ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 2, April 2014

Improvised Formal Specification-Based Inspection and Prediction

B.Arun Gunalan[1], G.Ramakrishnan[2]

PG Student [Software Engineering], Dept. of IT, Easwari Engineering College, Chennai, Tamilnadu, India [1]

Assistant professor, Dept. of IT, Easwari Engineering College, Chennai, Tamilnadu, India [2]

ABSTRACT: Inspection is commonly used for software error detection and correction. In the Formal Specification Based Inspection method, inspection is carried out to find out whether every functional scenario that is defined in the requirement specification is correctly implemented by a set of program paths. The method comprises of five steps: deriving functional scenarios from specification, deriving paths from program, linking scenarios to path, analyzing paths against the corresponding scenario, and producing an inspection report. In the proposed paper two things are dealt in the first, five more have been added to the analysis level namely: web application performance level, globalization, error handling, reusability and maintainability which detects more defects and improve efficiency of code, in the second inspection predictions are made such as defect prediction, which improves the inspection process.

KEYWORDS: Inspection, Defects, Prediction.

I.INTRODUCTION

Inspection is commonly used for software error detection and correction and it's proven traditionally for defect management. Conducting inspection for verification of program enhances the defect removal capability at the front end reduces error injection. Software has become an important part of business and intelligent system in the world, software industries work at several domains in the product development and interests of the client. Defect free is the important characteristic of the software, during pre and post development. A defect is bug or anomaly that arises due to human errors, incorrectness or incompleteness relative to software requirements during software development. Aim of the defect detection is to provide quality software that reduces the cost and to increase productivity which enables to achieve full customer satisfaction. Challenge faced by every software industry is to implement an effective defect management. Software inspection is the effective and efficient technique and it's the process of removing the defects as early as possible in the software life cycle.

Software inspection does not require running the programs, instead it can be done by human where he reads and checks the program to reveal defects. There are many software inspection methods and tools but most of these mainly focus on implementation related bugs, but in Formal Specification-Based Inspection (FSBI) method is a specification based inspection method, which helps to enhance efficiency and reduce human error during the inspection process.

II.FORMAL SPECIFICATION-BASED INSPECTION

A. Overview

Biffl.S, Halling.M [1] has proposed that inspection is the effective and efficient way for early defect removal in the software lifecycle. In the detection of defect, size of team, experience level of inspectors, development time, complexity of code and number of inspectors have a key effect on it. Further analysis have been done on what level the inspection could be stopped and its impact of stopping it. In the inspection, different reading techniques [3], [4], [7], [9], [11] are used rather than the single technique which is considered to be more effective in the inspection process. Here FBSI inspection technique is used, which focuses on whether functional scenario is implemented by a set of program paths and in turn whether every program path contributes to the functional scenario implementation.

Shaoying Liu, Yuting Chen [10] proposed the FSBI method where the specification based testing technique is used. Since it is difficult to derive all program paths only the representative of the paths are derived from the program by inspection. A scenario defines the functional scenario of a program. It can be defined by formal based specification language [6] (e.g., VDM-SL or Z) at the operational level or by system level. Using Parnas' SCR tabular notation [8] formal specification language is written to define the desired functions for the program. A program path is defined as a sequence of statements and/or conditions in a program.

B. Formal Specification Based Inspection Method

FSBI method comprises of five steps: Deriving functional scenarios from specification, deriving paths from program, linking scenarios to path, analyzing paths against the corresponding scenario, and producing an inspection report.

The deriving scenarios from specification deals about the derivation of functional scenarios from the requirement specification, then the next step is deriving program paths for that functional specification and then next step would be linking those functional specification with the paths of the program.
Expanding the analysis step, there are four levels:
1. Symbol.
2. Atomic condition.
3. Condition.
4. Scenario.
In the symbol level for example, whether the password is implemented correctly in a user registration page is checked, then in the atomic condition level, atomic condition is checked for its correct implementation, and then in the conditional level is the whole condition correctly implemented and then check for whether the whole scenario is implemented correctly using the checklist of questions prepared [5]. These were the analysis levels that are followed in the FSBI method.

III. IMPROVED FSBI METHOD

A. Overview

1. Web application
In this level, check is made whether the variable initialization is minimally kept, is the server is minimally utilized by using the cookie operation in the client side.
2. Globalization
In this level, check is made whether the database is minimally accessed during the operation of program.
3. Error handling
In this level, check is made whether the error message is displayed instead of displaying an error page.
4. Reusability
In this level, check is made whether the code written is reusable that is a code written for a module can be reused in another module in the software development, which reduces the cost and time in development.
5. Maintainability
In this level, check is made whether the code is written in standard level so that it is easy to maintain. If any other developer wants to make changes in a program other than the developer who developed it should be in ease to understand.

IV. AN EXAMPLE

To illustrate the inspection process, an application Deals For Me is taken, where it consists of operations such as Register user, Authenticating user, Display count, Display Advertisement content, View like counts, Share Advertisement, Password recovery, User account and Reward points.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 2, April 2014

Inspection process Formal Specification Based Inspection for Register user and Authenticating user are done as example process.

A. Inspection process for Register user and Authenticating user

1. Deriving Scenarios from Specification

a) Specification of Register user

Process : Register_user(F.name:First name, L.name: Last name, email:Email, pass: Password, R.pass:Repeatpassword)

```
Ext  wr  F.name : String
     wr  L.name: String
     wr  email:String
     wr  pass:String
     wr  R.pass:String

Pre    true

Post let X=Register_user(F.name, L.name, email, pass, R.pass)
        pass = R.pass
                 ∧
        Success message = "Registered successfully"
                 ∨
        pass ≠ R.pass
        warning message = "password mismatch"
End_process
```

b) Deriving Scenarios from Register user specification

```
   X=Register_user(F.name, L.name, email, pass, R.pass)
                 ∧
        pass = R.pass
        Success message = "Registered successfully"            (f1)
                 ∨
        pass = R.pass
        warning message = "password mismatch"                  (f2)
```

c) Specification of Authenticating user

```
Post  let V=Authentication_user (user, pass)in
         Given pass = password  ∧   user = username
                        ∧
        Success message = "Login successful"
                        ∨
        V.found = false
                 ∧
        Given pass ≠ password  ∧  user ≠ username ∧
        warning message = "Invalid username or password"
End_process
```

d) Deriving Scenarios from Authenticating user specification

```
        V=Authentication_user (user, pass)
                 ∧
        V.found  = true
                 ∧                                             (f3)
```

Given pass = password $\wedge$ user = username
$\wedge$
Success message = “Login successful”
$\vee$
V.found = false
$\wedge$
Given pass $\neq$ password $\wedge$ user $\neq$ username (f4)
warning message = “Invalid username or password”

2) Deriving program paths

a) Register user program paths

Program paths are derived from the program statements of Register user and Authenticating user.

Path 0:
1. (c) Password = Repeat password
2. Display success message

Path 1:
1. (c) Password <> Repeat password
2. Display warning message = Password mismatch

b) Authenticating user program path

Path 2:
1. (c) User = User name
2. (c) Pass = Password
3. Display success message

Path 3:
1. (c) User <> User name
2. (c) Pass <> Password
3. Display warning message

3) Linking Scenarios to path

Each of the scenarios and path derived for the Register user and Authenticating user is linked to each other as shown in Table. 1.

TABLE 1. LINKING SCENARIOS TO PATH.

No	Scenarios	Path set	Relation between path set & scenarios
1	f1	{ Path 0}	{Register user}{Path 0} {f1}
2	f2	{Path 1}	{Register user}{Path 1} {f2}
3	f3	{Path 2}	{Authenticating user}{Path 2} {f3}
4	f4	{Path 3}	{Authenticating user}{Path 3} {f4}

4) Analyzing paths

Paths are analysed with the analysis levels and with the corresponding checklist questions as shown in Table.2, the analysis is done for functional scenario f1

5) Producing inspection report

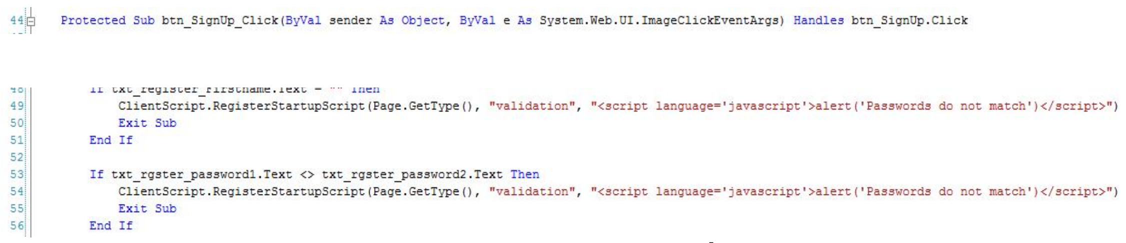
Inspection report has been produced from the previous step and the defect is shown in Fig. 1

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 2, April 2014



```
44  Protected Sub btn_SignUp_Click(ByVal sender As Object, ByVal e As System.Web.UI.ImageClickEventArgs) Handles btn_SignUp.Click


48      If txt_register_firstname.Text = "" Then
49          ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script language='javascript'>alert('Passwords do not match')</script>")
50          Exit Sub
51      End If
52
53      If txt_rgster_password1.Text <> txt_rgster_password2.Text Then
54          ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script language='javascript'>alert('Passwords do not match')</script>")
55          Exit Sub
56      End If
```

Fig 1. Erroneous code

Web application defect: 1.1
File name: login.aspx
Location: loginaspx.vb
Line number: 48 – 56

Suggested correction:
Use the script like below to reduce the sever performance
<script type="text/javascript">
function FBLikeCounter()
{ __doPostBack('btn_Triggerpostback','1');
}

TABLE 2. ANALYZING PATHS

Scenarios	Analysis levels	Questions
f1	1.Symbol	i) Is password correctly implemented? ii) Is Repeat password correctly implemented?
	2.Atomic condition	i) Is X = Register User(F.name, L.name, email, pass, R.pass) Correctly implemented?
	3.Condition	i) X = Register User(F.name, L.name, email, pass, R.pass) Pass = R.pass Correctly implemented?
	4.Scenario	i) Is the whole scenario f1 correctly implemented?
	5. Web application	i) Is Variable initialization minimally kept? ii) Is server minimally used, by using cookie variable, java script?
	6. Globalization	i) Is database access are minimal?
	7. Error handling	i) Is message box is displayed instead when error occurs? instead of displaying error page ?
	8. Reusability	i) Is common code is written so that it can be re-used across programs ?
	9. Maintainability	i) Is code easy to maintain?

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Special Issue 2, April 2014

B. Consolidated report

By improved FSBI method inspection have been done for many modules of the Deals for me application and a consolidated report has been prepared for modules Register user, Authenticating user Display count, Display advertisement content, View like counts, Share advertisement, password recovery, User account, Advertisement account and Reward points a sample of six modules have been shown in Table.3

TABLE 3. CONSOLIDATED REPORT.

	Register user	Auth user	Display count	Display Adv content	View like counts	Share Adv	Password recovery	User account	Adv account	Reward points
Defects captured	2	1	5	4	6	7	4	7	6	7
Development time	2	1	3	2.5	3	3.5	4	3	4.5	5
Complexity	4	3	7	6	8	8	8	9	9	8
Inspection time	0.5	0.30	1.10	1	1.15	1.05	1.15	1	1.25	1.30
No. of inspectors	3	3	3	2	3	2	2	3	2	3
Experience Level of Inspectors	8	8	8	5	7	8	6	6	7	8

V. PREDICTION

A. Overview

Prediction is made for how many defects to occur when the parameters such as development time, complexity, inspection time, number of inspectors and experience level of inspectors are given as input. The input values are taken from the consolidated report. For the prediction linear regression is used which is machine learning approach [2].

B. Linear Regression Method

In linear regression input data parameters are modeled using the hypothesis function, h (x) and unknown values are predicted, here in this we predict the number of defects. In linear regression X refers to the input variables and y is the predicted output variable. As shown in Fig. 2 the training set is the data got from the consolidated report such as development time, complexity, inspection time, number of inspectors and experience level of inspectors is taken as input, X and output of each is y which is number of defects, to this training set linear regression is applied. In the linear regression two methods are there, gradient descent and feature normalization, here in this feature normalization is used.

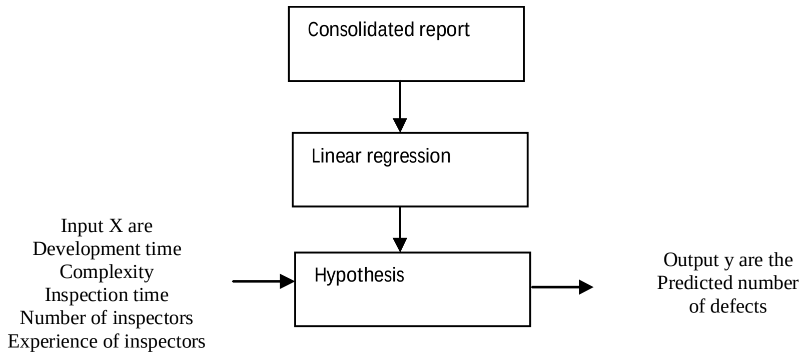


Fig 2. Flow chart of Prediction

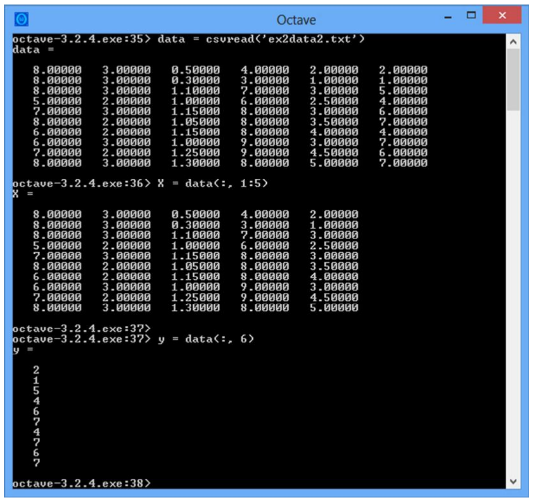
In feature normalisation input parameters are taken as vector X and output is taken as vector y and applying it in the equation.1 parameters θ_0, θ_1, θ_2 ….θ_n are found, where n is the number of features that is the number of input parameters, here five parameters are used so the n=5.
In this scheme, each node with message searches for possible path nodes to copy its message. Hence, possible path nodes of a node are considered. Using NSS, each node having message selects its path nodes to provide a sufficient level of end-to-end latency while examining its transmission effort. Here, it derives the CSS measure to permit CR-Networks nodes to decide which licensed channels should be used. The aim of CSS is to maximize spectrum utilization with minimum interference to primary system. Assume that there are M licensed channels with different bandwidth values and y denotes the bandwidth of channel c. Each CR-Networks node is also assumed to periodically sense a set of M licensed channels. Mi denotes the set including Ids of licensed channels that are periodically sensed by node i. suppose that channel c is periodically sensed by node i in each slot and channel c is idle during the time interval x called channel idle duration. Here, it use the product of channel bandwidth y and the channel idle duration x, tc = xy, as a metric to examine the channel idleness. Furthermore, failures in the sensing of primary users are assumed to cause the collisions among the transmissions of primary users and CR-Networks nodes.

$$\theta = (X^T X)^{-1} X^T y \qquad (1)$$

The values of θ are applied to hypothesis function $h_\theta(x)$ as in equation. 2 for prediction.

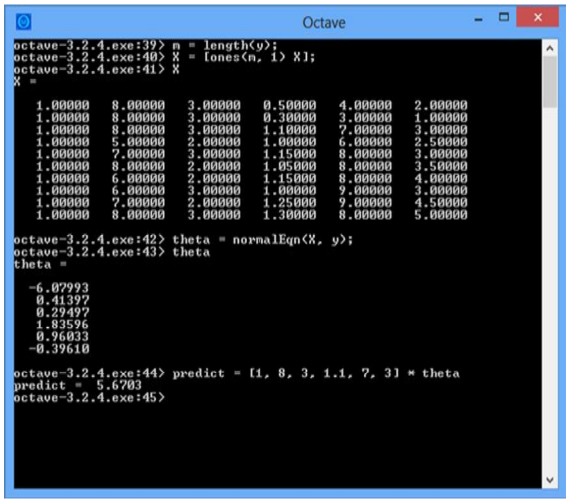
$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5 \qquad (2)$$

C. Implimentation
Prediction using the linear regression method feature normalization, is implemented using the Octave tool its like the mat lab tool.
1. Loading of data
Input parameters are loaded form the text file, input parameters are taken as X in matrix vector format and output parameters are taken as vector y as shown in Fig.3
2. Prediction of defects
To the vector X, 1 is added as for X0 always the value is 1, and the theta values are found using the equation.1 mentioned above and using the theta values prediction is made using the hypothesis function shown in equation. 2. Here the values taken for the parameters Experience level of inspectors, number of inspectors, inspection time, complexity and development time are 8, 3, 1.1, 7 and 3 respectively and it multiplied with the vector theta to find the predicted output, and the number of defects got as output is 5.6703 approximately.



```
octave-3.2.4.exe:35> data = csvread('ex2data2.txt')
data =

   8.00000   3.00000   0.50000   4.00000   2.00000   2.00000
   8.00000   3.00000   0.30000   3.00000   1.00000   1.00000
   8.00000   3.00000   1.10000   7.00000   3.00000   5.00000
   5.00000   2.00000   1.00000   6.00000   2.50000   4.00000
   7.00000   3.00000   1.15000   8.00000   3.00000   6.00000
   8.00000   2.00000   1.05000   8.00000   3.50000   7.00000
   6.00000   2.00000   1.15000   8.00000   4.00000   4.00000
   6.00000   3.00000   1.00000   9.00000   3.00000   7.00000
   7.00000   2.00000   1.25000   9.00000   4.50000   6.00000
   8.00000   3.00000   1.30000   8.00000   5.00000   7.00000

octave-3.2.4.exe:36> X = data(:, 1:5)
X =

   8.00000   3.00000   0.50000   4.00000   2.00000
   8.00000   3.00000   0.30000   3.00000   1.00000
   8.00000   3.00000   1.10000   7.00000   3.00000
   5.00000   2.00000   1.00000   6.00000   2.50000
   7.00000   3.00000   1.15000   8.00000   3.00000
   8.00000   2.00000   1.05000   8.00000   3.50000
   6.00000   2.00000   1.15000   8.00000   4.00000
   6.00000   3.00000   1.00000   9.00000   3.00000
   7.00000   2.00000   1.25000   9.00000   4.50000
   8.00000   3.00000   1.30000   8.00000   5.00000

octave-3.2.4.exe:37>
octave-3.2.4.exe:37> y = data(:, 6)
y =

   2
   1
   5
   4
   6
   7
   4
   7
   6
   7

octave-3.2.4.exe:38>
```

Fig.3 Loading of data



```
octave-3.2.4.exe:39> m = length(y);
octave-3.2.4.exe:40> X = [ones(m, 1) X];
octave-3.2.4.exe:41> X
X =

   1.00000   8.00000   3.00000   0.50000   4.00000   2.00000
   1.00000   8.00000   3.00000   0.30000   3.00000   1.00000
   1.00000   8.00000   3.00000   1.10000   7.00000   3.00000
   1.00000   5.00000   2.00000   1.00000   6.00000   2.50000
   1.00000   7.00000   3.00000   1.15000   8.00000   3.00000
   1.00000   8.00000   2.00000   1.05000   8.00000   3.50000
   1.00000   6.00000   2.00000   1.15000   8.00000   4.00000
   1.00000   6.00000   3.00000   1.00000   9.00000   3.00000
   1.00000   7.00000   2.00000   1.25000   9.00000   4.50000
   1.00000   8.00000   3.00000   1.30000   8.00000   5.00000

octave-3.2.4.exe:42> theta = normalEqn(X, y);
octave-3.2.4.exe:43> theta
theta =

  -6.07993
   0.41397
   0.29497
   1.83596
   0.96033
  -0.39610

octave-3.2.4.exe:44> predict = [1, 8, 3, 1.1, 7, 3] * theta
predict =  5.6703
octave-3.2.4.exe:45>
```

Fig.4 Prediction of number of defects

VI. CONCLUTION AND FUTURE WORK

In the existing FBSI method analysis have been done in the implementation view, that every functional scenarios are implemented in the program, but in the proposed improved FSBI method, view has been done in the developers view by surveying the working people in the software industry, their suggestions and remarks have been briefed and added as five more levels in the analysis and the efficiency of the code has been improved. For a given number of inspectors, experience level, complexity, inspection time and development time, prediction on how many defects could possibly occur can be predicted. In this paper deals for me application code is inspected in future more application code can be inspected and reliability can be tested.

REFERENCES

[1] Biffl. S and Halling. M, "Investigating the defect detection effectiveness and cost benefit of nominal inspection teams," Proceedings of IEEE Trans. Software Eng 2003.,vol. 29 , Issue.5, pp. 385-397.

[2] Challagulla V.U.B, Bastani F.B, I-Ling Yen, Paul R.A., "Empirical assessment of machine learning based software defect prediction techniques," IEEE International Workshop on Digital Object Identifier 2005, pp.263-270.

[3] Fagan.M.E, "Design and Code Inspections to Reduce Errors in Program Development," IBM Systems J., vol. 15, no. 3, pp. 182-211.

[4] Grechenig. T, Halling. M, Biffl. S and Kohle. M, "Using Reading Techniques to Focus Inspection performance," IEEE International Conf. 2001, pp.248-257.

[5] Hatton and Les, "Testing the Value of Checklists in Code Inspections," IEEE software 2008, vol.25, issue.4, pp.82-88.

[6] Laitenberger. O, "A Survey of Software Inspection Technologies," Handbook of Software Eng. and Knowledge Eng., 2002, pp. 517-556.

[7] Porter .A.A, Siy. H.P, and Votta. L.G , "A Review of Software Inspections," Advances in Computers 1996, vol. 42, pp. 39-76.

[8] Parnas D.L, and Weiss.D.M, "Active Design Reviews: Principles and Practices," J. Systems and Software 2005, vol. 7, no. 4, pp. 259-265.

[9] Soffa. M.L, Gupta. R, "A FrameWork for partial data flow analysis," Department of Computer Science, IEEE International Conf., 1984, pp.4-13.

[10] Shaoying Liu and Yuting Chen , "Formal Specification-Based Inspection for Verification of Programs, " IEEE Trans. Software Eng., 2012, vol. 38 , issue.5, pp.1100-1122.

[11] Saito .S, Takeuchi. M, Hiraoka. M, Kitani. T and Aoyama. M, "Requirements clinic: Third party inspection methodology and practice for improving the quality of software requirements specifications," IEEE Conf., publication, 2013, pp.290-295.