Taxonomy Based Testing and Validation of a new Defect Classification for Health Software

Journal:	*Journal of Software: Evolution and Process*
Manuscript ID	JSME-18-0102.R1
Wiley - Manuscript type:	Special issue – Empirical paper
Date Submitted by the Author:	06-Jul-2018
Complete List of Authors:	Rajaram, Hamsini; Dundalk Institute of Technology, Visual and Human Centred Computing Loane , John ; Dundalk Institute of Technology, School of Informatics and Creative Arts MacMahon, Silvana; Dundalk Institute of Technology, Visual and Human Centred Computing McCaffery, Fergal; Dundalk Institute of Technology, Computing and Maths
Keywords:	Defect taxonomy, Taxonomy based testing, Defect Classification for Health Software, Software Testing

Taxonomy Based Testing and Validation of a new Defect Classification for Health Software

H. Ketheswarasarma Rajaram[1], J. Loane[2], S. T. MacMahon[3], F. Mc Caffery[4]
Hamsini.Ketheswarasarma@dkit.ie[1], john.loane@dkit.ie[2], silvana.macmahon@dkit.ie[3], fergal.mccaffery@dkit.ie[4]
Dundalk Institute of Technology, Ireland

Abstract

Defect-based testing is a powerful tool for finding errors in software. Many software manufacturers avoid this method because it requires a detailed defect taxonomy that is expensive to construct and difficult to validate. The Association for the Advancement of Medical Instrumentation (AAMI) is developing SW91[1], a defect taxonomy to be published as a standard for health software. This paper details three methods to validate SW91 for its comprehensiveness. The initial validations of SW91 were conducted via mapping vulnerabilities from the Common Weakness Enumeration and a dataset from a medical device software development company in Ireland. Taxonomy based testing is another validation method proposed in this research and its applicability was investigated using empirical data from a medical device software development company in Ireland. Finally, the paper details future plans to implement taxonomy based testing to improve software quality in medical device software and to validate SW91. This validation will focus on the efficiency, reliability, ability to perform useful analyses and defect coverage of SW91.

Keywords

Defect taxonomy, Defect Classification scheme for health software, Validation, Taxonomy based testing, Testing

1. Introduction

This paper is an extension of the EuroSPI 2017 conference paper: Benefits of Defect Taxonomies and Validation of a new Defect Classification for Health Software [1]. Medical devices increasingly rely on software to provide functionality [2]. Due to the introduction of advanced technologies, the medical device software industry is facing massive growth of complex software [2]. This massive growth of medical device software leads to quality risks. Table 1 shows the number of adverse events related to medical device software failures between 2006 and 2011.

[1] This document is still under study and subject to change.

The US Food and Drug Administration (FDA) reports that from 2005 to 2011, 19.4% of medical device recalls were related to software [3]. Another study focused on recalls of medical devices related to computer-based failures such as software, hardware, inputs, outputs, or battery issues. This study reported that 2,303,441 recalls out of 12,024,836 were related to software. Software issues accounted for 33.3% of class I recalls, 65.6% of class II recalls and 75.3% of class III recalls [4], [5]. The Food and Drug Administration (FDA) recall process includes specifically identifying software-related recalls in order to improve medical device quality and to ensure patient safety [6].

The increasing complexity and the growth of medical device software make it difficult to control and prevent defects. Medical device software defects have resulted in increasing FDA recalls and they have potentially increased the risk of patient injuries including threats to life [7], [8]. In order to find software defects and to ensure software quality, software quality assurance practices have been integrated into software development. Software quality assurance processes aim to minimise software defects and show that software meets requirements. There are several software quality assurance activities, such as testing and inspections, which can be used to validate software [9]. However, medical device software development companies face challenges in improving software quality such as the inability to specify the software requirements properly, the lack of adequate software quality assurance procedures and the lack of relevant metrics to track software quality [10], [11].

Research studies suggest that a defect taxonomy is the best way to prevent and control defects [12]–[14]. Defect taxonomies were successfully used in various ways during the testing phase of software development, such as in system testing, brainstorming sessions for testers, measuring testing efficiency and classifying defects [14]–[18]. People use customised or original defect taxonomies in different domains such as the safety critical domain, the business domain and the telecommunications domain. Before they use defect taxonomies, they validate their defect taxonomies in terms of reliability, efficiency, and completeness [17], [19]–[21].

This paper explains the benefits of defect taxonomies, the development of a new defect taxonomy for health software called SW91 by AAMI and how this research contributed to the validation of SW91. Three different validation methods are proposed from his research including a new testing approach call taxonomy based testing. This paper is structured as follows: Section 2 explains what a defect taxonomy is and how industries have used defect taxonomies during software development. Section 3 explains the benefits of using defect taxonomies. Section 4 explains the development of new defect taxonomy for health software, SW91 and it briefly details the structure of SW91. Section 5 explains three different validation methods proposed from this research to validate SW91. It includes mapping defects from databases, mapping defects from medical device software development companies and a testing approach call taxonomy based testing which details how defect categories from a taxonomy can be used in testing. Section 6 outlines plans for future research and it details how taxonomy based testing will be used to validate SW91 and to improve software quality in medical device software development. Section 7 presents the summary and conclusions.

2. *State of the art – The use of defect taxonomies in industry*

A defect taxonomy is a system of hierarchical categories designed to be a useful aid for reproducibly classifying defects in the software development lifecycle [22]. There are many other terms for defect taxonomies including fault categorisation, defect classification, fault classification scheme and bug taxonomy. In this paper, the terms defect taxonomy and defect

classification scheme are used interchangeably. Table 2 lists a number of defect taxonomies which are widely used in different software domains [23], [24].

The remainder of this section explains how different industries have used various defect taxonomies for a variety of purposes in different phases of the software development lifecycle in order to improve software quality.

In 1998, at the Motorola Corporate Software Centre, the GSM Products Division's Base Station Systems (GSMBSS) conducted a study on how the orthogonal defect classification (ODC) scheme can be used to measure the progress of software development [25]. The ODC scheme was applied to an existing project with data collected by Fagan inspection [9]. After a successful feasibility study to verify the suitability of the ODC scheme, the team mapped defect data with minor modifications into the ODC scheme. This study proved that software development progress measurement and process improvement feedback can be produced from data which was collected using an existing inspection method by adopting the ODC scheme. The authors of this study believed the ODC scheme can easily be applied to enhance software quality and to improve customer satisfaction while developing defect prevention and qualitative process management techniques [25].

In 2004, Lutz and Mikulski [19] published work on the analysis of 199 anomalies from seven spacecraft at the Jet Propulsion Laboratory. The purpose of this study was to improve the safety of future missions. The ODC scheme was selected to classify the post-launch safety critical software anomalies in order to extract the defect signature. The following outcomes were highlighted from this study:

- Training on documentation of anomalies can limit the reoccurrence of anomalies.
- The benefit of maintaining the documentation of system requirements for the operational process has been identified.
- Anomalies' analysis enhances the reusability of knowledge from one system to another.
- When comparing the outcomes from other methods related to operational risk, the anomaly patterns obtained by classifying the anomalies using the ODC scheme provided additional understanding of operational risks.

Finally, the authors of this study stated classifying anomalies led to understanding the anomalies triggers and contributed to preventing operational anomalies.

Freimut et al. [20] conducted a study at Robert Bosch GmbH in the business unit for Gasoline Systems (GS) and published their work in 2005. Gasoline Systems developed electronic control units for gasoline engines with embedded software as a key component. It was decided to apply quantitative data management techniques in quality assurance strategies to overcome the lack of information related to quality assurance and overall system quality at Bosch GS. They defined, introduced and validated their customised defect classification scheme to track defects which are involved in software development and process measurement [20]. The following outcomes were obtained after applying their defect classification scheme:

- Defect flow distribution and its outputs were observed.
- Identification of the defects introduced in early stages and identified in later stages.
- Defect data from the case study which identified categories with a high number of defects.
- Providing the measurement outputs to management.

In 2007, Robillard et al. [26] published work detailing the measurable test efficiency in a software product due to changing testing practices. This research was conducted with a team that developed audio software for video games. Two different testing phases, A and B, were used to measure the test efficiency. Phase A used implicit testing practices to record defects.

In the second phase, B, an "Easy to follow" scheme was proposed to record the testing practices in order to make developers aware of the type of testing activities involved. A modified ODC scheme was used to record the defects in both testing phases and the following outcomes were observed [26]:

- The distribution of defects for the type of activity conducted in each phase, such as design review, code inspection, and unit test.
- The distribution of defects based on the discovery attributes. Discover attributes indicate who found the defects.
- The distribution of defects based on the activity qualifier attributes. The activity qualifier attribute indicates whether defects were found opportunistically or during planned testing.
- The statistical understanding of software process measurement using both types of testing.

In 2008, the ODC scheme was used in NASA flight projects. The ODC scheme was used as an extension of the COnstructive QUALity Model (COQUALMO) developed by Raymond Madachy and Barry Boehm from the University of Southern California, USA. The ODC COQUALMO model was used for critical NASA flight projects [18]. The ODC scheme was successfully adopted in defect reduction strategies. The ODC COQUALMO model helped in providing a highly detailed view of the defect profiles and their impact on specific risks. The ODC COQUALMO model with automated risk minimisation helped to meet the quality goals of NASA's flight projects in a shorter time with fewer resources [18].

In 2010, Li et al. [17] presented an extended and modified defect classification scheme named the Orthogonal Defect Classification scheme for Black-box Defects (ODC-BD) which was created based on the ODC scheme. This empirical study was aimed at helping black-box defect analysers and black-box testers to improve their testing efficiency and analysis. It was proved that the effort for defect analysis was reduced by 15% after applying the ODC-BD scheme. The test efficiency, measured as the number of detected defects per unit time, in the first week, without using the ODC-BD scheme, was 0.075. In the second month, the test efficiency increased to 0.125 using the ODC-BD scheme [17]. This empirical study with 1660 black-box defects is a good example of measuring the black-box testing process with the ODC scheme.

In 2012, Mellegård et al. [27] published their work on developing an effective and systematic software defect classification scheme at Volvo car corporation in Sweden. They developed a software defect classification scheme "the Light-weight Defect Classification scheme", which complements the IEEE standard classification for software anomalies [28], [29]. This study demonstrated the customisation of a generic defect classification scheme to classify defects. Adopting a customised defect classification scheme minimised the time required to find defects while helping to characterise the defects. The Hewlett-Packard defect categorisation scheme has been used within Hewlett-Packard departments for many different purposes such as root cause analysis and defect presentation [23], [30].

In 2014, Nuno Silva and Marco Vieira [31] published their work which demonstrated the importance of domain specific defect classification schemes. They focused on four systems from the aerospace and space industries. They demonstrated the following problems in adopting a generic classification scheme into a safety critical domain:

- There are problems with adopting a generic defect classification scheme without considering the defect propagation effects and the interconnection of defects from different phases of the software development lifecycle.
- Inability to cover all the defects with existing listed defect types, defect triggers and defect impacts.

- Differentiating dimensions to keep the orthogonality of the defects was not easy to achieve.
- Not showing the connection to the quality models.
- There were difficulties getting the necessary level of information related to each defect to map with the defect type, defect trigger or defect impact.
- Difficulties mapping non-functional defects.
- Difficulties mapping the defects to a related standard.

The above points demonstrate the problems in adopting a generic defect classification scheme into a safety critical domain. This study highlights the need for improved and domain specific defect taxonomies to classify defects in safety critical domains. Since medical device software is often safety critical, the necessity of a domain specific defect classification scheme has been identified and a defect classification scheme is being developed for healthcare software called SW91 [32]. The development of SW91 is explained in Section 4.

This section detailed how different defect classification schemes were used in different industries from 1998 to 2014 and the importance of domain specific defect classification schemes. The next section discusses the benefits of defect classification schemes.

3. *Benefits of defect classification schemes*

Bernd Freimut [23] has detailed various benefits of defect classification schemes including characterisation of the defects found, defect prevention, control inspections, evaluation and improvement of technologies, control testing, plan testing and reduce field defects.

Vallespir et al. [24] stated a defect classification scheme makes it easy to find the injected defects while providing information on phases, activities, and disciplines throughout the software development lifecycle. Robert B. Grady from Hewlett Packard stated that the benefit of classifying defects is to help find the correct quality assurance activity. He stated: “As you categorise the defects, you will uncover a variety of symptoms. A typical first step will be for you to decide to do better or different inspections or tests” [30]. When combining both Vallespir et al. and Robert B. Grady’s statements, injected defects could be identified and classified by a defect classification scheme. Those classified defects will inform the choice of the correct quality assurance activities. Detailed defect classification schemes play a significant role in understanding the software development process [33]. Defect classification schemes can be created for several different purposes. These include:

1. Making decisions during software development.
2. Tracking defects for process improvement.
3. Guiding the selection of test cases.
4. Analysing research results [33].

Vogel has detailed a procedure for medical device software defect management. “Classification” is the second of eight steps. Vogel’s defect procedure supports the need for defect classification schemes in medical device software development. He stated the importance of classification in medical device software as follows: “the classification is important for later determination on the recommendations and means for verifying any changes made to deal with the defect”. Safety critical domains have utilised defect classification schemes to reduce defects and to improve analysis [2], [18], [31]. Safety critical domains need a unique defect classification scheme tailored to their unique needs [31]. The medical device software industry is such a safety critical domain and hence should have its own domain specific defect classification scheme.

The Association for the Advancement of Medical Instrumentation (AAMI) is developing a defect classification scheme for health software which includes medical device software. Section 4 explains the development of the defect classification scheme for health software called SW91. Before the development of SW91, there has been no defect classification scheme specifically developed for use in the medical device software industry. It is hoped that applying a defect classification scheme into the medical device software industry will bring similar benefits observed in Sections 2 and 3.

4. AAMI and Development of SW91

AAMI is a non-profit organisation founded in 1967 and it is the primary source of consensus standards, both national and international, for the medical device industry, as well as practical information, support, and guidance for healthcare technology and sterilization professionals [34]. The development of SW91 started in 2014 and aims to provide a common language to classify defects and improve software quality in health software including medical device software [32]. SW91 was published in September 2016 for first public comment. The comments were resolved and SW91 was published in April 2017 for second public comment. The comments on the second draft were resolved and currently, SW91 is in the final stage of processing to publish as a standard and will be published in Q2-Q3 2018.

SW91 includes defect categories from the planning to the maintenance of a system. SW91 contains multi-level defect categories at the parent level and child level. Each parent level defect category includes several child defect categories. Each defect category has a defect code with a unique number. This numbering system uses a hierarchical system.

Table 3 shows the parent level defect categories from SW91. Each parent level category is numbered and each child level category is represented by appending a period and a number of the parent level number. Each defect category has annotations and some of the defect categories have an example as well. For example, Expression Evaluation (5.2.2.1) is one of the child level defect categories from Implementation Defects (5). It has the following annotation: "This category contains defects having to do with the way arithmetic, Boolean, string, and other expressions are evaluated."

SW91 contains annexes to demonstrate how to use SW91 defect categories and to show how user-friendly the standard is. For example, annex A of SW91 details how SW91 defect categories can be used in industries and academia. The following use cases are detailed in annex A:

- Development organisation looking to remove common defects.
- Development organisation performing an industry comparison.
- Business manager measuring return on investment.
- College professor studying software defect trends over time.

From the literature review conducted as part of this research it was observed that defect taxonomies were validated for their reliability, usability, defect coverage and completeness. Since SW91 is a new defect taxonomy, it is important to validate SW91 before it is used. This research has identified three validation methods which are detailed in the following section.

5. *Validation of SW91*

Bernd Freimut analysed different defect classification schemes based on their structure and their usability [23]. In the literature, it was found that defect classification schemes were validated for their reliability, ability to perform useful analyses, and efficiency [17], [19]–[21]. These features will be considered in the validation of SW91. In addition to the above terms, the defect coverage of SW91 will be validated in this research. Validation of SW91 has the following three different tracks:

1. Taxonomy based testing.
2. Mapping defects from medical device software companies.
3. Mapping defects from databases.

Section 5.1 explains the first track of the validation using taxonomy based testing. Section 5.2 explains the second track of validation and it details how data from a medical device software development company from Ireland was mapped to SW91 and how this mapping helped to conduct a retrospective analysis using a taxonomy based testing approach. Section 5.3 presents the third track of validation and it explains how open source defect data was mapped into SW91.

5.1 Taxonomy Based Testing

Defect taxonomies have been successfully used in software testing [15]. Creating the test cases for the defect categories from a defect taxonomy gives better test coverage [15], [35]. Michael Felderer and Armin Beer have conducted significant research on defect taxonomy-supported testing [14]. They stated, "Defect taxonomies can be applied to control the design of tests and the quality of releases to keep testing manageable although time and resources in projects are limited". In their research, a novel process of system testing using a defect taxonomy has been proposed and implemented. A case study was used to explain how a defect taxonomy can be integrated into the standardised test process defined by the International Software Testing Qualifications Board - ISTQB.

In taxonomy based testing, the requirements will be mapped into potential defect categories from the defect taxonomy and the test cases will be written based on the requirements and the mapped defect categories. The test cases will be executed to verify whether the software complies with the relevant requirements and does not contain the mapped defect categories from the defect taxonomy. Figure 1 explains taxonomy based testing. Taxonomy based testing will be used to validate SW91 in terms of the reliability, efficiency, useful analyses and the defect coverage. When it comes to the reliability of SW91, if a statistically significant number of quality assurance engineers at a medical device software company mapped the same given defects with the same defect categories from SW91, and then this will be used to demonstrate the reliability of SW91. At the end of the taxonomy based testing, if SW91 helped to increase the test efficiency and helped to reduce similar software defects in future phases, this will be considered as validation of SW91 in terms of efficiency and useful analyses. If SW91 covers all identified defects from the requirements to the final system, this will be considered as a validation of defect coverage. Taxonomy based testing will be implemented at different companies including medical device software companies and a testing company. Section 6 details how taxonomy based testing will be implemented at the companies.

Since this research is related to medical device software development, it is important to consider medical device software development processes. Successful medical device software development processes are required to demonstrate compliance with a set of medical device standards and regulations [36]. In that respect, EN 62304:2006+A1:2015 is a standard for the medical device software development process and it details the medical device software development processes that are required in order to build safe software. Annex C.4 of EN 62304:2006+A1:2015 explains a V-model for programmable electrical medical system requirements [37]. This V model includes the medical device software development lifecycle processes included in EN 62304:2006+A1:2015. Since SW91 has defect categories for each phase of the medical device software development process stated in EN 62304:2006+A1:2015, it can be linked with each phase of medical device software development process stated in the V model from Annex C.4 of EN 62304:2006+A1:2015. Figure 2 explains the modified V model from Annex C.4 of EN 62304:2006+A1:2015 [37] and SW91 defect categories. The grey colour V model shows the medical device software development phases and the red colour V model shows the SW91 defect categories.

When considering the V model shown in Figure 2, it is possible to implement taxonomy based testing at any phase of medical device software development such as software requirement analysis or software implementation. Test cases can be created at any phase of medical device software development and they can be executed in each phase's validation step. Section 6 details how this approach will be implemented at different companies.

This section explained the first track of SW91 validation. The next two sections explain two different mappings conducted in this study. Baldassarre et al. defined and followed a theoretical comparison process in their harmonisation process of ISO/IEC 9001:2000 and CMMI-DEV [38]. It has following steps:

- Analysing the models.
- Designing the mapping.
- Carrying out the mapping.
- Presenting outcomes.
- Analysing results.

The above steps were considered in this research and following mappings were carried out:

- Mapping data from a medical device software company into SW91 defect categories.
- Mapping Common Weakness Enumeration (CWE) vulnerabilities into SW91 defect categories.

5.2 Mapping data from a medical device software company into SW91 defect categories

Company A from Ireland develops medical device software applications. Company A was contacted and empirical data was requested to map to SW91 defect categories. The benefits of defect taxonomies, the taxonomy based testing approach and the need for defect taxonomies in the medical device software industry were presented to the management of company A and data was requested. Relevant data was obtained from company A and the following data has been used in this mapping:

1. Defects.
2. Software Design Specification.
3. User Requirement Documentation.
4. Risks.

5. Testing Protocols.

Figure 3 displays the mappings of data from company A into SW91 defect categories. In mapping A as shown in Figure 3, the defects from company A contained defect symptoms such as algorithm not working correctly and warning alerts not appearing. These symptoms were due to some defects in company A's application. Those defect symptoms were mapped into possible defect categories from SW91. Eight distinct defect symptoms were mapped into twenty distinct SW91 defect categories. Some slight changes of wording were observed between the defect symptoms from company A and SW91 defect categories. For example, a defect symptom from company A was described as "Units not converting correctly". This defect was mapped into the following SW91 defect category: Inappropriate Cast or Type Conversion (5.3.2.1.4). All defects were mapped into SW91 defect categories.

In mapping B from Figure 3, control flow diagrams from the software design specification document were mapped into SW91 defect categories. The appendix of the software design specification document from company A has very detailed control flow diagrams for their applications.

They describe the functionality of the application step by step. In this study, three different control flow diagrams were selected and each state of the control flow diagrams was mapped into distinct SW91 defect categories. In this mapping, all elements from the control flow diagrams were mapped into nineteen SW91 defect categories from architectural defects, design defects and implementation defects. Since SW91 uses a hierarchical numbering system for defect categories, it was easy to jump into the relevant defect category at the appropriate level. For example, if a control flow diagram has the following step: "Count <1", then searching for a relevant defect category from the implementation defects is straightforward. The following defect categories were assigned to the above processing step "Count <1":

- Mixed Sign (5.3.2.7.1).
- Use Before Check (5.3.2.6).
- Invalid Path (5.2.1.2.6).
- Operator (5.2.2.1.1).

In mapping C of Figure 3, the user requirement documentation from company A contains forty-one requirements including functional and non-functional requirements. Out of forty-one requirements from the user requirements documentation, thirty-nine were mapped into forty-three distinct SW91 defect categories. Two non-functional requirements were very general and could not be mapped into any existing SW91 defect categories. For every other requirement, the defect categories were selected from SW91 and mapped. Every requirement from the user requirement documentation has associated risks which have been assigned a priority. Since the requirements were already mapped into SW91 defect categories in mapping C, those SW91 defect categories used in mapping C were merged with the risks associated with their requirements. Despite the user requirements documentation and software design specification documents being prepared for the company's own use, it was possible to map them into SW91 defect categories.

A separate mapping of the testing protocols and SW91 defect categories was not performed because the testing protocols were already linked with the relevant software design specification and the user requirements from the user requirements document. The defect categories from SW91 used in both mapping B and C were directly linked with the respective testing protocols.

As we discussed in Section 5, in terms of the validation of a defect classification scheme, the defects from company A were mapped into SW91 defect categories and this mapping helped to validate the defect coverage of SW91. The reliability of SW91 can be observed here. Normally, the reliability of a defect classification scheme is determined by the mapping

of the same defects by different people. If different people map the same defects with the same defect categories from a classification scheme, this shows that the defect classification scheme has good reliability. Here, the same defects from three different documents including user requirements documentation, control flow diagrams from the software design specification document and defect symptoms from defect data mapped to the same categories in SW91. Due to the confidentiality of the data from company A, it is not possible to detail all the mappings here.

The next section explains how this mapping was used in the retrospective analysis of taxonomy based testing and the benefits that were observed from applying taxonomy based testing.

5.2.1 A retrospective study on taxonomy based testing

This mapping work was used in a retrospective study of taxonomy based testing. The first step of taxonomy based testing proposes to map the requirements into defect categories from a defect taxonomy. In this study, the requirements and the software design specification from company A were mapped into SW91 defect categories. Defect symptoms from company A were also mapped into SW91 defect categories to see if taxonomy based testing can be performed using the empirical data from a medical device software development company. From the three mappings (A, B and C), the following three sets of common SW91 defect categories were observed:

1. Ten common SW91 defect categories from mapping A and mapping B.
2. Eleven common SW91 defect categories from mapping A and mapping C.
3. Six common SW91 defect categories from mapping A, mapping B and mapping C.

When the requirements have been gathered and company A does this mapping, it will enable them to see the possible defect categories for each requirement. This type of mapping also allows goal oriented test cases to be written, consistent with taxonomy based testing. These goal oriented test cases will be based on the requirements and the respective mapped defect categories from SW91. Execution of the generated goal oriented test cases will save time in finding defects when executing the test cases.

Since company A has detailed control flow diagrams, mapping each stage of the control flow diagram into SW91 defect categories will help to minimise defects at the development phase. When we have the potential defect categories for every stage, developers can work to avoid those defects. Quality assurance engineers can run the test cases to find the mapped defect categories. This will minimise the time to find the defects and it will help to prevent defects at the earliest possible phase of software development.

Company A has risks for every requirement in their user requirement documentation. Those risks are prioritised by severity. If every requirement is mapped into defect categories from SW91, the risks can be used to prioritise which defects should be fixed first. This study also revealed the following benefits which are detailed in the following sections:

1. Defect reporting at the testing phase.
2. Defect minimisation.
3. Risk minimization.

5.2.1.1 Defect reporting at the testing phase

Defect data from company A contained defect symptoms such as warning alerts not appearing. Those defect symptoms must be reported by a software quality assurance engineer at company A. According to the current format of the defect reporting at company A, the developers do not know which defects caused the reported defect symptoms such as warning alerts not appearing. When the developers attempt to fix the reported defect symptoms, it will be hard for them to fix due to the current defect reporting format which does not provide details

of the defects causing the symptoms. Therefore, those defect symptoms could appear again in the second round of testing due to some other defect of which the developers or the software quality assurance engineer were not aware. In this situation, the developers and software quality assurance engineers wasted their time addressing a poorly defined defect. This situation can be addressed at company A by mapping the defect symptoms into actual defects. For example, in mapping A, the defect symptom warning alerts not appearing was mapped into the following SW91 defect categories:

- 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 defects with SW91 defect categories which could cause the reported defect symptom, warning alerts not appearing. Developers can try to fix the reported defect symptoms by addressing the mapped SW91 defects categories for the failed test. This type of mapping will minimise the reoccurrence of defect symptoms by checking all possible mapped SW91 defect categories. This type of mapping would reduce the development time and help to anticipate possible defects. If this mapping exists at company A, there will be a common and fixed language for software quality assurance engineers and developers to communicate the test failures.

5.2.1.2 Defect minimisation

Since company A has very detailed control flow diagrams, mapping each state of the control flow diagram into SW91 defect categories could minimise the occurrence of defects from the implementation phase. When company A has the potential defect categories for every state of the control flow diagrams and requirements, developers can work to avoid those mapped defect categories during the development. Software quality assurance engineers can execute test cases to find those mapped defect categories. 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.

5.2.1.3 Risk minimisation

Each requirement from the user requirements documentation has associated risks. Using the mappings explained in this study, each requirement has been mapped into its potential SW91 defect categories. If each requirement can be implemented with the minimum number of defects, then it is possible to minimise the associated risks as well. This type of software development with mapped SW91 defect categories at company A will help to minimise risks.

This section explained how an initial investigation was carried out with data from a medical device software development company and how the applicability of taxonomy based testing was investigated. The applicability of taxonomy based testing to the empirical data from a medical device software development company and benefits of taxonomy based testing was detailed in this section. Software Process Improvement (SPI) aims to improve quality and productivity levels [39]. Similarly, implementing taxonomy based testing will help to improve the quality of the product via achieving the above-stated benefits.

Section 5.3 presents another validation method of SW91 which is mapping defects from open databases.

5.3 Mapping Common Weakness Enumeration (CWE) vulnerabilities into SW91 defect categories

The CWE is an open source list containing common software weaknesses and their vulnerabilities. CWE Version 2.9 [40] was the latest version available when the mapping was started. CWE Version 2.9 contains 1004 vulnerabilities. This version of the CWE has multiple views such as full dictionary view, development view, research view and fault pattern view.

Prior to the mapping, it was necessary to select the appropriate view of vulnerabilities from the CWE. After carefully analysing the multiple views, the cross section view from the CWE was mapped into SW91 defect categories. The approach to the mapping and the selection of the cross section view was discussed and finalised with the SW91 development team. The SW91 development team is composed of members from a number of relevant disciplines such as medical device product development, software engineering, software quality, and regulatory policy with members considered expert in their field.

The cross section view contains a selection of software weaknesses which represent the range of weaknesses captured in the CWE. These weaknesses include a total of 158 vulnerabilities [40]. From the CWE cross section, out of 158 vulnerabilities, 150 vulnerabilities were successfully mapped into SW91 defect categories. This was a manual one to one mapping. In the initial mapping, it was not possible to find a suitable category from SW91 for the following eight vulnerabilities from the CWE cross section:

- 173: Improper Handling of Alternate Encoding.
- 175: Improper Handling of Mixed Encoding.
- 222: Truncation of Security-relevant Information.
- 434: Unrestricted Upload of File with Dangerous Type.
- 323: Reusing a Nonce, Key Pair in Encryption.
- 486: Comparison of Classes by Name.
- 502: Deserialization of Untrusted Data.
- 798: Use of Hard-coded Credentials.

The one to one mapping was reviewed by the SW91 development team who paid particular attention to the eight vulnerabilities that could not be mapped. The team members checked for the possibility of mapping them into existing SW91 defect categories and there was a discussion on adding new defect categories and changing the name of a defect category, in order to ensure that all vulnerabilities could be mapped into a suitable defect category from SW91.

Out of the eight vulnerabilities which could not initially be mapped, five vulnerabilities were mapped into newly added defect categories or defect categories that required name changes. Three vulnerabilities were mapped into an existing defect category. This one to one mapping established that all of the CWE cross section vulnerabilities could be mapped to at least one defect category from SW91.

From the completed one to one mapping, a subset of vulnerabilities was selected by an experienced team member. The selected subset included eighteen vulnerabilities from all phases of the software development lifecycle and one to many mappings were conducted for these vulnerabilities. In the one to many mappings, each vulnerability was potentially mapped into multiple different defect categories from SW91. The purpose of the one to many mapping was to show the usability of SW91. Figure 4 shows an example of a one to many mapping[2].

[2] This capture is from the draft version of the standard that was out for public comment. This is not intended to represent the final version of the standard.

In the validation process, out of the eighteen mappings, five mappings were accepted by the team without any changes. Three mappings were changed by adding additional defect categories into the existing mapping. Two mappings were changed by adding additional defect categories and deleting some mapped categories from the existing mappings. Three mappings were changed by adding additional defect categories including the reason for why those categories were selected. Two mappings were changed by deleting selected mapped defect categories from the existing mappings. One mapping was changed by deleting a defect category from existing mapping and including the reason for why other categories were selected. Finally, two existing mapping were deleted and replaced with new defect categories from SW91.

For example, vulnerability 642: External Control of Critical State Data was mapped by the researcher into the following two defect categories from SW91. The third defect category was added by the team members when they were validating the one to many mappings:

1. **Failure to Protect (5.3.2.3.3):** Software permits access to an object that should be protected (for security reasons rather than for coherency), assuming the design is correct.
2. **Private Data Declared Public (5.3.1.6.2):** An object is declared as public when it should be private. The object may be accessible to functions that should not use it, creating an unintended dependency or security vulnerability**.**
3. **Security (3.8):** The defects are related to security issues in the architecture, such as compromising of sensitive information, choosing an inappropriate authentication protocol, not using access control, or communication integrity. It may also include the use of unsigned software and the introduction of unknown changes after the software is deployed. Note that many issues related to security involve requirements inadequacies, and many security vulnerabilities might be caused by poor design or implementation activities. Capturing all the causes and contributing factors for a security defect should involve identifying possible Requirement Defects (2.*), Design Defects (4.*) and Implementation Defects (5.*) rather than categorising every failure associated with security as a Security (3.8) defect.

This one to many mapping shows how a user can select multiple different defect categories from SW91 to map into a particular vulnerability. This subset of mappings was added as annex D in SW91. This CWE mapping was conducted as part of the validation of SW91. This work was carried out to determine the defect categories and their coverage in SW91 when compared with publicly available vulnerabilities.

This section detailed the three validation methods for SW91. The next section explains how the first track of validation, taxonomy based testing will be implemented at different companies in order to improve software quality and to validate SW91.

6. Future work

This paper explains three validation methods for SW91 including a testing approach called taxonomy based testing, mapping data from a medical device software development company and the CWE mapping. The mapping work with company A was used to conduct a retrospective analysis of taxonomy based testing. Another retrospective analysis will be performed at a medical device software company, company B in Ireland to further investigate taxonomy based testing. The company B retrospective study will be conducted with the following participants:

1. Quality assurance engineers.

2. Developers.
3. Quality assurance manager.

Requirements and their identified defects will be collected from a completed project at company B. First software quality assurance engineers and developers will be asked to map the requirements into the potential SW91 defect categories. Then they will be asked to map the defects into SW91 defect categories. An analysis will be conducted on the mapped defect categories to investigate taxonomy based testing. The results of the analysis will be used to evaluate the following points:

- The implication of taxonomy based testing at a medical device software development company.
- Comprehensiveness of SW91.

A protocol has been written for the future implementation of taxonomy based testing. This protocol will enable the implementation of taxonomy based testing in any medical device software development company. The protocol has the following four phases:

1. Requirements mapping.
2. Test case writing.
3. Test execution.
4. Results analysis.

This protocol will be implemented at company C which develops medical device software and a software testing company, company D from Ireland. Data from each of the companies will be used to evaluate taxonomy based testing and to validate SW91. To continue the validation of SW91, this taxonomy based testing will consider the following points in terms of the validation of SW91:

- The efficiency of SW91.
- The reliability of SW91.
- Useful analyses enabled by SW91.
- Defect coverage.

The next section details the summary and the conclusion of this paper.

7. *Summary and Conclusion*

This paper explained software quality problems in the medical device software industry and why quality assurance practices may fail to identify defects. The benefits of defect taxonomies were outlined. The necessity for a domain specific defect taxonomy in safety critical domains was also briefly explained. The development of a new defect classification for health software, SW91, is underway to address this need in the medical device domain. Defect taxonomy validation methods were identified. These will be used to validate SW91. As a part of the validation of this newly developed defect classification for health software - SW91, two mappings were completed. Firstly, a mapping with the CWE's vulnerabilities and, secondly, with data from a medical device software company, company A. These mappings examined the reliability and the defect coverage of SW91. The company A mappings were used to conduct a retrospective analysis of taxonomy based testing. This paper detailed the applicability of taxonomy based testing and it's benefits. Future work will utilise the taxonomy based testing protocol to implement taxonomy based testing at two medical device software development companies and a testing company. The results of this implementation will be used to assess the benefits of taxonomy based testing and the results will also be used to validate SW91 in terms of efficiency, reliability, enabling of useful analyses and defect coverage of SW91.

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).

Literature

[1] H. K. Rajaram, J. Loane, S. T. MacMahon, and F. M. Caffery, 'Benefits of Defect Taxonomies and Validation of a new Defect Classification for Health Software', in *EuroAsiaSPI²*, 2017, p. 15.

[2] PTC.com, 'Software Development for Medical Devices.' pp. 1–9.

[3] L. K. Simone, 'Software-related recalls: An analysis of records', *Biomed. Instrum. Technol.*, vol. 47, no. 6, pp. 514–522, 2013.

[4] U.S. Food and Drug Administration, 'Recalls, Market Withdrawals, & Safety Alerts - Background and Definitions.' [Online]. Available: https://www.fda.gov/Safety/Recalls/ucm165546.htm. [Accessed: 03-Feb-2017].

[5] H. Alemzadeh, R. K. Iyer, Z. Kalbarczyk, J. Raman, and J. Raman, 'Analysis of safety-critical computer failures in medical devices', *IEEE Security and Privacy Magazine*, vol. 11, no. 4, pp. 14–26, 2013.

[6] FDA, 'Medical Device Recall Report FY2003 to FY2012', pp. 1–20, 2014.

[7] M. McHugh, F. McCaffery, S. T. MacMahon, and A. Finnegan, 'Improving Safety in Medical Devices from Concept to Retirements', *Handb. Med. Healthc. Technol.*, 2013.

[8] M. W. Bovee, D. L. Paul, and K. M. Nelson, 'A framework for assessing the use of third-party software quality assurance standards to meet FDA medical device software process control guidelines', *IEEE Trans. Eng. Manag.*, vol. 48, no. 4, pp. 465–478, 2001.

[9] J. Tian, *Software quality engineering*, vol. 32, no. 1. Dallas, TX: John Wiley & Sons, Inc, 1990.

[10] P.S.Cosgriff, 'Quality Assurance of Medical Care', *J. Public Health (Bangkok).*, 1990.

[11] FDA, 'Understanding Barriers to Medical Device Quality', *FDA Rev. Doc.*, p. 45, 2011.

[12] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, B. K. Ray, and D. S. Moebus, 'Orthogonal Defect Classification-A Concept for In-Process Measurements', *IEEE Trans. Softw. Eng.*, vol. 18, no. 11, pp. 943–956, 1992.

[13] M. Felderer and A. Beer, 'Using defect taxonomies for requirements validation in industrial projects', in *Requirements Engineering Conference (RE)*, 2013, pp. 296–301.

[14] M. Felderer and A. Beer, 'Using defect taxonomies to improve the maturity of the system test process: Results from an industrial case study', in *SWQD 2013*, 2013, vol. 133, pp. 125–146.

[15] R. Black, *Advanced Software Testing - Vol. 1*, 2nd ed., vol. 1. Santa Barbara: Rocky Nook Inc, 2008.

[16] G. Vijayaraghavan and C. Kaner, 'Bug taxonomies: Use them to generate better tests', in *STAR EAST 2003*, 2003, pp. 1–40.

[17] N. Li, Z. Li, and X. Sun, 'Classification of software defect detected by black-box testing: An empirical study', in *WCSE 2010*, 2010, vol. 2, pp. 234–240.

[18] R. Madachy and B. Boehm, 'ODC COQUALMO - A Software Defect Introduction and Removal Model using Orthogonal Defect Classification', 2008.

[19] R. R. Lutz and I. C. Mikulski, 'Empirical analysis of safety-critical anomalies during operations', *IEEE Trans. Softw. Eng.*, vol. 30, no. 3, pp. 172–180, 2004.

[20] B. Freimut, C. Denger, and M. Ketterer, 'An Industrial Case Study of Implementing and Validating Defect Classification for Process Improvement and Quality Management', in *International Software Metrics Symposium*, 2005, pp. 10–19.

[21] M. M. Manhães, M. C. P. Emer, and L. C. Bastos, 'Classifying Defects in Software Maintenance to Support Decisions Using Hierarchical ODC', *Comput. Beach*, pp. 283–292, 2014.

[22] International Software Testing Qualifications Board, 'Standard glossary of terms used in Software Testing', 2017. [Online]. Available: http://glossary.istqb.org/search/defect taxonomy.

[23] B. Freimut, 'Developing and Using Defect Classification Schemes', 2001.

[24] D. Vallespir, F. Grazioli, and J. Herbert, 'A framework to evaluate defect taxonomies', in *XV Argentine Congress of Computer Science*, 2009.

[25] N. Bridge and C. Miller, 'Orthogonal Defect Classification Using Defect Data to Improve Software Development', *Softw. Qual.*, vol. 3, no. 1, pp. 1–8, 1998.

[26] P. N. Robillard and T. Francois-Brosseau, 'Saying, "I Am Testing," is Enough to Improve the Product: An Empirical Study', in *ICCGI 2007*, 2007, p. 5.

[27] N. Mellegård, M. Staron, and F. Törner, 'A light-weight defect classification scheme for embedded automotive software and its initial evaluation', 2012.

[28] IEEE-SA Standard Board, 'IEEE Standard Classification for Software Anomalies', 1993.
[29] IEEE-SA Standard Board, 'IEEE Standard Classification for Software Anomalies (Revision of IEEE Std 1044- 1993)', 2010.
[30] Robert B. Grady, *Practical software metrics for project management and process improvement*. Prentice Hall PTR, 1992.
[31] N. Silva and M. Vieira, 'Experience report: Orthogonal classification of safety critical issues', *Proc. - Int. Symp. Softw. Reliab. Eng. ISSRE*, pp. 156–166, 2014.
[32] L. Simone and D. Rubery, 'Lisa Simone and Daniel Rubery: A Tower of Babel with Medical Device Software Failures', *AAMIBlog*, 2014. [Online]. Available: https://aamiblog.org/2014/10/10/lisa-simone-and-daniel-rubery-a-tower-of-babel-with-medical-device-software-failures/. [Accessed: 24-Jan-2017].
[33] D. Kelly and T. Shepard, 'A Case Study in the Use of Defect Classification in Inspections', *Conf. Cent. Adv. Stud. Collab. Res.*, 2001.
[34] Association for the Advancement of Medical Instrumentation, 'About AAMI', 2016. [Online]. Available: http://www.aami.org/membershipcommunity/content.aspx?ItemNumber=1292&navItemNumber=2906. [Accessed: 22-Dec-2016].
[35] C. Kaner, J. Falk, and H. Q. Nguyen, 'Testing Computer Software Second Edition APPENDIX: COMMON SOFTWARE ERRORS', 1993, pp. 1–89.
[36] P. Rust, D. Flood, and F. McCaffery, 'Creation of an IEC 62304 compliant software development plan', *J. Softw. Evol. Process*, vol. 28, no. 11, pp. 1005–1010, 2016.
[37] IEC, 'Medical device software — Software life-cycle processes', *Bs En 62304:2006 +a1:2015*, vol. 3, no. November 2008. p. 88p, 2015.
[38] M. T. Baldassarre, D. Caivano, F. J. Pino, M. Piattini, and G. Visaggio, 'Harmonization of ISO IEC 9001 2000 and CMMI DEV from a theoretical comparison to a real case application.pdf.' pp. 309–335, 2012.
[39] R. Messnarz, M. A. Sicilia, M. Biro, E. García-Barriocanal, M. Garre-Rubio, K. Siakas, and A. Clarke, 'Social responsibility aspects supporting the success of SPI', *Journal of Software: Evolution and Process*, vol. 26, no. 3. pp. 284–294, 2014.
[40] 'CWE - CWE-884: CWE Cross-section (2.9)', 2016. [Online]. Available: https://cwe.mitre.org/data/definitions/884.html. [Accessed: 11-Jan-2017].

Author CVs

Hamsini Ketheswarasarma Rajaram
Hamsini Ketheswarasarma Rajaram is a Doctoral Researcher in the Regulated Software Research Centre. She graduated with Honors in Bachelor Science in Information Technology (Software Engineering) in the year 2011, at Sheffield Hallam University, UK. She then completed her Master of Science degree in Bioinformatics in 2015, IBMBB University of Colombo, Sri Lanka. Currently, her research focus is on validating medical device software defect taxonomy.

John Loane
Dr. John Loane is a lecturer at the Dundalk Institute of Technology. He is a researcher in the Regulated Software Research Centre and formerly a researcher at the NetwellCASALA research centre. He has received EU FP7 research funding to develop an Indicator-based Interactive Decision Support and Information Exchange Platform for Smart Cities.

Silvana Togneri Mac Mahon
Dr Silvana Togneri Mac Mahon is a Postdoctoral Researcher in the Regulated Software Research Centre. She is currently working as a postdoctoral researcher on a Lero project which focuses on the development of a Medical Device Software Development Framework. She has

acted as international project leader, author and editor for the development of ISO/TR 80001-2-7 and is currently involved in the revision of the IEC 80001-1 standard.

Fergal MC Caffery
Dr Fergal Mc Caffery is a Lecturer with Dundalk Institute of Technology. He is the leader of the Regulated Software Research Group in Dundalk Institute of Technology and a member of Lero. He has been awarded Science Foundation Ireland funding through the Stokes Lectureship and Principal Investigator Programmes to research the area of software process improvement for the medical device domain. Additionally, he has received EU FP7 research funding to improve the effectiveness of embedded software development environments for the medical device industry.

Table 1: Medical device software related adverse events [5]

Device type	Number of adverse events		
	Deaths	Injury	Malfunction
Image processing system (LLZ)	1	0	4
Image-intensified fluoroscopic x-ray system (JAA)	0	1	2186
Implantable defibrillator (NIK/LWS MRM)	293	14,281	11,028
Physiological patient monitor/ arrhythmia detector or alarm (MHX/DSI)	4	79	276

Table 2: Defect taxonomies

Defect Taxonomies	Developed By
Orthogonal defect classification (ODC) scheme	IBM
Hewlett-Packard (HP) defect categorisation scheme	Hewlett-Packard company
IEEE standard classification for software anomalies	IEEE standard community
Taxonomy for software bugs	Boris Beizer

Table 3: SW91 defect categories

Parent level defect categories
Planning (1)
Requirements (2)
Architecture and Design (3)
Implementation (4)
Test (5)
Release Defects (6)
Maintenance (7)

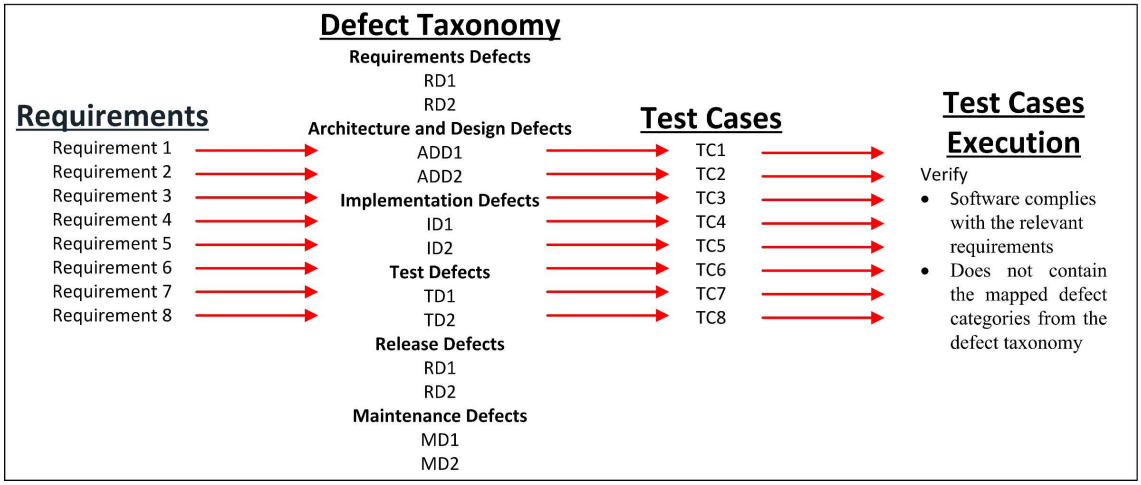

Figure 1: Taxonomy Based Testing

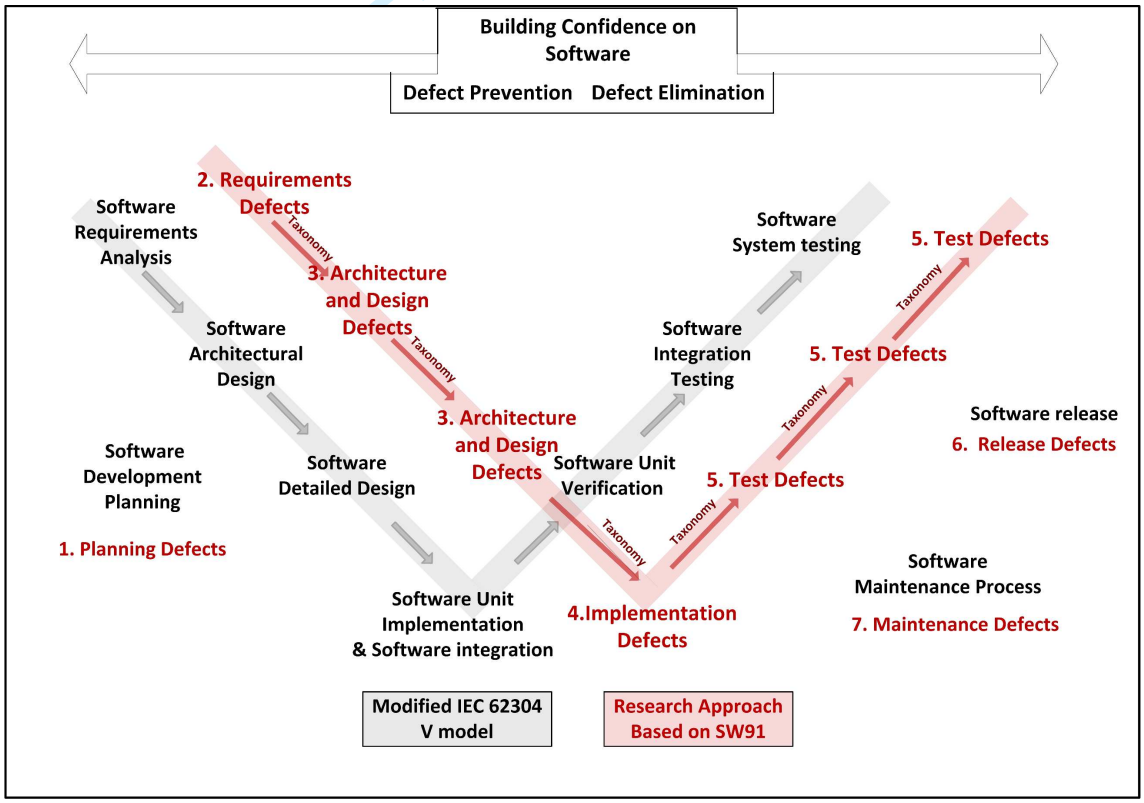

Figure 2: Modified IEC 62304 V model and taxonomy based testing

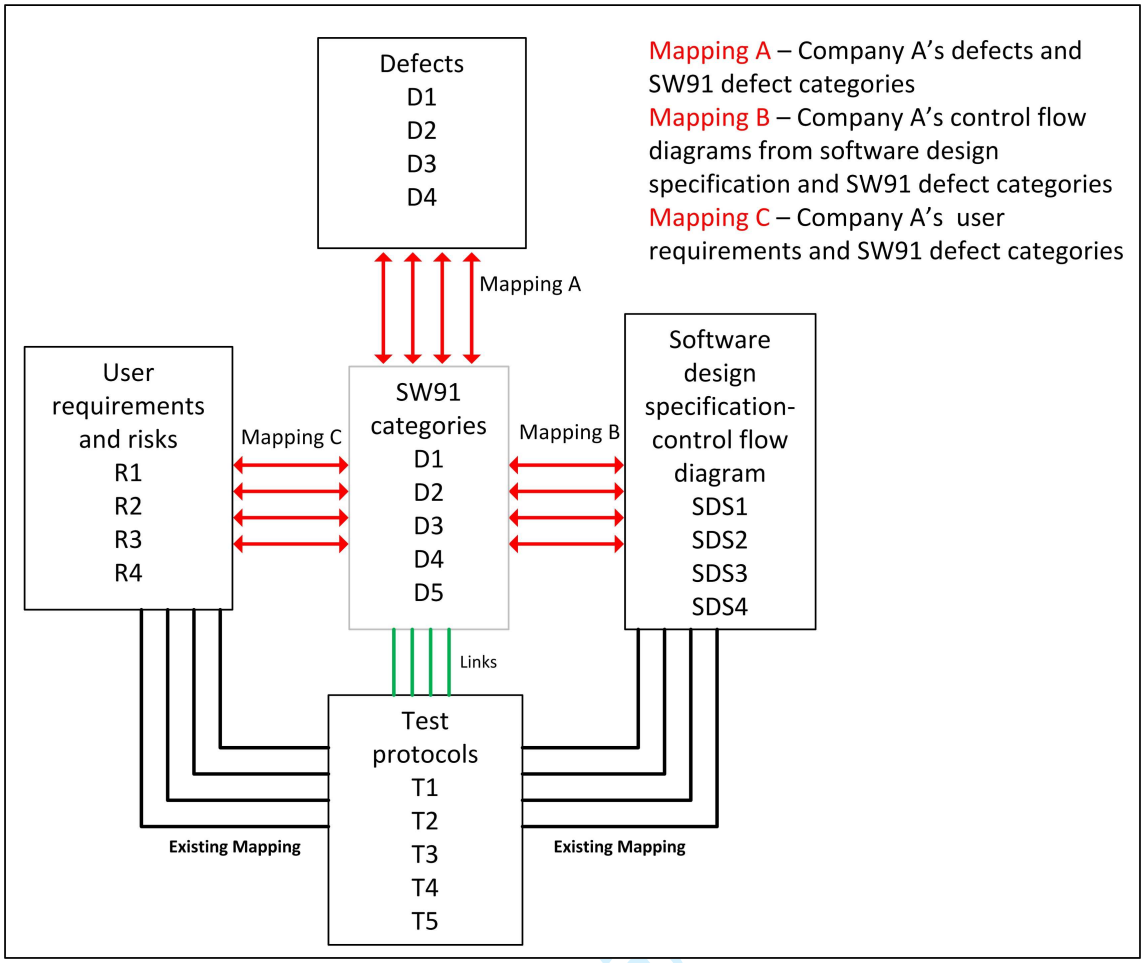

Figure 3: Mapping company A's data into SW91

CWE		SW91	
Vulnerability name	**Phases**	**Mapped categories**	**Phases**
642: External Control of Critical State Data The software stores security-critical state information about its users, or the software itself, in a location that is accessible to unauthorized actors	Architecture and Design Implementation	Security (3.8) Private Data Declared Public (5.3.1.6.2) Failure to Protect (5.3.2.3.3)	Architecture, Implementation

Figure 4: CWE Mapping[2]

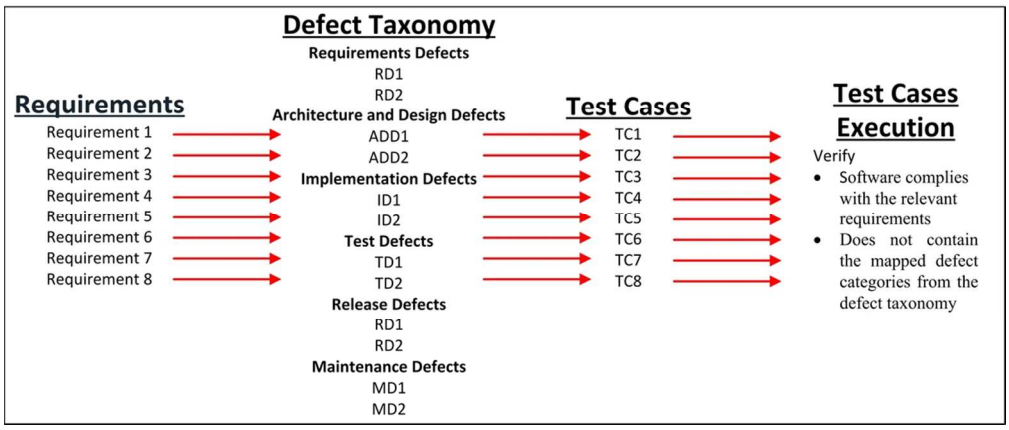

Figure 1: Taxonomy Based Testing

102x43mm (300 x 300 DPI)

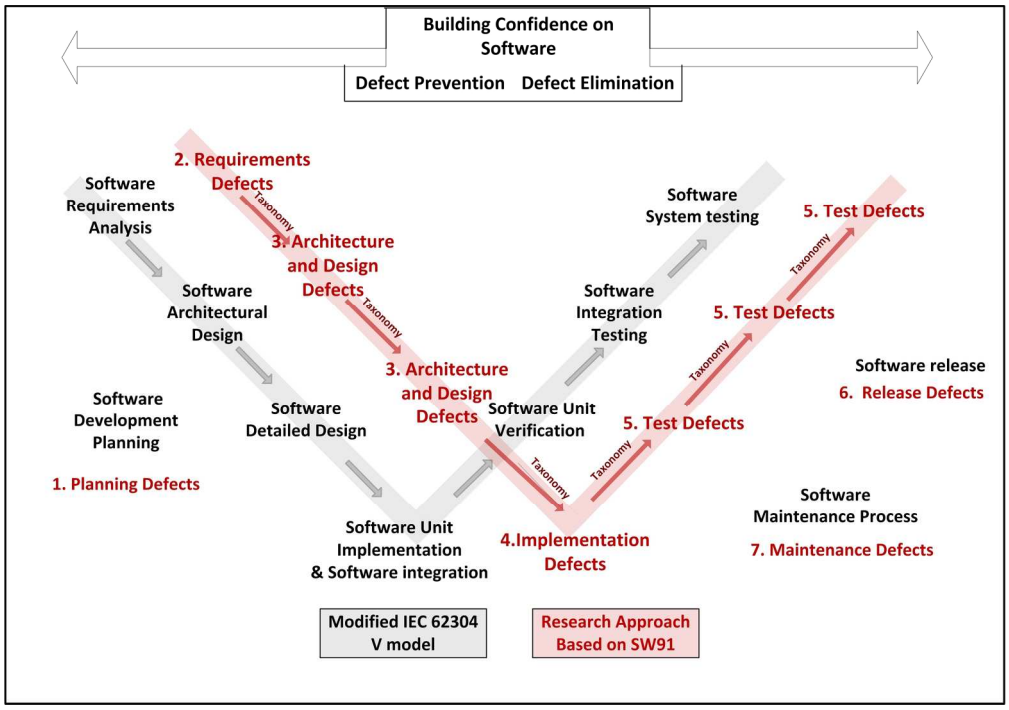

Figure 2: Modified IEC 62304 V model and taxonomy based testing

185x129mm (300 x 300 DPI)

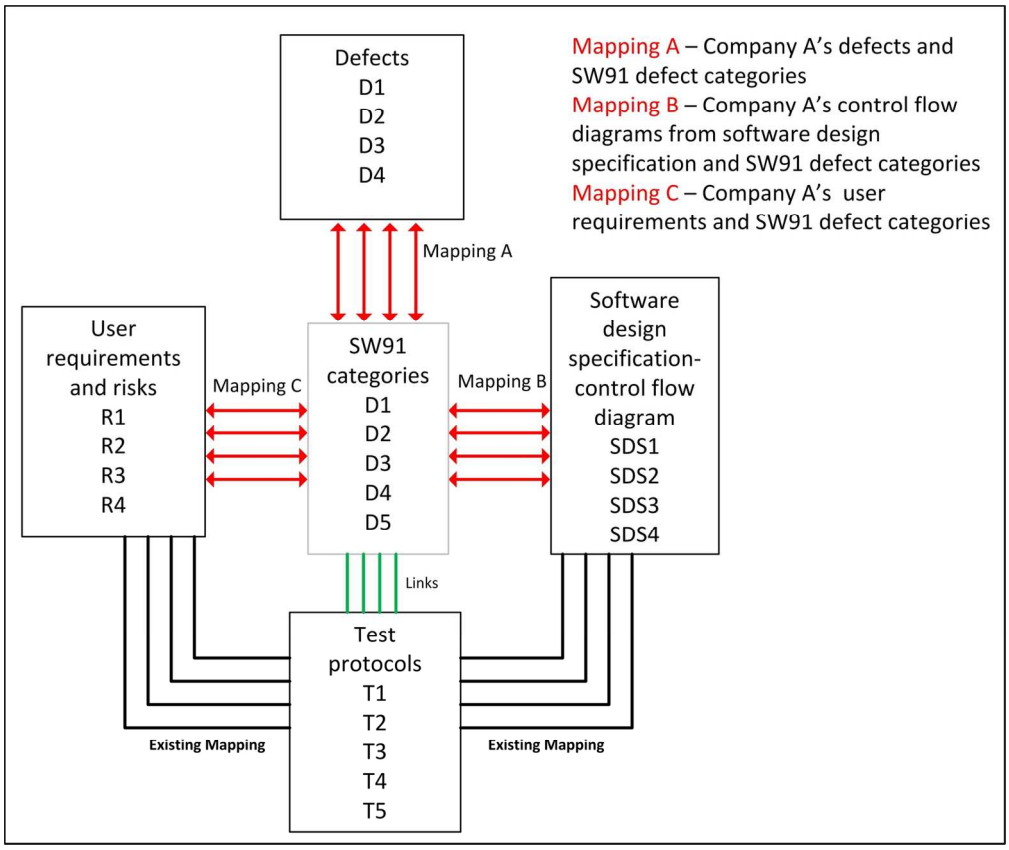

Figure 3: Mapping company A's data into SW91

145x121mm (300 x 300 DPI)

CWE		SW91	
Vulnerability name	**Phases**	**Mapped categories**	**Phases**
642: External Control of Critical State Data The software stores security-critical state information about its users, or the software itself, in a location that is accessible to unauthorized actors	Architecture and Design Implementation	Security (3.8) Private Data Declared Public (5.3.1.6.2) Failure to Protect (5.3.2.3.3)	Architecture, Implementation

Figure 4: CWE Mapping

444x118mm (300 x 300 DPI)