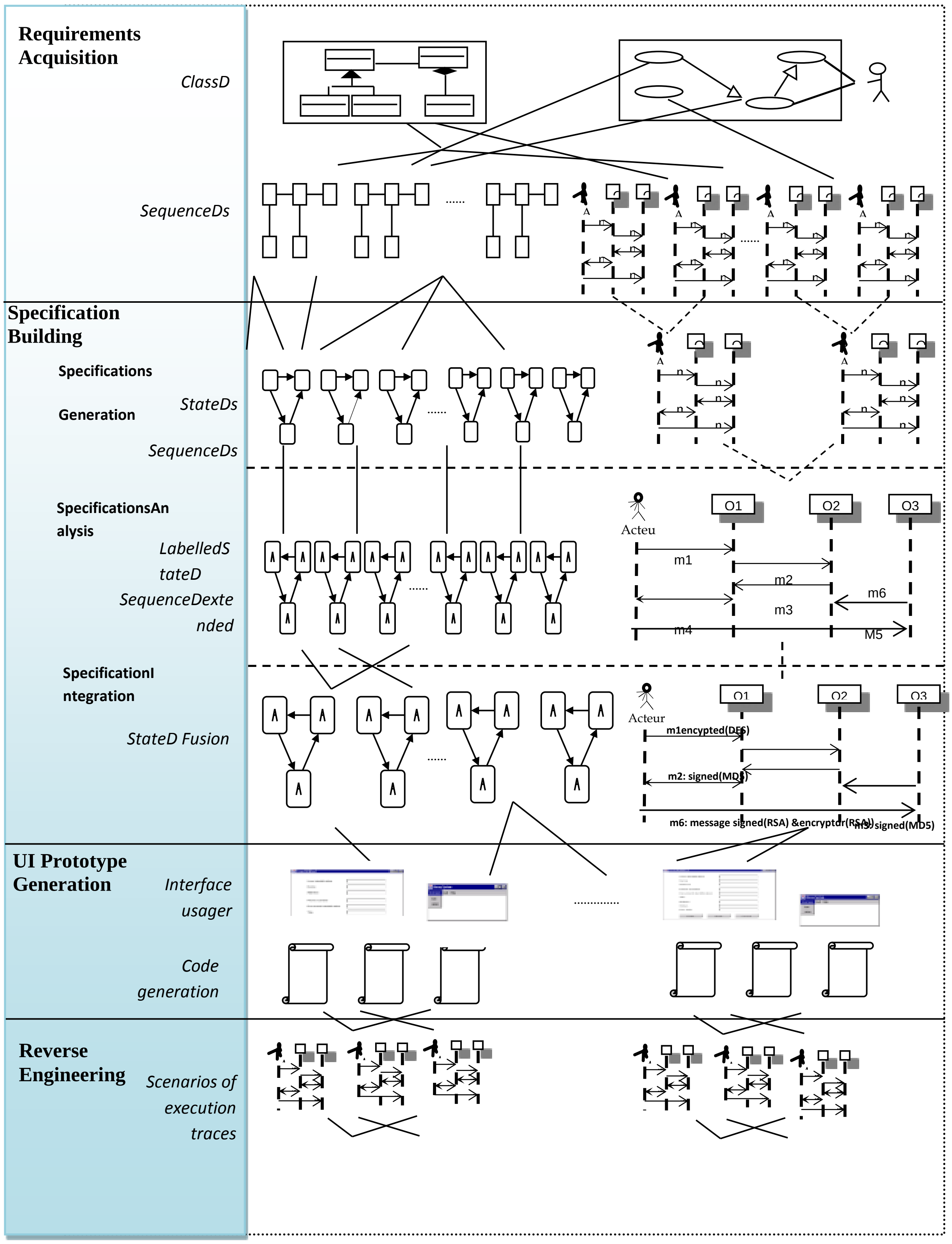
Program comprehension, analysis and evolution is often based on RE of the structure and behaviour of software to visual models such as UML diagrams, and much recent research has been focused on recovering and presenting the structure of programs as UML class diagrams (ClassDs). However, the recovery of dynamic behaviour, and particularly interaction behaviour, to model such as SequenceDs presents many challenges that have yet to be addressed.

The problem of recovering execution traces to UML behaviour (SequenceDs, StateDs…) for OO systems, written in languages such as C++ or Java, has already been extensively studied (Jakimi et al., 2011;Alalfi et al., 2009; Briand et al., 2006). In this work we use RE for goal to present a good methodology that can generate the interaction diagrams (SequenceDs or CommDs) or ClassDs from dynamic applications. Our method is not a perfect solution for all of these issues, but is an improvement to the extent that it delivers accurate results and supports the process of applications comprehension, analysis and evolution.

2. Description of the approach

In this section, we describe the overall approach to apply UML scenario and RE for develop a system. The approach consists of four activities (Figure 4), which are detailed below: (i) Requirements acquisition, (ii) Specification building, (iii) UIPrototype generation and (iv) RE. In our approach, we chose to use SequenceDs and UsecaseDs to acquire scenarios for their simplicity and wide use in different domains.

For Islamic application development and design, we can use UML scenarios (scenarios engineering) for developing the virtual learning environment of the Holy Quran. We must build an interactive model that explains the different scenarios of interaction (determine the dynamic requirements, agents and objects in the logical model of the system and facilitate the identification of design patterns that can be used in building design models of the components system). In this work, we have chosen to study the UsecaseD of the corrector of Quranic recitation integrated in an environment for self learning of the Holy Quran (Yahya et al., 2012).



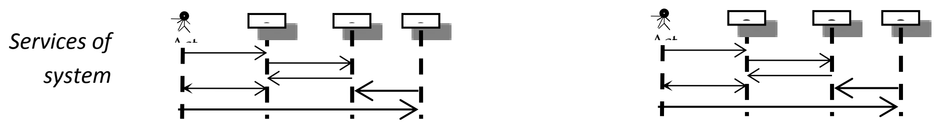


Fig.4. Overview of the approach

3.1 Requirements acquisition

Scenario modeling is the key technique mostly used in this activity. It is used in OO methodologies (OMG, 2007) as an approach to requirements engineering. The UML proposes a suitable framework for scenario acquisition using UsecaseDs for capturing system functionalities and SequenceDs/StateDs for describing scenarios. In this activity, the analyst first elaborates the ClassDmodeling the static aspect of the system. Then the analyst elaborates the UsecaseD capturing the system functionalities that consists to identify use cases and external actors interacting with.

A scenario shows a particular series of interactions among objects in a single execution (instance) of a use case. We can consider two types of scenarios: normal scenarios, which are executed in normal situations, and scenarios of exception executed in case of errors and abnormal situations (Elkoutbi et al., 2006, Jakimi et al., 2011).

Figure 5 shows an example of a UsecaseD corresponding to the Quranic recitation integrated in an environment for self learning of the Holy Quran system. In this UsecaseD, we find the actor (Student) interacting with five use cases (Identify, Data-modify, Recitation-register, Evaluation and Recitation-correct (Self-correcting, Automatic-correcting)). There are also several <<include>> relations; for instance, the use case Recitation-register relies on the services of the Identify use case.

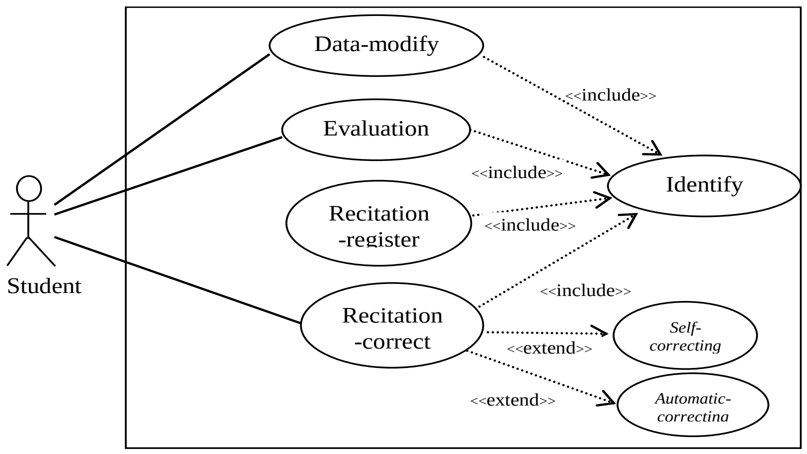


Fig.5. UsecaseD corresponding of the Quranic recitation

Interactions in a UsecaseD are modeled using some limited relations. These relations give only a simplified view of interactions that can really exist between services (use cases) given by a system. For example, in the case of the system “Quranic recitation system”, the use case “Identify” is used by four others use cases: Data-modify, Recitation-register, Evaluation and Recitation-correct. Within the relation includes, interactions between the four services of the system are not precisely defined. After executing the use case “Identify”, the student can repeatedly carry out one of the three other use cases (Data-modify, Recitation-register, Evaluation or Recitation-correct). These kind of constraints are not actually supported in the UML use case diagram .We think that artifacts provided by the UML standard in sequenceD with some extensions can be used to model interactions between use cases.

3.2 Specification building

A UsecaseD is very helpful in visualizing the context of a system and the boundaries of the system's behavior. A given use case is typically characterized by multiple scenarios.

This activity consists of deriving formal specifications from both the acquired UsecaseD and interaction diagrams modeling scenarios (SequenceDs). The resulting specification captures the behavior of the entire system in different forms (automatas, mathematics formulas, Statecharts, CPN…etc) (Jakimi et al., 2007; Nianhua etal., 2012; Bowles et al., 2010). We consider separately specifications at use case and scenario levels to capture hierarchy in the resulting specification. Indeed, this activity consists of deriving these forms both the acquired UsecaseD and SequenceDs.

In this activity, we consider three levels in building the system specification: (1) the specifications generation level, (2) the specifications analysis level, and (3) the specification integration level.

Specifications generation level. A scenario shows a particular series of interactions among objects in a single execution of a use case (execution instance of a use case). A scenario is defined as an instance of a given use case. In this activity, for each use case, the analyst acquires the corresponding scenarios into interaction diagrams and StateDs (statecharts). We apply to generate partial specifications for objects participating in scenarios of the system.

Specifications analysis level. The previous activity generates all forms (automata, statecharts, PNs…) with unlabelled scenarios and states. In this activity, the analyst must add state names to the generated StateDs and extend SequenceDs. In fact, our methodology is based on scenario and states names and can also add structural information like grouping scenarios and states.

Specification integration level. In this operation, we aim to merge all forms corresponding to the scenarios of a use case Uci, in order to produce an integrated form modeling the behavior of the use case (integrating many SequenceDs, StateDs or CPN into one form). The objective of this activity is to integrate for each object and each use case in which it participates all its partial forms into one single form per use case (Jakimi et al., 2011, elkoutbi et al. 2006).

In our example, the CPN corresponding to the UsecaseD is derived by mapping use cases into transitions. A place "Begin" is always added to model the initial state of the system. After a use case execution, the system will return back to its initial state for further use case executions. The place "Begin" may contain several tokens to model concurrent executions. Figure 6 depicts the CPN derived from the Quranic recitation system's UsecaseD (Figure 5) based on the following composition:

[Identify; (Data-modify, Recitation-register, Evaluation and Recitation-correct)]*

(The use case "Identify" is executed then it is followed by an iteration of one of four other use cases).

To obtain a global description of a given service (one use case) of the system or the description of the whole system, an operation of integration or composition between use cases and/or between scenarios is needed. The operation of integration looks like a generalization, where the analyst tries to identify and abstract some common parts in the system behavior. Composition constructs new behaviors from existing ones. This operation (composition) can be applied to different interaction objects like use cases, scenarios or messages. The difficulty of composition comes from the fact that interaction objects (use cases or scenarios specially) are being described independently one to each other's. For example, in the service 'Identify", we can consider two scenarios: a scenario corresponding to the regular identification "regularIdentify" and a scenario corresponding to the error identification "errorIdentify".

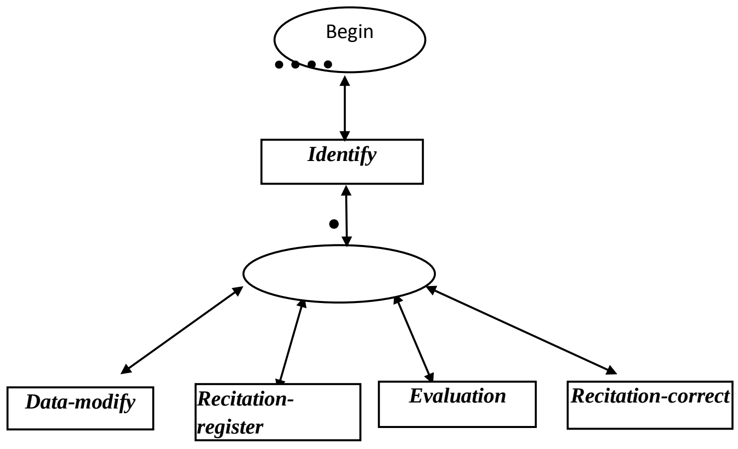


Fig.6. CPN corresponding to UsecaseD of the Figure 5

For each scenario of a given use case, we first derive the CPN structure; then the CPN semantic is built by the help of the analyst. The CPN structure is automatically obtained from the graph representing the sequence of messages in the scenario by adding places between each pair of sequential messages (Jakimi et al ,2007; Elkoutbi et al., 2005).

3.3 Prototype generation

In this activity, we can derive from the scenario specifications a UI prototype and code generation of the system. The generated prototype is stand alone and comprises a menu to switch between the different use cases. The various screens of the prototype represent the static aspect of the UI; the dynamic aspect of the UI, as captured in the specifications, maps into the dialog control of the prototype (Elkoutbi et al., 2006). In fact, we can generate UIprototypes from object specifications and develop graphical editors for StateDs and interaction diagrams, to ease scenario acquisition and allow for the visualization of the generated behavior specifications.

Finally, in this activity, we can also generate code from a SequenceD resulting (service) for the entire interface objects found in the system. This activity proposes a methodology to narrow the gap between multiple UML models and an implemented system. The narrowing of a gap is achieved by generating java code directly from multiple behavior aspect of the system. The code generation is achieved by creating a mapping between Behavioral diagrams basically capture the dynamic aspect of a system and the OO programming language (Jakimi et al., 2011;Aabidi et al., 2013).

3.4 Reverse Engineering

In this activity, we can to get back the requirements or the design specifications from a system. The reason behind this activity is better understandability and aid in maintenance of legacy systems. In other words, it is primarily intended to recover and record high-level information about the system.

RE for the static aspect of an OO system are already available in many UML CASE tools. However, there is a little work on RE of UML behavior (SequenceDs, statecharts...etc). The reason for performing RE is to maintain legacy code. Therefore, it should not be focused on program understanding but on system maintenance instead. This activity aims to provide program descriptions on higher levels of abstractions, such an abstract level could be a program description using UML diagrams. These program descriptions facilitate the understanding of program structures and program behavior and give an extraction of documentation and other higher-level description of software from the code itself.

We will focus the RE of UML behavior for to fully understand the behavior of a program. To build and generate these UML diagrams, we need to capture the systems state through dynamic analysis (Jakimi et al., 2011;Alalfi et al., 2009; Briand et al.; 2006, Flacarinet al., 2006).

4. Conclusion

The work presented in this paper proposes a new approach for UML scenarios engineering and RE. We suggested a new methodology for merging/integration scenarios, generation of UI prototypes from scenarios, code generation and reverse engineering from UML behavior. Scenarios are acquired as SequenceDs that transformed into diverse forms (PN, statecharts, automata…etc) for specification and building of the entire system.

As future work, we prospect to finalize an automatic corrector of Quranic recitation system and the study the possibility of RE from traces of code. We plan to generate UML diagrams from Java code that describe dynamic aspects of this system. We can also annotate additional information of non-functional requirements to the software design description diagrams as well, so that the resulting executable model can be used for evaluating those non-functional requirements.

5. References

Aabidi , M., Jakimi , A., El Kinani, E., Elkoutbi, M. (2013), A New Approach for Code Generation from UML State Machine, International Review on Computers and Software (IRECOS), Vol. 8 N. 2, February *2013*.

Alalfi,M., Cordy,J., Dean, T. (2009). Automated Reverse Engineering of UML Sequence Diagrams for Dynamic Web Applications. ICST Workshops 2009, pp. 287-294.

Briand,L., Labiche,Y., Leduc,J.(2006) Toward the Reverse Engineering of UML Sequence Diagrams for Distributed Java Software,Journal, IEEE Transactions on Software Engineering, pp. 642-663, v. 32 no. 9.

Booch G. (1994), Object Oriented Analysis and Design with Applications, Benjamin/Cummings Publishing Company Inc., Redwood City, CA.

Bowles, J., Meedeniya, D. (2010). Formal Transformation from Sequence Diagrams to Coloured Petri Nets. 2010 Asia Pacific Software Engineering Conference, pp. 216–225.

Elkoutbi, M. Khriss, I., Keller,R.K. (2006) "Automated Prototyping of User Interfaces Based on UML Scenarios". Automated Software Engineering Journal, 13, pp. 5-40

Elkoutbi M. and Keller R. K.,(1998) Modeling Interactive Systems with Hierarchical Colored Petri Nets, In Proc. of 1998 Adv. Simulation Technologies Conf., pp. 432-437, Boston, MA, Soc. for Comp. Simulation Intl. HPC98. Special session on Petri-Nets.

Falcarin, P. , Torchiano, M.(2006) A dynamic analysis tool for extracting UML 2 sequence diagrams. In ICSOFT (1), pp.171–176, 2006.

Glinz M. : An Integrated Formal Model of Scenarios based on Statecharts. In Fifth European Software Engineering Conference, Lecture Notes in Computer Science, Vol. 989, pp. 254-271, Springer-Verlag (1995).

Hsia P., Samuel J., Gao J., Kung D., Toyoshima Y., and Chen C : Formal Approach to Scenario Analysis. IEEE Software, 11(2):33-41, March 1994.

Jacobson I., Christerson M., Jonson P., and Overgaard G., Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, 1992.

Jakimi A., Sabraoui A., Elkoutbi M., Idri A., (2007). A new approach for composing UML scenarios and code generation, IEEE SETIT'2007, Tunisia 2007.

Jakimi, A., Elkoutbi, M. (2011) UML Scenarios engineering and automatic code generation", Arab Computer Society Journal, Vol. 4, No. 2.

Jakimi A., Elbermi L. El Koutbi M., "Reverse Engineering of UML Sequence Diagrams". Proceeding in International WOTIC'11, pp. 13-15, ENSEM, Casablanca, Morocco.

Nianhua, Y., Huiqun Y., Hua S.(2012). Modelling UML sequence diagrams with aspect-oriented extended Petri nets. IJCAT 45(1): pp.57-65.

OMG. (2007), "UML superstructure, v2.1.2,http://www.omg.org/spec/UML/2.1.2/ Superstructure/PDF.

Rolland, C., Ben Achour, C., Cauvet, C., (1998) "A Proposal for a Scenario Classification Framework". The Requirements Engineering Journal, Volume 3, Number 1.

Elhadj, Y. O.M., Alghamdi, M., Elkanhal, M., Alansary, A.,Toward an Automatic Corrector of Quranic Recitation Integrated in an Environment for Self Learning of the Holy Quran, Journal of Computer Research,v11, n° 1, 2012.