Defect Tracking Best Practices

Abstract:
Whether an organization is developing a new system or maintaining an existing system, implementing best practices in the defect tracking and management processes will save time and effort. In this paper we discuss typical issues and lessons learned, and map these to preventive measures.

Bio:
Employed since April ’02 as Bureau Chief for the Engineering Compliance Bureau, Information Technology Division, New Mexico Human Services Department. In this position I oversee testing and process improvement for several large maintenance phase applications as well as new projects. Previously employed and consulted for 18 years in test engineering and test management for NASA and private satellite communications projects, including NASA TDRSS and Iridium.

Defect Detection Distribution and the SDLC

For the near future at least, software projects will invariably require defect tracking and management. Until software engineers come up with a method to completely prevent defects from getting created, defects will require review, fixing, testing, and re-release to the production systems.

Figure 1 illustrates a typical defect detection distribution for a new project. Figure 2 illustrates the desired detection distribution for a new project. Although the Software Development Lifecycle (SDLC) methodology in place may somewhat affect the project schedule and defect count at release, the project will still likely be delivered with a residual defect count that looks more like Figure 1 than Figure 2. For example, a project using the Spiral SDLC model may have the project activities overlapping each other rather than in discreet phases as shown. And an Agile Scrum management approach may have development iterations of subsystem functionality. However, all SDLC models have common constraints: the resources available (skilled staff and tools), accurate requirements management, the schedule available, and the capabilities of the management team. These constraints result in delivery of the project with a non-zero defect count. It is therefore imperative that the most critical and show stopper type of issues be quickly identified and tagged for priority fix and retest. For a project that has achieved maintenance phase, the objective is to efficiently correct the deficiencies, especially those affecting the customer. These must also be appropriately identified and tagged for priority rework, retest, and re-release.

Defect Lifecycle

Figure 3 illustrates a typical defect lifecycle and the relative amount of time for the rework. The desire to limit the total amount of lifecycle time requires an examination of each defect state and the transition between states. The time within each state is dependent on the project management practices and resources (development tools, staffing levels, etc). The time to transition between states is incurred by the communications and hand-off processes, at the heart of defect tracking.

Defect Tracking Issues

Having supported many types of software intensive projects, I have observed many types of issues with the defect tracking processes, even for projects with expensive tracking tools in place. In the

following discussion we list and define the types of issues resulting from poor defect tracking, and the resulting impacts to the defect lifecycle.

1. Defect Repository Uses Ad Hoc Spreadsheet
   New projects sometimes are late in acquiring their defect tracking tools and processes, and can find themselves in this situation. The quick ad hoc response is the spreadsheet repository. This method requires all the project participants to periodically open the spreadsheet to update and/or read state changes and status. It also requires a single controlled version so that everyone uses current information. This method is only helpful while the number of open defects is small and relatively stable. It is unwieldy and unreliable when there are many program participants and/or many defects to work. It is a certain recipe for disaster for complex or large projects.
2. Defect Repository via Email
   The use of email to report and store defects is another ad hoc method. All participants must keep copies of the emailed defects. As status updates are provided, there can become very many emails for each defect. It gets very difficult to track status as the defect goes through its fix lifecycle. Utilizing email for a defect repository opens up the possibility for multiple points of failure.
3. Defect Repository Inaccessible
   If the defect repository is not accessible for some project members, they will need to request those who have access to retrieve or update defect information. This situation can result from an insufficient number of tool licenses, from inaccessible servers, or from security privilege issues.
4. Insufficient Issue Description
   If the analyst cannot understand the nature of the defect issue, they must contact the defect originator for clarification. No forward progress can be made on the defect until the originator provides the missing information.
5. Supporting Data Unavailable
   The analyst often requires the supporting data for an issue to fully understand it. If this was not provided, the analyst must contact the originator for the location of the data. No forward progress can be made on the defect until the originator provides the supporting data. The analyst may have no choice but to declare the defect as closed due to insufficient data. Time is wasted in either trying to recreate the test results data or in reopening the defect when it eventually recurs in the test lab or with the customer.
6. Sequence of Events Unclear
   The analyst sometimes requires information regarding the sequence of events leading up to the issue. If it is not properly documented in the defect write up, and if the tester cannot recreate the sequence in the test environment, the analyst may have no choice but to declare the defect as closed due to insufficient information. Time is wasted in either trying to recreate the sequence or in reopening the defect when it eventually recurs in the test lab or with the customer.
7. Insufficient Defect History
   An insufficient record of the changes and state transitions can create confusion. Suppose for example that a defect has gone through the fix-test cycle twice and failed both times. If the history record does not properly indicate the information for each failure, then the assigned analyst might not use the current test information for the failure analysis.
8. Unclear Assignment
   If the team members cannot determine who is currently assigned to the defect, chances are that it is not being worked. The defect may languish until the next review meeting.
9. Incorrect Assignment

If the wrong team member is assigned the defect, it may not be getting the attention it deserves. If the team member is not the most proficient in the system domain for the defect (software or hardware domain, subsystem, or function), the defect might not get resolved correctly or efficiently.

10. Incorrect Priority Assigned
    The effects of incorrect priority assignment are obvious – the lower priority work is being tasked before the higher priority work. Once this situation is discovered and the team is re-tasked, the actual higher priority defect is completed later than when it should have been completed. The lower priority item gets placed onto the "to do later" queue and may not be revisited for some time, resulting in the Old Defects issue.
11. Poor Prioritization Scheme
    The defect prioritization scheme for the development phase of a new project is slightly different from the prioritization scheme for the maintenance phase. Since the project phase is concerned with on-time delivery of a not yet used system or product, the prioritization scheme is based on the affect to the implementation schedule. During the maintenance phase, the prioritization scheme is based on the impact to the customer. In addition, adding or changing priority levels mid project creates confusion and can leave defects incorrectly prioritized. Assigning too many defects the highest priority defeats the purpose of having priorities, as it creates confusion in determining which tasks to work first.
12. Old Defects
    A low priority defect that has not been worked for a long time can become problematic. The product users may develop a work around for the defect, which over time may become part of their standard operating procedure. In addition, the data or knowledge for the defect may be lost over time, making it more difficult to work once reassigned.
13. Defect State Incorrect
    This issue can easily occur when the assigned participant completes their task but fails to update the defect status with the change of state. The defect may not be properly assigned or tasked until the incorrect state is corrected.
14. Defect Not Classified
    System defects can result from a number of issues, and can originate during all phases and from all realms of the project. Classifying defects by root cause (code, design, requirement, CM, etc) and by domain (software or hardware subsystems) helps to sort and assign them. More importantly, classification metrics can help reveal systemic issues. For example, many recent defects classified as CM related may indicate poor code migration processes.
15. Backlog Trend Unknown
    Figures 4 and 5 provide example defect trending metrics, used to analyze defect density distributions. The defect density is the number of defects per size of the application or domain. If you don't know the trends in the domains or subsystems, it may not be clear whether you may need hire new resources or reassign resources from one domain to another.
16. Increasing Defect Backlog
    The backlog of defects may indicate a resource tasking problem or a systemic problem (such as architecture, implementation, or design) as seen in Figure 5. The backlogs must be reviewed periodically.
17. Defects Renumbered or Deleted
    Deleting or renumbering of the defects affects the capability to research historical issues and trends.
18. Infrequent Defect Review Meetings
    Periodic defect review meetings are required to assign new defects and to review the status of the backlog as needed for possible state changes or reassignments. Failure to review often

enough can delay assignments or reassignments, can delay state changes, and can delay escalation of serious system or resource issues.

19. Insufficient Defect Review Team Membership
    Like an Agile Scrum team, the right parties must work together for these reviews to be productive. Development, Test, and the customer must be represented at the reviews.

Mapping of Issues to Preventive Measures & Best Practices

Figure 6 takes the issues as identified above and maps them to preventive measures. These can be described within the context of the project plans or deliverables described below.

Project Plan Scheduled Tasks and/or Deliverables:

- Plan for Acquiring a Defect Tracking Tool (software, licenses, admin manuals)
  - Tool reviews / decision to purchase or use home grown tools
  - The contractor should be tasked with procuring the tool and assigning the rights to the customer after project warranty.
  - Tool Installation & Setup
  - Tool Training
- Plan for Platform Needs (infrastructure for supporting tool and defect database)
  - Acquisition
  - Installation / Network Integration
  - Configuration of Security & Access Rights
- Plan for Development of a Defect Tracking Plan to address each of the following *
  - Defect Report Content
    - Originator / Origination Date
    - Defect Description
    - Sequence of Events
    - Supporting Data
    - Defect Lifecycle History
    - Subsystem or Domain Category
    - Root Cause
  - Defect Tracking Process
    - Numbering Scheme
    - Prioritization Scheme
    - Categorization Scheme
    - Root Cause Assignment Scheme
    - Assignment to Team Members
    - Defect Lifecycle Flow (States and State Transitions)
    - State Transition Communications
  - Defect Reviews
    - Required Attendees
    - Defect reviews should begin during unit testing.
    - Review Schedule (meet more often for new projects)
    - Defect Review Agenda (assignment of priorities, assignment of defects to team members, reassignments, review of backlog, reporting to management)
  - Process Training
    - Training and User Manuals

*** Additional Considerations for Best Practices**

1. Defect States. The work flow for each project may be different. The defect management tool and process should allow for a project team to determine the defect states and the milestones necessary to transition from state to state. Typical defect workflow states are:
    a. New (to be assigned)
    b. Analysis (analysis of issue in progress, Analyst assigned)
    c. Invalid (duplicate issue, or "works as designed"; may be closed)
    d. Ready For Dev (analysis complete, ready for software coding)
    e. Development (software coding in progress, Developer assigned)
    f. Ready For Test (software change complete)
    g. System Test (system testing in progress, Tester assigned)
    h. Failed Test (failed system test or UAT, to be returned to Analysis)
    i. UAT (Ready for UAT, or UAT in Progress)
    j. Migrate (UAT Complete, Ready to Migrate)
    k. Closed
2. State Change Communications. The processes and tools should provide for efficient communication of defect state changes throughout the defect workflow, so that a defect is never left in limbo. Ideally, the users are provided a notification the instant a defect is assigned to them. Some defect tracking tools provide this feature.
3. Prioritization. The process and tool should allow for assigning the appropriate priority level. Most priority schemes use four levels – numbered 1 to 4 where 1=high, 2=medium, 3=low, and 4="nice to have". During the system development phase of a project, defect prioritization is a function of impact to the implementation schedule (Project Risk), whereas during the operational phase, defect prioritization is a function of impact to the business.
4. Defect Root Causes. It is useful to assign a defect root cause to the defects. This helps to facilitate metrics analysis and reporting of defect density distributions for the various functional areas or software domains. The categories of root causes should include at a minimum the following process areas:
    a. Requirements (Requirements definition error or issue)
    b. Design (Design error or design documentation error or issue)
    c. Code (Coding error or issue)
    d. CM (Configuration Management error or issue, including migration errors)
    e. Test (Tester error, works as designed, or test issue)
    f. Documentation (Incorrect user manual or operational documentation)
    g. Environment (Issue resulted from external or environmental condition)
    h. Not Reproducible (Root cause unknown)
    i. Invalid (duplicate issue, or "works as designed")
    j. TBD (Root cause is yet to be determined)
5. Metrics Reports. The tools should provide for generation of sorted reports, user specified, to facilitate team review of the following:
    a. Defect backlog (all non-closed defects for the current date).
    b. Number of defects opened and closed during a reporting period.
    c. Total defects generated by a given date.

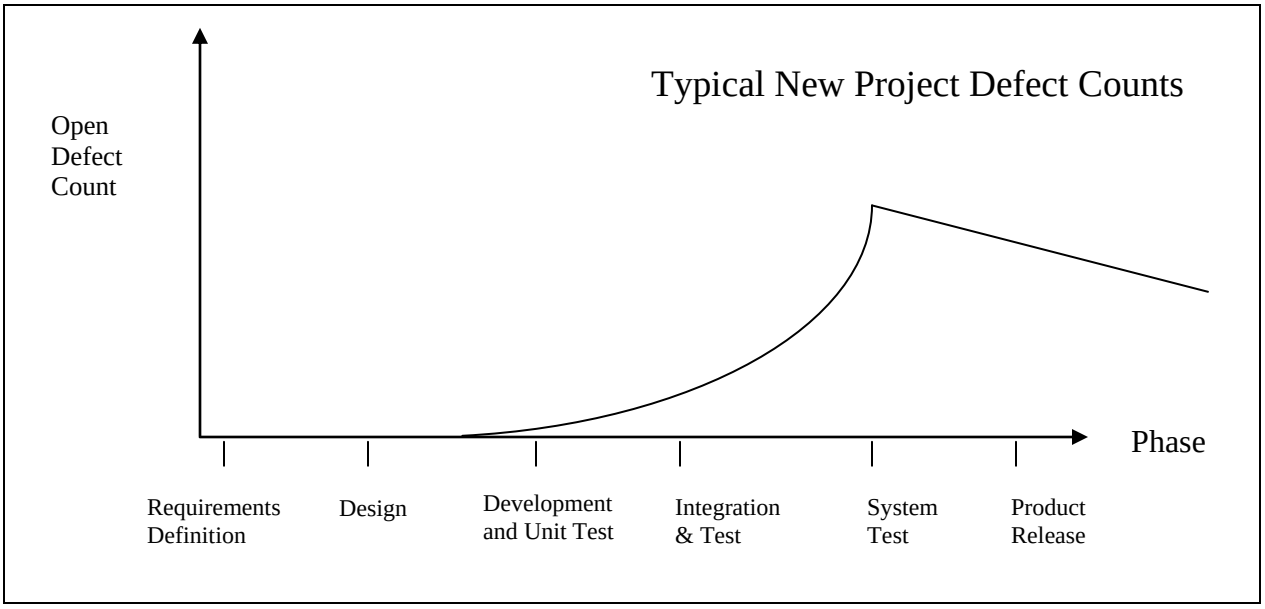


Figure 1

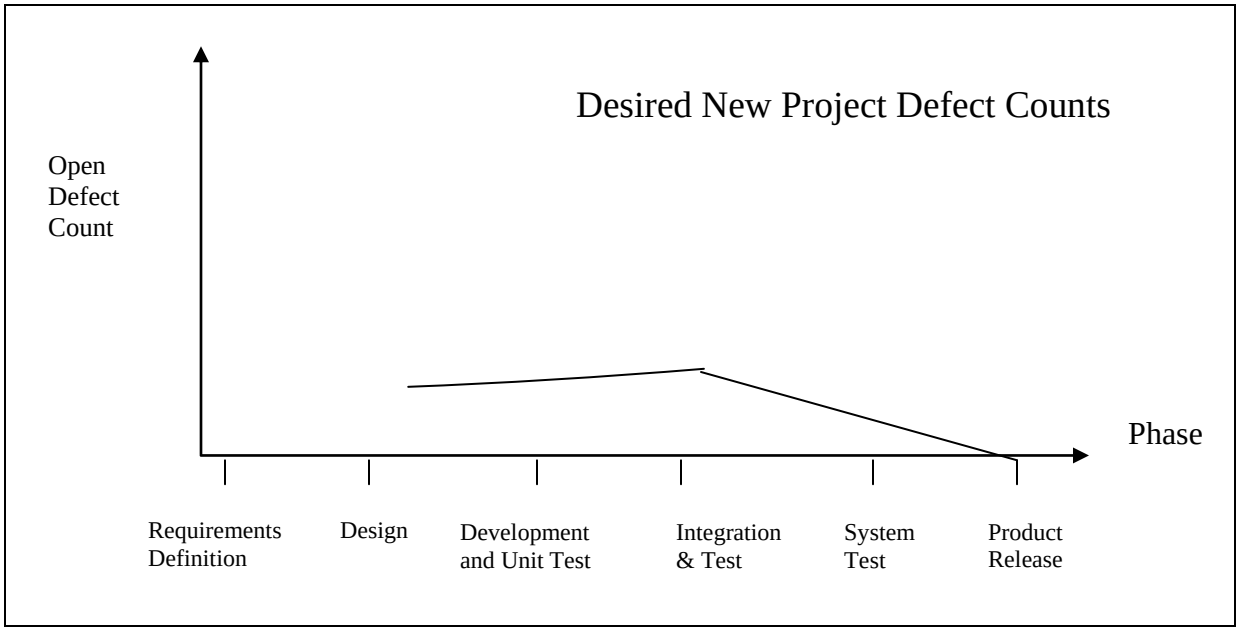


Figure 2

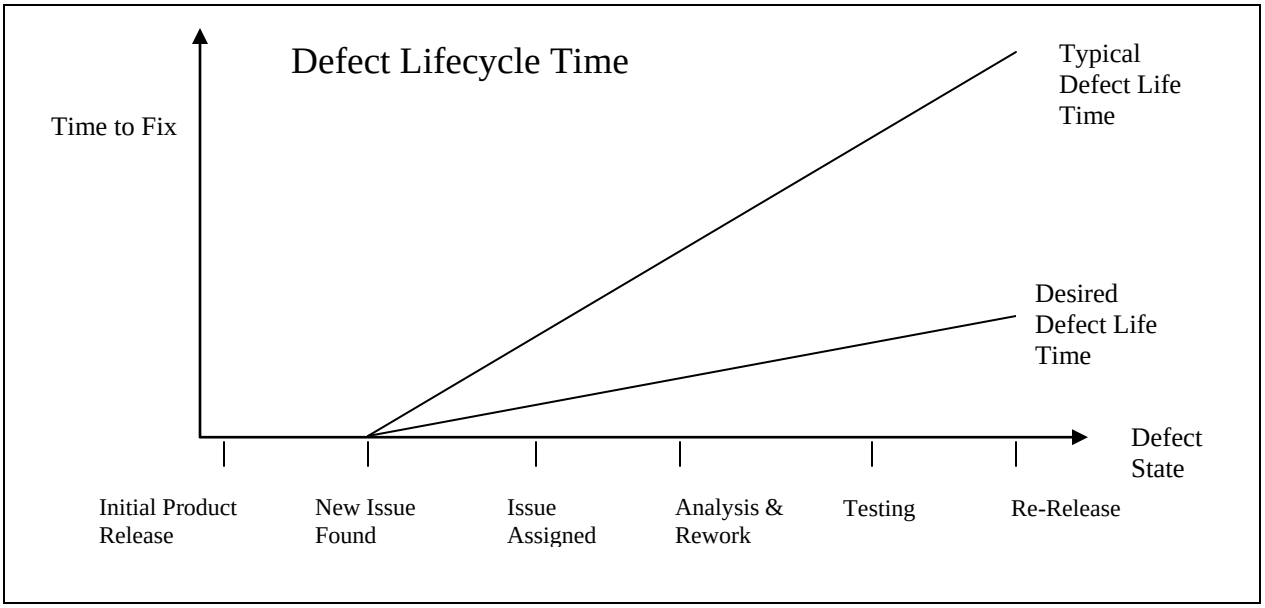


Figure 3
Defect life time is a function of the time within each state and the time to transition between states.

Figure 4. A proper defect backlog trend

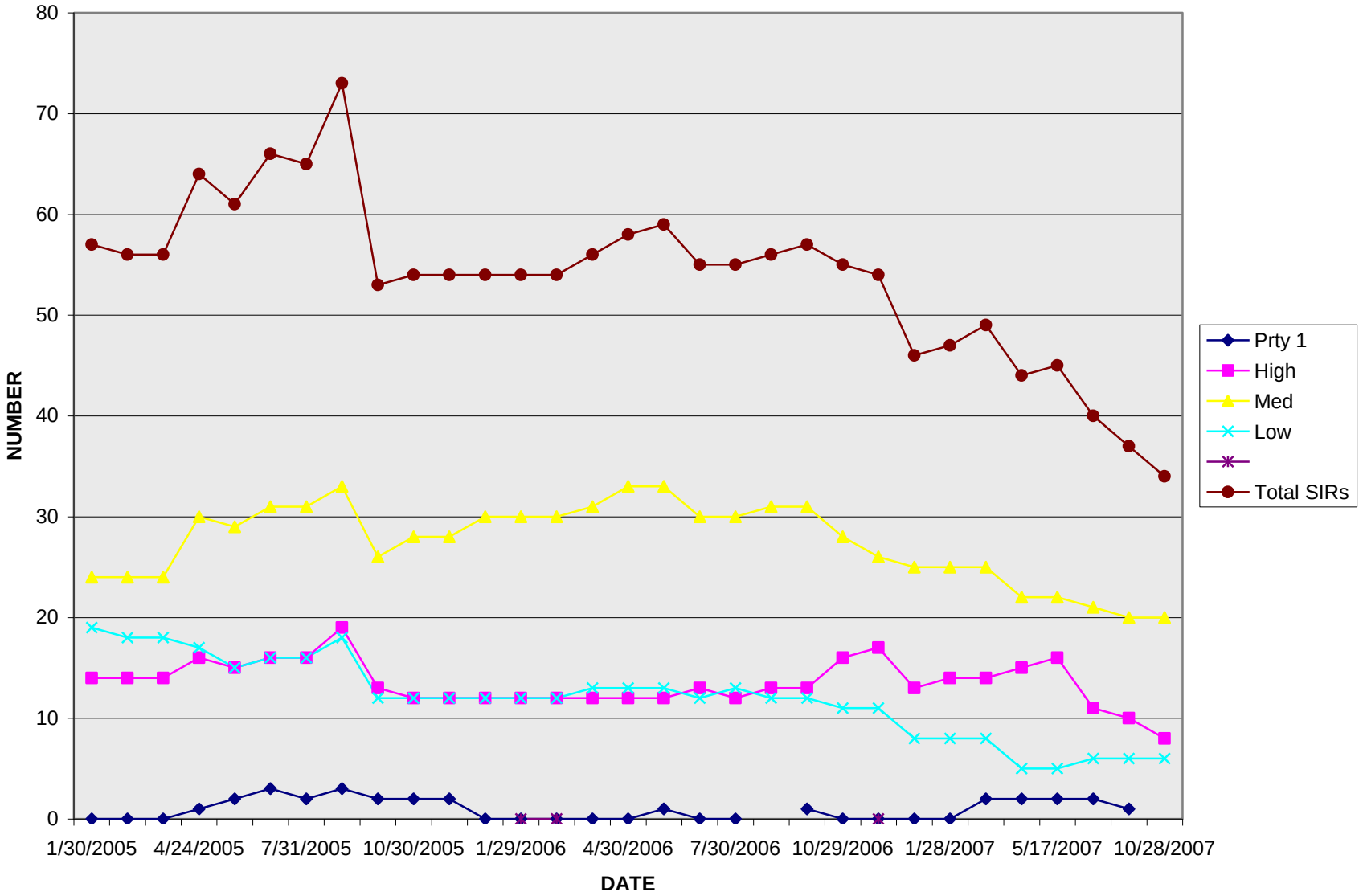
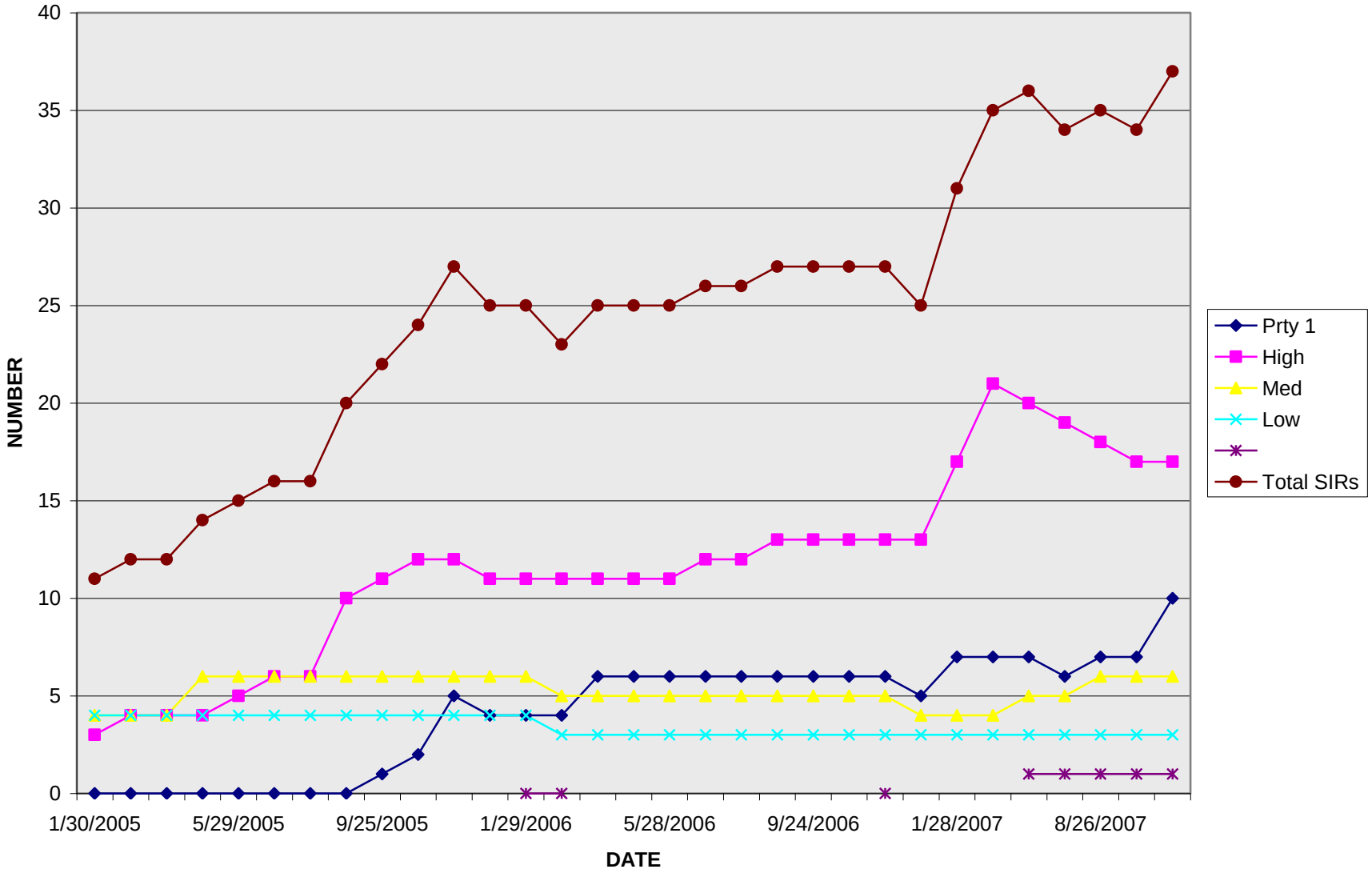


Figure 5. A problematic defect backlog trend



Tracking Issue	Source	Corrective Action	Preventive Measures
1. Ad Hoc Repository	Project Planning	Defect Tracking Plan	Require the Defect Tracking Plan and any tools as project deliverables. Add tool setup and training to project plan.
2. Email Repository	Project Planning	Defect Tracking Plan	Addressed above
3. Repository Inaccessible	Project Planning	Server and Network Plans	Determine accessibility requirement during project planning, and translate to server & network requirements.
4. Issue Description	Process	Process & Training	Describe level of detail for defect description field, in the Defect Tracking Plan, schedule process training in the project plan.
5. Data Unavailable	Process	Process & Training	Describe inclusion of defect supporting data and/or location of data, in Tracking Plan and training.
6. Sequence Unclear	Process	Process & Training	Describe inclusion of details for sequence of events for issue description field, in Tracking Plan and training.
7. Insufficient History	Process	Process & Training	Describe the inclusion of historical info for the Tracking Plan, and training.
8. Unclear Assignment	Process	Process & Training	Describe the process for assignment of defects to project team members, in the Tracking Plan, and provide for training.
9. Incorrect Assignment	Process	Process & Training; Reviews	Describe the process for assignment of defects to project team members, in the Tracking Plan, and provide for training.
10. Incorrect Priority	Process	Process & Training; Reviews	Describe the prioritization scheme in the Tracking Plan; and the process for assigning priorities at the periodic Defect Review meetings; and provide for training.
11. Priority Scheme	Process	Process & Training	Addressed above
12. Old Defects	Project Planning	Management Reviews	Tracking Plan should describe the process for management review of old defects at the periodic Defect Reviews.
13. Defect State Incorrect	Process	Process & Training; reviews	Describe defect states and (state transitions in Tracking Plan; provide for training, and include in periodic reviews.
14. Defects Not Classified	Process	Process & Training	Describe classification scheme for root causes and classifications for domains in Tracking Plan; and provide training.
15. Trends Unknown	Project Planning	Management Reviews	Plan for trending in Tracking Plan; assign trending to a project team member; plan for periodic review at Defect Reviews.
16. Increasing Backlog	Project Planning	Management Reviews	Addressed above
17. Renumbered / Deleted	Process	Process & Training	Describe numbering scheme in Tracking Plan; provide for training.

Figure 6. Issues mapped to Preventive Measures