Bugs management

Bugs management is the process of reporting and tracking the progress of bugs/defects from discovery through to resolution. The bug, in general, might be defined as a deviation from requirements or the abnormal behavior of the software. As bugs are part of every development process, there are several activities, to which bugs can usually be traced:

- Unfinished requirements
- Requirements that are not detailed enough
- Requirements with unclear or multiple meanings
- Logic errors in the design documents
- Coding errors
- Not enough sufficient testing
- Misunderstanding of user needs
- Lack of documentation

While software defects might be inevitable, their number can be minimized by using a sound management procedure and implementing suitable bug management process. This process should focus on preventing the defects, catching the defects as early as possible, and minimizing their impact on the project. The bugs management process should be based on the following principles:

- Prevent the defects: if it is not possible or practical, the defect should be found as quickly as possible in order to minimize the impact of the defect.
- The process should be risk driven i.e. the priorities, and resources should be based on the extent to which risk can possibly be reduced.
- The bugs measurement should be integral part of the software development process and to be used by the project team to improve the process.
- The process of reporting and analyzing the bug-related information should be automated as much as possible.

If a bug tracking system is in place, it should encapsulate these activities. Most software developments incorporate a bug tracking system or repository that allows both developers and users to post problems encountered with the software, suggest possible enhancements, and comment upon existing bug reports. One potential advantage of a bug repository is that it may allow more bugs to be identified and solved, improving the quality of the software produced. The reports that appear in this repository must be triaged to determine if the report is one which requires attention and if it is, which developer will be assigned the responsibility of resolving the report.

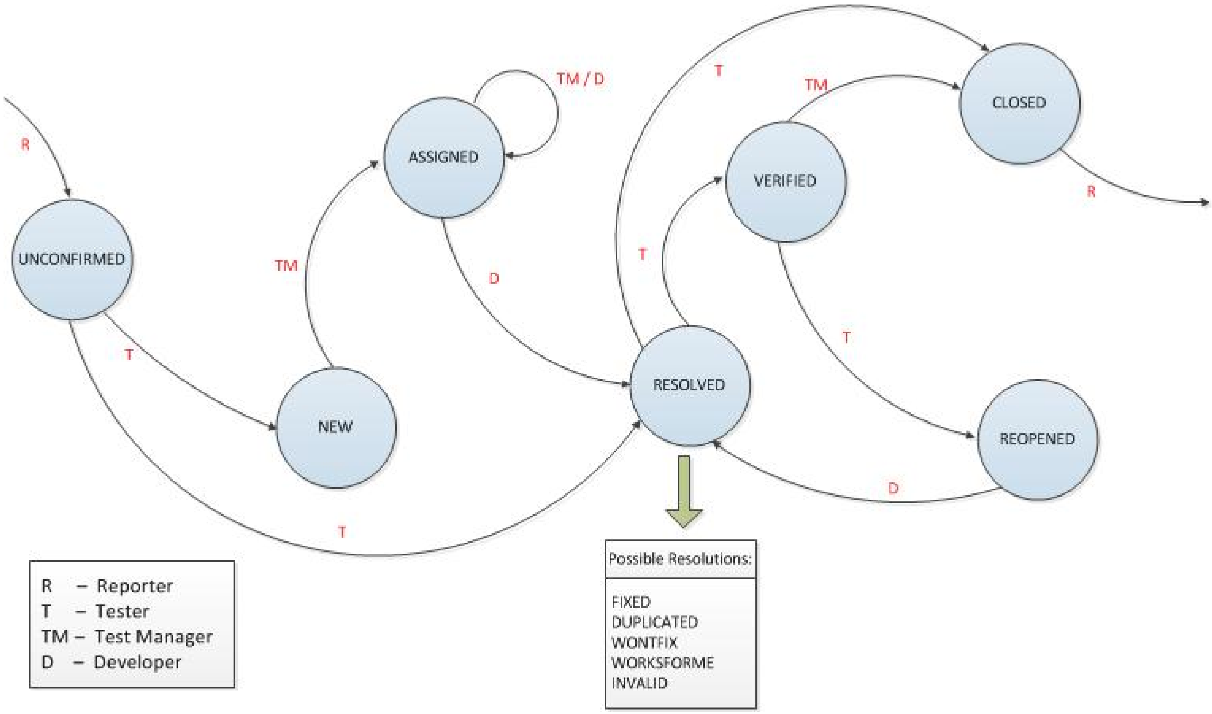
The bug tracking system is particularly important in open source software development, where the project team members can be dispersed. Therefore, the proposed bug tracking repository to be established in the OSOR environment will become one of the main channels of communication between SA Tool Kit Testing Group users and Developers Team for purposes of reporting bugs. It might be used not only to keep track of problem reports, but also to coordinate work among the two teams.

The other benefits of bug tracking system include:

- Offering the means of solving application's bug: Unless bugs are tracked, they can't be fixed, leading to failed projects.
- Increase visibility of development process: Improve the customer satisfaction by increasing communication and allowing them to monitor the progress of development.
- Traceability of bugs and their resolutions: Maintain audit trails to ensure all changes are accounted for.
- Release planning: Manage the bugs and enhancements that are to be resolved for the next product release.
- Resource scheduling: Manage the bugs that are assigned to each team member.
- Prioritization: Assign priorities to the bugs to ensure critical errors are addressed before minor issues, such as the wording of an error message.
- Improved control of a project: Monitor the status and progress of bugs, to follow the improvement in stability of a product or to ensure early detection of failing projects.
- Information consolidation: Capture all bugs in one place to promote the sharing of information project wide.
- Improve the quality of the software by increasing productivity: Notification of bug creation and status change to team members increases awareness and responsiveness.

The bugs life cycle

In software development process, the bug has a life cycle. The time span between the first time any bug is detected till the point when the bug is fixed (or closed or rejected or deferred) is called a bug life cycle. In that life cycle, the bug attains different states, which can be diagrammatically shown as follows:



Bug state diagram.

- The Reporter finds a bug and reports it bug tracking environment (bug status: Unconfirmed; responsible: Tester).
- The Tester evaluates the issue and tries to reproduce the bug. After successful confirmation the issue is registered as new bug (bug status: New; responsible: Test manager).
- The Test Manager assigns the bug to the developer (bug status: Assigned; responsible: Developer).
- The developer solves the issue and assigns resolution status (bug status: Resolved; responsible: Tester).
- The tester verifies the solution:
    - if the problem persists, the bug is passed back to the development team for fixing (bug status: Reopened; responsible: Developer).
    - If the solution is verified positively, the bug is closed (bug status: Closed; responsible: Reporter).

The bugs management is the process of organizing and managing these states. The process mainly depends on the bug tracking system which records bugs reported by the reporter. The bug is usually described by the time a bug was reported, its severity, the description of the incorrect program behavior, the step-by-step description how to re-create the bug, the identity of the reporter and any developers who fix it, the name of the component it have been found, the version of the software and the operating system on which the bug occurs.
Bug life cycle is closely related to software development life cycle. At any time during the software development life cycle errors can be made during the gathering of requirements, requirements

analysis, functional design, document preparation, coding, test planning, unit and integration testing, maintenance, release preparation, etc. Bug life cycle begins when a software developer or architect makes a mistake, creates an unintentional software defect, i.e. a bug, and ends when the bug is fixed, and the bug is no longer in existence.
After the issue is resolved, fixes should be re-tested. Additional effort should be made regarding requirements, software, hardware, safety impact, etc., for regression testing to check the fixes didn't create other problems elsewhere in the system.

Bug Status Standards

- **Unconfirmed:** The bug is added, and it's not yet validated.
- **New:** The bug is validated, and it must be processed.
- **Assigned:** The bug is not yet resolved, but is assigned to the developer.
- **Resolved:** The bug is resolved in a way described below.

Resolution	Description
-Fixed	The bug is checked and tested.
-Wontfix	The bug is described a bug which will never be fixed.
-Duplicate	The bug is repeated more than once.
-Worksforme	All attempts at reproducing this bug were futile, and reading the code produces no clues as to why the described behavior would occur.
-Invalid	The bug is in some way not valid.

- **Verified:** The bug is duly fixed.
- **Reopened:** The bug is detected again.
- **Closed:** The bug is fixed and confirmed its absence.

Bug priority standard:

Bug priority standard refers to the need as to how urgently bug is required to be fixed. It describes the importance of the bug. Bug priority may change according to the schedule of testing.

There are five levels of Bug Priority:

- **Immediate** – **(5):** The bug blocks development or testing work and should be fixed ASAP, or is a security issue in a released version of the software.
- **Urgent** – **(4):** The bug blocks usability of a large portion of the product, and should be fixed before the next planned release.
- **High** – **(3):** Seriously broken, but not as high impact. Should be fixed before next major release.
- **Normal** – **(2):** Either a fairly straightforward workaround exists or the functionality is not very important and/or not frequently used.
- **Low** – **(1):** The bug is not all that important.

Bug severity standard:

Bug severity standard refers to the quantum of danger as to how badly the bug can harm the system. Severity is a feature of constant nature associated with the bug.
There are seven levels of Bug Severity:

- **Blocker** – The bug is so important that prevents development or testing on the affected product.
- **Critical** – The bug causes the product's crash or a function does not work at all.
- **Major** – The bug that affects the product's feature to be operational.
- **Normal** – The bug impacts the product to work improperly.
- **Minor** – The bug causes the product to not work optimal, but it is possible to workaround such bug.
- **Trivial** – The bug irritates user, something is clearly no right, but it does not affect usability.
- **Enhancement** – A proposal of new functionality that improve the product in the future.

Roles and Responsibilities

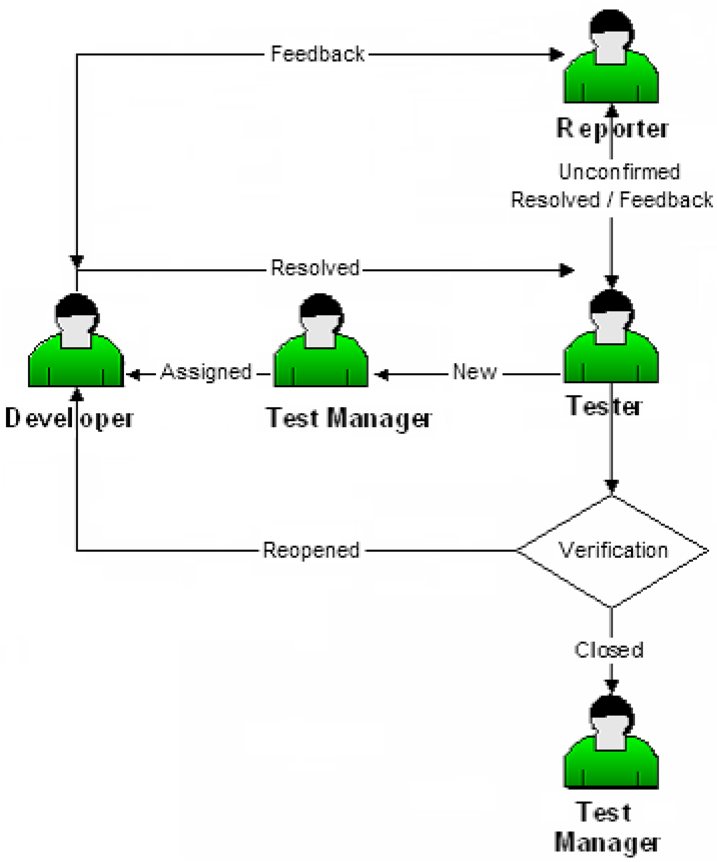
The following roles can be distinguished with regard to the organization of SA Tool Kit bug management process:

Role	Description and responsibilities
Reporter	Person(s) who reports the bug.
Test manager	Person who acknowledges the bug and assigns it to a developer.
Tester	Person(s) who tests the bugs and verifies the solution.
Developer	Person(s) who resolves the bugs.

In general, several of the roles might be performed by the same individual and – whenever necessary - several people could fulfill one role. However, given the relatively small number of people and organizations dealing with SA Tool Kit software and no external developer communities foreseen, it is proposed to create small development and testing teams with clearly defined tasks and responsibilities, as probably the most effective way to handle bug management process.

Bugs management process

The Bugs Management process allows the project team to collect, evaluate, correct, and track reported bugs and issues. When a bug is first reported, it is registered as unconfirmed and managed by Test Manager. Test Manager is responsible to periodically go through any new bugs that may have been reported and for accepting or rejecting new bug. He is also in charge to assign accepted bugs to developers and set priority and severity of the bug based, for example, on how many entities it affects and whether there are any workarounds.
When a bug is fixed by a member of the development team, it is considered as resolved. The testing team uses the information provided by Reporter to verify that a bug has been fixed in a later build of the product. After successful verification, the bug is closed; however when there are still issues unresolved, the bug might be reopened for further development.



The bugs management will be handled in accordance with the following policy:

- Any registered SA Tool Kit OSOR user can report a bug to bug-tracking system.

- The bugs must be filed correctly and with as much information as possible. If the bug is reported in an improper way, the testing and development teams might not be able to reproduce it and /or fix it, and the bug might be closed promptly.
- If the reported issue is further evaluated as an enhancement in the product, it will be considered as an enhancement and the reporter will be asked to file it as a change request.
- All bugs must follow the process. If a bug is not submitted in accordance with this process, it won't be considered.
- The content of the bugs repository will be visible to all OSOR SA Tool Kit users.
- The bugs might be rejected, because they are duplicates or are instead considered to be a feature request.

Bugs reporting guidelines

The main information required when reporting a bug is as follows:

- Short but clear and explanatory description of an issue

- Environment: The specifics of the environment in which the issue occurred are necessary to allow the problem to be reproduced. This may include details such as:

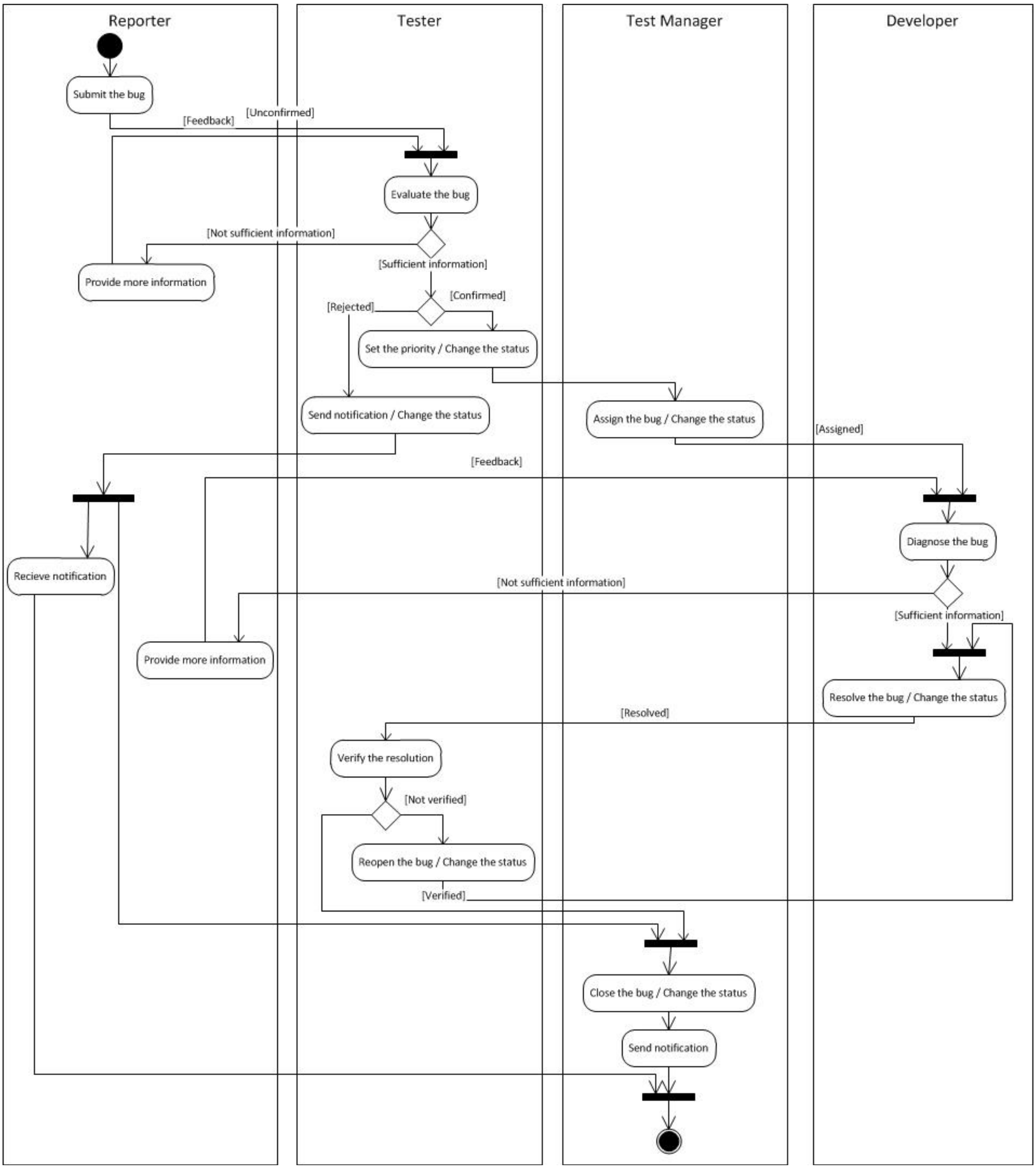
  - hardware configuration;
  - operating system;
  - dependent software;
  - browser version;
  - product version;
  - the component which failed.
- Steps to reproduce the problem: A concise and minimal set of steps which can be followed to reproduce the problem.

- Expected results: This should describe the expected behavior or expected results which will readily explain why the problem is being reported when compared to the actual results.

- Actual results: This should describe what actually happened, complete with any error messages, stack traces, screen shots, log files that show the outcome. It may be the case that the behavior is correct but has been misunderstood when compared to the expected results, in this case the resolution may be that clearer documentation is required.

The workflow

The bug management workflow describes the actions needed to report, evaluate and fix the bug in bug tracking environment.
It is expected that the workflow might vary depending on the amount of available project resources, documentation procedures, and/or complexity of the bug, with a bug perhaps moving between a number of developers for different stages of resolution, followed by updates to the documentation by a technical writer before a tester can verify its resolution.
While the amount of available resources constrains the rate at which issues can be resolved, the open source projects should acknowledge each issue as soon as it appears in the tracking environment. The need for rapid notification requires that form for filing new issues should be able to capture the Reporter’s email address, so he/she can be notified about the changes in issue status, or contacted for more information.



Bug activity diagram.

- **Submit the bug:** The new bug is submitted to SA Tool Kit OSOR by the Reporter. The person who files the issue (reporter) may be totally unknown to the project—bug reports are as likely to come from the user community as from the developers.