When Test Drives the Development Cycle Bus

Having finished our last project, which ended as a fire drill as usual, the managers of development and test concluded that we didn't ever want to go through that again. All agreed the test team had been riding the development cycle bus long enough.

Our new paradigm has test driving the development cycle bus instead of engineering. The process of change is very difficult in any company, the changes that we have undergone have totally reengineered our Research and Development department. It took courage for the test team to go from passively waiting for whatever bug the engineers decided to fix (or what piece of code they decided to complete) to the active role as facilitator of the entire process. The engineering team has been helped by having test push for dates and specific bug fixes in weekly meetings. In these weekly meetings each person knows that if they do not do their part as promised other people will not be able to deliver either. The process on the whole has brought the entire development team together working toward one common goal "To Release Quality Software".

The test teams first task was to organize and sell the idea to product and project management and a skeptical engineering team. We all looked back at our past release record and knew that we wanted change for the better. All knew this new process would involve a lot of effort to get started but, it was better than doing things the way we always had in the past.

Everyone including engineering, management, and especially test are now fully on board and wouldn't want it any other way. Our innovative formula for success is broken down into the following sections:

- **Establish Daily Events/Goals to Meet Weekly Build Objectives**
- **Make Bug Triage a Weekly Process**
- **Grade the Software With a Two-Tiered Approach**
  - **Grade the Weekly Cycle Build**
  - **Grade Each Module of the Software Weekly**
- **Let Everyone On the Team Know the Status of the Entire Project**
- **Manage The Project With Flexibility**

Establish Daily Events/Goals to Meet Weekly Build Objectives.

During the early phase of the software development cycle, engineering builds the software every week. Closer to the Beta release, the software can and is often built semiweekly or even more frequently. Each weekly build is treated as a release candidate in the sense that the state of the software is always known. If the software has to ship, it is known which areas have been tested and which modules are or are not ready to ship. Both engineering and test are assigned daily deliverables for which they are accountable to deliver to each other. Commitment of delivery by each team as well as each individual allows the build to occur successfully, and improves the state of the software weekly.

All members on the project from test, engineering, project management and documentation attend a daily morning briefing. The Test or Engineering Manager direct the meeting, relaying pertinent information such as the condition of the build and most importantly specifically asking engineering bug fixes promised will make the build deadline. We then go around the room and each person tells what they will be doing for the day. This allows everyone to know what everyone else is doing. Telling the group individual objectives is a type of reinforcement because the commitment seems to be more effective for follow through and delivery. The additional byproducts of the meetings are that it often uncovers important pieces of information and many times matches people up to solve problems together. The meeting is generally held in a small room with only a few chairs so that the attendees will not prolong the meeting but, get to the point.

Each individual owns weekly build objectives always conscious that others are relying on follow through and delivery. Starting this cycle you are essentially practicing making the Gold CD weekly.

Make Bug Triage a Weekly Process

The bug triage process is a weekly meeting where the lead engineer and lead tester agree on the bugs that will be fixed in the upcoming week's build cycle. This is the chance for the lead tester and lead engineer to reach a consensus and address the most severe bugs. In most cases, addressing the most severe bugs enables testing to continue. This also increases test's ability to improve the depth of its reporting on the state of the software each week. Triage also provides a time for discussion and risk analysis between the test and engineering teams.

The test lead will generally start the meeting with comments or questions about the weekly build, the testing effort or technical questions that the engineer is best suited to answer. Test produces a "Hot List", which are the top bugs as designated by the test team. This list is set up in a spreadsheet and includes the following information:

The title of the bug in the bug repository
A bug repository tracking number
The module that the bug resides
The status of the bug (bug verified, fixed etc.)
What build the bug was promised to be fixed
Who owns the bug (engineering or test)

Engineering and test then decide which bugs will be worked on during the week. In addition it is confirmed if the past weeks commitments have been met. A very important item to note is that engineering should not be using their precious time fixing bugs not on the list without the approval of test. If test can anticipate what will be in the build they are ahead of the planning game. Additionally if an engineer has extra time they should fix bugs that are in the best interest of the test team.

The expected outcome of the meeting is test and engineering have addressed the highest priority bugs and decided which ones will be in the next build. This is executed by assigning a specific engineer the responsibility to fix the bug. If an engineer sees their name beside a bug it places necessary pressure to fix it in the allotted time. It is imperative that the list be maintained since many decisions are based on this list. Once the meeting has adjourned test will update the bug repository and begin to build their test suite. The test lead maintains the list, which is always kept on a shared drive so that everyone has access to the information anytime.

The idea here is to weekly prioritize the bugs into a "Hot List" and have weekly fix commitments that everyone agrees to.

Grade the Software With a Two-Tiered Approach.

Grading the software is a two-tiered approach. Grades consist of weekly build cycle grades and module grades. The grades are calculated by using the bug repository counts and a scale everyone is familiar with: A to F, just like school. Grades are distributed to each team member and posted where all can see.

Grade the Weekly Cycle Build.

The test team grades the weekly cycle build. Most bugs are found during the sanity testing of the weekly build. If an engineer introduces showstoppers in the build, which prevent testing, the build receives a letter grade of F. If an engineer breaks the weekly build, it receives the same grade and the engineer who breaks the build has to buy the entire team bagels.

The weekly cycle build report is a way to look at software as a whole and how it has changed over the past week. This report is set up as a spreadsheet and compiled by the test lead through the bug repository and has the following information:

Cycle Build Number
Release Date
Overall Grade
Bugs submitted during the past week broken down by severity
Current open bug counts broken down by severity
Total bugs on the project opened and closed
Number of bugs closed during the past week

The test lead is again responsible for maintaining the build grades which are also kept on a shared drive. This ensures everyone has access to the information anytime.

The key point is everyone knows the overall state of the software at any given time.

Grade Each Module of the Software Weekly.

The software is also segmented into modules. Test also grades each module of the software weekly. Some module's grades will remain static while test recommends engineering to focus on more severe modules. The ultimate goal is to have all modules at a letter grade agreed upon in the test plan. The module grades are calculated using the same counts and scale as the weekly builds.

The module grades are a way to look at software by module and to lend additional insight in to the weekly build grade. Everyone focuses on how individual modules have fluctuated over the past week. An important item to note is that it is imperative to have modules set up in the bug repository correctly, meaning too few or too many modules will skew the module reporting. The size of the module and quantity of bugs are considered when evaluating the modules and assigning grades.

This report is created in a spreadsheet and compiled by the test lead utilizing the bug repository database with the following information:

The Module Name
The Number of bugs by module broken down by severity

The Number of total bugs by module
A weighted average
Total Score
Grade
Comments

The weighted average - We have four levels of severity for bugs and assign scores in the following way:

Showstopper	=	5
High	=	4
Medium	=	3
Low	=	2

We multiply the number of bugs in each module by their respective weight and then add each to get the Weighted Total. This can be easily set up as a formula in the spread sheet. The Total Software Score is calculated by an average of each module.

The Grade is calculated using the following :

E	0 - 30	Fail (Bug count contains showstopper bugs, unstable condition)
D	31 - 50	Below Average (Bug count for module contains excessive severity High bug count, needs work)
C	51 - 70	Average (Bug count for module contains moderate severity high, needs improvement)
B	71 - 90	Good (Bug count for module is low and severity is medium, may be released)
A	91 - 100	Excellent (Bug count for module is low and severity low, "Zero Defect")

The test lead is also responsible for maintaining the module grades, which are also kept on a shared drive so that everyone has access to the information anytime.

The key point is everyone knows the state of each module at any given time.

Let Everyone On the Team Know the Status of the Entire Project.

This type of grading system goes beyond the use of metrics, everyone on the team knows the state of the overall software as well as the individual modules. All on the project are striving for an A and each team member can see the progress and the readiness of the software.

This reinforces we are all working toward the same goal and allows each person to have the same data on the project. Perceptions may be different but, the grades should align most perception differences. Slips and broken builds can never be totally eliminated but, if we all agree where we are, it becomes much easier to determine success and reach the next milestone and the ultimate goal of releasing quality software.

Manage the Project With Flexibility

Finally, once the process is in place, it is easy to accelerate or decelerate the cycles as needed. At the start of many projects we will not start the module grades until we have built the software a few weeks, this is due to unit testing has not been completed and the grades would not reflect the true state of the software. As we get closer to the Beta release we may build three or four times a week depending on the necessary iterations and if any last minute fixes are required.

This formula needs to be adapted for each organization and even each project. In the planning stages each project weekly cycles, release grades and even module names must be evaluated and agreed upon by everyone on the project. We can't give you exact module or cycle grade sheets because even in our organization some Test Leads have adjusted the sheets to more efficiently fit their project.

The Test Manager can not just copy grade sheets and sit back. Each project will be different and require collecting different information. The Test Manager should continually tweak the process and adjust it as needed. It's hard work but, effort well worthwhile when you see the outcome of using this process.

We are not saying that the test should take over in a coup, that would only exacerbate the rocky relationship between engineering and test. As with any change it takes time and buy in. However, there are tremendous benefits and someone has to drive the bus so why not test. Some of the benefits to the test team are having modules fixed in an order that allows the test schedule to be adhered to, this allows testing to be performed more efficiently. Finally, test is always pressed for time toward the end of a project, if they start controlling the software’s destiny from the beginning the end is not so painful.

The hardest part is starting, which includes getting organized, developing the grading criteria, the tools to perform the bug tracking, and of course, selling the idea to management and engineers. In the end, we all win. We are ready for show time and can produce a gold CD with a few finishing touches—just like we’ve done numerous times before in our weekly dress rehearsals.

Cindy Necaise

Cindy Necaise has over 15 years experience with Point-of- Sale software in the hospitality and retail industries. Cindy joined Micros in 1996. Her current role is Test & QA Manager for two POS applications for the Leisure and Entertainment division. Micros L & E products can be found in most major airports, hotels, stadiums, and theme parks around the world. Most recently, Cindy successfully led the testing of a porting project of a POS application from UNIX to Windows NT. She holds a BA in Business Management from the College of Notre Dame of Maryland.