REFERENCES

[1]D. Starr, "Defining Code Quality", *Blog.smartbear.com*, 2015. [Online]. Available: http://blog.smartbear.com/code-review/defining-code-quality/.

[2] "Code Quality - How It Affects Your Bottom Line", Castsoftware.com. [Online]. Available: http://www.castsoftware.com/glossary/code-quality. [Accessed: 04- Jan- 2017].

[3] D. Athanasiou and J.Visser, "Test Code Quality and Its Relation to Issue Handling Performance".

[4] R. Baggen, J. Correia, K. Schill and J.Visser, "Standardized code quality benchmarking for improving software maintainability"

[5] B. Luijten and J. Visser, "Faster defect resolution with highertechnical quality of software," in 4th International Workshop onSoftware Quality and Maintainability (SQM 2010), 2010.

[6] J. McGonigal, Reality is broken: Why games make us better and how they can change the world. Penguin, 2011.

[7] Philipp Lombriserand Roald van der Valk , "Improving the Quality of the Software Development Lifecycle with Gamification"

[8] G. Zichermann and C. Cunningham, Gamification by design: Implementinggame mechanics in web and mobile apps. " O'Reilly Media, Inc.," 2011.

[9] O. Pedreira, F. García, N. Brisaboa, and M. Piattini, "Gamification in software engineering – A systematic mapping," Information and Software Technology, vol. 57, pp. 157–168, Jan. 2015.

[10] J. Fernandes, D. Duarte, C. Ribeiro, C. Farinha, J. M. Pereira, and M. M. daSilva, "iThink: A Game-Based Approach Towards Improving Collaboration and Participation in Requirement Elicitation," *ProcediaComput. Sci.*, vol. 15,pp. 66–77, Jan. 2012.

[11] E. De Bono, *Six thinking hats*. Taylor & Francis, 1999

[12] N. Tillmann, J. Bishop, R. N. Horspool, D. Perelman, and T. Xie, "Code hunt: searching for secret code for fun," in *Proceedings of the 7th International Workshop on Search-Based Software Testing*, 2014, pp. 23–26.

[13] K. Januszewski, "Announcing Visual Studio Achievements Beta," 2012. [Online]. Available: http://channel9.msdn.com/Blogs/C9team/AnnouncingVisual-Studio-Achievements. [Accessed: 15-Sep-2014].

[14] Steve Bygren, Greg Carrier, Tom Maher, Patrick Maurer, David Smiley, Rick Spiewak, Christine Sweed, "Applying the fundamentals of quality to software acquisition", Systems Conference (SysCon) 2012 IEEE International, pp. 1-6, 2012.

[15] A. Aurum, H. Petersson, and C. Wohlin, “State-of-the-art: software inspections after 25 years,” SoftwareTesting, Verification and Reliability, vol. 12, no. 3, pp. 133–154, 2002.

[16]M. Harman, "Why Source Code Analysis and Manipulation Will Always Be Important".

[17] “Minimizing code defects to improve software quality and lower development costs” [online]. Available: ftp://ftp.software.ibm.com/software/rational/info/do-more/RAW14109USEN.pdf

[18] M. Gegick and L. Williams, “Towards the use of automated static analysis alerts for early identification of vulnerability- and attack-prone components,” in Proc. ICIMP, 2007, pp. 18–23.

[19] V. Barstad, V. Shulgin, M. Goodwin, and T. Gjøsæter, “Towards a Collaborative Code Review Plugin,” in Proceedings of the 2013 NIK conference, 2013, pp. 37–40.

[20] D. Hovemeyer and W. Pugh. Finding Bugs is Easy. In Onward!, OOPSLA’04, Vancouver, BC, October 2004.

[21] V. Barstad, M. Goodwin, and T. Gjøsæter, "Predicting Source Code Quality with Static Analysis and Machine Learning,".[Online]. Available: http://ojs.bibsys.no/index.php/NIK/article/download/26/22. Accessed: Aug. 15, 2016.

[22] N. Meghanathan, "Source Code Analysis to Remove Security Vulnerabilities in Java Socket Programs: A Case Study", International Journal of Network Security & Its Applications, vol. 5, no. 1, pp. 1-16, 2013.

[23] G. Murphy, M. Kersten, and L. Findlater, “How are Java software developersusing the Eclipse IDE?”Software, IEEE, vol. 23, pp. 76–83, 2006.

[24] W. Snipes, B. Robinson, and E. Murphy-Hill, "Code Hot Spot: A Tool for Extraction and Analysis of Code Change History".

[25] E. Murphy-Hill, B. Johnson, and Y. Song, "Why Don’t Software Developers Use Static Analysis Tools to Find Bugs?,". [Online]. Available: http://people.engr.ncsu.edu/ermurph3/papers/icse13b.pdf.

[26] J. Foster and M. Hicks, "Improving Software Quality with Static Analysis ∗".

[27] M. Gegick and L. Williams, “Towards the use of automated static analysis alerts for early identification of vulnerability- and attack-prone components,” in Proc. ICIMP, 2007, pp. 18–23.

[28] T. Menzies, J. Greenwald and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors", IEEE Transactions on Software Engineering, vol. 33, no. 1, pp. 2-13, 2007.

[29] "FindBugs Bug Descriptions", Findbugs.sourceforge.net, 2018. [Online]. Available: http://findbugs.sourceforge.net/bugDescriptions.html. [Accessed: 24- Aug- 2017].

[30] A guide to research evaluation ...". [Online]. Available: http://www.rand.org/pubs/monographs/MG1217.html. [Accessed: 28- Nov- 2017].

[31] R. Powell, "Evaluation Research: An Overview", Library Trends, vol. 55, no. 1, pp. 102-120, 2006.

[32] C. Catal, U. Sevim, and B. Diri, “Practical development of an Eclipse-based software fault prediction tool using Naive Bayes algorithm,” Expert Systems with Applications, 2011.

[33] "Findbugs - Static Code Analysis of Java", *Methodsandtools.com*, 2018. [Online]. Available: http://www.methodsandtools.com/tools/findbugs.php. [Accessed: 28- Nov- 2017].

[34] J. Zheng, L. Williams, N. Nagappan, W. Snipes, J. Hudepohl and M. Vouk, "On the value of static analysis for fault detection in software", *IEEE Transactions on Software Engineering*, vol. 32, no. 4, pp. 240-253, 2006.

[35] I. Stamelos, L. Angelis, A. Oikonomou and G. Bleris, "Code quality analysis in open source software development", *Information Systems Journal*, vol. 12, no. 1, pp. 43-60, 2002.

[36] P. Emanuelsson and U. Nilsson, "A Comparative Study of Industrial Static Analysis Tools", Electronic Notes in Theoretical Computer Science, vol. 217, pp. 5-21, 2008.

APPENDIX A

Data collected for two months using log files. The data are presented in weekly format.

Week 1	Developer 1	Developer 2	Developer 3	Developer 4	Developer 5	Developer 6
Bad Practice	3	4	2	2	1	1
Correctness	3	2	1	2	0	0
Malicious code vulnerability	1	1	0	0	0	0
Multithreaded correctness	0	0	0	0	0	1
Performance	2	3	1	1	4	0
Security	1	1	1	0	0	3
Dodgy code	2	1	1	4	1	1

Week 2	Developer 1	Developer 2	Developer 3	Developer 4	Developer 5	Developer 6
Bad Practice	2	2	1	1	1	1
Correctness	2	2	2	0	1	0
Malicious code vulnerability	1	1	1	0	0	1
Multithreaded correctness	2	1	1	0	0	0
Performance	3	1	1	2	3	1
Security	0	1	0	0	0	2
Dodgy code	1	1	1	3	2	1

Week 3	Developer 1	Developer 2	Developer 3	Developer 4	Developer 5	Developer 6
Bad Practice	3	2	2	2	2	1
Correctness	3	2	0	1	0	2
Malicious code vulnerability	0	0	0	1	0	0
Multithreaded correctness	1	1	0	0	1	0
Performance	2	3	2	1	3	1
Security	1	0	1	0	0	1
Dodgy code	3	2	2	3	1	2

Week 4	Developer 1	Developer 2	Developer 3	Developer 4	Developer 5	Developer 6
Bad Practice	2	2	1	1	1	1
Correctness	2	1	2	2	1	1
Malicious code vulnerability	0	0	0	1	0	1
Multithreaded correctness	1	1	0	0	0	0
Performance	1	4	2	2	4	2
Security	1	1	1	2	1	2
Dodgy code	2	1	1	2	1	1

Week 5	Developer 1	Developer 2	Developer 3	Developer 4	Developer 5	Developer 6
Bad Practice	3	3	3	3	1	2
Correctness	2	2	3	3	3	1
Malicious code vulnerability	1	1	0	0	1	0
Multithreaded correctness	0	0	0	0	0	1
Performance	2	2	2	1	2	1
Security	1	1	1	1	1	2
Dodgy code	2	3	2	3	2	1

Week 6	Developer 1	Developer 2	Developer 3	Developer 4	Developer 5	Developer 6
Bad Practice	2	3	2	1	2	1
Correctness	1	1	1	2	1	2
Malicious code vulnerability	1	1	0	0	0	1
Multithreaded correctness	1	0	0	0	1	0
Performance	2	1	1	0	2	1
Security	1	0	0	0	1	1
Dodgy code	2	3	1	2	2	1

Week 7	Developer 1	Developer 2	Developer 3	Developer 4	Developer 5	Developer 6
Bad Practice	2	3	3	2	1	1
Correctness	2	1	2	2	1	1
Malicious code vulnerability	1	0	0	0	1	0
Multithreaded correctness	2	2	1	1	1	0
Performance	2	2	1	2	2	
Security	1	0	0	0	0	3
Dodgy code	2	2	2	3	1	2

Week 8	Developer 1	Developer 2	Developer 3	Developer 4	Developer 5	Developer 6
Bad Practice	3	2	1	2	1	1
Correctness	2	2	3	1	1	1
Malicious code vulnerability	1	0	0	0	0	1
Multithreaded correctness	1	0	1	0	0	0
Performance	2	1	0	1	3	2
Security	1	0	0	0	0	1
Dodgy code	2	3	2	1	2	1

APPENDIX B

Sample Java implementation of a basic parser for FindBugs log files.

```
import java.io.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class BasicFindBugsLogParser {

    public static void main(String[] args) {
        ArrayList<Bug> bugs = new ArrayList<>();
        Map<String, Integer> typeToNumberofBugs = new HashMap<>();

        File fXmlFile =
                new
File("path/to/log_file/Foresight.fbwarnings.xml");
        DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder;

        FileWriter fw;
        BufferedWriter bw;
        PrintWriter out = null;
        try {
            dBuilder = dbFactory.newDocumentBuilder();
            fw = new FileWriter("path/to/data", true);
            bw = new BufferedWriter(fw);
            out = new PrintWriter(bw);
            Document doc = dBuilder.parse(fXmlFile);
            doc.getDocumentElement().normalize();

            NodeList nList =
doc.getElementsByTagName("BugInstance");

            for (int index= 0; index< nList.getLength();index++) {
                Node nNode = nList.item(index);

                // Traverse and collect the relevant data
                if (nNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element eElement = (Element) nNode;
                    String type = eElement.getAttribute("type");
                    String rank = eElement.getAttribute("rank");
```

```
                    String priority =
eElement.getAttribute("priority");
                    String category =
eElement.getAttribute("category");
                    out.println(type + "\t" + rank + "\t" + priority
+ "\t" + category);
                    String key = priority;
                    if (typeToNumberofBugs.containsKey(key)) {
                        typeToNumberofBugs.put(key,
typeToNumberofBugs.get(key) + 1);
                    } else {
                        typeToNumberofBugs.put(key, 1);
                    }
                    bugs.add(new Bug(rank, type, priority));
                }
            }
            // Just to log the collected data
            for (Entry<String, Integer> entry :
typeToNumberofBugs.entrySet()) {
                System.out.println(entry.getKey() + "-" +
entry.getValue());
            }

        } catch (ParserConfigurationException | IOException |
SAXException e) {
            throw new Exception("Parsing failed", e);
        } finally {
            if (out != null) {
                out.close();
            }
            bw.close();
            fw.close();
        }
    }

    static class Bug {
        String rank;
        String type;
        String complexity;

        Bug(String rank, String type, String complexity) {
            this.rank = rank;
            this.type = type;
            this.complexity = complexity;
        }
    }
}
```