# Classifying defects in software maintenance to support decisions using hierarchical ODC

**.Marcelo M. Manhães, Maria Claudia P. Emer, Laudelino C. Bastos**

Departamento de Informática – Universidade Técnica Federal do Paraná (UTFPR)
Caixa Postal 80230-901 – Curitiba– PR – Brazil

marcelo.manhaes@hotmail.com, mclaudia@dainf.ct.utfpr.edu.br,
bastos@dainf.ct.utfpr.edu.br

***Abstract.*** *Software maintenance is a critical part in software life cycle where fast production deployment and effectiveness for high quality software are pressure factors. For cloud and regular Service Providers such as out sourcing enterprises it is a key factor of success due to service level agreements closed to customer needs. Although, more high level tools of system monitoring are available into market to increase high availability, a lot of software packages go to service providers with fatal bugs. The effectiveness in terms of service components failure identification and priorities definition focused on incidents information are few used. This paper presents a method to extend orthogonal defect classification (ODC) in terms of triggers and sources attributes in a hierarchical way to improve technical support decisions in service providers. This approach classifies better errors on this phase using service components in software maintenance from incidents history. The ODC original triggers and sources are decomposed in values closely related to software maintenance service components without change the orthogonality essence of original ODC using a drill-down feature. These attributes are used with ODC defect type and ODC impact in a multi-dimensional modeling. This method reaches better service errors identification, remove it earlier than standard methods and better process feedback to manage service provider decisions. The method suggests an improvement in service providers functionality based on experiments.*

## 1. Introduction

In service providers today, there are several cases where customer software is inserted in a selected infrastructure service, for instance, in a cloud environment. Discover where problem is found becomes very important to drive next release tests and directing problem to customer's software or infrastructure or still if the problem is clearly detected to customer code, service provider can offer a refactoring code service. It is a fundamental point where software changes have huge frequency to follow time to market.

Quality and cost are key points to guarantee success of service providers where rely on factors such as component and board quality, efficient application deployments, high availability, a good support of system diagnostics and behind these factors a process that controls quality assured, resource and costs [Toai et al. 2006]. To achieve

these key performance indicators a correct and functional classification of errors improve a correct prioritization and hence a faster resolution. The information about errors would be valuable to classify and prioritize service components in software maintenance or when software goes to production.

In classification of defects field several studies were described using flat classification as described in [Knuth 1989] or hierarchical model in [Beizer and Vinter 2001]. In [Gray 1986] concept of activations in software production failures were introduced based on software tested and used for a long time defining type of bugs in two categories transients (easily reproduced) and no-transients (not easily reproduced).

[Chillarege et Al., 1992] introduced ODC (Orthogonal Defect Classification) that turned a rich and multi-faceted technology that provides many benefits. One of them is the prevention of defects as decreasing the number of defects being injected into design and code [IBM 2013]. There were some works related to improve some test phases where ODC was extended to accomplish this improvement as described in [Li et al. 2010] for black box testing and for code inspection as described in [Kelly 2000].

Defect classification varies according to different purposes, but on this paper the goal is presents an ODC framework to software maintenance, purposing an extension of ODC classification for it called ODC-SC (ODC Service Components). This method is based on fact that in production there is a huge volume transaction. Moreover software maintenance is an important slice in software development process with high costs related. All classification is extracted from incident tickets and applied techniques of multi-dimensional modeling and drill-down in some ODC attributes.

Classification and prioritization are key points to follow continuous and high volume change process where service providers need to provide an efficient defect prevention process in order to follow a huge demand for business agility. Also it is fundamental to provide good measures about production environment and if a problem is from customer code or if is from infrastructure provisioning and how to improve for deployments.

The main purpose of ODC-SC method is described as follows: 1) provides service providers managements and outsourcing enterprises controls with a comprehensive analysis method to define if errors are infra-structure based or from customer data or code based. 2) Provides infra-structure prioritization of defects in order to inspect defects types for individual components or still to prioritize class of problems during reviews. 3) Help new deployments to improve high availability in production.

This paper is organized as follows. Section 2 presents a discussion on related works. Section 3 proposes a method to increments ODC triggers and ODC source values for software maintenance using a service components concept. Section 4 provides an experiment to validate of method ODC-SC from software maintenance records. Discussions about some decisions are explained in section 5. Some conclusive remarks are given in section 6.

## 2. Background and related work

It was done a bibliographic review around 11 years (2002 to 2013) for journals on reference sites Science Direct (http://www.sciencedirect.com/), ACM Library

(http://dl.acm.org/) and IEEE Xplore (http://ieeexplore.ieee.org) for similar or related works. From initial articles, past studies were gathered before the initial time frame. In classification of defects field, several studies were described. These studies in literature start by one of the initial materials on this field in [Knuth 1989] where it was designed from experiences in projects of medium size about an evolution of a typesetting system using a flat categorization mixing faults with enhancements. Studies described in [Beizer and Vinter 2001] provided a comprehensive schema for software faults categorization using a hierarchical model with 10 initial categories and 100 leaf categories in software development process. These initial works contributed on software fault categorization, however, provided a model with high difficulties to classify due to ambiguities in several software faults becoming difficult to use.

Other studies produced another approach focused in report of fails (activations) that affected availability and reliability when software was in production [Gray 1986]. The state production was considered when software has been tested and used for long periods of time. On referred research it was introduced concepts of **Heinsenbugs** or no deterministic faults where it was related to fails not reproduced with repeated execution and **Bohrbugs** or deterministic faults where bugs are reproduced easily. This study concluded most of production software faults were transients. However software introduced in production was not considered on related study.

The term ODC was first described in [Chillarege et Al., 1992] with a proposal to classify defects to improve software development life cycle process. The main idea would be extract semantic information from defects. From defect information it was extracted a relationship between cause and effect. Still in [Chillarege et Al., 1992] the natural extension of ODC is the defect prevention process where fits very well with defect data collected from stability or system tests where provides most accurate information requirement for the software quality assessment and if is classified well can focusing in priorities and also in defect fix effectiveness.

Related works as described in [Li et al. 2010] shows a method called ODC-BD that is a black box testing using ODC. On this solution, ODC is totally customized to black-box expanding the defect attributes and describing specific defects and therefore improving effectiveness as research says. In other article, in code level scope, it was proposed a new defect classification from original ODC that was called ODC-CC (Orthogonal Defect Classification Computational Codes) to be used for code level defects [Kelly 2000].

Similar approaches from this research are described in [Podgurski et al. 2003] where fails that are caused from same defect and have a considerable number of occurrences can be elected to defect prevention according to fail criticality. The method used is a manual clustering to define similar fails in a bi-dimensional space where the points of failures are positioned. The similarities of fails are measured from the distance between then according specific criteria. Group of failures can be identified from a graphic visual inspection. The users can judge what fails are more related instead of believe in automatic clustering method. A disadvantage of this model is that only 2 dimensions are exposed and little differences between failures are poor presented. ODC can improve this approach because it is a multi-dimension classification of defects.

Recent related studies in other areas using ODC was found such as garment industry it can be highlighted a quality management system using an Online Analytical Processing and mining association rules. This system extracts garment defects through hidden patterns in order to work on defect prediction, root cause identification and proactive actions to improve quality. The quality improvements are reached by fix what steps on production process raise more failures [Lee et al. 2013].

Based on this state of the art cited above, with few studies focused for software maintenance with respect to classification and analysis of defects, but a lot of studies related to ODC, the ultimate goal of this paper is to optimize the ODC defect classification for software maintenance to allow support decisions to define what the scope of problems and defect prioritization to improve maintenance process.

## 3. Classification schema for software maintenance

The method used on this research keeps the original structure of ODC and componentize ODC triggers and ODC sources. This will change initial flat characteristic of ODC putting a hierarchical mode. Also it can be considered a drill down method of ODC triggers and ODC sources. The information about what attributes make in hierarchical mode were motivated from incident information gathered from several customers supported in Enterprise “A”.

The Enterprise “A” is a big player in software outsourcing where manages hardware and software located in other enterprises such as banks, cred card holdings, electronic enterprises, retail business and so for. If is counted only middleware products supported from a local team, it reaches around 20 customers. The support data is shown in a central tool to control all work tasks. The available defect´s data history shows a high difficulty to avoid ambiguity in report issues and also a difficulty in prevent issues where several products are involved such as databases, middleware, operational system and hardware types. Also the volume of incidents are high, around 2000 by month if counts all middleware products from customers where Enterprise “A” manages. The hardware and software sets have high complexity configurations and they are mixed in multiple combinations and versions as customer environment is completely third part managed.

Before the deeper classification definition, it will be defined a big ODC picture regarding a defect report. To define a defect report from an incident, there are four main attributes that it will focus in ODC as described below (Figure 1) [Chillarege 2006].

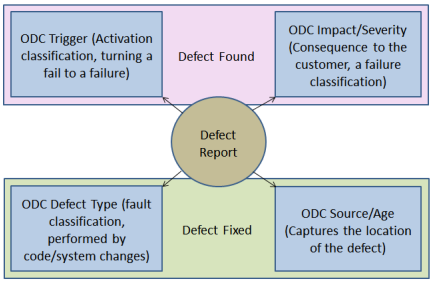


**Figure 1. A defect report formed by ODC trigger and ODC impact when open a defect and ODC defect type and ODC Source/Age when close a defect [Chillarege 2006].**

ODC triggers are very important because can show a classification of defect activation [Chillarege 2011]. ODC defect type measures the software development process where defects are fixed regarding faults representing what correction was applied [IBM 2013]. ODC impact represents the failure classification to the customer view. ODC source indicates the location of defect [Chillarege 2006].

Based on these four main attributes in a multi-dimensional mode and based on data gathered from incidents in Enterprise "A". It can be defined a general mapping as described on Figure 2.

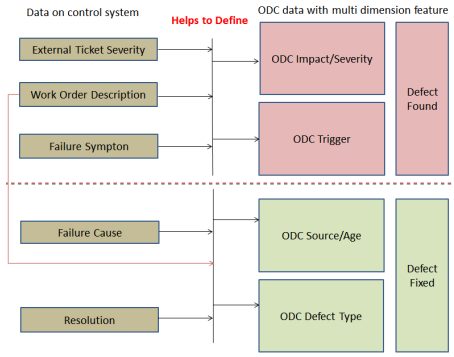


**Figure 2. Mapping from original support data and ODC attributes in a multi-dimensional feature.**

The initial mapping consists to map External Ticket Severity to ODC impact and Resolution to ODC Defect Type. Work Order Description and Failure Symptom map to ODC trigger and Impact. Finally, Work Order Description, Failure Cause and Resolution map to ODC Source. These mappings can be generalized to other service providers, considering that most of incident´s report has at least severity, description of incident (work order) and resolution. An important point to mention is that from the four ODC attributes is that ODC trigger and ODC source will have a drill-down feature. On the other hand, ODC impact is mapped from numbers (1 to 4) and ODC defect type are mapped directly from operational variables source data based on some criteria.

When we classify field defects, on drill-down feature, trigger is the first point, by selecting the trigger that best represents the environment or condition that exposed the defect in the customer's or service provider´s environment. The activity is selected based on the task associated originally to the trigger [IBM 2013].

Software maintenance was a base for this research but based on closely activity that is system test. These are the triggers in ODC scope [Chillarege and Ram 2002]:

- Workload Volume/Stress: it is related to the limits of performance, resources, users, queues, traffic and so for.
- Recovery/Exception: Invoke exception handling, recovery, termination, error percolation for instance.
- Startup/Restart: Relevant events of turning on, off or changing the degree of service availability.
- Hardware Configuration: Incidents surfaced as a consequence of changes in hardware setup.
- Software Configuration: Incidents surfaced as a consequence of changes to software setup.

- Blocked Test/Normal Mode: Customer found nonspecific trigger where test could not run during production deployment.

On software maintenance phase, the environment can be so complex and trigger mitigation is a fundamental point to find a problem as soon as possible and provide a deeper classification and mitigation of risk and focus on priorities. This demand is being a fundamental point to enterprise that provides software services such as web site hosts, public or hybrid cloud services where applications that are deployed on the service providers need to deliver a fast resolution for problems. Based on this demand triggers subtypes are spread around ODC triggers as described in Table 1.

**Table 1. The table is a proposal of trigger sub types for software maintenance and verification with original ODC triggers associated.**

| Trigger sub type | Related to what Original Trigger? | Description |
|---|---|---|
| Application | Software Configuration, Workload Stress, Recovery Exception, Startup/Restart | Application Code that belongs to customer or any third part code. It is related to any application code error, configuration of application on the system. |
| Security Infrastructure | Software Configuration, Workload /Stress, Recovery Exception, Startup/Restart | Errors related to certificates (hardware or software), firewalls, data access, Single Sign On (SSO), authorization and authentication. |
| CPU | Hardware Configuration, Workload /Stress | Triggers related to CPU problems caused by low capacity sizing or physical problems. |
| Memory | Hardware Configuration, Workload /Stress | Triggers related to Memory problems caused by low capacity sizing or physical problems. |
| Storage | Hardware Configuration, Workload /Stress | Problems closely to bad dimension of file system or some hardware problem. Application code can generate a lot of logs files raising File system full. |
| Other Hardware | Workload /Stress | Hardware that is not storage, network, print services or security infra. |
| Network | Hardware Configuration, Workload /Stress, Recovery Exception, Startup/Restart | Problems closely related to hardware firmware problem, network sizing or other that are not CPU, Memory, Storage, printing services for instance. |
| Printing | Software Configuration, Hardware Configuration, Workload /Stress, Recovery Exception, Startup/Restart | Printing problems related to hardware or software drivers involved. |

| | | |
|---|---|---|
| OS | Software Configuration, Workload /Stress, Recovery Exception, Startup/Restart | Problems related to incorrect configuration and patching (version update), slow response related to capacity, resource contention, file system settings, ulimits settings for UNIX . In short all OS settings. |
| Middleware | Software Configuration, Workload /Stress, Recovery Exception, Startup/Restart | All configuration problems related to these products such as application servers, main frames, message queue servers products, HTTP servers, Web servers and so for. |
| Database | Software Configuration, Workload /Stress, Recovery Exception, Startup/Restart | All problems related to database products itself or any data access configuration such as drivers. |
| Unknown Errors | Blocked Test/Normal mode | There is not enough information to define where the issue occurred such as when a migration is in place and new software version is not working. |

It is very useful for software maintenance to determinate what component in service provider is causing more problems regarding availability in production. This conclusion is based on incidents report from Enterprise “A”.

From Table1 described earlier it is provided a distribution of software and hardware components gathered from service providers in Figure 3 where is it shown the distribution along the ODC base system test triggers. Again this information is collected from incident reports on Enterprise “A” where shows the components involved.

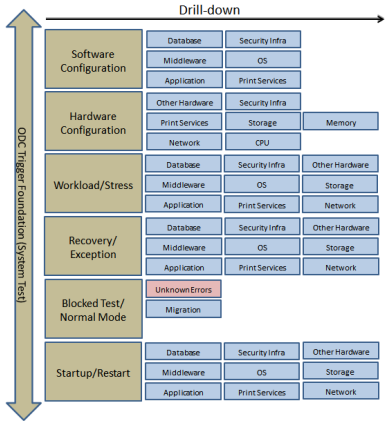


**Figure 3. Components in service providers distributed in foundation ODC triggers using a drill-down technique**

Another point where ODC attributes will be deeper on this work is related to ODC source that describes where defect has been found when fixed. It can be considered a mitigation of ODC triggers where describes the surface of fail.

It will be used the ODC foundation source values in the first level. In the second level the parameters will be imported from ODC triggers subtypes to form second level

of ODC sources as described in Figure 4. This approach guarantees the tight relationship with surface when defect is found with high level location when defect is fixed.

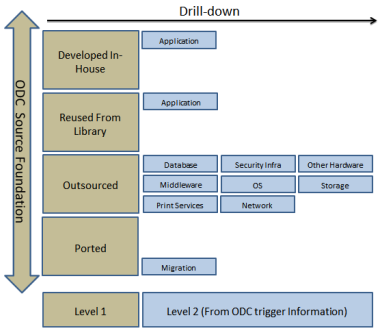


**Figure 4. ODC triggers being placed as second level for ODC source attribute**

In the third level these values are expanded in sub components related to products or configuration components and following same source data in Enterprise "A". This third level can be related to area where it needs to focus such as support for test service providers or software development. On this level describes in depth where it can be found the problem when was fixed. The information about what and how was fixed is described in ODC defect type. The Figure 5 explains the third level for ODC source. It can be use optionally depending on complexity of maintenance services. Despite third level into ODC source necessary or not, the important point is to position service components into ODC source foundation for instance, developed In-House and Reused From library (Customer) or Outsourced (Service Provider). This point can be very important to define responsibilities in a regular or cloud service provider.

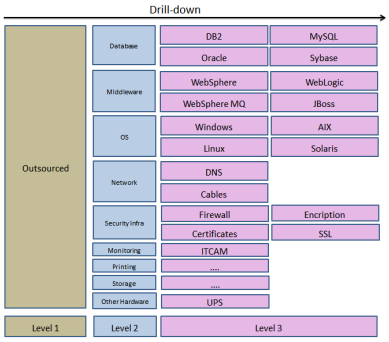


**Figure 5. Third level of ODC sources related to specific products or components**

## 4. Validation of ODC-SC

The method was applied from service work orders extracted from different teams such as database, middleware, network main frame teams and others in the outsourcing Enterprise "A". This data set was delivered in comma-separated values (CSV) format. Indeed, it was applied a manual classification at first phase to understand better business case. Using a set of 50 entries of work orders that could be considered a training set for future data mining purposes. From method it was defined four ODC attributes: trigger, impact, and source and defect type. Besides original ODC, it was applied service components (drill-down) into triggers and sources. With final data, using a simple graphic tool, such a Microsoft Excel, it shows the result as described on Figure 6.

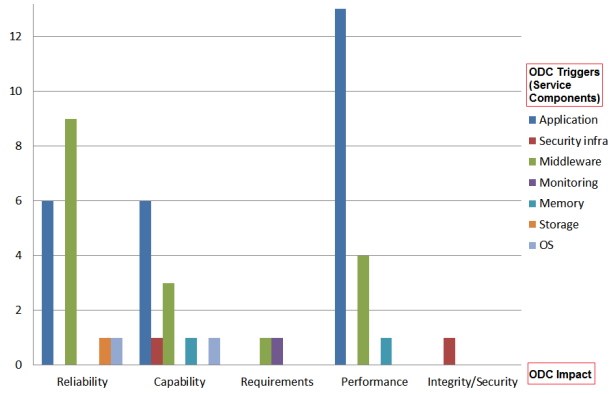


**Figure 6. The data classified in a graphic**

This two dimension graphic is extracted from tickets in Enterprise "A" and shows the distribution of errors using ODC triggers with service components feature, axis y. It was used a kind of analysis dice between ODC impact attribute and ODC trigger, with trigger service components drilled down on this experiment. These service components on axis y are distributed around axis x (Impact ODC attribute). The impact ODC represents what was impacted on customer environment. The conclusion on this chart suggests that application code is impacting of system in a considerable value ( Performance, Reliability and Capability). The customer code needs to be improved in performance. Reliability in original ODC represents when system is down by a general failure and performs high availability impact. Capability in ODC means that the system is being impacted partially in some functions. On the other hand, Middleware from service provider is impacting system reliability in high levels and needs to be improved.

## 5. Discussion

As defect is closed in software maintenance, service components are included on second level of ODC source making easier to discover defects in complex environment of service providers, increasing defect fix hate by time. With this information, a prioritization of defects can be executed from customer impact, for instance, defining fails surfaced by application code and middleware as maximum priority. The solution can be expanded for other service providers in software maintenance since incident data being accurate reported with description, impact and resolution data as minimum required. The third level on ODC source is optional but can improve defect prevention.

## 6. Conclusion and Future Work

This research works better ODC triggers and sources drill-down classification measures to improve effectiveness in software maintenance. Also works with ODC defect type and ODC impact directly in a multi-dimensional way. With this method it is easier to define priority of fixes and can be a guarantee that software will have a better quality assured using process feedback feature of ODC. Also with an ordinated and multi-dimensional data it is possible to fill weakness in service providers such as better resources provisioning and defect prevention for future deployments such as application code. The solution is expansible to regular or cloud service providers. In cloud providers for instance it have a high number of used could services (private or public) to be used

by customer. Decide if customer or service provider is problem cause is a key point to fix defect rates. About future work, the method can be improved to drill-down ODC impact in order to define what business applications are more important to fix for incidents and provides defect prevention prioritization work.

## 7. References

Beizer, Boris, Vinter, Otto (2001) “Bug taxonomy and statistics. Technical report”, Software Engineering Mentor, p. 2630.

Chillarege, R., (2011) "Understanding Bohr-Mandel Bugs through ODC Triggers and a Case Study with Empirical Estimations of Their Field Proportion," Software Aging and Rejuvenation, 2011 IEEE Third International Workshop on , vol., no., p.7-13

Chillarege, R., (2006) “ODC - a 10x for Root Cause Analysis”, Proceedings. RAM 2006 Workshop.

Chillarege, R., Ram Prasad, K., (2002) "Test and development process retrospective - a case study using ODC triggers," Dependable Systems and Networks. Proceedings. International Conference on, vol., no., p. 669-678.

Chillarege, R., Bhandari, I. S., Chaar J. K., Halliday, M. J., Moebus, D. S., Ray, B. K., Wong, M. (1992) “Orthogonal defect classification-a concept for in-process measurements”, IEEE Transactions on Software Engineer, EUA, vol. 18, p. 943-956.

Gray, J. (1986) “Why do computers stop and what can be done about it?”, In Proceedings of Symposium on Reliability in Distributed Software and Database Systems (SRDS-5). IEEE CS Press, vol. no., p. 3–12.

Kelly D., (2000) “An experiment to investigate a new software inspection technique”, Master’s Thesis, Royal Military College of Canada.

Knuth D. E. ,(1989) “The errors of TEX. Software–Practice and Experience”, V. 19, Issue 7, July 1989, p. 607-685.

Lee, C.K.H., Choy, K.L., Ho, G.T.S., Chin, K.S., Law, K.M.Y., Tse, Y.K. (2013) “A hybrid OLAP-association rule mining based quality management system for extracting defect patterns in the garment industry”, Expert Systems with Applications, vol. 40, Issue 7, 1 June, p. 2435-2446.

Li, N., Li, Z. and Sun, X. (2010) “Classification of Software Defect Detected by Black-box Testing: An Empirical Study”, Second WRI World Congress on Software Engineering. China, vol. 2, p. 234-240.

IBM research center for Software Engineering (2013) “Examples of the Use of ODC”, http://www.research.ibm.com/softeng/ODC/ODC.HTM, March.

Podgurski, A., Leon, D., Francis, P., Masri, W., Minch, M. (2003) "Automated Support for Classifying Software Failure Reports." In: Proceedings of the 25th International Conference on, vol., no., p. 465-475.

Toai, V., Zhiyuan W., Eaton, T., Ghosh, P., Huai L., Young L., Weili W., Rong F., Singletary, D., Xinli G., (2006) "Design for Board and System Level Structural Test and Diagnosis," Test Conference, ITC '06. IEEE International, vol., no., p.1,10, 22-27.