

A Retrospective Study of Taxonomy based Testing using Empirical Data from a Medical Device Software Company

Hamsini Ketheswarasarma Rajaram^a, John Loane^b, Silvana Togneri MacMahon^c
and Fergal Mc Caffery^d

Regulated Software Research Centre, Dundalk Institute of Technology, Co Louth, Dundalk, Ireland

Keywords: Defect Taxonomy, Defect Classification, Validation, Taxonomy based Testing, Empirical Data, Root Cause Analysis, Defect Minimisation, Medical Device Software.

Abstract: Software defects in medical devices have caused serious injuries and deaths to patients. Medical devices are facing an increasing number of the U.S Food and Drug Administration (FDA) recalls due to poor quality software. Research studies suggest that defect taxonomies are powerful tools to prevent and control defects. Defect taxonomies have been used to improve software quality in the safety critical, business and telecommunications domains. Defect taxonomies can be used in testing and are more efficient at finding defects at the earliest possible stage of software development. This paper discusses taxonomy based testing in medical device software (MDS) development. SW91 is a new defect taxonomy for health software developed by the Association for the Advancement of Medical Instrumentation. This paper details a retrospective study conducted to investigate taxonomy based testing by mapping empirical data from a MDS company in Ireland to SW91 defects. It explains the process and shows the benefits of taxonomy based testing, which include defect minimisation and root cause analysis. It provides recommendations which can be followed when using taxonomy based testing. It also details interviews conducted with the CEO, developers and the quality assurance engineer from Company A. Finally, it briefly details how taxonomy based testing will be implemented at a MDS company by applying a framework which was developed from this research.


1 INTRODUCTION


Medical devices increasingly rely on software to provide additional functionality (L. K. Simone, 2013). Since medical device functionality directly impacts patient safety, it is important to ensure high quality software in medical devices. Software quality is measured by the number of defects found in software (Ioan Mihnea Iaco & Radu, 2008). In order to find software defects and to ensure software quality, software quality assurance processes have been integrated into software development. Software quality assurance processes aim to minimise software defects and show that software meets requirements. However, organisations face challenges in improving software quality such as the inability to specify the software requirements properly, the lack of adequate software quality assurance processes and the lack of


relevant metrics to track software quality (FDA, 2011; P.S.Cosgriff, 1990).


Research studies suggest that a defect taxonomy is the best way to prevent and control defects (Chillarege et al., 1992; Felderer & Beer, 2013a, 2013b). Defect taxonomies have been used successfully in various ways during the testing phase of software development, such as in system testing, measuring testing efficiency and classifying defects (Black, 2008; Felderer & Beer, 2013b; Li, Li, & Sun, 2010; Madachy & Boehm, 2008). This research has proposed a testing approach called taxonomy based testing to improve MDS quality (Rajaram(&), Loane, MacMahon, & Fergal, 2019; Rajaram, Loane, MacMahon, & Mc Caffery, 2018).

Taxonomy based testing is a defect based testing technique. In taxonomy based testing, the requirements will be mapped into potential defects

^a  <https://orcid.org/0000-0002-3294-3906>

^b  <https://orcid.org/0000-0002-9285-5019>

^c  <https://orcid.org/0000-0003-0179-2436>

^d  <https://orcid.org/0000-0002-0839-8362>

from a defect taxonomy and the test cases will be written based on the requirements and the mapped defects. The test cases will be executed to verify whether the software complies with the relevant requirements and does not contain the mapped defects from the defect taxonomy. Taxonomy based testing uses the Classification of Defects in Health Software - SW91 as its defect taxonomy.

SW91, Classification of Defects in Health Software is a defect classification scheme and a standard for health software which has been developed by the Association for the Advancement of Medical Instrumentation (AAMI) in collaboration with the U.S FDA (Association for the Advancement of Medical Instrumentation, 2016). SW91 development work started in 2014 and aims to provide a common language to classify defects and improve software quality in health software including MDS (L. Simone & Rubery, 2014). SW91 was published on 22 of October 2018 as a standard (AAMI, 2019).

SW91 includes defects from the planning to the maintenance phases of a system. It contains multi-level defects such as parent level and child level. Every parent level defect includes several child defects. Each defect has a defect code with a unique number. This numbering system follows a hierarchical structure. Every parent level defect is represented by a number. Every child level defect is represented by appending a period and a number to the parent level category. Each defect has annotations and some of the defects have examples. For example, Failure to Save/Restore (5.3.2.5.1) is one of the child level defects from Implementation Defects (5). It has the following description: “Context was not saved or restored when it should have been.”

The version of SW91 which was open for public comment in September 2016, was used in this study. It is not much different from the final version.

This paper details how a retrospective study was conducted on data received from a completed project from a MDS company in Ireland. This study investigated the applicability and benefits of taxonomy based testing in the MDS industry.

The next section explains the process followed in this retrospective study. Section 3 explains the results obtained from this study. Section 4 outlines the benefits of this study and recommendations provided to Company A. Section 5 outlines the interview results obtained from Company A. Section 6 presents future work and Section 7 presents the summary and conclusions.

2 PROCESSES FOLLOWED IN THE RETROSPECTIVE STUDY

Company A develops MDS applications. This study uses defects, software design specification (SDS), user requirements, risks and test protocols from a completed project at Company A.

For each type of document, potential SW91 defects were searched and mapped. Out of the documents received from Company A, the SDS and requirements were used to do the direct mapping into SW91 defects. Test protocols were linked with requirements and the SDS. Therefore, SW91 defects used in the mapping of the SDS and the requirements were used in the test protocol mapping. The remainder of this section explains each of the mappings.

2.1 Mapping A – Defects from Company A and SW91 Defects

Defects from the Company A contain defect symptoms such as algorithm not working correctly and warning alerts not appearing. Repetition in the defect symptoms was observed and removed. Eight distinct defect symptoms were identified after removing the duplicates. These symptoms were due to some defects in company A’s application. The defects for each symptom were manually searched from SW91 and mapped. Eight distinct defect symptoms were mapped into twenty distinct SW91 defects.

Table 1 details a sample mapping of a defect symptom and its SW91 defects.

Table 1: Defect symptom and SW91 defects.

Defect symptom	SW91 defects
Warning alerts not appearing	Invalid Path (5.2.1.2.6)
	Parameter Type (4.2.2.2)
	Wrong Algorithm Selected (4.2.4)
	Parameter Structure (4.2.2.3)
	Use Before Check (5.3.2.6)
	Bad Translation (5.1)
	Invalid Path (5.2.1.2.6)

The next section explains how the SDSs were used in this study.

2.2 Mapping B – SDS and SW91 Defects

The SDS document has very detailed control flow diagrams for the application. It includes forty control

flow diagrams. Out of forty, three diagrams were randomly selected to map into SW91 defects. Not all of the diagrams were mapped into SW91 defects due to the similarity in their structure.

The defects for each development stage of the control flow diagrams were manually searched for in SW91 and mapped.

Table 2 shows a sample of the mapping conducted between the development stage from a control flow diagram and SW91 defects.

Table 2: Corrected calcium SW91 Defects.

Development stages of control flow diagram	SW91 defects
Input variables to text field	Use before check (5.3.2.6)
	Inappropriate cast or type conversion (5.3.2.1.4)
Has the user filled the inputs	Control state (5.2.1.4)
	Inappropriate cast or type conversion (5.3.2.1.4)

This section explained how the SDS was mapped into SW91 defects. The next section explains how requirements were mapped into SW91 defects.

2.3 Mapping C - Requirements and SW91 Defects

The company shared forty-one requirements, including both functional and non-functional requirements. Every requirement has associated risks which have been assigned a priority. Out of forty-one requirements, thirty-nine were mapped into forty-three distinct SW91 defects. For every requirement in this mapping, the potential defects which could occur in the development of the requirement were manually searched for in SW91 and they were mapped. Table 3 shows a sample mapping of the requirement into SW91 defects. Two non-functional requirements were very general and it was not possible to map them into any SW91 defects. For example, the application must be endorsed by users, government bodies and cancer research bodies is one of the non-functional requirements and it was not mapped into any SW91 defects.

Table 3: Requirements and SW91 defects SW91.

Requirement	SW91 defects
R1. The application must allow the user to calculate a number of medical formulas	Bad Translation (5.1)
	Expression Evaluation (5.2.2.1)
	Operator (5.2.2.1.1)
	Grouping (5.2.2.1.2)
	Scalar Type (5.3.1.1)
	Incorrect Timeout (5.2.2.5)

This section explained how requirements were mapped into SW91 defects. The next section explains how Company A test protocols were mapped into SW91 defects. The test protocols are the last type of document used in this study.

2.4 Test Protocols and SW91 Defects

Each test protocol contains test cases and procedures to be followed in the testing phase. Test protocols are associated with their respective SDS and requirement. For example, if the requirement ID is R1 and the SDS ID is S1, then the respective test protocol T1 will include the requirement R1 and SDS S1. Figure 1 details the links between test protocols, requirements and the SDS.

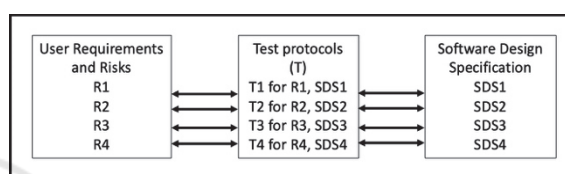


Figure 1: Test protocols and links.

When starting the mapping of a test protocol, the following defects for the test protocol are already there:

- SW91 defects used in the mapping of the control flow diagram from the SDS, Mapping B.
- SW91 defects used in the mapping of functional requirements, from Mapping C.

For each of the test protocols, the above mappings were merged and repeated defects were removed. This merging and removal of duplicate defects provided distinct potential defects for each test protocol.

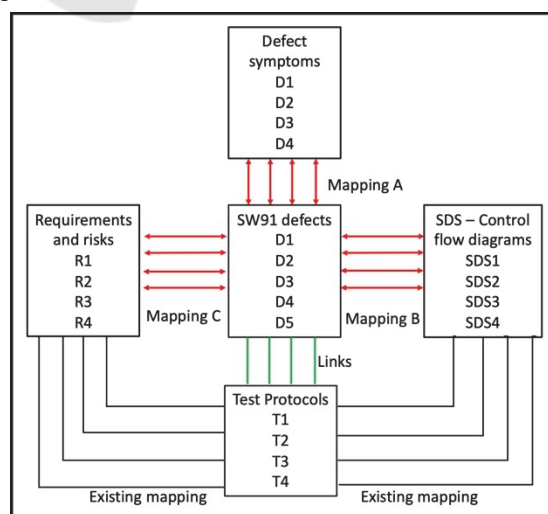


Figure 2: Company A data and SW91 defects.

At this stage of the paper, we have seen how the defect symptoms, control flow diagrams, requirements with risks and test protocols were used to do the mappings. Figure 2 shows the mappings and it summarises all four types of data and their mappings into SW91 defects.

The next section details the results observed from this study.

3 RESULTS

First, eight distinct defect symptoms were mapped into twenty distinct SW91 defects. This mapping is labelled Mapping A.

Secondly, all development stages of the selected three control flow diagrams were mapped into nineteen SW91 defects. This mapping is labelled Mapping B.

Thirdly, the thirty-nine functional requirements and their risks were mapped into forty-three distinct SW91 defects and it is labelled Mapping C.

From the three mappings (A, B and C), the following three sets of common SW91 defects were observed:

1. Common SW91 defects from Mapping A and Mapping B.
2. Common SW91 defects from Mapping A and Mapping C.
3. Common SW91 defects from Mapping A, Mapping B and Mapping C.

The remainder of this section explains the above three results.

3.1 Common SW91 Defects from Mapping A and Mapping B

Ten distinct SW91 defects were observed from Mapping A (Defect symptoms and SW91 Defects) and Mapping B (SDS and SW91 defects). This mapping is represented in Figure 3. The coloured triangle in Figure 3 represents common SW91 defects which are listed in column 1 of Table 4.

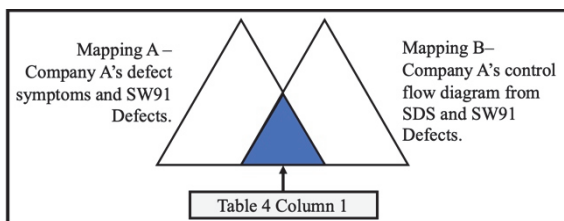


Figure 3: Overlapping SW91 defects from mappings A and B.

3.2 Common SW91 Defects from Mapping A and Mapping C

Eleven distinct SW91 defects were observed from both Mapping A (Defect symptoms and SW91 Defects) and Mapping C (Requirements and SW91 defects). This overlap is detailed in Figure 4. The coloured triangle from Figure 4 represents common SW91 defects which are listed in column 2 of Table 4.

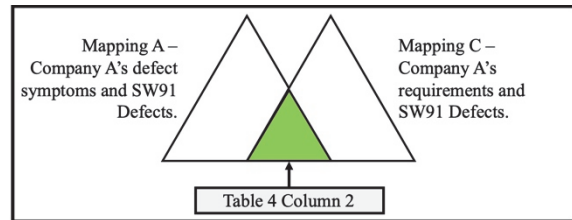


Figure 4: Overlapping SW91 defects from mappings A and C.

These defects can be highlighted during the requirements gathering phase of the application development. The requirements gathering phase failed to consider the defects listed in Table 4, column 2 and they reoccurred at the testing phase.

If Company A has the mappings at the requirement gathering phase, all the requirements will have been mapped into their potential SW91 defects from different phases. The software architect could consider the defects related to the design phase and it will help to avoid the mapped defects for each requirement. Developers could consider the defects related to the implementation phase and it will help to avoid the mapped defects for each requirement.

This mapping will help to avoid defects at earlier phases. The quality assurance engineer can also get ideas on possible defects for each requirement by considering the mapped defects and it will help to generate goal oriented test cases.

3.3 Common SW91 Defects from Mappings A, B and C

Six distinct SW91 defects were observed from all three mappings (Mapping A, Mapping B, Mapping C). These mappings are represented in Figure 5. The coloured triangle in Figure 5 represents the common SW91 defects which are listed in column 3 of Table 4.

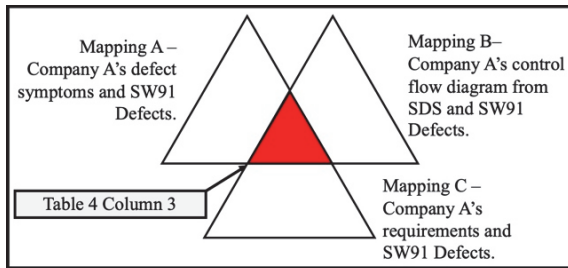


Figure 5: Overlapping SW91 defects from mappings A, B and C.

These defects can be highlighted either at the requirements gathering phase or the software design phase of the application development. Both the requirements gathering and design phases failed to find the defects listed in Table 4, column 3 and they reoccurred at the testing phase. If this mapping existed at Company A, it would provide a common language for all stakeholders to discuss the potential defects for each requirement.

Table 4: All Overlapping SW91 defects and Company A Data.

Mappings A and B	Mappings A and C	Mappings A, B, and C
Bad Translation (5.1)	Bad Translation (5.1)	Bad Translation (5.1)
Inappropriate Cast or Type Conversion (5.3.2.1.4)	Corrupted Database Upgrade (7.8)	Inconsistent Requirement (2.3.5)
Inconsistent Requirement (2.3.5)	Inconsistent Requirement (2.3.5)	Internal Interfaces (4.2)
Internal Interfaces (4.2)	Interface Parameter, Invocation (4.2.2)	Operator (5.2.2.1.1)
Invalid Path (5.2.1.2.6)	Internal Interfaces (4.2)	Use Before Check (5.3.2.6)
Operator (5.2.2.1.1)	Operator (5.2.2.1.1)	Wrong Algorithm Selected (4.2.4)
Parameter Structure (4.2.2.3)	Wrong Algorithm Selected (4.2.4)	
Parameter Type (4.2.2.2)	Size (5.3.1.2)	
Use Before Check (5.3.2.6)	Transactions (3.4.4)	
Wrong Algorithm Selected (4.2.4)	Use Before Check (5.3.2.6)	
	Scalability (3.3)	

The next section discusses the benefits of this study and recommendations provided to Company A from this study.

4 BENEFITS AND DISCUSSION

This section details the following benefits from this study:

1. Defect reporting at the testing phase
2. Defect minimization
3. Risk minimization
4. Root cause analysis

4.1 Defect Reporting at the Testing Phase

Company A reports defect symptoms from their testing. These defect symptoms must be reported by a quality assurance engineer at Company A. The developers need to work to fix the defect symptom. According to the current format used for defect reporting, the developers do not know about the actual defects for reported defect symptoms. When the developers attempt to fix the reported defect symptom, it will be hard for them to fix due to the poorly defined defect symptom. Therefore, warning alerts not appearing could appear again in the second round of testing due to some other defect of which the developers or the quality assurance engineers were not aware. This situation can be addressed at Company A by mapping the defect warning alerts not appearing into the following SW91 defects:

- Invalid Path (5.2.1.2.6)
- Parameter Type (4.2.2.2)
- Wrong Algorithm Selected (4.2.4)
- Parameter Structure (4.2.2.3)
- Use Before Check (5.3.2.6)
- Bad Translation (5.1)

The developer can be informed of the possible SW91 defects which could cause the defect symptom warning alerts not to appear. Developers can work to fix the possible SW91 defects when fixing the reported defect symptoms. Developers can fix the reported defect symptom by addressing the mapped SW91 defects for the failed test. This type of mapping will minimise the reoccurrence of defect symptoms by checking all possible mapped SW91 defects. This type of mapping would reduce the development time and help to anticipate possible defects. If this mapping is used at company A, there will be a common language for quality assurance engineers and developers to communicate the test failures.

4.2 Defect Minimisation

Since Company A has very detailed control flow diagrams, mapping each development stage of the control flow diagram into SW91 defects could minimise the occurrence of defects at the implementation phase. When Company A has the potential defects for every development stage of the control flow diagrams and requirements, the software architect and developers can work to avoid those mapped defects during the design and implementation. Quality assurance engineers can execute test protocols to find those mapped defects. Again, this will minimise the time to find defects in the application and will help to prevent the defects at the earliest possible phase of software development.

4.3 Risk Minimisation

Each requirement has associated risks. Using the mappings explained in this study, each requirement has been mapped into its potential SW91 defects. If each requirement can be implemented with the minimum number of defects, then it is possible to minimise the associated risks as well. For example, as shown in Section 2.3, when the requirement, the application must allow the user to calculate a number of medical formulas is being developed, the risk of this requirement will be mitigated by avoiding the mapped defects. When Company A minimises the occurrence of the defects for each requirement, the associated risk of the requirement also will be mitigated. This type of mapping at Company A will help to lower risks associated with requirements.

4.4 Root Cause Analysis

The defect symptoms from Company A appeared when the quality assurance engineer executed the test protocols. Those symptoms were due to the defects. Table 5 displays all potential defects of all the identified defect symptoms from Mapping A. This list of defects will help in finding the root causes of the identified defect symptoms. Table 5 includes root causes from the requirements gathering phase to the maintenance phase.

The hierarchical numbering system of SW91 enables the identification of the phase of the defect. For example, Incompatible Requirement (2.3.4) is one of the root causes listed in Table 5. It has a defect code starting with 2, meaning that this defect belongs to the Requirement Defects (2) from SW91.

When the quality assurance engineer reports the defect to the developer, the developer can address the

Table 5: Distinct SW91 defects from Mapping A.

Distinct SW91 defects used in Mapping A
Requirement Completeness (2.2)
Incompatible Requirement (2.3.4)
Inconsistent Requirement (2.3.5)
Scalability (3.3)
Transactions (3.4.4)
Internal Interfaces (4.2)
Interface Parameter, Invocation (4.2. 2)
Component Invocation (4.2.1)
Wrong Algorithm Selected (4.2.4)
Parameter Structure (4.2.2.3)
Parameter Type (4.2.2.2)
Operator (5.2.2.1.1)
Incorrect Save/Restore (5.3.2.5)
Invalid Path (5.2.1.2.6)
Data Definition (5.3.1)
Bad Translation (5.1)
Size (5.3.1.2)
Inappropriate Cast or Type Conversion (5.3.2.1.4)
Use Before Check (5.3.2.6)
Corrupted Database Upgrade (7.8)

root causes related to the implementation phase. In Table 5, the defects starting with number 5 are related to the implementation phase. Other root causes not related to the implementation phase such as Inconsistent Requirement (2.3.5) or Requirement Completeness (2.2) can be investigated by other people involved in the development of the application such as the business analyst or the software architect. When Company A records the root causes for a release, it will enable finding and eliminating common root causes from future releases. If Company A used this mapping at the testing phase of their application, the following benefits could be observed during and after the testing phase:

1. The quality assurance engineer can report the identified defect symptoms along with the potential root causes which are detailed in SW91.
2. The developer can see and fix the actual root causes of defect symptoms by fixing the SW91 defects used in the mapping.
4. Company A can minimise the occurrence of the same root causes in future releases.

A detailed report of this study was submitted to Company A. This report includes recommendations in order to maximise the benefits of taxonomy based testing. The recommendations and their benefits are listed in Table 6. This section discussed the results and benefits of the taxonomy based testing approach using data from Company A. This section also listed the recommendations provided to Company A based on this study. The next section details interviews conducted with Company A.

5 INTERVIEW WITH COMPANY A

After submission of the report, it was presented to employees from Company A. The employees included the CEO, two developers and a quality

Table 6: Recommendations and benefits.

Recommendation	Benefits
Map the existing identified defect symptoms into SW91 defects.	Minimize the occurrence of the same defects.
	Save test execution time.
	Increase test efficiency by reducing the test cycle.
	Find the possible root causes.
Map each development stage of the control flow diagram into SW91 defects.	Identify defects at an earlier phases.
	The developer can work to avoid mapped SW91 defects when implementing each stage.
	Write test cases to cover those mapped SW91 defects for each control flow diagram.
Map the requirements into SW91 defects.	All the requirements will have been mapped into SW91 defects.
	Write test cases to cover those mapped defects for each requirement.
	Identify the defects at an earlier phase of software development.
	Brainstorm with the quality assurance engineers with possible defects for each requirement.
Map the risks into SW91 defects.	The developer can work to avoid the mapped SW91 defects when implementing the requirements.
	All the risks will have potential defects.
	Avoid the mapped SW91 defects and minimise the risks which are associated with those requirements.
Map the test protocols into SW91 defects.	Brainstorm with the quality assurance engineers with possible defects for each risk.
	Brainstorm with the quality assurance engineers with possible defects.
	Save test execution time.

assurance engineer. Three separate interviews were conducted with the CEO, developers and the quality assurance engineer. The interviews were mainly focused on getting their opinion on the benefits and recommendations. Also, the possibilities for implementing taxonomy based testing at company A were discussed.

The CEO agreed with the recommendations and benefits except for the root cause analysis. He said that root cause analysis was not straightforward because of defects related to the organisation's cultural and environmental change such as lack of communication between employees and lack of internet access. SW91 contains only software defects and it is not focused on defects related to environmental or cultural change.

The developers agreed with the recommendations and the benefits. They preferred to have the defect mapping when moving from user requirements to system requirements. They stated that this mapping would help to minimise the risks when implementing a requirement from scratch.

The quality assurance engineer has accepted the recommendations and the benefits of this study and he wondered how this approach would work at a small or medium-sized organisation. He suggested that a tool to implement taxonomy based testing would save time and effort. He also said that this mapping should take place at the risk management stage to get the maximum benefit from taxonomy based testing. He would like to see how this mapping would benefit a project that is in development.

All four interviewees were asked about the implementation of taxonomy based testing at Company A and its limitations. They said that it is useful to implement, but the main limitations are time and resources with their current project. They agreed that this kind of mapping would help to save time and it would also save the project manager's time.

6 FUTURE RESEARCH

This paper explained a retrospective study conducted using data from Company A. A framework for taxonomy based testing was developed for future implementation and this framework was validated by experts from the software testing industry. This framework will enable the implementation of taxonomy based testing without the researcher's involvement in any MDS companies. The next step of this research will involve implementing this framework in a MDS company, Company B.

The necessary data will be requested from the implementation and the data will be used to evaluate the benefits of taxonomy based testing. The next section details the summary and conclusion of this paper.

7 SUMMARY AND CONCLUSION

Poor quality software in medical devices has caused serious harm to patients' health and increased FDA recalls. Defect taxonomies have been used successfully in software development to prevent and control defects. This paper explained what a defect taxonomy is and how a defect taxonomy can be used in MDS testing to minimise defects and to improve software quality. "Defect classification scheme for health software – SW91" is a standard and defect taxonomy for health software. This research proposed a testing technique, taxonomy based testing using SW91. By using the taxonomy based testing approach, each requirement can be mapped into its potential defects. These mappings at the requirements gathering phase will help to avoid the defects related to the design phase and implementation phase. It will improve software quality by eliminating defects at an earlier phase of software development. Also, this mapping will help to write goal oriented test cases by considering the mapped SW91 defects. If we can write goal oriented test cases based on the mapped defects against the requirements, then it will save test execution time.

This paper explained a retrospective study of taxonomy based testing with data from a MDS company, Company A. The data includes defect symptoms, SDS, requirements and test protocols. The data from Company A was mapped into SW91 defects and benefits were observed. Based on this study, a detailed report was submitted to Company A. This report includes the process used in this study, benefits and recommendations to Company A. This study explained how taxonomy based testing could be used to conduct root cause analysis, improve defect reporting and minimise defects and risks at a MDS company. This paper also discussed the interview on this study conducted with employees from company A and its results. Finally, this paper discussed how this research will be continued with the taxonomy based testing framework.

ACKNOWLEDGEMENTS

This work was supported with the financial support of the Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie)

REFERENCES

- AAMI. (2019). Consensus Standards. Retrieved February 19, 2019, from <http://www.aami.org/newsviews/newsdetail.aspx?ItemNumber=7367>
- Association for the Advancement of Medical Instrumentation. (2016). Future AAMI SW91, 29-September-16 Committee Draft for Vote, Classification of Defects in Health Software. USA. Retrieved from <http://www.aami.org/standards/downloadables/aamirevf.pdf>
- Black, R. (2008). *Advanced Software Testing - Vol. 1* (2nd ed., Vol. 1). Santa Barbara: Rocky Nook Inc.
- Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Ray, B. K., & Moebus, D. S. (1992). Orthogonal Defect Classification-A Concept for In-Process Measurements. *IEEE Transactions on Software Engineering*, 18(11), 943–956. <https://doi.org/10.1109/32.177364>
- FDA. (2011). Understanding Barriers to Medical Device Quality. *FDA Review Document*, 45.
- Felderer, M., & Beer, A. (2013a). Using defect taxonomies for requirements validation in industrial projects. In *RE*. <https://doi.org/10.1109/RE.2013.6636733>
- Felderer, M., & Beer, A. (2013b). Using defect taxonomies to improve the maturity of the system test process: Results from an industrial case study. In *SWQD 2013* (Vol. 133, pp. 125–146). https://doi.org/10.1007/978-3-642-35702-2_9
- Ioan Mihnea Iaco, & Radu, C. (2008). Testing: First Step Towards Software Quality. *Journal of Applied Quantitative Methods*, 3(3), 241–253.
- Li, N., Li, Z., & Sun, X. (2010). Classification of software defect detected by black-box testing: An empirical study. In *WCSE 2010* (Vol. 2, pp. 234–240). <https://doi.org/10.1109/WCSE.2010.28>
- Madachy, R., & Boehm, B. (2008). *ODC COQUALMO - A Software Defect Introduction and Removal Model using Orthogonal Defect Classification*. University of Southern California Center for Systems and Software Engineering. Retrieved from <http://csse.usc.edu/TECHRPTS/2008/usc-csse-2008-817/usc-csse-2008-817.pdf>
- P.S.Cosgriff. (1990). Quality Assurance of Medical Care. *Journal of Public Health*. Retrieved from <http://elibrary.ru/item.asp?id=10628982>
- Rajaram(&), H. K., Loane, J., MacMahon, S. T., & Fergal, M. (2019). A Framework for Taxonomy Based Testing

- Using Classification of Defects in Health Software-SW91. In *Systems, Software and Services Process Improvement* (pp. 606–618). Springer Nature Switzerland AG. Retrieved from https://link.springer.com/chapter/10.1007/978-3-030-28005-5_47
- Rajaram, H. K., Loane, J., MacMahon, S. T., & Mc Caffery, F. (2018). Taxonomy-based testing and validation of a new defect classification for health software. *Journal of Software: Evolution and Process*, 31(1), 1–13. <https://doi.org/10.1002/smr.1985>
- Simone, L. K. (2013). Software-related recalls: An analysis of records. *Biomedical Instrumentation and Technology*, 47(6), 514–522. <https://doi.org/10.2345/0899-8205-47.6.514>
- Simone, L., & Rubery, D. (2014). Lisa Simone and Daniel Rubery: A Tower of Babel with Medical Device Software Failures. Retrieved January 24, 2017, from <https://aamiblog.org/2014/10/10/lisa-simone-and-daniel-rubery-a-tower-of-babel-with-medical-device-software-failures/>

