A Framework to Evaluate Defect Taxonomies

Diego Vallespir[1], Fernanda Grazioli[1], and Juliana Herbert[2]

[1] Instituto de Computación, Facultad de Ingeniería
Universidad de la República, Montevideo, Uruguay.
[2] Herbert Consulting
Porto Alegre, RS, Brazil.
dvallesp@fing.edu.uy , fgrazioli@adinet.com.uy,
juliana@herbertconsulting.com

Abstract. This paper presents a framework for evaluate and compare different defect taxonomies. Six well-known taxonomies are evaluated with it and the results are showed. We found deficiencies in every taxonomy once they are evaluated with the framework.

Key words: Software engineering, Software verification and validation, Defect taxonomies

1 Introduction

There are different reasons for using defect taxonomies. In *software development* they can be used to know which defect types are normally injected, improving both the verification activity and the software development process. Defect taxonomies have been used in many ways in *empiric software engineering* (ESE).

Knowing what defect types are injected, allows to look for them in a particular way. So, the verification activity could take less time and find more defects. For example, if 90% of the defects are in the requirements specification, it is possible to thoroughly verify this specification and reduce the time used in verifying other software products. In other words, to guide the verification considering the knowledge about the defects that are injected.

Also, the classification of defects gives an idea of the phases, activities, disciplines, etc. of the development process where most of the defects are injected. With this data the software development process can be improved by reducing the quantity of defects injected during the development.

For example, in ESE the defect taxonomies have been used to study the impact that different verification techniques have on different defect types. The goal is to know if different verification techniques find different defect types. By knowing this it can be possible to optimize the combination of different verification techniques in order to maximize the number of defects found.

Not only these reasons but many others that have not been exposed here show the importance of defect taxonomies in software development and in ESE. Unfortunately, there is not an universally used taxonomy, neither in software

development nor in ESE. This situation causes a lot of problems, for example, it is difficult or even impossible to compare some results between researchers.

A wide variety of taxonomies exists on the industry and in the literature. It is important to compare them from different points of view, trying to identify strengths and weaknesses of each one.

In this paper we analyze the most important taxonomies we found: Hewlett-Packard Taxonomy [1], Kaner, Falk and Nguyen's Taxonomy [2], Robert Binder's Taxonomy [3], IEEE Standard Classification for Software Anomalies [4], Orthogonal Defect Classification [5] and Beizer's Taxonomy [6]. Due to space reasons the taxonomies are not presented so we must assume they are known.

The paper is organized as follows. Section 2 presents a meta-taxonomy. A framework for the taxonomies comparison is presented in section 3. The taxonomies' comparison is presented in section 4. Section 5 presents the conclusions and the future work.

2 Meta Taxonomy

There is a lack of literature on meta taxonomies. We developed our comparison framework based on a previous work of Freimut [7]. From section 2.1 to 2.3 the Freimut proposal is presented. This proposal is divided in different aspects: attributes[3], structure types and properties.

Section 2.4 presents a taxonomy classification proposed in [8].

2.1 Attributes

A taxonomy is composed of attributes. Each attribute has a set of possible values. The values represent defect characteristics that must be registered at the classification moment.

The attributes to be considered must be those that are relevant to a future analysis. The attributes proposed by Freimut are listed below.

Location It refers to where the defect is located. The amount of detail in the specification may vary either indicating the system's high level entity on which the defect is found (Design, Code) or indicating the name of the software product it belongs to.

Timing It refers to the project's phase in which the injection, detection and/or correction of the defect is produced.

Symptom It refers to what it is observed when a failure is generated or to a description of the activity that is causing the failure.

End Result It refers to the description of the failure caused by the defect. This attribute usually contains values such as performance and usability among others.

Mechanism It refers to how the defect was injected, detected and/or corrected.

[3] In the original they are called *key elements*

Cause (Error) It refers to the human error that caused the injection of the defect (communication, lack of time and others).

Severity It refers to how serious the defect or failure is.

Cost It refers to the time and effort devoted to finding and/or correcting a defect.

Sometimes, the difference between symptom and end result is not clear. Symptom refers to what it is observed, maybe without knowing what is really happening, when a failure occurs. On the other hand, end result refers to what it really happens when the failure occurs.

For example, a simple error message could be a symptom of a failure. Once the defect (that causes the failure) is found it can be known the real extent of the effects of the failure (end result). For example, the database is corrupted and a error message is shown (the same one as in the symptom).

2.2 Structure Types

The existing relationships between the attributes in a taxonomy determine its structure type.

One of the possible structures is a hierarchy. The values of an attribute in a certain level are refined by values of attributes in the next level of the hierarchy. An example is Beizer's taxonomy [6].

A tree is another structure in which the attributes are not independent. The choice of a value in an attribute determines the possible values of the following attributes.

Another possible structure is the orthogonal in which values of each attribute are assigned independently of the other attributes. An example of this is the ODC taxonomy [5].

Another structure is the semi-orthogonal. There is no dependence among some attributes while among others the choice of a value determines the possible values of another attribute. The HP taxonomy [1] is an example of this.

2.3 Properties

If a taxonomy is not properly defined, problems may arise in order to get correct, reliable and good quality results. Therefore, the analysis of the results may be questionable. To avoid these problems, it is desired that taxonomies include the following requirements and properties.

Mutually exclusive attribute values When choosing the value of an attribute, only one value is appropriate.

Orthogonal attributes As it was previously explained, orthogonality refers to the independence between attributes.

Complete attribute values The set of values must be complete for each attribute. This means that when classifying an attribute for a given defect it is possible to choose an appropriate value for it.

Small number of attribute values The set of possible values for each attribute should not be very big. Having a small set of values not only makes the classification process simpler but also makes it less probable of mistakes. However, if the taxonomy is meant for a thorough analysis, it is possible that a bigger set of values may be needed in order to achieve an adequate precision in the results.

Description of attribute values It is important that all the possible values are defined clearly and stated with examples of defects that classified under that value.

2.4 Taxonomy Classification

In [8], it is presented a classification of what they call defect classification systems. These are divided in three categories: defect taxonomy, root cause analysis and defect classification.

The defect taxonomies are defects types categorizations. An example is the Beizer's taxonomy.

In root cause analysis not only defects are analyzed, but also their cause. The main goal is to identify the roots of these causes and eliminate them in order to avoid more defects. This approach is considered rather elaborated, so the cost/benefit relation is not clear. There are proposals for this in [9] and [10].

The defect classification uses multiple dimensions to classify defects. The goal is to reduce the costs and maintain the benefits of root cause analysis. Examples of this are ODC, HP and IEEE taxonomies.

In this paper we use the term taxonomy for any of these classification systems.

3 Comparison Framework

In this section it is presented our comparison framework. The best way to evaluate taxonomies in a correct and coherent way is to build a framework in which the characteristics, strengths and deficiencies of each taxonomy can be objectively and evenly evaluated.

Freimut also made an interesting comparison since he evaluates the attributes of various taxonomies, including HP, IEEE and ODC. However, as taxonomies can also be evaluated by their properties it is not enough to achieve our goal. Besides, this comparison does not include all the taxonomies evaluated here nor is completely coherent with its definitions.

In order to reach the objective of carrying out a complete and correct theoretical balance, we propose the development of a comparison framework. The framework will consist of two views: Attribute and Property.

The **Attribute view** takes Freimut's proposal as the main idea, which is modified in the framework by the addition of more specific attributes. The target of this view is to evaluate the capacity of a taxonomy to describe defects. The description of each attribute of the framework is as follows. The original name

of those which derive from Freimut's proposal appears between brackets at the end of the description. Otherwise appears "new".

Suspected location It refers to the place where the defect is suspected to be. (Location)
Real location It refers to the place where the defect really is. (Location)
Failure's causing phase It refers to the project phase in which occurs the failure. (Timing)
Injection's phase It refers to the project phase in which the defect is injected. (Timing)
Symptom It refers to what is observed when the failure occurs. (Symptom)
Type It refers to the defect type that was detected. (New)
Failure's causing mechanism It refers to the activity that drives to the failure. (Mechanism)
Injection's mechanism It refers to how the defect was injected. (Mechanism)
Correction's mechanism It refers to the activity that corrected the defect. (Mechanism)
Failure's impact It refers to the impact the failure has on the product when the failure occurs. (End result)
Defect's impact It refers to the impact the defect has on the project. (End result)
Cause (Error) It refers to the human error that caused the defect's injection. (Cause)
Responsible source It refers to where the product (in which the defect was injected) was developed (In house, external, etc.). (New)
Re-correction It refers to whether or not the defect was introduced when fixing a previous defect. (New)
Occurrence's Probability It refers to the estimated odds in favour of the failure to occur. (New)
Severity It refers to how serious the failure is. (Severity)
Priority It refers to how fast the defect has to be corrected. (New)
Detection's cost It refers to the time consumed in the detection of the defect after the failure appeared. (Cost)
Estimated cost of correction It refers to the estimated time its correction will take. (Cost)
Real cost of correction It refers to the time consumed in the defect's correction. (Cost)

The **Property view**, is based on the desired properties proposed by Freimut including as well other properties that we believe are necessary and useful. Below are described the properties that have not been previously presented and the possible set of values for each one.

Quality of attribute value's description It refers to expressing what level of quality the descriptions of the attributes' values have. For this property there are no possible values defined, therefore the existing quality level for each taxonomy must be expressed.

Quality of attribute value's examples It refers to expressing what level of quality the examples of attributes' values have. The possible values for this property would be "Good", "Bad" or "No examples contained".

Classification It refers to the way of classifying taxonomies according to [8]. The possible values for this property would be "Defect taxonomy", "Root cause analysis" or "Defect classification".

Structure It refers to the structure of the taxonomy. The possible values for this property would be "Hierarchical", "Tree", "Orthogonal" or "Semi-Orthogonal".

Generality's level It refers to the capacity of a taxonomy to be applied in different software projects or processes. The possible values for this property would be "Specific" (can be applied only for particular software), "Intermediate" (can be applied only in a phase of software development) or "Generic" (can be applied in any phase of the development process).

Adaptability It refers to the capacity of expansion and modification of the taxonomy depending on the requirements. The possible values for this property would be "Adaptable", "Adaptable with terms" or "Non adaptable".

Learning time It refers to the time it takes to learn how to use certain taxonomy. The possible values for this property would be "High", "Medium" or "Low".

Classification time It refers to the time it takes to classify a defect in a taxonomy after knowing how to use it. The possible values for this property would be "High", "Medium" or "Low".

The properties proposed in section 2.3 and the possible set of values for each one are listed below.

Mutually exclusive attribute values "Yes" or "No" are the possible values for this property.

Orthogonal attributes The possible values for this property would be "Yes", "No" and "Does not apply". The last value applies only when the taxonomy is composed by one attribute.

Complete attribute values The possible values for this property would be "Yes", "Aparent" and "No".

4 Comparison

The way the attribute view should be used is as follows: for each attribute in the view, look for a corresponding attribute in the taxonomy under evaluation. Table 1 shows the evaluation of the six taxonomies against the framework's attribute view. It is observed that the only attribute included in all taxonomies, is the defect type.

Binders, Kaner, Beizer an HP classify the defect only when it has been detected. ODC and IEEE carry out the classification once the failure occurs and also after the defect is detected. IEEE also carries out the classification once the defect is corrected and when the tracking of the defect is finished.

Table 1. Taxonomies' comparison against the Attribute view

Attribute	**HP**	**Kaner**	**Binder**	**IEEE**	**ODC**	**Beizer**
Suspected location				Suspected Cause		
Real location	Origin			Source, Actual Cause	Target	
Failure's causing phase				Project Phase		
Injection's phase						
Symptom				Symptom, Product Status		
Type	Type	Type	Type	Type	Type	Type
Failure's causing mechanism				Project Activity	Trigger, Activity	
Injection's mechanism	Mode		Type		Qualifier	
Correction's mechanism				Corrective Action, Resolution		
Failure's impact				Customer value, Mission/Safety	Impact	
Defect's impact				Project Cost, Project Quality/Reliability, Project Risk, Societal		
Cause (Error)						
Responsible source					Source, Age	
Re-correction					Age	
Occurrence's Probability				Repeatability		
Severity				Severity		
Priority				Priority		
Detection's cost						
Estimated cost of correction				Project Schedule		
Real cost of correction						

We can observe that there are attributes of the framework that are not contemplated in any of the taxonomies. The following presents each of them and the reasons for the addition in the framework.

Injection phase It is interesting to register the phase in which the defect is injected because that information can be used to prevent the injection from similar defects in future projects.

Cause (Error) It is interesting to register which was the human error that led to the defect. By being aware of the problem, defects in future projects can be avoided. Some results of root cause analysis studies give possible values for this attribute. For example, in [9], a Cause attribute is proposed with the values Education, Communication and Tools. In [10] an attribute that captures different types of causes is proposed: Human causes, Project causes, Revision Causes.

Detection's cost It is interesting because a costs' analysis could be made depending on the different defect types. This allows to estimate changes in the project's schedule.

Real cost of correction Such as the detection's cost, its registration is interesting for a future analysis of costs.

Table 2 presents the comparison of the six taxonomies against the framework's property view.

In HP, Kaner, IEEE and ODC taxonomies the values for an attribute are mutually exclusive. However, in Binder and Beizer's taxonomies, the set of values for an attribute does not present this property, which can generate inconsistent data since a defect could be classified in different ways.

Observing the orthogonality of attributes, this property is not classified by Kaner, Binder or Beizer's taxonomies because these constituted of only one attribute. For IEEE and ODC, this property is fulfilled since the value of each attribute is completely independent of the values of the other attributes. However, for HP this does not apply because the values of the Type attribute depends on the value chosen for the Origin attribute.

The property of having a complete set of attribute values is a difficult one to demonstrate. The Beizer is the only taxonomy that fulfills this property due to a special value that means "others". This value is used when a defect characteristic does not correspond to any of the other values.

Regarding the quality of the descriptions and examples of the values of an attribute, a variation among the different taxonomies is observed. Anyhow, the HP, IEEE and ODC taxonomies can be grouped under a level of good quality since they present complete and clear descriptions, also they include concise examples. On the other hand, Kaner, Binder and Beizer's taxonomies can be grouped under a level of bad quality given their attributes are ambiguous or incomplete and they do not have descriptions or examples.

Regarding the learning time, the taxonomies that are considered easily understandable were grouped with "Low", either because of the clarity and no ambiguity of its presentation, and/or because the taxonomies have a small amount of attributes. The taxonomies presented by Binder and Beizer were tagged with a "High" value. Although they are composed by only one attribute, many possible values exist for it. This, added to vague descriptions and examples of poor quality, result in a longer learning time than the other group. The taxonomy presented by IEEE is also considered to have "High" learning time because it contains such a large amount of attributes that they have to be studied and understood.

Table 2. Taxonomies' comparisons against the Property view

Property	**HP**	**Kaner**	**Binder**	**IEEE**	**ODC**	**Beizer**
Mutually exclusive attribute values	Yes	Yes	No	Yes	Yes	No
Orthogonal attributes	No	Does not apply	Does not apply	Yes	Yes	Does not apply
Complete attribute values	Aparent	Aparent	Aparent	Aparent	Aparent	Yes
Quality of attribute value's description	Good: clear descriptions, no ambiguities	Bad: not all values are explained	Values are not described	Good: clear descriptions, no ambiguities	Good: clear descriptions, no ambiguities	Bad: superficial description, ambiguous in some cases
Quality of attribute value's examples	Good	No examples contained	No examples contained	Good	Good	Bad
Classification	Defect classification	Defect Taxonomy	Defect Taxonomy	Defect classification	Defect classification	Defect Taxonomy
Structure	Semi-Orthogonal	Does not apply	Does not apply	Orthogonal and Hierarchical	Orthogonal	Hierarchical
Generality's level	Generic	Generic	Specific	Generic	Generic	Generic
Adaptability	Adaptable	Adaptable	Adaptable	Adaptable	Adaptable	Adaptable
Learning time	Low	Low	High	High	Low	High
Classification time	Low	Low	High	High	Low	High

The proposed reasoning for the time of classification is analogue. Those taxonomies, with no ambiguous descriptions, good examples and/or few attributes to classify, are grouped under the value "Low". The taxonomies with higher amount of attributes as well as those that are not clearly defined or exemplified, take a longer time for the user to classify because of a lack of certainty among two or more values, are considered to have a "High" time of classification.

It is clear that classifying time as "High", "Medium" or "Low" is subjective, with the connotations that it implies. In the future, an average of the times of learning and classification could be calculated for each taxonomy, and then be able to formulate a better comparison, more representative of the reality.

It is observed that the amount of attributes, the quality of the descriptions' values of the attributes and the quality of the examples presented in each taxonomy; significantly influence both the learning and the classification times of the taxonomy.

5 Conclusions

We developed and presented an original comparison framework for defect taxonomies. This framework extends and improves Freimut ideas. It contains two relevant views: attribute and property. The first view is useful to evaluate those defects' characteristics that are considered for a given taxonomy. The second view is useful to evaluate desired properties each taxonomy should have.

Using the framework we evaluate and compare six well-known taxonomies. The results show that each taxonomy considers different characteristics of a defect. Some of them are more complete, from an attribute point of view, than others. However, those less complete have a bigger set of values for their attributes, for example, Beizer and Binder. The analysis also shows that there are differences in the properties among the taxonomies.

As a future work it is important to: analyze the impact of whether having or not a specific framework's attribute in a taxonomy, analyze the values proposed in each taxonomy for each attribute, analyze if different test levels (unit, integration, system) require different taxonomies and finally analyze the way taxonomies have been used in the industry. We are currently working on these last two points.

References

1. Grady, R.B.: Practical Software Metrics For Project Management and Process Improvement. Hewlett-Packard (1992)
2. Kaner, C., Falk, J., Nguyen, H.Q.: Testing Computer Software (2nd. Edition). International Thomson Computer Press (1999)
3. Binder, R.V.: Testing Object-oriented Systems Models, Patterns, and Tools. Addison-Wesley (1999)
4. IEEE: IEEE 1044-1993 Standard Classification for Software Anomalies. Institute of Electrical and Electronics Engineers (1993)
5. Chillarege, R.: Handbook of Software Reliability Engineering. IEEE Computer Society Press, McGraw-Hill Book Company (1996)
6. Beizer, B.: Software Testing Techniques, Second Edition. Van Nostrand Reinhold Co. (1990)
7. Freimut, B.: Developing and using defect classification schemes. Technical report, Fraunhofer IESE (2001)
8. Wagner, S.: Defect Classifications and Defect Types Revisited. Technische Universitt Mnchen (2008)
9. Mays, R.G., Jones, C.L., Holloway, G.J., Studinski, D.: Experiences with defect prevention. IBM SYSTEMS JOURNAL (1990)
10. Leszak, M., Perry, D.E., Stoll, D.: A case study in root cause defect analysis. Proceedings of the 22nd international conference on Software engineering (2000)