COMMONWEALTH OF PENNSYLVANIA
DEPARTMENT'S OF PUBLIC WELFARE, INSURANCE, AND AGING

INFORMATION TECHNOLOGY STANDARD

Name Of Standard: **Defect Management and Reporting**	Number: **STD-EASS007**
Domain: **Application Domain**	Category: **Standards**
Date Issued: **04/29/2010**	Issued By Direction Of:
Date Revised: **02/25/2014**	 Shirley A. Monroe, Dir of Div of Technical Engineering

Abstract:

Software applications are developed with varying degrees of complexities beginning with business requirements through development, testing and implementation into production environments. The business requirements, technical design logic and respective application layers are complex requiring specialty knowledge and skills to develop a software solution. These complexities may introduce defects throughout the end to end phases affecting the intended use of the application systems.

The purpose of this standard is to establish processes used by the Department of Public Welfare (DPW) to eliminate or minimize defects impacting the intended use of software application systems delivered to our customers.

Adherence to these policies is mandatory. When a DPW and/or contracted solution provider believe there is a need to deviate from these standards/policies, the agency and/or Offeror must first receive written approval to do so from the DPW, Chief Information Officer (CIO), Contract Administrator, or designated BIS Director.

General:

In support of our Bureau of Information Systems (BIS) mission to deliver value-added software systems solutions to effectively support DPW's Programs Offices business needs; BIS has established process a standard to manage and report software application systems defects.

Standard:

Defect Management and Reporting Standard:

1. Defect Definition: Defects are any errors that result in a **"failure to conform to specifications"** or a **"failure to function/operate properly"**.

2. Defining and qualifying defects must include the following parameters:

    a. **Category**: Defect categorized types: (e.g. SEP, Configuration, Hardware, Code, Product, Systems, Operations, Undetermined /Other)
    b. **Priority**: the relative importance of addressing and resolving this defect. Prioritize within category (e.g. 1- Mission Critical, 2 – Mission Essential (ok with work-around), 3 – Cosmetic)
    c. **Severity:** the relative impact of this defect to the end-user, organization, business partners, and citizens, etc.
    d. **Status:** the current state of the defect (i.e., open, closed, reopened etc.)

3. Defect Category Type Definitions:

    a. **Software Engineering Process (SEP) Defect**: refer to errors that relate to: SDLC phase anomalies, design flaws, testing flaws, documentation errors, standards incompliance, missing or flawed requirements definition, ill-conceived or flawed implementations, tools, etc).

    b. **Configuration Defect:** refer to errors related to software (or a combination of software products), patches, or upgrades that are installed, administered, compiled, resourced, or misconfigured in a manner that results in improper systems operations, introduce vulnerabilities, or degrade systems performance.

    c. **Hardware Defect**: refer to defects related to any hardware malfunction (e.g., infrastructure, telecommunications, network, server, desktop, SAN, printer, scanner, MFD, mailer/stuffer/sorter, etc)

    d. **Code Defect**: are defects in code that result in the software not functioning properly or operating as designed or evinces modes that were not specified in the design. Code defects refer to software errors that result in or associated with: unexpected errors or

undesirable results, program control and logic errors, introduce vulnerabilities, interface handling errors, db synchronization anomalies, subroutines, stored procedures, and database errors, data anomalies, memory leaks, error handling, data structures errors, as well as:

- **Allocation Management**: One module de-allocates a region of memory before it has completely finished with it. After the region is reallocated, the original module continues to use it in its original capacity.

- **Copying Overrun**: The program copies bytes past the end of a buffer.

- **Pointer Management**: A variable containing the address of data was corrupted. Code using this corrupted address caused the overlay.

- **Register Reused**: In assembly language code, a register is reused without saving and restoring its original contents.

- **Type Mismatch**: A field is added to a message format or a structure, but not all of the code using the structure is modified to reflect the change. Type mismatch errors could also occur when the meaning of a bit in a bit field is redefined.

- **Uninitialized Pointer**: A variable containing the address of data is not initialized.

- **Undefined State**: The system goes into a state that the designers had not anticipated. In overlay errors, the bad state caused the program to mistake the contents of a memory region.

- **Data Error**: An arithmetic miscalculation or other error in the code makes it produce the wrong data.

- **PTF Compilation**: Individual bug fixes are distributed together on a PTF (Program Temporary Fix) tape. Occasionally, a bug is repaired in one PTF but lost when the next one is compiled (a later bug fix is applied to an earlier version of the software).

- **Sequence Error**: Messages were sent or received in an unexpected order. The system deferred an action, such as

an acknowledgement message, but then forgot to execute the action.

- **Statement Logic**: Statements were executed in the wrong order or were omitted. For example, a routine returns too early under some circumstances. Forgetting to check a routine's return code is also a statement logic error.

- **Synchronization**: A error occurred in locking code or synchronization between threads of control.

- **Interface Protocol**: failure to comply with the interface protocols by sending erroneous requests or responses

e. **Product Defect**: refer to third party software product anomalies, interoperability and integration errors that results in improper systems operations, introduce vulnerabilities, or degrade systems performance. Includes all domains: database software, Web 2.0, Portal, network/Infrastructure software, security software, desktop or server Operating Systems (O.S.) software, and utility software, etc.

f. **Systems Defect:** is a fault, error or failure with embedded systems software or firmware; custom built or factory provided user and control interfaces to embedded systems and/or peripheral devices used to support business and/or technical operations; platform integration and/or configuration errors associated with embedded systems and/or peripheral devices.

g. **Operations Defect**: refer to defects related to user misunderstanding or incorrect use of the software and/or computer systems operational staff error (e.g., scheduling, batch run sequence anomalies, systems clean-up routines, etc)

h. **Undetermined Defect**: status of defect has not been determined

4. Defect Severity Definitions:

Defects must be classified by severity level. Defect Severity is classified into four categories:

a. Level 1: **Fatal Defects**
b. Level 2: **Major Defects**
c. Level 3: **Minor Defects**
d. Level 4: **Cosmetic Defects**

Fatal Defects are the defects, which results in the failure of the complete software system, of a subsystem, or of a software unit so that no work or testing can be carried out after the occurrence of the defect. Fatal defects could result in critical loss of data, critical loss of system availability, critical loss of security, critical loss of safety, or cause very serious consequences to citizens and/or agency mission. Multiple functions are severely broken, cannot be used, and there is no workaround. This defect must be resolved prior to approval of work product.

Major Defects are one, which also causes failure of entire or part of system, but there are some processing alternatives, which would allow further operation of the system. Major defects could cause significant consequences for the system and disruptions in business operations. One or more functions are badly broken, needs to be fixed but there is a workaround. This defect must be resolved prior to approval of work product.

Minor Defects does not result in failure but causes the system to produce incorrect, incomplete, or inconsistent results, or the defect impairs the system usability. Minor defects may cause small or negligible consequences for the system, minor disruptions in business operations and would be relatively easy to recover. Minor defects can be released into production provided there is a workaround or a waiver has been granted by DPW. This defect should be resolved prior to approval of work product.

Cosmetic Defects are small errors that do not prevent or hinder functionality. Cosmetic defects are trivial defects that can cause no negative consequences for the system or business operations. Resolution of this defect needs to be negotiated with impacted personnel.

Severity Determination Considerations:

a. **Answer the following questions before determining the severity:**
   1. Does the system allow me to work even after defect occurs?
   2. Does the system recover from the defect by any means?
   3. If the defect is recoverable, does the system do this on its own or any external effort is needed to recover from the defect?
   4. Did I check whether the same defect is reflected in all other related sections (or entire system)?
   5. Can I be able to repeat the defect in some other system having same configuration (O/S, Browsers etc.) as that of the system where I found the defect?
   6. Can I be able to repeat the defect in other configurations also?
   7. Does the defect affect only particular category of users or all?
   8. How frequently the defect occurs?

9. Which inputs make the defect?

b. The severity level increases if the answer for some of the above question is 'Yes and for some others 'No. For example, if the answer for question 1 is 'Yes, then further testing of the system is not possible and hence severity is high. Also the defect should be generalized as far as possible (i.e., after you find the defect, try to find out that the defect is repeated in reuse components, cross-browsers, cross-O/S etc).

5. Defect Reporting:

a. Required defect tracking and reporting data elements:

- Total number of open defects per category per defect severity
- Total number of defects resolved, outstanding, and re-opened per category per severity
- Total Number of Defects per SDLC Phase per Defect Severity
- Total number of defects per software release by category
- Defect fix rate per software release by category
- Average fix cycle time per category per severity

b. Defect Data Analysis:

- Test effectiveness and coverage (defect discovery and systems functionality)
- Use defect data to improve SDLC phase processes
- Collect defect data and subsequent analysis of defect, root-cause, fault detection, and failure modality
- Use defect data to assess product migration and Go or No-Go decision points

c. Reporting Frequency: Data (i.e., items outlined in 5. a.) should be collected and tracked monthly for all Peer Reviews conducted.

d. Defect management and reporting systems must be unique, separate, and operate independent of change control systems. The change control system should be used exclusively to track systems changes that are associated with systems modifications and maintenance activities tied to software release cycles. However, the systems maintenance PCRs are linked to a software release incorporating a specific or group of defects for resolution depending on the complexity and criticality.

e. DPW makes the final determination of assigning a defect categorization and criticality. DPW and/or contracted solution providers must provide input and specific data and/or information upon request by the Department with regards to defect prevention, discovery, resolution, management, tracking, reporting, SDLC processes and test reports. DPW and/or contracted solution providers are not authorized to make deletions or modifications to defects outside of status indicators without approval from the Department. Changes to a particular defect's categorization or criticality is not authorized and requires formal written authorization from DPW-BIS director, designated BIS SQA manager, or DPW Contract Administrator. The change control system is primarily used to track systems changes that are associated with systems modifications and maintenance activities tied to software release cycles. However, the systems maintenance PCRs are linked to a software release incorporating a specific or group of defects for resolution depending on the complexity and criticality.

Exemptions from this Standard:

There will be no exemptions to this standard.

Refresh Schedule:

All standards and referenced documentation identified in this standard will be subject to review and possible revision annually or upon request by the DPW Information Technology Standards Team.

Standard Revision Log:

Change Date	Version	Change Description	Author and Organization
04/29/2010	1.0	Initial creation	Thomas King
05/112010	1.1	Changes per DPW Domains' review.	Thomas King
05/27/2010	1.2	Changes per DPW Domains' review.	Thomas King
12/28/2010	1.3	Reviewed Content – No Changes	Thomas King
02/25/2014	1.4	Reviewed Content – Revised for DEA division director change, content is current	Michael Light