Managing Workflow for Software Defects

For teams managing software quality, it is crucial to manage the workflow around the the defect reporting process so that everyone understands how a defect moves from recognition to resolution. Below are some tips for defining the workflow for software defects.

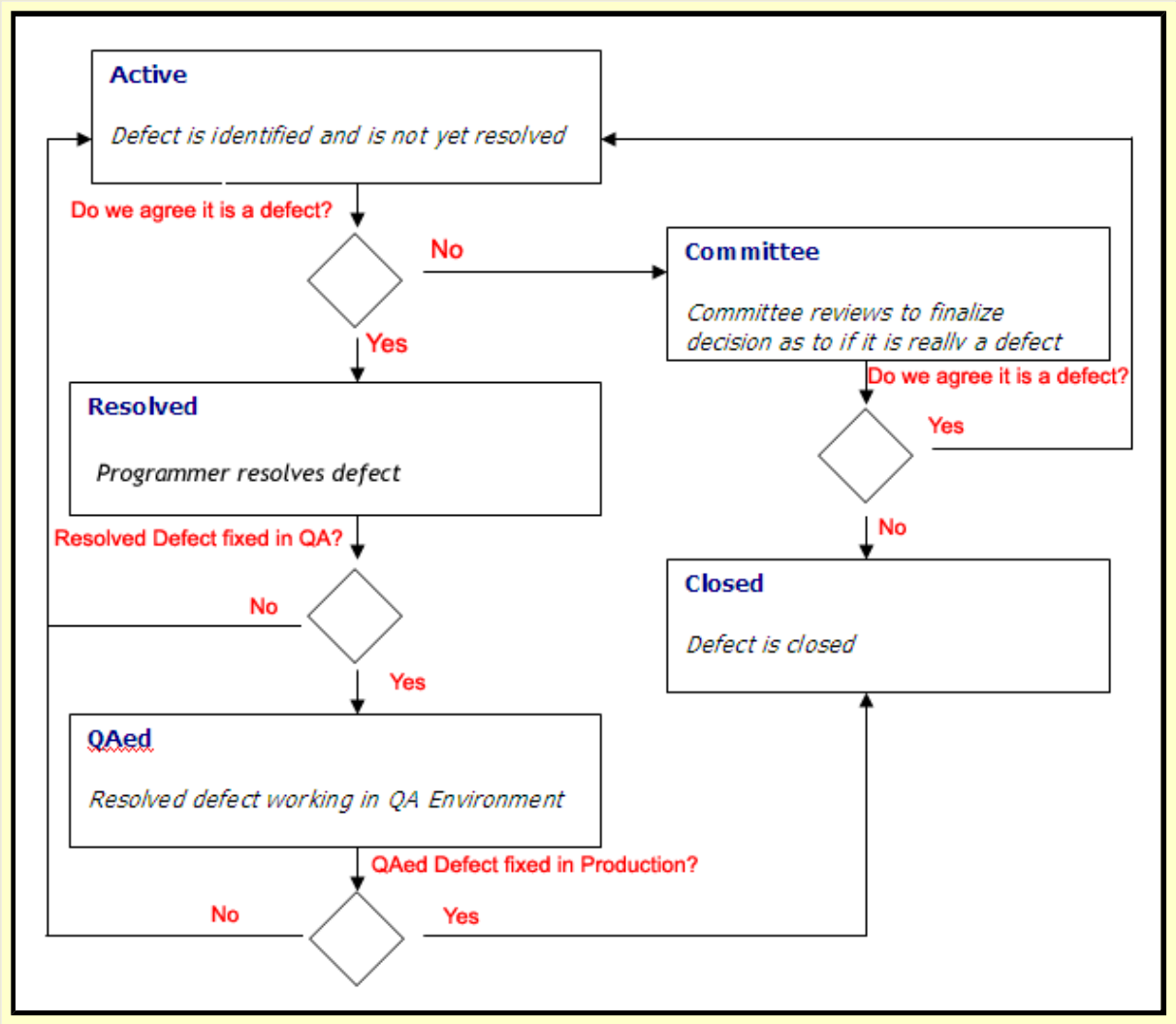
1. **Define the Workflow Statuses** - When tracking software defects, it is important to define the workflow. Workflow is normally tracked via the "status". Let's create a simple workflow for a development team, where the tester finds a defect and follows it through resolution, quality assurance and closure. Below are some possible sets of statuses (workflow) for this process.

   Workflow Statuses:

   Active
   Resolved
   QAed
   Committee
   Closed

(continued on next page)

2. **Flowchart the Workflow** - Flowcharting the workflow allows team members to understand the process in full. We created the flowchart below using Microsoft Word.



3. **Advanced Workflow** - In our example above, we used simple workflow. However, if your team uses software to manage defects, you should be able to implement more robust workflow. For example, the software should allow you to define "state transitions". This identifies how a status can transition from one status to another. In our example, above, you may want to setup these transitions:

   » **Active** - Can only transition to Resolved or Committee
   » **Committee** - Can only transition to Active or Closed
   » **Resolved** - Can only transition to Active or QAed
   » **QAed** - Can only transition to Active or Closed
   » **Closed** - No transitions allowed

   Likewise, the software should also allow you to define what fields (or items) you wish to make required upon different states. In the example above, if the defect is changed to **Resolved**, we may want to require that the programmer enter the **resolution information** (resolution code and description of how they resolved it)**.** Robust defect tracking software will allow you to define the field attributes for each state transition. **Software Planner** (http://www.SoftwarePlanner.com) does this nicely, you can see how this is handled from **Software Planner** by viewing this movie:

http://www.pragmaticsw.com/GuidedTours/Default.asp?FileName=Workflow

4. **Defect Severity** - Another important aspect of defect tracking is to objectively define your defect severities. If this is subjective, team members will struggle classifying the severity. Below are severities that are objective:

   » **1-Crash** - Set when the defect causes the software to crash
   » **2-Major Bug** - Set when there is a major defect with NO workaround
   » **3-Workaround** - Set when there is a defect but it has a workaround
   » **4-Trivial** - Not a major bug, trivial (e.g. misspelling, etc)

5. **Defect Priority** - Similar to severity, the priority for resolving the defect should be objective, not subjective. Below are priorities that are objective:

   » **1-Fix ASAP** - Highest level of priority, must be fixed as soon as possible
   » **2-Fix Soon** - Fix once the priority 1 items are completed
   » **3-Fix If Time** - Fix if time allows, otherwise, fix in a future release

6. **User Acceptance Test Release Template** - Upon entering User Acceptance Testing, it is wise to create a document that describes how your QA process went. Here is a User Acceptance Test Release Report template:

   http://www.PragmaticSW.com/Pragmatic/Templates/UATRelease.rtf

Helpful Templates

Below are some helpful templates to aid you in developing software solutions on-time and on-budget:

- **Project Management Guidelines -** http://www.PragmaticSW.com/Pragmatic/Templates/ProjectMgtGuidelines.rtf
- **Functional Specifications -** http://www.PragmaticSW.com/Pragmatic/Templates/FunctionalSpec.rtf
- **Architectural Overview -** http://www.PragmaticSW.com/Pragmatic/Templates/ArchitectureOverview.rtf
- **Detailed Design** - http://www.PragmaticSW.com/Pragmatic/Templates/DetailedDesign.rtf
- **Strategic Planning Document** - http://www.PragmaticSW.com/Pragmatic/Templates/StrategicPlanning.rtf
- **Test Design** - http://www.PragmaticSW.com/Pragmatic/Templates/TestDesign.rtf
- **Risk Assessment** - http://www.PragmaticSW.com/Pragmatic/Templates/Risk%20Assessment.rtf
- **Weekly Status** - http://www.PragmaticSW.com/Pragmatic/Templates/WeeklyStatusRpt.rtf
- **User Acceptance Test Release Report** - http://www.PragmaticSW.com/Pragmatic/Templates/UATRelease.rtf
- **Post Mortem Report** - http://www.PragmaticSW.com/Pragmatic/Templates/PostMortem.rtf
- **All Templates** - http://www.PragmaticSW.com/Templates.htm
- **Prior Newsletters** - http://www.PragmaticSW.com/Newsletters.htm
- **Software Planner** - http://www.SoftwarePlanner.com
- **Defect Tracker** - http://www.DefectTracker.com
- **Remoteus (Remote Desktop Sharing)** - http://www.PragmaticSW.com/Remoteus.asp

About the Author

Steve Miller is the President of **Pragmatic Software** (http://www.PragmaticSW.com). With over 20 years of experience, Steve has extensive knowledge in project management, software architecture and test design. Steve publishes a monthly newsletter for companies that design and develop software. You can read other newsletters at http://www.PragmaticSW.com/Newsletters.htm. Steve's email is steve.miller@PragmaticSW.com.

Pragmatic Software Co., Inc.
383 Inverness Parkway
Suite 280
Englewood, CO 80112

Phone: 303.768.7480
Fax: 303.768.7481
Web site: http://www.PragmaticSW.com
E-mail: info@PragmaticSW.com