Periodicals of Engineering and Natural Sciences ISSN 2303-4521
Vol. 9, No. 2, March 2021, pp.430-440

Designing and implementing a tool to transform source code to UML diagrams

Rasha Gh. Alsarraj[1], Atica M. Altaie[2], Anfal A. Fadhil [3]
[1, 2, 3] College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq

ABSTRACT

Currently, reverse engineering is considered as a significant process to extract the design information and abstractions of a system from the present software. The extracted source codes have been converted into a certain structure to be easily analyzed in the following procedure. For facilitating the software process development, it might be optimum to have tools beingcertain which generate automatically or help UML generating models from the codes as a source. The current study focuses on the reverse engineering process regarding the python and java source codes to the behavior diagrams: the use case and the activity diagrams which might be of high importance in the process of software maintenance. This approach is carried out in the current study in the application referred to as the RCUML tool with the use of the python language which helped in understanding python and java source codes in the software application, and enabling visualization regarding the software behavior.

Keywords: Activity diagram, Reverse Engineering, UML, Use case diagram, Software Engineering

Corresponding Author:

Rasha Gh. Alsarraj
College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq
Email: rasha.alsarraj@uomosul.edu.iq

1. Introduction

Reverse engineering (RE) can be defined as the procedure used to analyze the subject systems to identify the components of the system and their inter-relationships and also to create system representations in other forms or at high abstraction levels. Reverse engineering is required in the case of understanding long-term software systems because of out-of-date, inadequate software saved information, system complexity, and insufficient knowledge regarding the system's maintainer. Moreover, it is utilized for recovering the missed data, or enhance the provided one, detect effects as side ones, reutilize the components in addition to reduce attempts of maintenance. Concerning the field of computer sciences, reverse engineering dynamic models convert codes to models like UML (Unified Modeling Language) state diagrams, sequence diagrams, and so on. The essential aim is to increase the abstraction's level for excellently understanding the software's behavior. The communities of software engineering focus on generating tools that help the analysts of the system gaining insight into the system structure . Also, the significant role related to the reverse tools of engineering in engineering software has been as followings [2]:

(1) Objectives of program analysis are to analyze the source codes as well as extracting associated information, such as methods and classes.
(2) Plan recognition will identify the patterns which might reflect the system structures or behaviors.
(3) Concept assignment has been developed for searching the patterns of systems and source code structures and identifying the relations among the system components.
(4) System's re-documentation.

This study specifies the diagrams of UML as target behavior models concerning reverse engineering. The chief study aim of the current is to provide the tool support to automatically extract the UML activity and the use case diagrams from the python and java source codes. To realize such aim, RCUML has been developed and estimated by using the method of reverse engineering for transforming python and java source codes into the models of UML. The contributions of the present study were as follows:

- A tool helping the developers of Computational Science and Engineering (CSE) extract the behavior diagrams of UML from the python and java source codes by enabling them to make excellent decisions related to maintenance tasks and software development.
- Describing the process of transformation utilized for developing tools to help other authors create tools about the community of CSE.
- Evaluation results and the experiences by utilizing tools on the real CSE projects to highlight its limitations and benefits.
- Workshop feedback might help software engineers in developing tools and practices which are
- adequate to be used in the domain of CSE.

This study has been prepared in the following way; the related works are pointed out in section 2 where UML description will be provided in section 3, the methodology of the tool will be given in section 4, the case study for testing the tool's implementation is explained in section, 5. the comparison of the rcuml tool with other tools will be given in section 6 as for the future works and major conclusions, they are mentioned in section7.

1.1 Related work

UML models reverse engineering has been majorly used in software engineering; it might majorly reduce the efforts needed for constructing the UML diagrams that could be divided into two categories. The first category describes a software system's structure, while the second one will define its behavior. Furthermore, the structural UML diagrams might be reverse-engineered from the codes or other static structures. RE is simultaneously related to behavioral diagrams, namely use case and Activity diagrams, which are not majorly used. Yet, there have been considerable attempts to create a working approach regarding RE of the UML behavioral diagrams [3].

A study conducted by [4] pointed out the drawbacks of RE with the Rational Software Architect and described the approaches for overcoming them. Such tricks and technical tips will be of high importance for identifying components and generating high-level abstractions as UML class and the sequence diagrams from the Java classes.

Another study conducted by [5] has suggested a method that is dynamic according to Labeled System of Transition (LTS) concerning the merger collected traces and generates diagrams of sequence of high-level. Such methods have been effective in generating UML representative behaviors. Yet, it was recognizing certain drawbacks, namely information filtering problems. Therefore, the resulted sequence diagrams contain much insignificant data of no help in understanding the software.

Moreover, another study conducted by [6] has provided a tool that accepts the Java programs as input and to determine the program's structural characteristics including the creation of an activity diagram and class diagram. Accordingly, the class diagram includes the dependency association between different classes which are not indicated using other different tools. The methods analyze various proposed methods and implement new techniques to draw both the class as well as the activity diagram.

A study conducted by [7] has presented automatic tools that are abstract the requisite elements out of the program and build a format of UML diagram. Such a process has been extremely easy and has been extracted directly from the code itself. It does include RE support which analyzes codes of java source and attempts to produce comparable formats of UML model. There are five major structural and behavioral diagrams that have been retrieved in simple formats with a lot of drawbacks regarding the implementations refered to them.

A study conducted by [8] has presented the framework of Flowgen that creates the flow-charts from the marked C^{++} source code as a set regarding the activity diagrams of high-level interconnected (UML), one for every one of the methods or functions in sources of C^{++}. It offers visual and an easy overview regarding the complex numerical algorithms implementations. Flowgen has been paired to generally apply tools of Doxygen.

It is important to refer to the study conducted by [9] which describes a novel approach concerning the automatic extraction and visualization of the software behavioral models from the execution traces. Lengthy traces have been specified by filtering out the low-level software components using algorithms that apply dynamic as well as static data. Eight algorithms have been compared in the present work. Traces have been visualized with the use of Use Case Map (UCM) scenario notation. The tool-supported approach has been customized via various algorithms of filtering and parameters, and thus allows the generation of models at various abstraction levels.

A study conducted by [10] generated the sequence diagrams from the Abstract Syntax Tree (AST). It concluded that the AST structure can assist the process of RE. Statement visits in AST have been properly ordered based on the source code's sequence. There have been certain complexities in carrying out the AST algorithm. The

complexity has been found when the algorithm must find another node from unidentified locations. To solve such a problem, this study will be put in the form of recording or marking each one of the nodes of a potential to be searched again. Concerning such solution, in the case when a node has been required, there will be no requirement for searching it via AST; yet this study might be searching it at nodes that were previously recorded.
Another study conducted by [11] showed the UML extension for extracting the sequence UML diagrams from the FORTRAN code as well as for offering their ability through the utilized open-source platform. The study argues that the suggested ability could increase the abstraction levels at the computational science community which is considered as an up-to-date FORTRAN.
A study conducted by [3] reflects an approach that enables the creation of a UML use case and the activity diagrams from recorded user activity and the HTML files on websites. The utilized algorithm analyzes recorded user activity and transforms it into UML Use Case and Activity diagrams. The prototype which is related to the suggested algorithm has been carried out as a chrome extension. Such a prototype has been first ever of its type. It generates the use case as well as activity diagrams, yet it cannot detect the extension in addition to include relations.
A study conducted by [2] extends for UML tool via accumulation of capability to create diagrams of sequence to aid developers of software that create better judgments throughout their software processes development. The current study offered tool of transformation which changes the code as source in Fortran being object-oriented to a sequence diagram of UML as an assistance in systems understanding and analyzing developed in the language of Fortran. Once a utilizer imports to the tool a FORTRAN source code, the system produces an XMI file demonstrating sequence of UML diagram and then makes the sequence diagram via utilizing Modelio.
A study conducted by [12] proposed approach designates that when utilizers come in the code of source into the tool of visualization, it might changed into the diagram of class and sequence as per utilizer requirement. Also, it define algorithm to create suitable sequence and class diagram to the object-oriented C^{++} source code that is entered via utilizer.
Yet, there have been some tools on the market which are comparable to the current study, but they have been at all paid applications. There are also other tools with open-source lacking in some aspects that might be overcome through our tool which focuses on UML behavioral diagrams concerning the python and java source codes and using python.

1.2 UML

UML is considered as one of the standard modeling notations that utilized for representing software as graphical notations. It has been initially presented in 1997 and extremely utilizes modeling language in the industry [13]. Thus, UML is used to specify, visualize, construct, as well as corroborate artifacts related to the software systems, business modeling, and other non-software systems [14]. There has been evidence related to its advantages throughout development of software for it upsurges the understanding among developers and customers and expands communications between the crew members. Furthermore, UML recovers the source quality codes by decreasing its defect density. There are a few indications respecting UML benefits throughout software maintenance. Additionally, diagrams having high detail levels are of high importance in software development [15], whereas the ones with low detailed levels are better in the case of conducting tasks of maintenance [13].
UML comprises ten diagram types that are divided into three categories. There are four types of diagrams representing structures, including deployment, object, class, as well as component diagrams. There are also five types of diagrams representing various aspects regarding the dynamic behavior. They include state chart, activity, collaboration, as well as use case diagrams. Finally, package diagrams represent approaches for organizing and managing application modules. There is also a system which is modeled through UML and includes many distinctive that overlapping and related diagrams of different types.
A lot of the automatic code generators have been created to generate codes from the UML models for different programming languages (particularly object-oriented). Such, a UML model's usages need to be consistent, complete, formal, and unambiguous, to get qualitative systems [16]. Furthermore, there have been advantages of using the tools of RE for generating the diagrams of UML from source codes like improving productivity, process, and code quality as well as costs for software maintenance [1]. Forward Designed (FD) diagrams (which are the diagrams created throughout forwarding development) have been provided for the maintainers in the maintenance stage; yet in the case when it is not applied, diagrams might be reconstructed via the RE method [17]. The differences in diagrams' origins (RE or FD diagrams) as well as various approaches that could be utilized for generating the RE diagram lead to various diagrams styles that might impact the source codes' quality which has been maintained.

The RE diagrams can be easily acquired in comparison to FD ones since they can be automatically generated along tool of reversing with no high investment in the developer's efforts. Since they can be automatically and easily generated at whatever time, the maintainers might have recent diagrams which are modeling the system in the required case. Yet, the obstacle with such diagrams is their extremely high detailed levels resulted from the obstacles to understand them. There are many problems associated with acquiring diagrams with high detail levels in the case when they are originating from the source codes following the RE approach [13]:
The abstraction level has been extremely low since each one of the elements in the code of source has been specified in the diagrams of UML. The such advantage is that there has been extremely traceability elevated to source codes from diagrams. Rules of business permit designers to form UML diagrams through certain design objectives. Furthermore, the developer’s device the source codes via adoting such diagrams.
Such RE diagrams, different from the FD diagrams which are considered as platform-dependent, and thus have details related to implementations, patterns, and utilized frameworks. Following acquiring the diagrams of RE, cleaning as well as layout processes must be conducted to adapt them to the matching audience.

1.2.1 Use-case diagram

Such diagrams are of high significance in terms of software development [18]. They are also considered as artifacts generated throughout the initial stages which are important to formulate, corroborate the behavior of the system [19]. The system's functional requirements have been depicted with the utilization of a use case diagram and might be specified as a contract among customers and developers [20]. Excellent understandability regarding the use case diagram has been needed to have efficient contributions as far as an artifact in the software development process.
Their criticisms regarding systems focu on the requirement developers, and those important changes must be done in the requirement model. Also, the use case diagrams help users evaluate the behavior of the system before writing the code and could be utilized as blueprints throughout the entire software development [21].
The quality related to the conceptual models or the artifacts like use case diagrams is of high importance in suitable and efficient execution and implementation regarding the system requirements. Thus, it is of high pre-ponderance to guarantee the quality of use case diagram, and work must be carried out to achieve such purpose [20].
The majority of the requirement models which provided are through a use case diagram include two parts. One can be considered as a textual description and the other one is considered as the diagram part. The diagram part will provide relations between actors and use cases. The textual description part will present descriptions regarding each one of the use cases. So, a decrease in a misunderstanding between the software developers and the users could be of high advantage for improving the software quality [22].

1.2.2 Activity diagram

The UML activity diagram [23] has been utilized for modeling dynamic behavior (control flow and data flow) [24]. It explains the sequence of activities and actions specific to an operation or a use case. It provides a set of elements that allow a very rich expression of any sequence in a system. Its notation is relatively close to the state transition diagram in its presentation, but its interpretation is significantly different. The activity diagram is essentially composed of activities and transitions. An activity specifies a behavior described by an organized sequencing of units whose basic elements are actions [25].
Even though the major aim of diagrams is communications means and for the ease in the overall understandings, the available information has been of high importance input for the following development tasks, namely elaboration as well as the corroboration in terms of the detailed requirement [26]or derivations related to the test cases [27]. However, there are various tasks with a lot of information requirements that are significantly based on having certain information that is contained in activity diagrams [22].

2. Methodology

Concerning the present section, testing and implementations as for the suggested tool for creating UML diagrams from legacy python and java programs are discussed. It is effectively working with the executable codes, yet it might be also consuming non-executable declarations. This study tool has been developed to visualize the procedural code regarding any complexity. It further optionally visualizes the comments and saves files in form of a picture. The tool supports reverse engineering by importing existing python and java programs and creates UML diagrams so you can see how elements such as system processes, relationships, and other objects relate to each other. The tool will enable the user to navigate rapidly from the model directly to the source code in the same environment.

The tool has been programmed by using a python programming language which is used widely in researches since 2008. There is a huge standard library in python, majorly indicated as one of the greatest advantages of python that offers tools suited for different tasks. The standard library of python can support different types of platforms and operating systems.

It is worth-noting that in 2019, the rank of python has been the third. The tool consists of three interfaces, one to open the file, the second to display the use case diagram, and the third to display the activity diagram. When a program is input in RCUML, it reads the program from left to right and also from top to the bottom and generates two behavior diagrams; namely use case and activity. Concerning Figure 2, the created activity diagram displays coding parts thoroughly. It has been primordial in the case when developers or engineers comprehend the details of code. The tool extracts knowledge regarding the program control flow such as decision points (if-else condition statements), Function/method calls, for/while loops, Switch-cases, nested-if-statements, Statements (variable-initialization) for generating a single activity diagram concerning each one of the methods or functions. This study is considered as the test of the initial implementations of RCUML on various source files. It has been effective concerning all the programs and the possibility of parallel processing.

Properties of UML models are extracted through multiple steps:

- For each program, spaces are removed from the beginning of lines, and code analysis is started line by line.
- The code line is splited into words and searched on reserved words.
- All constants are recognized in the program (by comparing it with the pre-built dictionary for reserved words). These constants, namely all java or python reserve words (loop, condition, variable type …, etc.)
- If reserved words are variable types, the tool draws a model element corresponding to it containing the line.
- If reserved words are loops or conditions, the tool draws a model element (circle, rectangle ...etc.) corresponding to it containing the conditions with the deletion of a reserved word.
- Search on the end of loops or a condition is to be the model element end.
- After completing all the lines of the program, the UML model is shown to the user.

The comments are marked with "//, /* and #". This makes the programmer distinguishs annotation from regular python and java code. RCUML takes annotations that describe software processes that specify the succeeding line of code. This allows RCUML to generate a rich description that represents the use case diagram.

Properties of UML use-case models are extracted by the following multiple steps:

- For each program, remove spaces from the beginning of lines and start code analysis line by line.
- The tool recognizes all actors in the program.
- The tool recognizes all comments in the program marked with //, /*, and for java programs # for python programs.
- Categorize relationship links between use-cases and actors.
- The tool draws the use-case element corresponding and contains the line to it.
- Link use cases with actors.
- After completing all the lines of the program, the UML model is shown to the user.

Methodology for extracting UML models described in Figure 1.

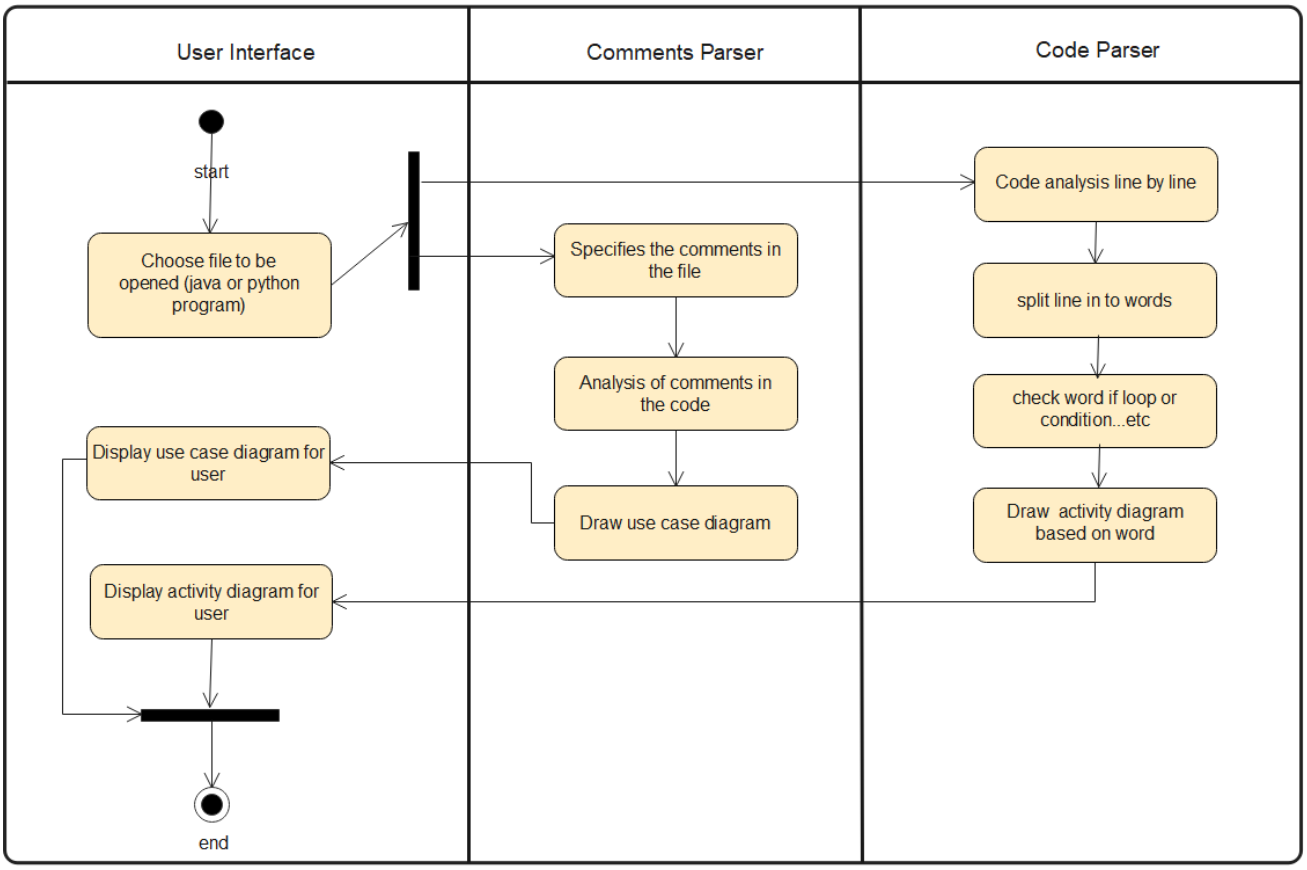

Figure 1. Activity diagram of the RCUML tool methodology

2.1 Case stuides

To test if the proposed method could be utilized in practice, we had chosen a case study and implemented it. First, the user can choose any program. Figure 2 represents the java program that will be entered into the tool. Then, the tool summarizes and analyzes the overall results produced by the tool as two Figures; Figures 3 and 4 represent the use case diagram and activity diagram that is generated from the tool for the java program.

```
import java.util.Scanner;
class PrimeCheck{
public static void main(String args []){
int temp;
boolean isPrime=true;
Scanner scan= new Scanner(System.in);
System.out.println("Enter any number:");
// capture the input in an integer
int num=scan.nextInt();
scan.close();
for (int i=2;i<=num/2;i++){
temp=num%i;
if (temp==0){
isPrime=false;break;
} }
// If isPrime is true then the number is prime else not
if(isPrime)
System.out.println(num + " is a Prime Number");
else
System.out.println(num + " is not a Prime Number");
} }
```

Figure: 2. Java case study program

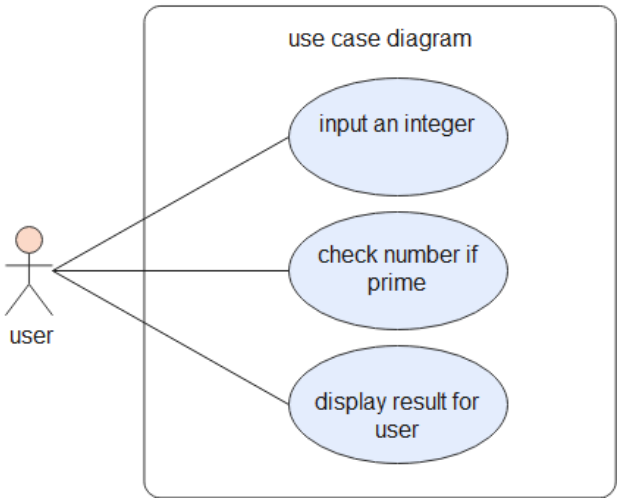

Figure: 3. Use case diagram generated from the RCUML tool

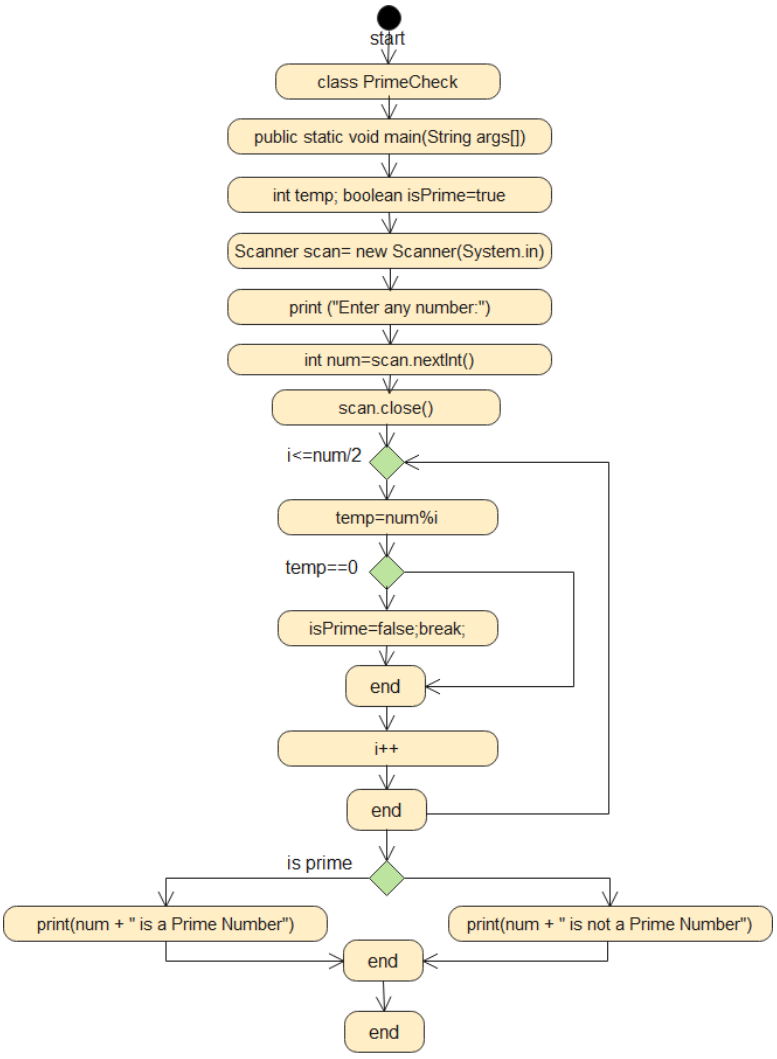

Figure: 4.Activity diagram generated from the RCUML tool

Figure 5 represents the python program that will be entered into the tool. Then, the tool will come out with the results as two figures; Figures 3 and 6 represent the use case diagram and the activity diagram that is generated from the tool for the python program.

```
class PrimeCheck:
  temp=0
  isPrime= 'true'
 # input in an integer
  num= int(input("Enter any number: "))
# check number if prime
  for i in range(2,num):
      temp=num%i
      if temp==0:
          isPrime='false'
          break
  # Display result for user
  if isPrime=='true':
     print(num , " is a Prime Number")
  else:
     print(num , " is not a Prime Number")
```

Figure: 5. Python case study program

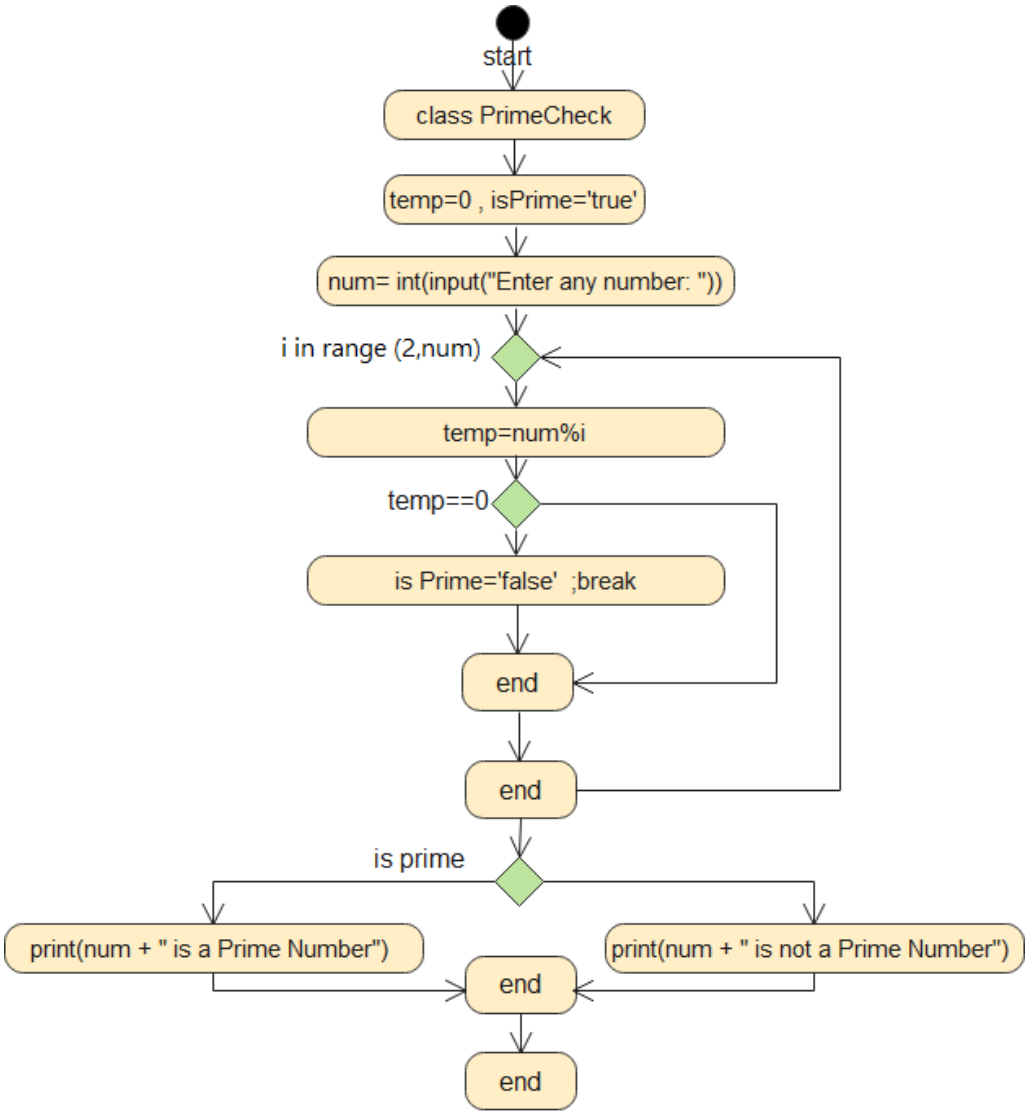

Figure 6: Activity diagram generated from the RCUML tool

3. Results and discussion

In this section, we compare our tool Rcuml to other tools and-that reverse-engineer code, and the comparison is shown in Table 1.

Reverse engineering includes high risk, particularly when the teams of project development and maintenance are different. Many tools and models of reverse engineering are proposed in previous studies [2,3,4,5,6,7,8,9,10,11,12]. However, each of them focuses on only at a subsection of operational requirements. Many researches inspect the reverse engineering of UML static models that are limited in their functionality, like class diagrams. It is significant to recognize the attributes of software quality can be estimated from the software models that have dynamic behavior. However, there are few studies on reverse engineering software models that have dynamic behavioral, and most of them are in the sequence diagram. Dynamic behavioral models of software are vital as static models because they help to recognize interactions that are multiple and complex among objects. The proposed model of the current research offered a tool that transforms the source code of object-oriented python and java to UML diagrams that are the use case and activity diagrams are as assistance in studying, understanding, and analyzing programs written in the python and java languages and to assist the developer of software to create better decisions through their processes.

These two diagrams were chosen for their importance in explaining the behavior of the software and due to their important characteristics where the use case diagram provides a complete summary of the system software in one drawing that can be used in many aspects of software development. The activity diagram can implement a functional goal from beginning to the end graphically of any software. So, activity diagrams act more suitable for catching the behavioral aspects that can be directly performed at the model-level.

Table 1: comparison with RCUML tool

Year	Authors	Tool / Approach	UML diagram	Source code	Covering language concepts
2008	Xu, and Wood [4]	Approach	Class and sequence diagrams.	Java	Global and local variable declaration Method call Return statement
2011	Ziadi et al [5]	Approach	Sequence diagram	execution traces of Java	Global and Local variable declaration If else statement For - while loops Method call Return statement
2013	Gade et al [6]	Tool	Class and activity diagram	Java	Global and Local variable declaration If else statement Method call
2014	Kulkarni et al [7]	Tool	Simple (class diagrams, object diagrams, use case diagrams, sequence diagrams and activity diagrams).	Java	Global and Local variable declaration Object declaration Method call Return statement
2014	Kosower et al[8]	Flowgen	Activity diagram	C++	Global and Local variable declaration If else statement Object declaration Return statement
2015	Braun et al[9]	TraceToUCM	Sequence diagram	Execution trace of java	Variable declaration For - while loops Return statement
2016	Fauzi et al[10]	REVUML	Sequence diagram	Abstract Syntax Tree (AST)	For - while loops If else statement Object declaration Method call
2017	Leatongkam et al [11]	ForUML	Simple sequence diagram	Fortran	Method call Return statement
2018	Drungilas et al [3]	Prototype Approch	Use case and activity diagrams	HTML files on websites	Global and Local variable declaration If else statement Equality comparison
2019	Nanthaamornphong and Leatongkam [2]	ForUML	Sequence diagram	Fortran	Global and Local variable declaration Object declaration Method call Return statement
2020	Singh[12]	Visualization	Class and sequence diagram	C++	
2021	Alsarraj et al	Rcuml	Use case and activity diagrams	Java and Python	Global and Local variable declaration For - while loops If else statement Equality comparison Object declaration Method call Return statement

4. Conclusion and future work

RE objectives are providing current software from models. As such, it might facilitate the program comprehension o and duly the evolution and maintenance regarding the systems. The static models have been limited in their importance. There is a good realization of the quality attribute like reliability and performance could be suggested from the system's dynamic behavioral models. This work presents an overview regarding the RE of the behavioral diagrams and simultaneously suggesting the novel tool for extracting UML dynamic behavior diagrams from python and java source codes with the use of dynamic analysis. The dynamic analysis has been according to system behavior data that might be captured and specified in use case and activity diagrams. In the future, the aim is to develop tool support for more types that are related to the diagrams of UML (beyond use case and activity diagrams) to help developers carry out the system analysis and evaluate this method on the extra complex system like the embedded ones.

Acknowledgments
The authors are very grateful to the University of Mosul/ College of Computer Sciences and Mathematics for their provided facilities, which helped improving this work quality.

References

[1] M. H. Aabidi, *et al.*, "Benefits of reverse engineering technologies in software development makerspace", ITM Web of Conferences13, 2017.

[2] A. Nanthaamornphong, and A. Leatongkam, "Extended ForUML for Automatic Generation of UML Sequence Diagrams from ObjectOriented Fortran", *Hindawi Scientific Programming*, 2019.

[3] V. Drungilas, *et al.*, "Reverse Engineering of UML Use case Model from Website Usage Records", *International Conference on Information Technologies, IVUS*, 2018.

[4] F. Xu, and A. Wood, "Reverse engineering UML class and sequence diagrams from Java code with IBM Rational Software Architect", *IBM Corporation*, 2008.

[5] T. M. Ziadi, *et al.*, "A Fully Dynamic Approach to the Reverse Engineering of UML Sequence Diagrams", *16thIEEE International Conference on Engineering of Complex Computer Systems*, 2011.

[6] A. Gade, *et al.*, "Reverse Engineering of Object-Oriented System", *International Journal of Scientific and Research Publications*, Vol.3, Issue4, 2013.

[7] R. N. Kulkarni, *et al.*," Abstraction of UML Diagrams from Java code", *International Journal of Combined Research & Development (IJCRD)*, Vol.2; Issue: 4, 2014.

[8] D. A. Kosower, *et al.*, "Flowgen: Flowchart-Based Documentation Framework for C++", IEEE *14th International Working Conference on Source Code Analysis and Manipulation*, 2014.

[9] E. Braun, *et al.*, "Generating Software Documentation in Use case Maps from Filtered Execution Traces", *17th International SDL Model-Driven Engineering for Smart Cities, Springer*, 2015.

[10] E. Fauzi, *et al.*, "Reverse engineering of source code to sequence diagram using abstract syntax tree", *International Conference on Data and Software Engineering (ICoDSE)*, IEEE, 2016.

[11] A. Leatongkam, *et al.*, "WIP: Generating Sequence Diagrams for Modern Fortran", *IEEE/ACM 12th International Workshop on Software Engineering for Science*, 2017.

[12] K. Singh, "Transformation of Source Code into UML diagrams through Visualization Tool", *International Journal of Advanced Science and Technology* Vol. 29, No. 8, pp. 4861-1114, 2020.

[13] A. M. Fernández-Sáez, *et al.*, "Are Forward Designed or Reverse-Engineered UML diagrams more helpful for code maintenance? A family of experiments", *Information and Software Technology*, 2014.

[14] C. baidada, *et al.*, "An analysis and new methodology for reverse engineering of UML behavioral", *International Journal of Advanced Engineering, Management and Science (IJAEMS)*, Vol.2, Issue-7, 2016.

[15] A Nugroho, "Level of detail in UML models and its impact on model comprehension: A controlled experiment," *Information and Software Technology*, vol. 51, no. 12, 2009.

[16] S. J. Niepostyn," The sufficient criteria for consistent modelling of the use case realization diagrams with a new functional-structure-behaviour UML diagram", *Przegląd Elektrotechniczny Sigma NOT*, vol.2,PP. 31-35, 2015.

[17] R. Perez-Castillo, *et al.,* "Reengineering Technologies", *IEEE Software*, vol. 28, no. 6, 2011.

[18] R. Klimek, and P. Szwed, "Formal Analysis of Use Case Diagrams", *Computer Science*, Vol. 11, 2010.

[19] G. Booch, et *al.*, "The Unified Modeling Language User Guide", *Addison Wesley*, 2001.

[20] S. Sabharwal, *et al.*, "Empirical and Theoretical Validation of a Use case Diagram Complexity Metric", *International Journal of Information Technology andComputer Science (IJITCS), MECS Press*, Vol.9, No.11, 2017.

[21] U. Sabir, *et al.*,"A Model Driven Reverse Engineering Framework for Generating High Level UML Models From Java Source Code ", *IEEE access*, Vol. 7, 2019, DOI: 10.1109/ACCESS.2019.2950884.

[22] P. Obilikwu , *et al.,* "The Practicality Of Engineering Principles In Soft Ware Engineering" *International Journal of Advanced*, 2019, 7(12),PP. 923-934, DOI: 10.21474/IJAR01/10234.

[23] Object Management Group (OMG), "Unified Modeling Language (UML)", *Superstructure,* v2.5, http://www.omg.org/, 2017.

[24] E. V. Sunitha, and P. Samuel," Automatic Code Generation From UML State Chart Diagrams", *IEEE Access*, vol. 7, pp. 8591-8608, 2019.

[25] A. Belghiat, and A. Chaoui, "A Graph Transformation of Activity diagrams into πcalculus for Verification Purpose", *ICAASE*, 2018.

[26] B. Rajabi, and SP Lee, "Change management framework to support uml diagrams changes," *Int. Arabic J. Inf. Technol*., Vol. 16, no. 4, pp. 720–730, 2019.

[27] D. Kundu and D. Samanta, "A Novel Approach to Generate Test Cases from UML Activity Diagrams", *Journal of Object Technology*, vol. 8, no. 3, pp.65–83, 2009.