The Path to Successful Defect Tracking

One of the challenges all software development organizations face is how to handle the defect-tracking (also known as "bug" tracking) process. Unfortunately, often, little thought is put into designing and implementing such a system. Among the reasons that a well-designed defect-tracking system is necessary:

- Traceability – You need to know what areas of the system produce the most defects.
- Accountability – Do you have weaker developers and/or QA engineers? For example: a developer who writes sloppy code will cause more defects, and a QA engineer who lacks attention to detail may miss a critical bug during the testing process.

Choosing a Defect-tracking System

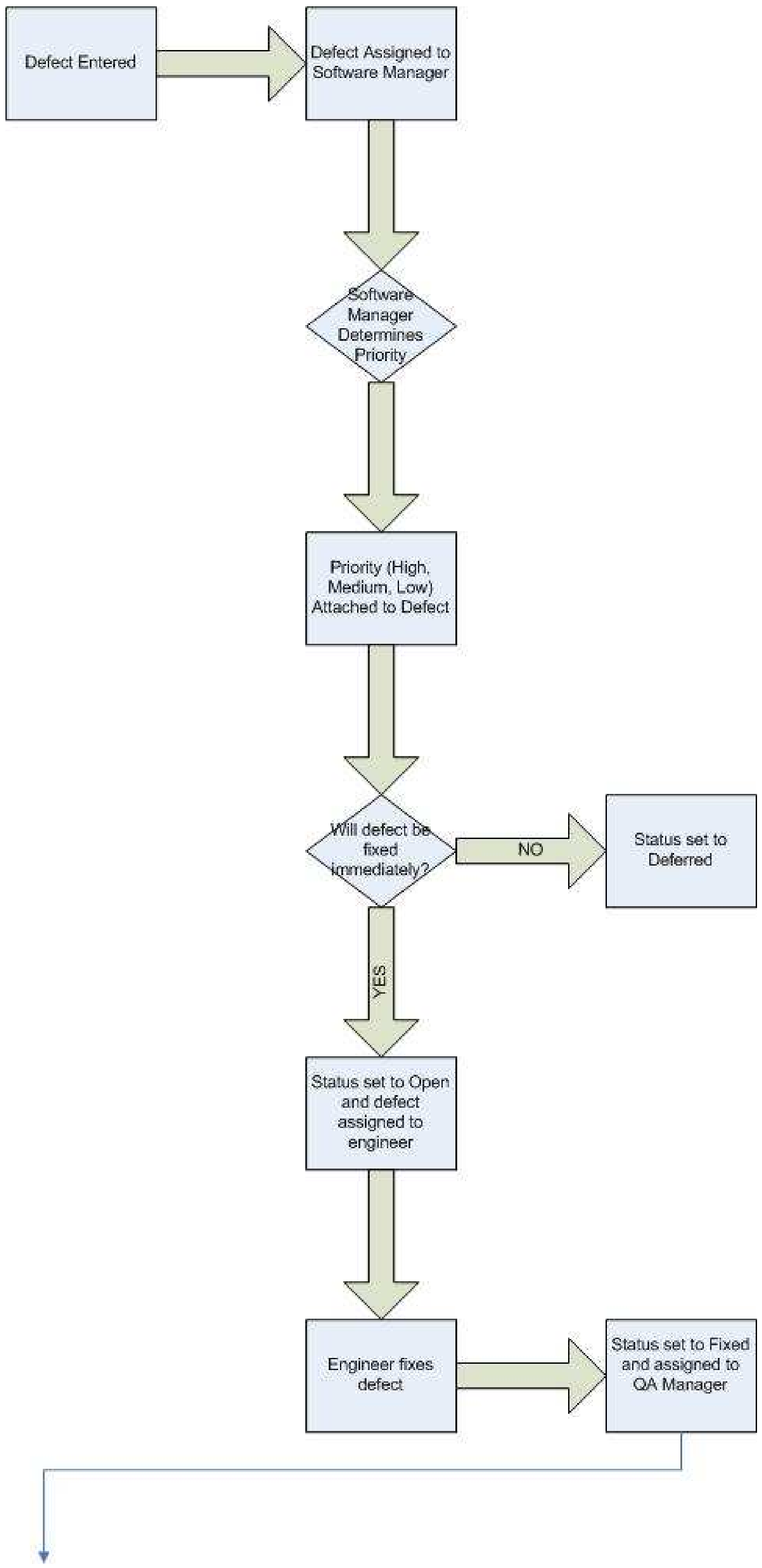
When determining what type of defect-tracking system to choose, you will need to consider several elements:

- Buy or build. Depending on your budget and development resources, you will have to decide between buying a commercially-available defect-tracking system and building your own. While building your own system may be less expensive in the short-run, in the long-run, it may end up costing you more in time and money.
    - a. If you decide to buy…
        - i. How much are you willing to spend? Basic software packages run from $100/license to more than $2000/license.
        - ii. What kind of platform do you want to use? A hard-core, commercial database solution using Oracle? Or, a scaled-down, basic system using Access? (Of course, this corresponds directly with your budget.)
        - iii. Nowadays, many vendors offer hosted solutions. These solutions are ideal for companies who do not have the resources to implement and maintain a system on their own.
    - b. If you decide to build…
        - i. Do you have developers who can be dedicated to the task? Depending on the complexity of the system you are going to build, you will need at least one full time developer for an extended period of time.
        - ii. What programming language will you use? Chances are you will be building a web-based application with a database backend. You can choose any number of languages that work well with databases; for example, Perl and Cold Fusion are two possibilities.
        - iii. Do you have hardware that can be used to host your application? You will most likely need a web server and a database server.

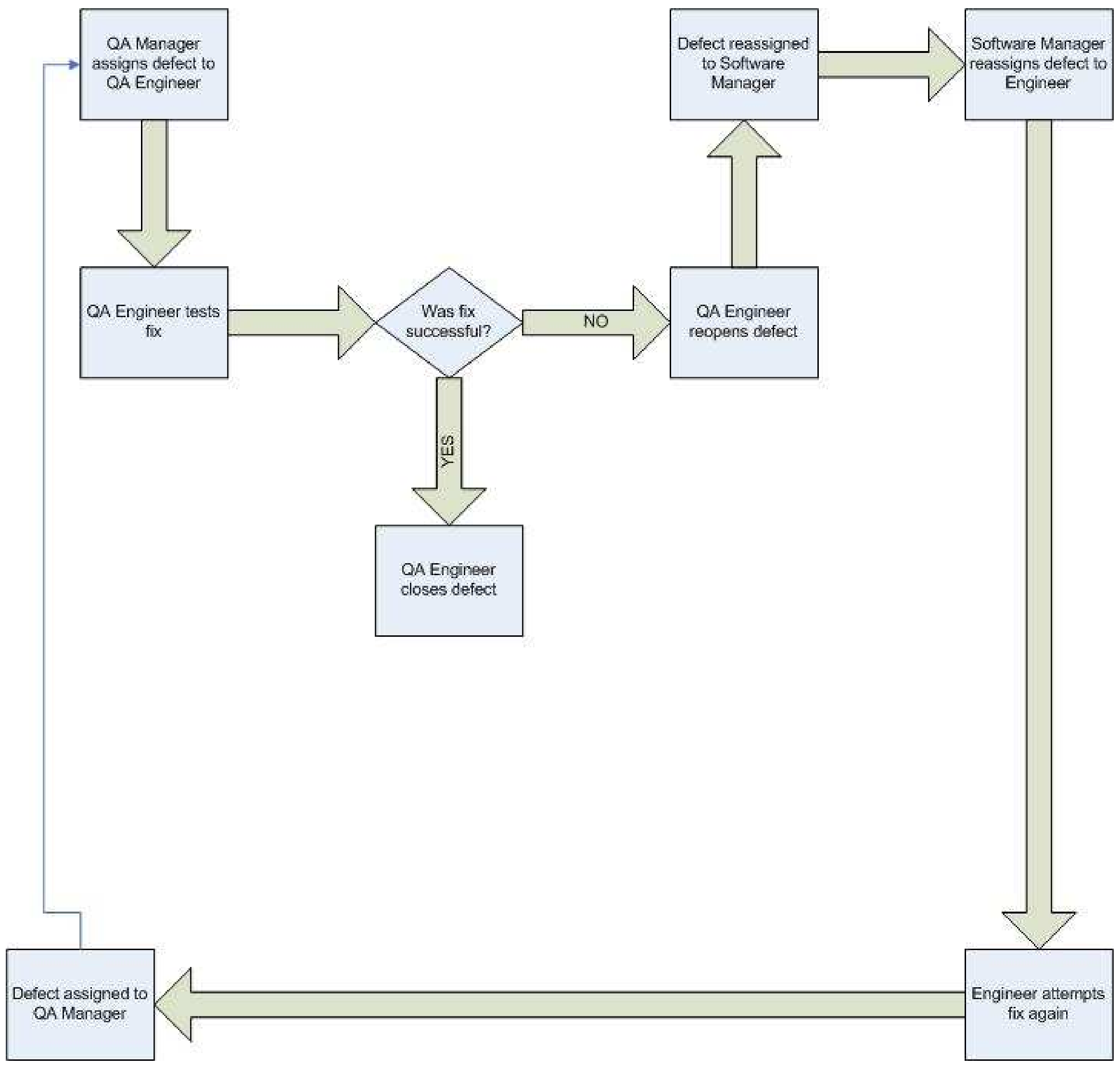
Prior to Implementation

Once you have purchased or decided to build a system, you will need to answer some basic questions:

1. Who will have access to your defect-tracking system? Which employees will have the ability to add defects? Edit defects? Which users should have the ability to change a defect's status? These are important questions which should be determined BEFORE implementing the system.
2. How much information should you collect upon initial bug entry? If users outside of the development group (for example, support personnel) are going to enter defects, you will want to limit the required information to that easily known. Also, requiring too much information may lead to reluctance from some employees to enter defects at all. On the other hand, it is important to collect enough information for your developers to be able to research and fix the problem.
3. What should be the basic path of a defect? That is, in most defect systems, bugs are routed through various individuals/groups depending on the status, type of defect, and area of the system affected, among others. A common path of a defect traveling through the systems is as follows:



Defect Entered
Defect Assigned to Software Manager
Software Manager Determines Priority
Priority (High, Medium, Low) Attached to Defect
Will defect be fixed immediately?
NO
Status set to Deferred
YES
Status set to Open and defect assigned to engineer
Engineer fixes defect
Status set to Fixed and assigned to QA Manager


QA Manager assigns defect to QA Engineer
QA Engineer tests fix
Was fix successful?
NO
YES
QA Engineer reopens defect
QA Engineer closes defect
Defect reassigned to Software Manager
Software Manager reassigns defect to Engineer
Engineer attempts fix again
Defect assigned to QA Manager

After Implementation

Once you have implemented the defect-tracking system (whether you have purchased a system or built one in house), you are ready to begin entering and assigning defects. However, what do you do with this data? After all, you will soon have a database full of useful information about your software products, including the number and severity of defects found by both your development staff (most likely QA Engineers) and technical support personnel.

As stated in the introduction, this data can help answer questions like:

1. What areas of the application are most vulnerable to defects (For example: Is the UI stable? Does the backend database connection work reliably?)
2. Which developers have the most reopened bugs?

3. Which QA engineers find the most defects? Which find the most severe defects (as opposed to superficial issues, such as spelling and other typographical errors)?

In addition, upper management may want a report on the quality and stability of your product(s). As long as you are storing your data in an ODBC-compliant database, you should be able to easily extract this data into spreadsheets, which can in turn be formatted and sorted in several ways.

Based on these reports, management can make decisions like whether to upgrade the development staff, place additional resources on specific (more vulnerable) areas of the system, and perhaps most importantly, determine the status of a project.

Conclusion

Many companies make the mistake of not spending enough time choosing a defect-tracking system. Think carefully about what information is most important, who will be using the system, and how you will report the results. The bottom line is that a strong defect-tracking system is critical to the success of a software development organization.