Volume-4, Issue-6, June-2015 • ISSN No 2277 - 8160

Research Paper Engineering

Improvement in Automated Model Based Testing by Natural Language Approaches

Priyanka Nanda Student, Dept. of CSE Lovely Professional University, Phagwara, Punjab

Mr. Makul Mahajan Assistant Professor, Department of CSE Lovely Professional University, Phagwara, Punjab

ABSTRACT

Models in particular finite state machine models – provide an invaluable source of information for the derivation of effective test cases. However, models usually approximate part of the program semantics and capture only some of the relevant dependencies and constraints. As a consequence, some of the test cases that are derived from models are infeasible. The primary objective is to generate model-based system and acceptance test cases considering Natural Language requirements deliverables. The generation of Executable Test Cases which predicted behaviors that did not exist in the expert's approach.. Model Checking combined with k permutations of n values of variables and specification patterns were used to address this goal. Models in particular finite state machine models – provide an invaluable source of information for the derivation of effective test cases. However, models usually approximate part of the program semantics and capture only some of the relevant dependencies and constraints. As a consequence, some of the test cases that are derived from models are infeasible. We will use NLP-MBT tool in which different test case will execute according to N-gram statistics.

KEYWORDS : Software engineering, Software testing, Model based testing, Natural language processing, N-Gram.

Introduction

A.Software Engineering

Software engineering is the learning and presentation of engineering to the plan, progress and keep of software. It is an engineering reprimand that is concerned with all facets of software creation. Software engineering is a great, multifaceted, and nonfigurative subject it is problematic to hypothesis vigorous learning trainings that build on the student's basic knowledge of programming and tranquil teach elementary software engineering ideologies. It is also the case that launch students stereotypically know how to build small programs, but they have petite skill with the procedures necessary to harvest consistent and continuing maintainable components. It mainly focus on step-by-step that points students toward the structure of exceedingly reliable trivial components using well known, greatest-applies software engineering performances [21]. They are:

- Requirements engineering
- Software design
- Software construction
- Software testing
- Software maintenance
- Software configuration management
- Software engineering management
- Software engineering process
- Software engineering tools and methods
- Software quality management[22]

B. Software Testing

Software testing is a learning showed quality of the product. The process of defining the behavior of system and check whether the system is to be work properly. In software testing, following are the different properties specify the degree to which the component or system under test:

- meets the requirements that focused its design and development,
- replies properly to all kinds of inputs,
- performs its functions within an suitable time [23].

C. Model Based Testing

Model-based testing is used for designing model-based and executes artifacts that are to perform software testing or system testing. Models are used to represent testing approaches. In a model based testing, a model concerning a SUT which is typically an abstract that defines the behavior of system under test. [8][11].

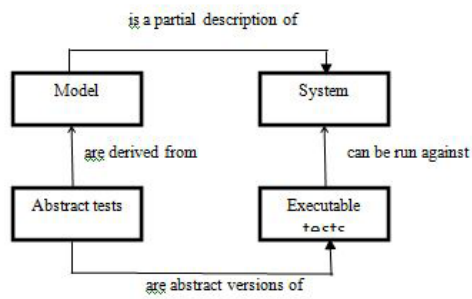

Figure 1. Graphical view of Model Bases Testing

Model based testing provides a system for automatic generation of test cases using models mined from software artifacts. MBT approach has threeessentials:

- Software comportment
- Measures
- Generate supporting substructure for the tests.

Basically, MBT has two main approaches:

- Offline MBT
- Online MBT

Offline MBT:

- It permits automate test execution in third party test execution platform.
- It makes possible to create a tool chain.
- It yields determinate sets of tests and executes [25].

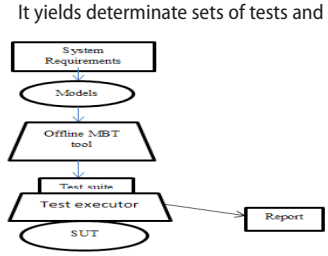

Figure 3: Offline Model Based Testing

Online MBT:

- It produce test case which are in performance.
- It stimulating non-deterministic system.
- In it, infinite test suite is repeatedly [25].

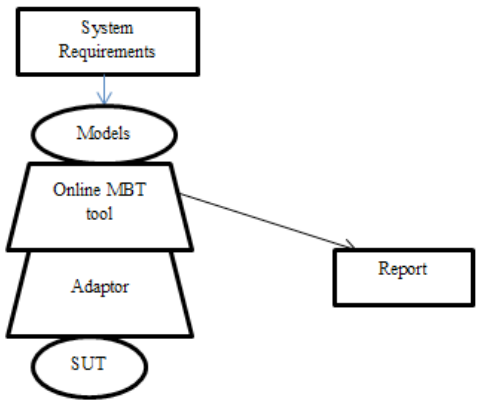

Figure 2. Online model Based Testing

D. Natural Language Processing

Natural language is defined to the language spoken by people, e.g. English, Japanese, as different to artificial languages, like C++, Java, etc.

Natural Language Processing is the field of computer science and phonology which is concerned with the communications between computers and human languages [23]

II. LITERATURE SURVEY

A.Pretschner*et al* (2012) proposed that the models of system under test which are depends on model based testing that develop test cases for the system. Here authors discussed the classification of main features that cover the model based testing methods that show that how to classify so that it should be considerate the comparisons and modification of model based testing methods. In this paper, authors have discussed different tools and methods to improve the scalability which increase the performance of test generation [1].

H.Samih*et al* (2014)proposed a Model based testing for PL-usage models has proposed to reinforce model based testing that provides automatic test case generation which furnish with variability information. , it show a supported tool that allow model based testing which generate test cases for not only one product but different developments [111

Bernhard Rumpe(2014) proposed to check what properties a modeling UML sequentially support extreme programming technique well. It uses XP which is an explicit reaction to the complexity of today's modeling methods like the Unified Process, the Open Toolbox of techniques are needs to make UML suitable for an extreme modeling approach [3].

CyrillaArtho*et al* (2013) proposed to display test sequences of application programming interface calls which engross model based testing with different system configuration. It proposed to try SAT solvers techniques which used for verification back-ends that generates sequences of valid API for progressive feature of SAT solvers [5].

Julien Botella *et al* (2013) proposed the procedures of model based testing has proposed that where some application is done in the scene of a qualification testing phase made by at autonomous designers, developers and sponsors of the cryptographic components under test appeal on security cryptographic components. It will work upon the bid of MBT techniques which use MBT for pure functional testing that is the test generation model and the test selection criteria [12].

Mark Harman*et al* (2013) proposed to use the slant of Oracle automation that is the main key to isolate present constriction which hampers unstinting general automation for tests and Oracle automation includes modeling, specifications, contract driven development and metamorphic testing. This paper also tells the ample report of Oracles in software testing which describe implied attitude that gives some endowment for the lack of Oracle [15].

Briand *et al* (2012) have proposed commonly used FSM model which is use in UML, class and sequence diagram. Here author can represent the values of class attributes and the graphical objects. Here author discussed how to represent abstract and concrete applications where each FSM represents Authors discussed different transitions in FSM which signifies action or event related to an application. Here mainly work depends on how to perform event or action in an application and how to call method so that application state can change during execution. [2].

Dudekula Mohammad Rafi *et al* (2012) have proposed that how space is near between both views by inspecting in respect of the advantage and bound of test automation. This paper builds analysis of some advantages and drawback of software test automation in educational information. This tells how to unify actor view of software test automation [8].

Mohamed Mussa*et al* (2012) have proposed that how to brings some idea based on model based performances that use UML2 Testing Profile for generating integration test cases from unit test models and how to construct integration test model that use for UTP models [16].

Petra Brosch*et al* (2012) have proposed that how to use of overlapping information which innate in multiple views of models for automatic testing has proposed. The authors have proposed to use multi-view modeling languages like UML that offer different diagram types to lower the complexity of re-counting software systems where each diagram allowing for splitting a complex model into various areas of concern. So, in that way, the diagrams are complemented with one another, that work together to provide a holistic representation of the system. Here we find that how the information can be used as test data [17].

Yoav Bergner*et al* (2012) have proposed that how collaborative filtering is applied to use dichotomously scored student response data and find optimal parameters for each student and item based on cross-validated prediction accuracy. To use CF, it is fast, stretchy and firm [20].

Cristran Cadar*et al* (2011) proposed to use symbolic executions which is a program analysis performance used for solving restraint in technology that increased availability of computational power. In this paper, it become able to all plainly that how modern symbolic execution slant empower organized testing for bug finding and symbolic execution used for handle the expanding number of paths in the code [6].

Gervazi*et al* (2011)proposed a formal framework for identifying, analyzing and managing inconsistency in natural language requirements derived from multiple stakeholders. In this article, a particular inconsistency namely logical contradiction (any situation in which some fact α and its negation ¬α can be simultaneously derived from the same specification) was concentrated [10].

Dias Neto*et al (*2011)proposed to define the behaviors which are appropriate for measuring the testing. In this phase is also known as group related to test. Here authors take an example of FSM models which is used as an test case which showing output that involves sow to classify an events [7].

ChristelBaier*et al* (2010)proposed the checking of Model. Gervazi also proposed a methodology for the lightweight validation of natural language requirements. In this paper it tells validation as a decision problem [4].

Kedian Mu*et al* (2008) proposed the priority -based scoring vector, which participates the measure of the degree of inconsistency with the measure of the significance of inconsistency. Here author discussed for checking the inconsistencies in natural language requirements [14].

El-Far *et al* (2007) proposed the model-based testing which is an method that bases common events of the software testing process such as test case generation and test results evaluation [9].

Utting*et al* (2006) proposed a model-based testing which consists of a test strategy in which test cases are derived completely from a model that describes some feature of software. In this paper authors discussed the behavior or structure of the software which has been formalized by means of models with well-defined rules such as UML diagrams. Here authors also discussed that a model-based testing technique can be applied to any type of testing (functional, structural, etc[19].

Sarma*et al* (2005)In this paper, authors proposed that the technique which are depend on the source based on UML show system state graph where use case models, sequence diagrams, and State chart models represent. In this paper, authors discussed to cover the transition path coverage Here authors discussed that how the work can interact with users during the process. [18].

Jurafsky*et al* (2004) proposed to differentiate six categories of the knowledge of language that is needed to engage in complex language behavior: Phonetics and Phonology, Morphology, Syntax, Semantics, Pragmatics, and Discourse. Here authors discussed that how to use a novel semantic encoding of the CNL behavior in the form of a timed transition relation [13].

III.PROBLEM FORMULATION

A. N-Gram Approaches

- This is prediction model using the n gram for prediction.
- NLP uses in sentence derivation which is use for prediction of sentences, which have many possibilities.
- If we resemble above give point and model based testing is the same because model based

testing have many test cases which can use any point but some test cases is useful ,so we can use n gram statics for predicting these useful test cases.

- N-gram approach useful for reduce the complexity of model based testing and increasing the feasibility of model based testing.

B. Problem statement

The model based testing have many way to select the test cases but some test cases is useful, so we can say model-based testing is non deterministic approach, we can reduce this by N-gram statistics of test cases and reduce the complexity of model based testing.

Avoiding the generation of infeasible event sequences using the N-gram statistics which predict the feasible sequences of test cases it is same as sentences and N-gram statistics can be used in a similar way as in NLP to achieve such purpose. generating event sequences that contain N-gram previously observed in real executions, the likelihood that such sequences will in turn be executable is increased.

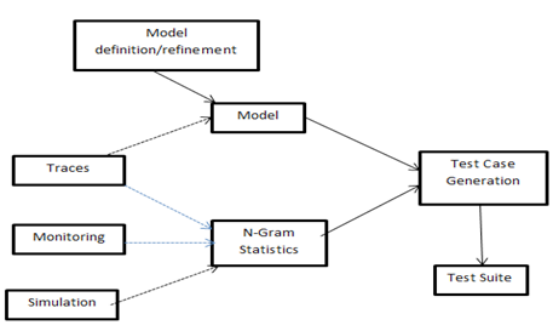

Figure 4: Problem Statement of proposed work

IV. PROPOSED METHODOLOGY

N -gram statistics play important role in prediction next test cases in test suite. But it is important to using this in model based testing.

Probabilistic model for prediction next word in testing predict best test cases in test suite.

N-gram statistics best possible combination of states in test suite.

N-gram Statistics reduce the event sequences by prediction approach. Avoiding the generation of infeasible event sequences is very similar to avoiding the derivation of sentences.

Depending on the application the most appropriate among these three data collection methods may different.

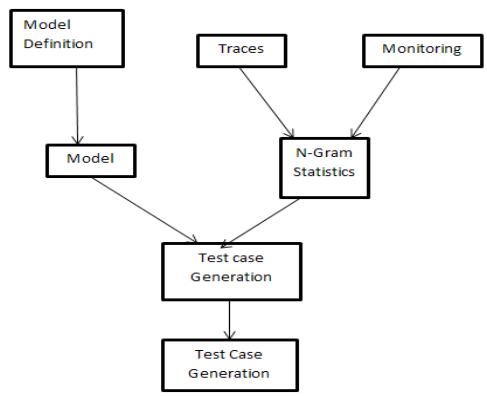

Figure 5: Flowchart of Research Methodology

Figure 2 shows a high level view of the proposed approach.While graph visit test case generation algorithms (Random, Depth first and Breadth first) require just one input (i.e., the model), N-Gram based test case generation needs two inputs: model and N-gram statistics. The model can be defined manually by the user; it can be inferred automatically from execution traces using state abstraction or event sequence abstraction or a mixed approach can be followed, in which the model is first inferred and then it is manually refined by the user.

V.OUTCOMES AND DISCUSSION

Parameters for testing results:

Feasibility: This is the Ratio between feasible test sequences and total number of test sequences generated by each test strategy.

Coverage: This is the Ratio between covered transitions and total number of transitions in the model.

Size of test cases: this is the Number of test sequences in the test suite. (Test suite size)

Length of test: This is the Average number of events in the test (Test case Length) sequences added to each test suite

1. Feasibility with all test cases in Flexi- store software

	Feasibility	Coverage (%)	Size of test cases	Length of test case	
Depth first	2%	58	4	25.34	
Breadth first	11%	87	34	3.38	
Random	72%	65	4.5	55.01	
Ngram-2	65%	88	4	47.73	
Ngram-3	4%	73	4.3	49.66	
Ngram-4	27%	88	5	45.48	
Interpolated3	39%	79	14.33	54.15	Flexi-store
Interpolated4	78%	89	13	53.41	
Interpolated5	57%	8	12	51.27	
Interpolated6	81%	89	10	55.95	
Interpolated7	46%	8	9.7	53.63	
Interpolated8	71%	88	9	55.01	
Interpolated9	43%	8	8.39	57.18	
Interpolated10	86%	87	8	56.5	

1(a).

	Feasibility
Depth first	2%
Breadth first	2%
Random	11%
Ngram-2	72%
Ngram-3	65%
Ngram-4	4%
Interpolated3	27%
Interpolated4	39%
Interpolated5	78%
Interpolated6	57%
Interpolated7	81%
Interpolated8	46%
Interpolated9	71%
Interpolated10	43%

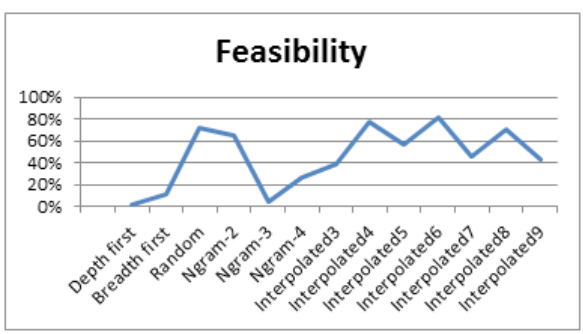

Figure 6: (Feasibility VS total no of test sequences)

Feasibility of Breadth first, Depth first and Random Approaches less than N-gram and interpolated N-gram approaches (INTERP), it will increase when increase N-gram value of n. It show prediction increase when N-gram increase

	Coverage (%)
Depth first	58
Breadth first	87
Random	65
Ngram-2	88
Ngram-3	73
Ngram-4	88
Interpolated3	79
Interpolated4	89
Interpolated5	8
Interpolated6	89
Interpolated7	8
Interpolated8	88
Interpolated9	8
Interpolated10	87

1(b).

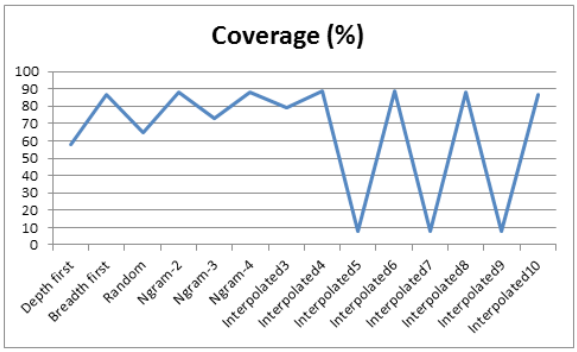

Figure 7: (Coverage VS total number of test sequences)
Here step by step feasibility of Breadth first, Depth first and Random approaches increases when N-Gram value increase and simultaneously N-Gram approaches (interpolated) increases.

	Size of test cases
Depth first	4
Breadth first	34
Random	4.5
Ngram-2	4
Ngram-3	4.3
Ngram-4	5
Interpolated3	14.33
Interpolated4	13
Interpolated5	12
Interpolated6	10
Interpolated7	9.7
Interpolated8	9
Interpolated9	8.39
Interpolated10	8

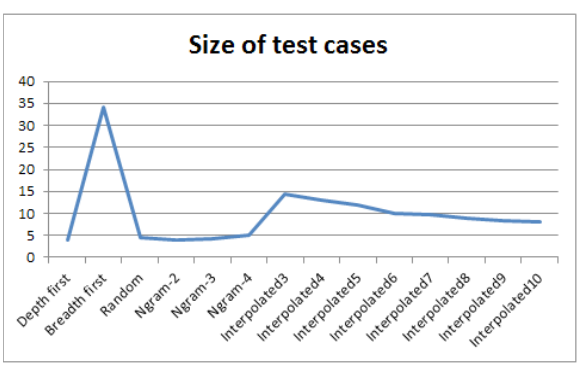

Figure 8: (Size of test cases VS total number of test sequences)

Feasibility of Breadth first, Depth first and Random Approaches more than N-gram and interpolated N-gram approaches (INTERP). where N-gram decrease when increase interpolated N-gram value of n.

1(d)

	LENTH
Depth first	25.34
Breadth first	3.38
Random	55.01
Ngram-2	47.73
Ngram-3	49.66
Ngram-4	45.48
Interpolated3	54.15
Interpolated4	53.41
Interpolated5	51.27
Interpolated6	55.95
Interpolated7	53.63
Interpolated8	55.01
Interpolated9	57.18
Interpolated10	56.5

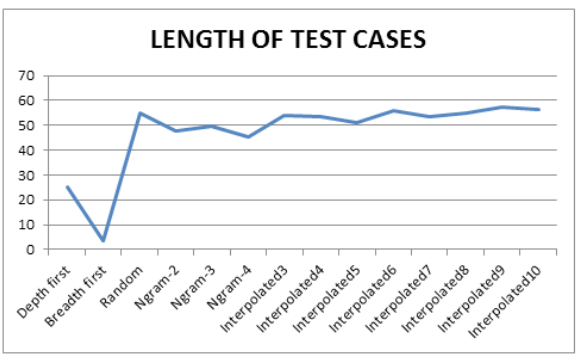

Figure 9: (Length of test case Vs total number of test sequences)

Feasibility of Breadth first, Depth first approaches less and Random Approaches increases and simultaneously interpolated N-gram approaches (Interpolated) will increase, it will increase when increase N-gram value of n. It show prediction increase when N-gram increase.

1(e)

	Feasibility	Coverage (%)	Size of test cases	Length of test case
Depth first	2%	58	4	25.34
Breadth first	11%	87	34	3.38
Random	72%	65	4.5	55.01
Ngram-2	65%	88	4	47.73
Ngram-3	4%	73	4.3	49.66
Ngram-4	27%	88	5	45.48
Interpolated3	39%	79	14.33	54.15
Interpolated4	78%	89	13	53.41
Interpolated5	57%	8	12	51.27
Interpolated6	81%	89	10	55.95
Interpolated7	46%	8	9.7	53.63
Interpolated8	71%	88	9	55.01
Interpolated9	43%	8	8.39	57.18
Interpolated10	86%	87	8	56.5

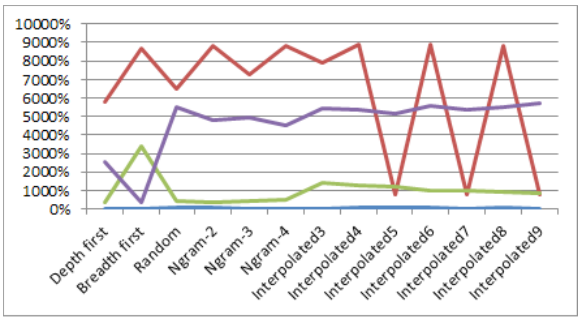

Figure 10: (Ratio between all test cases Vs total test sequences)

Feasibility of Breadth first, Depth first and Random Approaches less than N-gram and interpolated N-gram approaches (Interpolated), it will increase when increase N-gram value of n. It show prediction increase when N-gram increase.

V. CONCLUSION AND FUTURE SCOPE

In this thesis we have working on statics of N-gram because we resemble these two problem one of prediction of word in sentence by using of previous words and second one model based testing in which predict the path of next test case by using previous test cases by n gram approaches.

We have check four metrics for analysis our results, these are coverage, feasibility, length of test cases, number of test cases in one test suite. We have compare with previous approach like depth first, Breadth first and random .N-gram approach given significance difference from previous approaches.

In future we can use the N gram approach to web application and verification of hardware because in both cases same problem as model based test cases , so we can generalize our model.

REFERENCES

[1] A. Pretschner, M.Utting (2012) "A taxanomy of model based testing approaches" | | [2] Andrews, J.H, Briand, L.C.Labiche, Y, (2012) "Is mutation an appropriate tool for testing experiments? In: International Conference on Software Engineering." , pp. 402–411. ACM, New York, NY, USA | [3] Bernhard Rumpe , (2014) " Executable Modelling with UML- A vision or a nightmare". | [4] ChristelBaier, Joost–Pieter Katoen, (2010) " Invariant-based automatic testing of modern web applications. IEEE Transactions on Software Engineering", 38(1):35–53. | [5] Cyrilla Artho, Armin Biere, Martina Seide, (2913) "Model-Based Testing for verification Back-ends". | | [6] Cristran Cadar, Patrice Godefroid, Sarfraz Khurshid, Corina S. Pasareanu,, Kaushik Sen, Willeam Visser, (2011) "Symbolic execution for Software Testing in Practice". | | [7] Dias Neto, Vorobev, E., Lapschies, FZahlten, (2011) "Automated model-based testing with RT-Tester.", Tech. rep., Universit¨ t Bremen | [8] Dudekula Mohammad Rafi, Katem Reddy, Kiran Moses, Kai Petersen, (2012). "Benefits and limitations of Automated Software Testing: Literature Review & practitioner survey". | | [9] El-Far, Whittaker, (2007) "Word sense disambiguation: A survey. ACM Computing Surveys", v. 41, n. 2, p. 1–69. | [10] Gervazi, Zowghi, (2011) "Testing Web Applications by Modeling with FSMs. Software and System Modeling", Vol 4, n. 3, pages 326–345. | [11] H.Samih, H. Lguen, R. Bogushk, (2014) "Deriving Usage Model Variants for Model-Based Testing". | | [12] Julien Botella, Fabrice Bouquet, Jean Francois, Capuron, FranckLebeau, Bruno Legard, Florence Schadle, (2013) "Model-Based Testing of Cryptographic components lessons learned from experience". | [13] Jurafsky, Martin, (2004) "Principles of model checking" . Cambridge, MA, USA: The MIT Press,.975 p. | [14] Kedian Mu, ZhiJin, Ruqian Lu, Weiru Li, (2008)."Combining model-based and combinatorial testing for effective test case generation. In Process of the ACM International Symposium on Software Testing and Analysis (ISSTA"), pages 100–110. | [15] Mark Harman, Phil Mcminn, Muzammil Shahbaz, Shin Yoo (2013) "A comprehensive survey of Trends in oracle For Software Testing". | | [16] Mohamed Mussa, Ferhat Khandek, (2012) "Towards a Model Based Approach for Integration Testing". | | [17] Petra Brosch, U. Egly, Schastian Gabmeyer, Yerti Kappel, Martina Seisi, Hans Compits, Magdalena Widl, Maneel Wimmer, (2012) "Towards scenario-based testing of UML Diagrams" | . | [18] Sarma, Mall, (2005) "Using model checking to generate tests from specifications". | [19] Utting, Legear, (2006) "Formal methods in industry: achievements, problems", future. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), 28, Shanghai, China. Proceedings... New York, NY, USA: ACM, 2006. P.761–768. | [20] Yoav Bergner, Stefan Droschler, Gerd Kortemeyer, Saif Rayyan, Daniel Seaton , (2012) "Collaborative filtering Analysis of Student Response Data: Machine- Learning Item Response Theory" | | [21] Ackerman, A.F, (2014) "An Introduction to Software engineering", | [22] RS Pressman, (2005) "Software engineering: practitioner's approach" | [23] V Ambriola, G Tortora, (1993) "An Introduction to SoftwareArchitecture, Advances in Software | Engineering and Knowledge Engineering", Volume I | | [24] http://en.wikipedia.org/wiki/Software_testing | | [25]http://www.cs.tut.fi/tapahtumat/testaus08/Olli-Pekka.pdf |