A Novel Weight Assignment Approach for Detection of Clones in Simulink Diagrams

S.MYTHILI
Research Scholar, Department of Information Technology,
Bharathiar University,
INDIA.
smythili78@gmail.com
DR.S.SARALA
Assistant Professor, Department of Information Technology,
Bharathiar University,
INDIA.
sriohmau@yahoo.co.in

Abstract :-Clone detection is a process of detecting duplicate patterns which resembles the original. The process of clone detection has been carried out for several purposes like code clone identification, clone software identification, clone image detection, clone object detection and clone language identification. Textual techniques like dynamic pattern matching, latent semantic indexing, dot plots and, Lexical Techniques like token based , line based approaches, and Syntactic Techniques like tree based approaches, metric based approaches and Semantic Techniques like Program Dependency Graph and Hybrid approaches are used for detection of clones . The proposed method detects clones in Simulink based block diagrams. Still now this process has been carried out with graph based technique. The proposed model uses a weight assignment method to identify the clones in a faster and accurate manner. It can identify both exactly matched and similarly matched clones. The proposed method is evaluated with various experimental setup and the results are compared with the existing tools.

Key-Words :- Clone Detection, Reusability , Model Driven Architecture, Simulink, UML, Software Maintenance

1. Introduction

Model-driven architecture (MDA) is an approach for designing software and is used for the development of software systems. It provides a set of guidelines for the structuring of specifications, which are expressed as models. Model-driven architecture is a kind of domain engineering, and supports model-driven engineering of software systems. In recent years many organizations have started to focus its attention on MDA as an approach to application design. This is a very positive development for several reasons.

MDA encourages well-organized use of system models in the software development process, and it reuses these models when creating families of systems. Matlab-Simulink is a popular MDA tool for designing software for small scale embedded systems to large scale flight control systems. An increasing demand for the embedded system, have created the need for automated system design. The MDA models thus created will be finally transformed in to source code which in turn is transformed in to the executable files. Since models become the main artifacts it has become a big concern to maintain the quality of the models. Duplications in such models have to be reduced to increase the maintainability and reusability of the models.

Code cloning is a method of detecting clones in the code fragment with the help of the some metrics of similarity. The previous work done concentrated on detecting simple clones [1] and the higher level clones [2] in the software. Now the focus of the clone detection has been directed to MDA. In general, code clones may be described using the topology [3] [4] mentioned in Table 1.

Like code cloning the duplication in the models are used to identify the duplicate parts in the models. Such duplicate parts when found can be included in the library and can be reused to reduce model size. Similarly the model similarity for the MDA can be categorized in to 4 types as mentioned in Table 2.

Table 1 Code Clone Types

Types	Description
Type 1	Identical code fragments except for variations in whitespace and comments.
Type 2	Syntactically identical fragments except for variations in identifiers, literals, types ,layout and comments.
Type 3	Copied fragments with further modifications. Statements can be changed, added or removed in addition to variations in identifiers, literals, types, layout and comments.
Type 4	Two or more code fragments that perform the same computation but implemented through different syntactic variants.

Table 2 Proposed Clone Types for Diagrams

Types	Description
MS1	An exactly same model with same blocks, subsystems, and same naming sequences
MS2	An exactly same model with same blocks, subsystems but with different naming sequences
MS3	Isomorphic models
MS4	Different Model architectures with similar subsystems

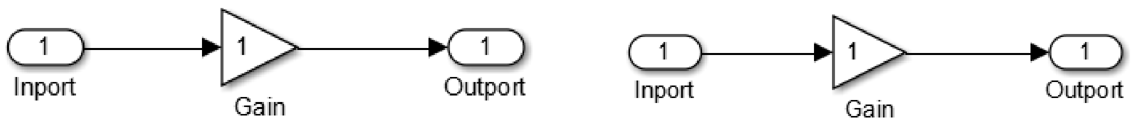


Fig. 1(a), 1(b) Compared Simulink Models for Clone Type MS1

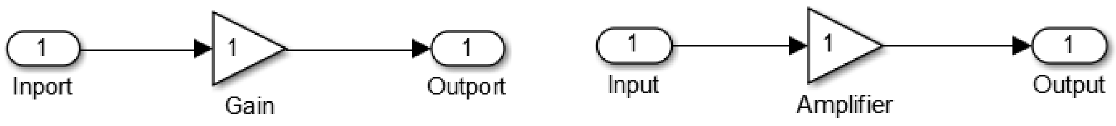


Fig. 2(a), 2(b) Compared Simulink Models for Clone Type MS2

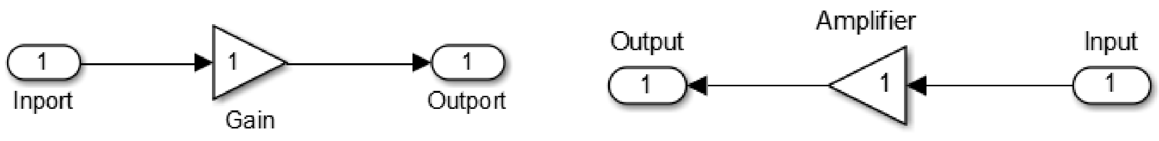


Fig. 3(a), 3(b) Compared Simulink Models for Clone Type MS3

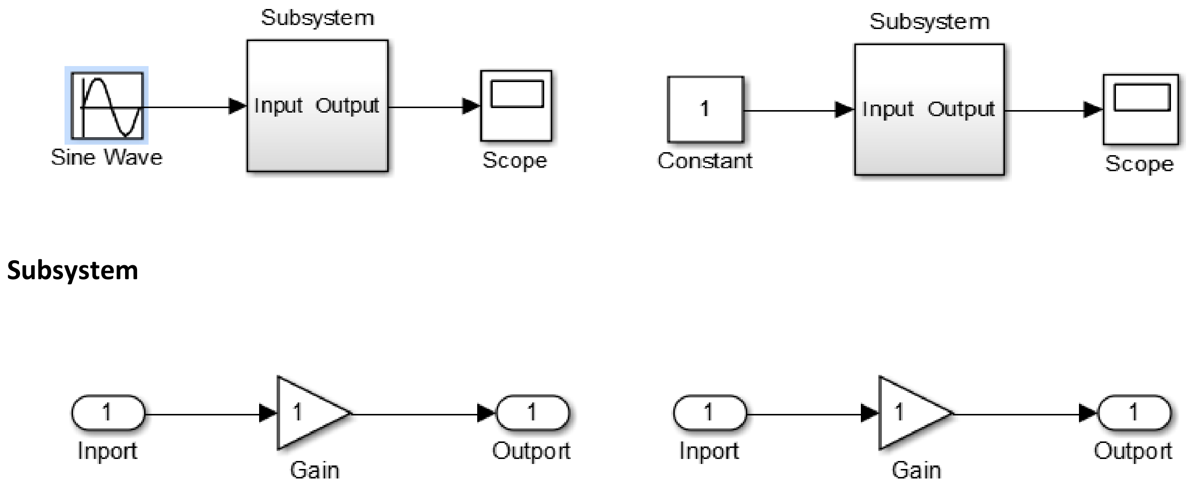


Fig. 4(a), 4(b), 4 (c) , 4(d) Compared Simulink Models for Clone Type MS4

The proposed system is designed to find the clones in Simulink models and the clone thus found are categorized into the 4 types as given in Table 2 and the diagrammatic illustrations are shown in Fig. 1, Fig. 2, Fig. 3 and Fig. 4.

Simulink is a environment for simulation and model–based design for creating dynamic and embedded systems. It is an extension to Matlab that helps us to build computer models using block diagram notation rapidly and precisely. Block diagram notation is a graphical representation of the dynamic models. A flowchart is also a graphical representation which describes the sequence of operations. In a flow chart only one block is active at a given time. In the case of simulink block diagrams, all the blocks in the block diagram may be active at one point of time. So the block diagram can be represented as a set of simultaneous equations. Simulink provides an interactive graphical environment and a customizable set of block libraries that helps us to design, simulate and implement a variety of systems. Some of the systems designed using this are communication systems, signal processing systems, control , video processing and Image processing systems.

Simulink models are made up of entities called blocks. Blocks may be mathematical, boolean or structural. User–defined blocks may be defined, and the functionality can be defined by the user. Blocks can be connected to each other through input and output ports. The number of ports in a block can be fixed or variable depending on the function of the block. Blocks in a simulink models are connected using signal lines. The Signal lines connect the block’s output port to one or more input ports.The signals carry data through the signal lines.One important feature of simulink is that very complex models can be created with ease without losing the overview.

In contrast to the text based programming languages where a lot of time is spent for developing the code, simulink is a quick way of developing models. The process of clone detection in models helps us to identify the duplicate patterns of models in earlier stage of program development. By removing duplicates at this stage will reduce the time and cost spent for the development and code thus developed will be free from clones and is easy for maintenance.

This work is organized as follows. Section 2 describes about the related work, Section 3 explains about the methodology for the proposed system, Section 4 describes the experiments and compares the results with the existing results and Section 5 gives the conclusion and future work.

2. Related Work

A model is a collection of logical entities which describe the system at multiple levels of abstraction. There is a lot of previous work carried out to detect clones in the models.

Source Code classification for clone detetction has been given by Roy and Cordy [5] and classification for the source code has been well defined but the author in [6] has given a new classification of clones in visual dataflow languages. The clones in dataflow languages are classified into 4 types from DF0 to DF3.

Clone Detective [7] is a state-of- the-art tools for detecting clones in models. It covers all the groups form DF0- DF4 to detect exact clones. The clone detective tool is open source tool for clone detection in models. The tool is implemented as ConQAT[8] and is able to detect exactly matched and approximate model clones.This tool uses a graph based approach whereas the proposed work uses the weight assignment method for detecting the clones.

Model CD is a clone detection framework for detection of clones in Simulink models proposed by Pham.et.al [9] in the year 2009. The framework comprises of two algorithms escan and ascan . The escan algorithm is designed to find the clones of the type DF0 to DF2 and the ascan algorithm finds the clones of DF3.

The work by Liu *et al.* [10] finds out duplicate patterns in UML sequence diagrams. With special preprocessing the 2 dimensional diagram is converted in to a one dimensional array. A special algorithm is designed to detect common prefixes of suffixesThey represent sequence diagram as an array and then build a suffix tree for it. The algorithm ensures that every duplication detected with the suffix tree can be extracted into a separate reusable sequence diagram.

Ren *et al.* [11] is used to detect clones in sequence diagrams and then to refractor them. The detection process is not fully automated. It extends the concept of refactoring to scenario based specifications and are described using a Message Sequence Chart.

The paper in [12] detects and visualize differences between different versions of UML documents such as class and object diagrams. It shows a unified document which contains the common and specific parts of both base documents and the specific parts are highlighted. In this approach the software documents are stored as syntax trees in XML files or in a repository system, and the version management system supports fine-grained data. The difference computation algorithm detects structural changes and enables their appropriate visualization.

UMLDiff proposed in [13] is an algorithm for automatically detecting structural changes between the designs of subsequent versions of object-oriented software. It takes as input two class models of a Java software system. It produces as output a change tree and reports it in terms of additions, removals, moves, renaming of packages, classes, interfaces, fields, methods, changes to their attributes and changes of the dependencies among these entities. UMLDiff produces an accurate report of the design evolution of the software system, and enables subsequent design-evolution analyses from multiple perspectives in support of various evolution activities. It can assist software engineers in their tasks of understanding the rationale of design evolution of the software system and planning future development and maintenance activities.

Komondoor and Horwitz [14] uses Program Dependence Graphs to detect clones in models. The novel aspect of this approach is the use of program dependence graphs and program slicing to find isomorphic PDG subgraphs that represent clones. It also finds non-contiguous clones, clones in which matching statements have been reordered, and clones that are intertwined with each other. The clones that are found are likely to be meaningful computations, and thus good candidates for extraction.

Many approaches for code clone detection like Token-based approaches, Tree-based approaches, AST–based approaches and PDG based approaches has been proposed and a survey of all these approaches can be found in [15].

E.Arora *et al.* [16] have conducted a study to all the models defined in UML including internal and External Structure of UML They have also reviewed some of the techniques available for the Model Clone Prevention and Detection.

Stephan M *et al.* [17] have presented a state of model comparison and it applies to Model-Driven Engineering. They have concentrated specifically at model matching approaches, the application of these approaches, and the types of models that these approaches are intended to work with. It also also indicates future trends and directions.

Alalfi M. H *et al*. [18] describes the plan to adapt mature code-based clone detection techniques to the efficient identification of near-miss clones in models. The goal of their work is to leverage successful source text-based clone detection techniques by transforming graph-based models to normalized text form in order to capture semantically meaningful near-miss results that can help in further model analysis tasks. NiCad code clone detector is used to identify near-miss Simulink model clones at the “system” granularity and has been extended to Simulink “model” and “block” granularities as well.

Al-Batran B *et al.* [19] have used the concept of normal forms, to find the clones in models and has also extended it to cover semantic clones with identical behavior but different structures. It also presents a generalized concept of clones for Simulink models, describes a pattern-based normal-form approach, and discusses about results and implementation.

The work in [20] uses Mutation Analysis for devising and validating new clone detection techniques and tools. It also implements a framework for evaluating Simulink model clone detectors. It includes a taxonomy of Simulink mutations, Simulink clone report transformations, and more. It outlines the method for calculating precision and recall, also discusses areas of future work, including semantic clone mutations, and developing framework implementations for other model types, like UML.

3. Clone Detection for Simulink

3.1 Methodology

The architecture for clone detection is shown in Fig 5. It takes the Query model as the input, identifies the weight of the query model and the database model and finally compares it. The algorithm for the proposed model is shown in Table 3.

Table 3 Algorithm for clone detection

Algorithm for Clone Detection in Diagrams

```
Step 1: Get theQuery model as input.
Step 2: Identify  the real blocks in the query model
Step 3: Check  for connections between block in
        Query Models.
Step 4: Separate the subsystems and identify the
         inner  blocks.
Step 5: Assign Weights for the Query Model
Step 6: Assign weights for all the models in the
        database .
Step 7: Compare weights of Query model with
         database  models.
Step 8: Identify the clone based on compared weights.
```

3.2 Various Stages of Clone Detection

3.2.1 Query model

The query model is a Simulink model which is to be checked for clone with the existing Simulink models in the database. It is the input for the whole process of clone detection. The query model can be one among the database models or a new unknown model. For the query model which is unknown to the database, the result will not display any clone. If the query model is a model in the database with some changes, the result will display the query model and the cloned database model. After getting query model further processing is carried out.

3.2.2 Weight Identification

This process assigns weights for each block in the model. It consist of four modules Real block identification, Connection Checking, Subsystem checking and Weight assignment.

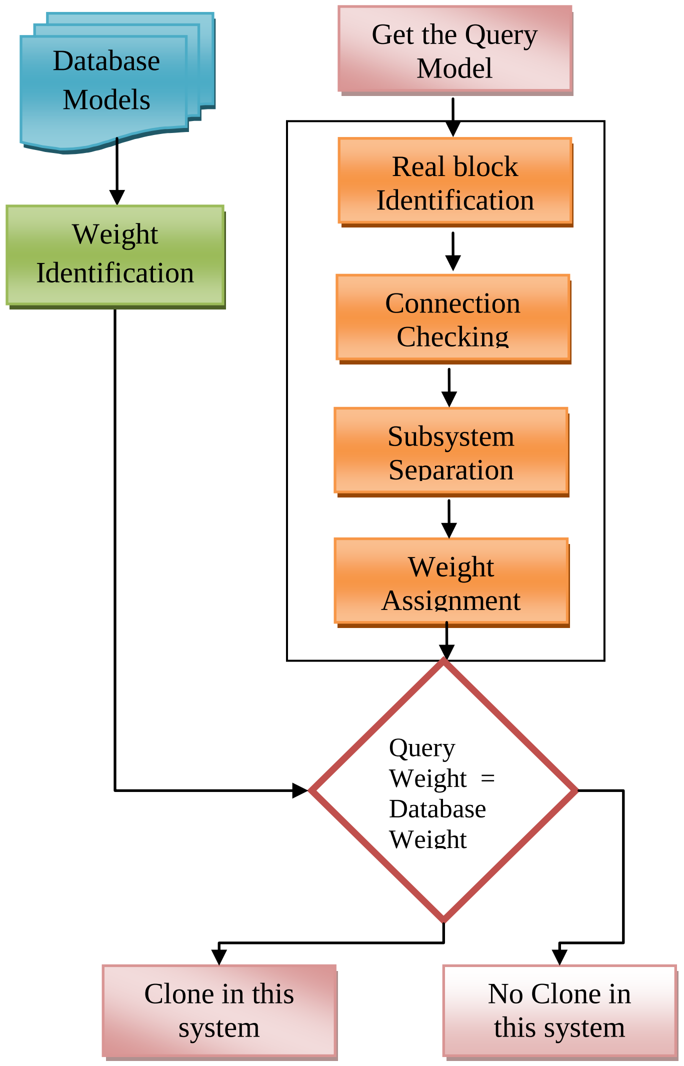


Fig. 5 Architecture for clone detection in Simulink models

3.2.2.1 Real Blocks Identification

Simulink model is made up of both inbuilt blocks and subsystems. The inbuilt block is a pre-defined block present in the Simulink library, and the subsystem is a group of inbuilt blocks. This module converts the existing blocks names into real block names. The process of identifying the blocks and converting their names into original names as present in the Simulink library is called as real block identification. For example if a Simulink query model consists of three blocks named Input, Subsystem and Output, each of these blocks is to be checked for its presence and name in the Simulink library.

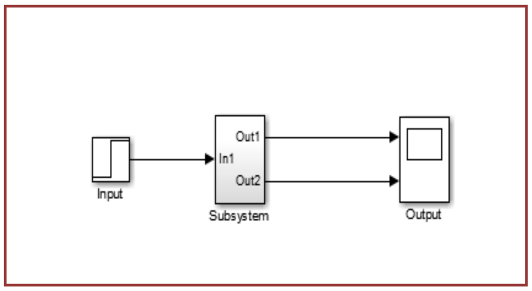


Fig. 6 Simulink Query Model

Initially in Fig. 6.the first block input is taken, and compared with all blocks in Simulink library and its real name is identified as step. Similarly the other two blocks are identified as subsystem and scope respectively and is shown in Fig. 7. For each and every block in the query model the user-defined names specified and the identified real names are collected and saved for future use.

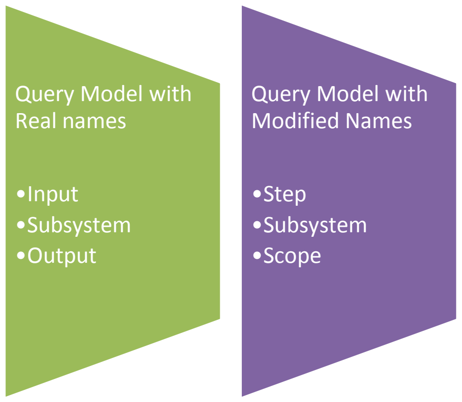


Fig. 7 Real Names and Modified Names

3.2.2.2 Connection Checking

After identifying real names, the next step is to check for connections. The block with the real names, identified from the previous part are taken and placed in an empty model and their corresponding connections are drawn (i.e.) step's output terminal is connected to the input terminal of subsystem block and output terminal of subsystem is connected to the scope's input terminal. After designing this, the block is checked with the query model for presence of similar connections. For the model mentioned in Fig. 6 , based on the connections the paths are represented as Step\ Subsystem \1\Scope and Step\Substystem \2 \Scope".

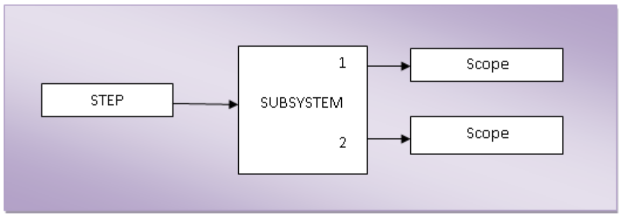


Fig. 8 Connection Checking

3.2.2.3 Subsystem Separation

The query model given as input may contain real blocks or both real blocks and subsystems. Subsystems are group of the inbuilt blocks in the simulink library grouped for a particular purpose. Since all the blocks in the query model are re-named with their real names , further the subsystems has to be re-named. The sub-systems are re-named as subsystem1, subsystem2 as in Fig. 8. After the overall naming process is completed, the subsystems are taken separately to re-name the inner blocks. The same process used for naming real blocks in query model is followed for subsystem re-naming. After subsystem re-naming paths for the Fig. 9 are mentioned as "Inport\Gain\Outport1", "Inport\PID\Transfer Function\Outport2".

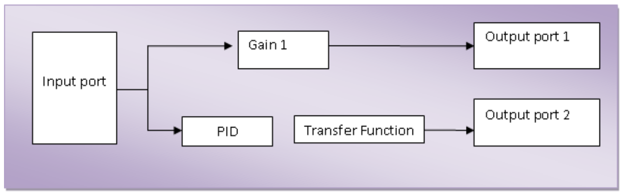


Fig. 9 Subsystems inside the main system

3.2.2.4 Weight Assignment

After the completion of assigning real names for the entire query model, the whole system is to be assigned with weights in order to compare them with all database models. From the real names created and path formed with those real names the weights are calculated. It is calculated by finding the sum of the ASCII values of all the characters in the path.A list of ASCII values are shown in the Table 4. The path for the main system is identified as step\subsytem\scope.The ASCII values of the step is 412 , the ASCII value for subsystem is 943 and the ASCII value for scope is 506. Therefore the weights for Fig. 6 is shown in Table 5.

Table 4. ASCII Values

ASCII Code	Value
45	hyphen(-)
46	.
47	/
48 to 57	0 to 9
58	:
59	;
60	<
61	(=)
62	>
63	?
64	@
65 to 90	A to Z
97 to 122	a to z

Table 5. Weight assignment for the main system

Block	Step	Step-Subsystem	Step-Subsystem-Scope
Weight	412	412-943	412-943-506

3.3 Weight Assignment for Subsystem

The above Fig. 6 has two subsystems mentioned in Fig. 9. The path for the above subsystem is "subsystem1\input port\gain\output port1", and "subsystem2\input port\PID\Transfer Function\ output port2" and their weights are mentioned in Table 6 and Table 7.

Table 6. Weight assignment for the subsystem 1

Block	Inport	Inport-PID	Inport-PID-Transfer Function	Inport – PID-Transfer Function-Outport
Weight	636	636-221	636-221-1084	636-221-1084-765

Table 7. Weight assignment for the subsystem 2

Block	Inport	Inport-Gain	Inport-Gain-Outport
Weight	636	636-415	636-415-765

3.4 Weight Identification of Database Model

The query model is now assigned real names, paths and weights. In order to identify whether the given query model is a clone or not we need to compare it with original models. This comparison is carried out only with the help of weights assigned to query model. So the original models in the database have to be assigned weights in the same manner such that the comparing process becomes easier. Weights assigned for the database models are collectively saved as another database with weights and its corresponding model names.

3.5 Comparison

The final process in the clone detection is the comparison process. For comparison two parameters are required, they are the weights assigned for query model and the weights assigned for the database models. In the above mentioned model example the weights assigned finally are 412-943-506. The weight of the main model and the weight of the subsystems are compared with the weights in the database. If any database model matches with the query models main system or subsystem then they are displayed as clones.If both the main model and the subsystem matches with the query model then the whole system is displayed as a clone.

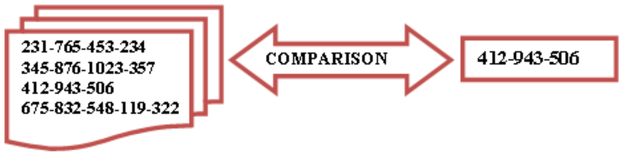


Fig. 10 Comparing Database Model Weights With the Query Model Weights

4. Experiments and Results

The Open source Simulink model based systems Sim_labs and the Seminar Designs taken for our study are available from Source-Forge and MATLAB Center . For the two systems considered number of clones and its clone pairs are extracted from the ConQat, aScan , eScan, and from the proposed method. Fig. 11. Shows the results from the Clone Detective tool for the dataset Sim_Labs. Fig 12. Shows the results from the proposed model. Clone Detective identifies 3 clone pairs in 6 seconds for the above dataset. Further for the same dataset the proposed model identifies 116 clone pairs in 10 seconds which are listed in Fig. 12. Similarly the proposed model is compared with the other exisiting tools eScan and aScan which are depicted in Table 8.

When comparing with the identification of number of clone pairs in simulink_labs with the existing tools, Clone Detective tool identifies 3 clone pairs, eScan identifies 60 clone pairs, aScan identifies 105 clone pairs and the proposed model identifies 116 Clone Pairs Fig. 13. Shows that the proposed method identifies maximum number of clones pairs. Similarly for the dataset Seminar Design Clone Detective tool identifies 29 clone pairs, eScan identifies 46 clone pairs, aScan identifies 67 clone pairs and the proposed model identifies 74 Clone Pairs.

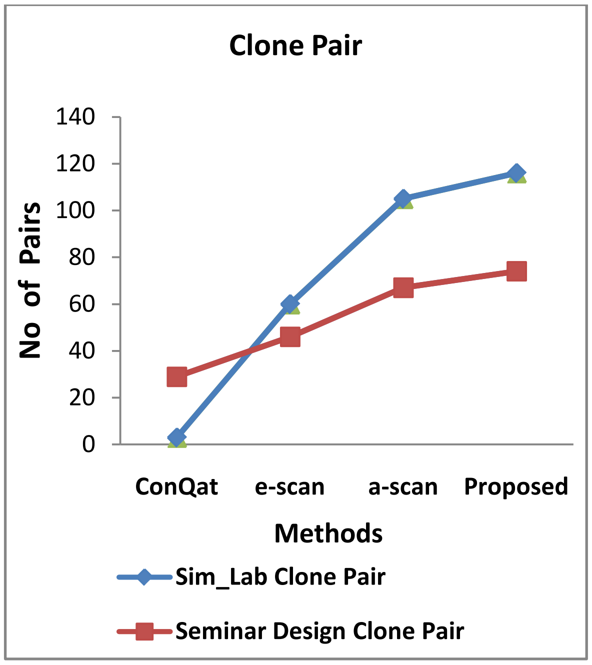
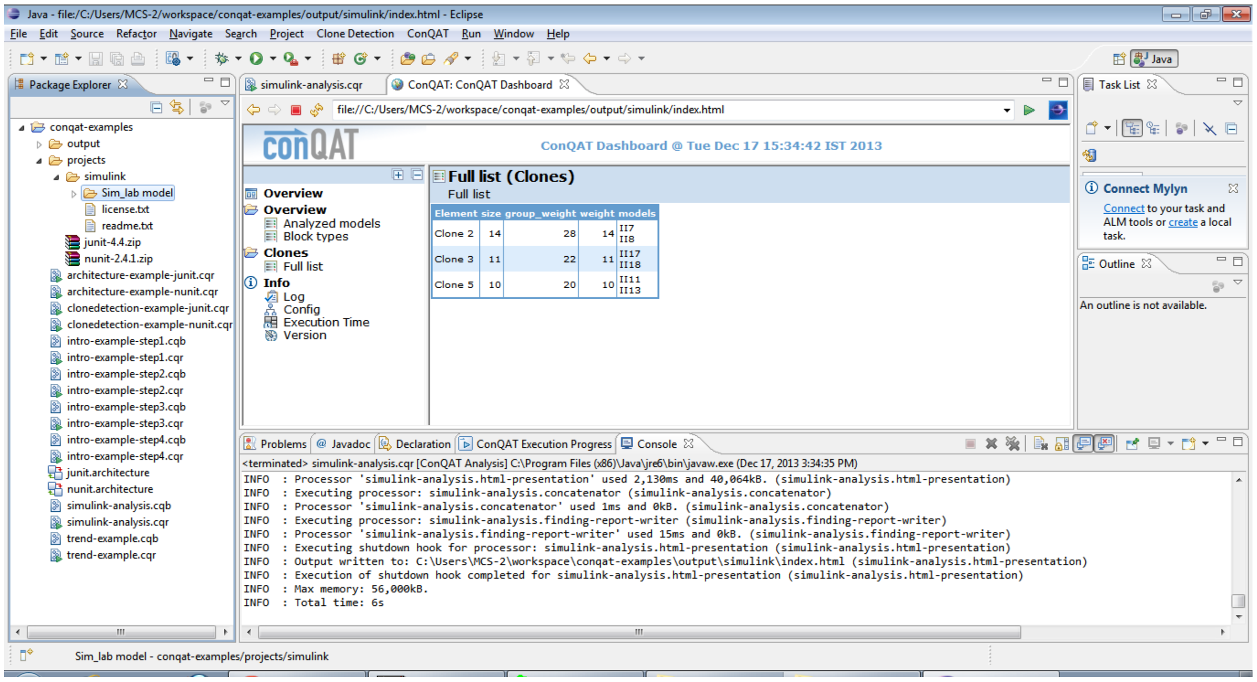


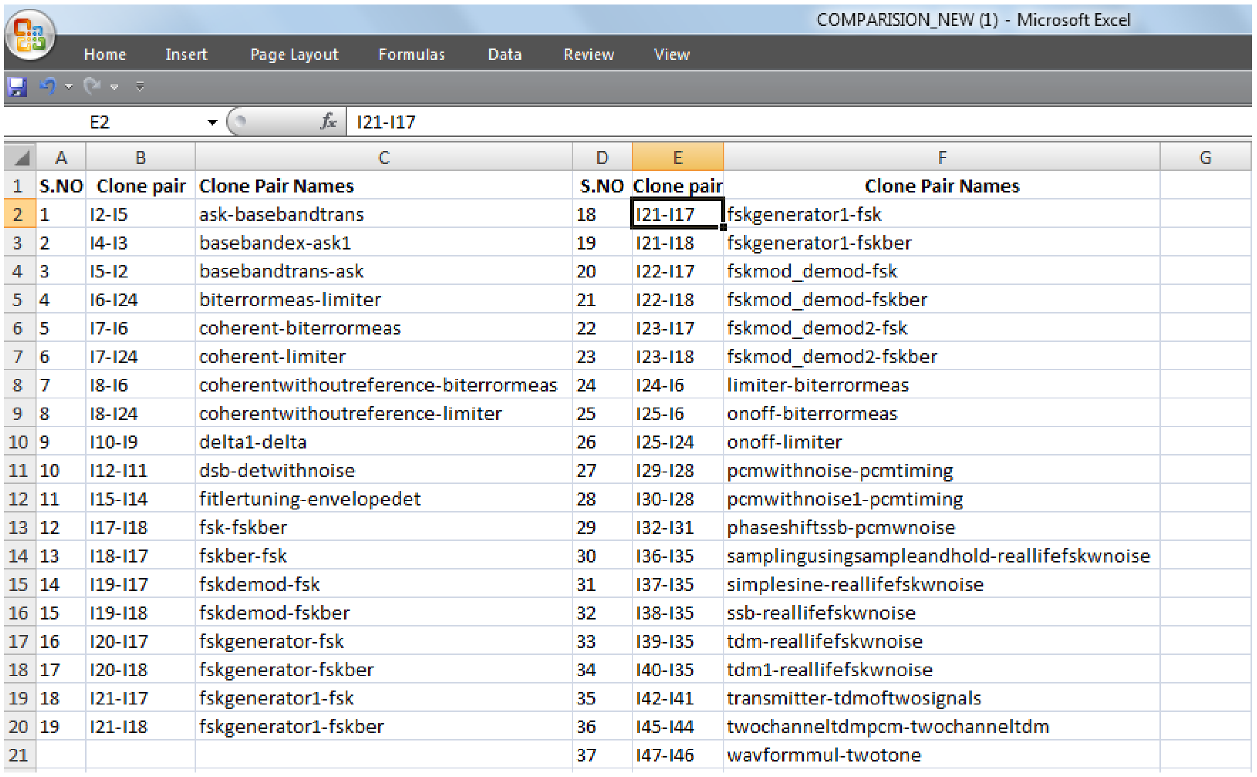
Fig. 13. Comparision of Clone Pairs



Element	size	group_weight	weight	models
Clone 2	14	28	14	II7 II8
Clone 3	11	22	11	II17 II18
Clone 5	10	20	10	II11 II13

```
<terminated> simulink-analysis.cqr [ConQAT Analysis] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (Dec 17, 2013 3:34:35 PM)
INFO  : Processor 'simulink-analysis.html-presentation' used 2,130ms and 40,064kB. (simulink-analysis.html-presentation)
INFO  : Executing processor: simulink-analysis.concatenator (simulink-analysis.concatenator)
INFO  : Processor 'simulink-analysis.concatenator' used 1ms and 0kB. (simulink-analysis.concatenator)
INFO  : Executing processor: simulink-analysis.finding-report-writer (simulink-analysis.finding-report-writer)
INFO  : Processor 'simulink-analysis.finding-report-writer' used 15ms and 0kB. (simulink-analysis.finding-report-writer)
INFO  : Executing shutdown hook for processor: simulink-analysis.html-presentation (simulink-analysis.html-presentation)
INFO  : Output written to: C:\Users\MCS-2\workspace\conqat-examples\output\simulink\index.html (simulink-analysis.html-presentation)
INFO  : Execution of shutdown hook completed for simulink-analysis.html-presentation (simulink-analysis.html-presentation)
INFO  : Max memory: 56,000kB.
INFO  : Total time: 6s
```

Fig. 11 Result of ConQat - CloneDetective for Sim_labs


COMPARISION_NEW (1) - Microsoft Excel
Home
Insert
Page Layout
Formulas
Data
Review
View
E2
I21-I17

S.NO	Clone pair	Clone Pair Names	S.NO	Clone pair	Clone Pair Names
1	I2-I5	ask-basebandtrans	18	I21-I17	fskgenerator1-fsk
2	I4-I3	basebandex-ask1	19	I21-I18	fskgenerator1-fskber
3	I5-I2	basebandtrans-ask	20	I22-I17	fskmod_demod-fsk
4	I6-I24	biterrormeas-limiter	21	I22-I18	fskmod_demod-fskber
5	I7-I6	coherent-biterrormeas	22	I23-I17	fskmod_demod2-fsk
6	I7-I24	coherent-limiter	23	I23-I18	fskmod_demod2-fskber
7	I8-I6	coherentwithoutreference-biterrormeas	24	I24-I6	limiter-biterrormeas
8	I8-I24	coherentwithoutreference-limiter	25	I25-I6	onoff-biterrormeas
9	I10-I9	delta1-delta	26	I25-I24	onoff-limiter
10	I12-I11	dsb-detwithnoise	27	I29-I28	pcmwithnoise-pcmtiming
11	I15-I14	fitlertuning-envelopedet	28	I30-I28	pcmwithnoise1-pcmtiming
12	I17-I18	fsk-fskber	29	I32-I31	phaseshiftssb-pcmwnoise
13	I18-I17	fskber-fsk	30	I36-I35	samplingusingsampleandhold-reallifefskwnoise
14	I19-I17	fskdemod-fsk	31	I37-I35	simplesine-reallifefskwnoise
15	I19-I18	fskdemod-fskber	32	I38-I35	ssb-reallifefskwnoise
16	I20-I17	fskgenerator-fsk	33	I39-I35	tdm-reallifefskwnoise
17	I20-I18	fskgenerator-fskber	34	I40-I35	tdm1-reallifefskwnoise
18	I21-I17	fskgenerator1-fsk	35	I42-I41	transmitter-tdmoftwosignals
19	I21-I18	fskgenerator1-fskber	36	I45-I44	twochanneltdmpcm-twochanneltdm
			37	I47-I46	wavformmul-twotone

Fig. 12 Results of Proposed Model for Simulink_labs

Table 8. Comparison of the results from tools.

System	Simulink_labs			Seminar_Designs		
Methods	Time (sec)	No.of Clones	Clone Pair	Time (sec)	No.of Clones	Clone Pair
Clone Detective	6	6	3	77	101	29
eScan	120	183	60	300	180	46
aScan	1.4	272	105	6.8	424	67
Proposed	10	298	116	15	458	74

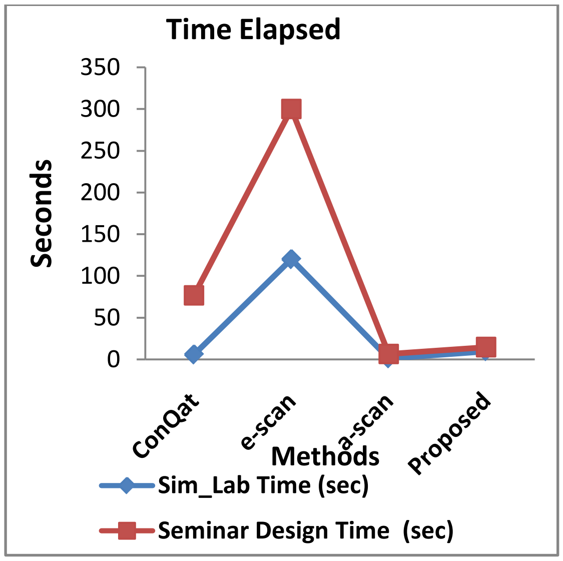


Fig. 14. Comparision of the Simulink models with Time (Sec)

When comparing the execution time of simulink_labs to identify the number of clone pairs with the existing tools, ConQat tool executes in 6 seconds, eScan executes in 120 seconds, aScan executes in 1.4 seconds and the proposed model executes in 10 seconds Fig. 14. Shows that the proposed method takes minimum time to find maximum number of clones pairs. Similarly for the dataset seminar_designs the existing tools, ConQat tool executes in 77 seconds, eScan executes in 300 seconds, aScan executes in 6.8 seconds and the proposed model executes in 9 seconds.

5. Conclusion

UML diagrams and MDA have become popular in recent days, so that with the advent of these diagrams and architectures it becomes necessary to detect duplications in such formats. In this research we have analyzed the reasons for duplications in Simulink diagrams and a new methodology has been proposed to detect the clones in UML diagram using the method of weight assignment. Previously lot of tools of available for the detection of clones in models, but the proposed method puts forth a new algorithm which uses the weight factor for detection. Most of the approaches which have been previously studied, have also detected the clones in models but they all use a graph based technique to detect the clones. While comparing with the other approaches, this work involves real block identification, subsystem separation, connection checking and puts forth a new algorithm which uses the weight factor for detection. The weight assignment method performs better when compared to the other methods which converts the query model in to a graph for the detection process. It can find out more clones in very less execution time when compared with the other tools like ConQat , eScan and aScan. Further, this process is likely to be implemented to check for clones in various industrial projects for evaluations. Perhaps, it can also be extended to detect clones in process-oriented models.

References:

[1] S.Mythili, Dr.S.Sarala, A Language Independent Approach for Method Level Clone Detection Using Fingerprinting in *International Journal of Advanced Research in Computer Science*, Volume 3, No. 2, March-April 2012.

[2] S.Mythili, Dr.S.Sarala, Enhanced Technique to Identify Higher Level Clones in Software in the *Proceedings of the Second International Conference on SocProS* , December 28-30, 2012 published by Advances in Intelligent

Systems and Computing- Springer, Vol. 236 , ISBN 978-81-322-1601-8.

[3] Chanchal Kumar Roy, James R. Cordy , and R.Koschke, Comparison and Evaluation of Code Clone Detection techniques and tools: A qualitative approach, *Journal Science of Computer Programming*,Vol.74, pp 470-495, May, 2009.

[4] N Schwarz, M Lungu, R Robbes ,On How Often Code Is Cloned Across Repositories, *Proceedings of the International Conference on Software Engineering*, pp 1289-1292, 2012.

[5] C. K. Roy and J. R. Cordy, A survey on software clone detection research, *School of Computing* TR 2007-541, Queen's University, pp. 115, 2007.

[6] N. Gold, J. Krinke, M. Harman, and D. Binkley, Issues in clone classification for dataflow languages,*Proceedings of the 4th International Workshop on Software Clones*, IWSC '10, pp. 83-84, 2010.

[7] Technical University Munich. Continuous Quality Assessment Toolkit Website http://www.conqat.org

[8] E. Juergens, F. Deissenboeck, and B. Hummel. Clonedetective - a workbench for clone detection research,*Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pp. 603-606, 2009.

[9] N. H. Pham, H. A. Nguyen, T. T. Nguyen, J. M. Al-Kofahi, and T. N.Nguyen. Complete and accurate clone detection in graph-based models,*Proceedings of the 31st International Conference on Software Engineering*,ICSE '09, pp.276-286, 2009.

[10] Liu, Hui, Zhiyi Ma, Lu Zhang, and Weizhong Shao,Detecting duplications in sequence diagrams based on suffix trees, *13th Asia Pacific Software Engineering Conference*, APSEC, pp. 269-276, 2006.

[11] Ren, Shengbing, Kexing Rui, and Greg Butler,Refactoring the scenario specification: A message sequence chart approach, *Object-Oriented Information Systems*. Springer Berlin Heidelberg, pp. 294-298, 2003.

[12] Ohst, Dirk, Michael Welle, and Udo Kelter. Differences between versions of UML diagrams, *ACM SIGSOFT Software Engineering Notes* ,vol. 28,no.5, pp. 227-236, 2003

[13] Xing, Zhenchang, and Eleni Stroulia,UML Diff: an algorithm for object-oriented design differencing, *Proceedings of the 20th IEEE/ACM International Conference on Automated software engineering*, ASE'05, pp. 54-65.

[14] Komondoor, Raghavan, and Susan Horwitz, Using slicing to identify duplication in source code, *Static Analysis*, Springer Berlin Heidelberg, pp. 40-56,2001.

[15] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo.Comparison and Evaluation of Clone Detection Tools, *IEEE Transactions on Software Engineering*, vol. 33, no. 9, pp. 577-591, 2007

[16] E.Arora and Y. Sarita Choudhary, A Review of Clone Detection Techniques using Model Semantics,*Global Journal of Computer Science and Technology* ,vol 13, no 6, 2013.

[17] Stephan, M, & Cordy, J. R, A Survey of Model Comparison Approaches and Applications. *In Modelsward*,pp. 265-277, 2013.

[18] Alalfi M. H., Cordy, J. R, Dean, T. R, Stephan, M & Stevenson, A Near-miss model clone detection for Simulink models. *In the 6th International Workshop on Software Clones (IWSC)*, pp 78-79,2012 .

[19] Al-Batran B., Schätz, B., & Hummel, B,Semantic clone detection for model-based development of embedded systems, *In Model Driven Engineering Languages and Systems*, pp. 258-272, 2011.

[20] Stephan M., & Cordy, J. R. Model clone detector evaluation using mutation analysis, *In International Conference on Software Maintenance and Evolution*,pp. 1-6,2014.