Software Development Bug Tracking – a Case Study:

"Tool isn't user friendly" or " User isn't process friendly"

Leah Goldin / Golden SoLutions, & Lilach Rochell / Nice Systems Ltd.
{goldinl@netvision.net.il; lilach.rochell@nice.com;}

1. Introduction

Today's improvement of software development processes is heavily dependent on supporting CASE tools, which contain the knowledge about the developed products along with process knowledge used to track and verify the process itself. The commercial of-the-shelf (COTS) tools are usually customized to fit the local environment and process. Such customization includes modifying statuses, re-design permissions and transitions, etc.

Using the implemented tools, e.g. bug tracking; the software development team reflects the realities of its processes and products into the tool content. The tool's users, therefore, are a key factor in successful process implementation. This is yet another special case of the truism that quality derives from the human factor.

In this article we describe the implementation of software development bug tracking via WebPT/Continuus, in a development organization of about 200 developers, testers and managers of various ranks. We emphasize the interplay of technology, procedures and human nature, and describe our experiences and lessons on how this interplay affects the continuous quality improvement process. The implementer often hears complaints about the tool not being user-friendly. In some cases, the implementer – committed to the tool – will perceive this as a case of a user being not process-friendly. Both responses may have a high emotional content. Who is right?

2. Background

About two years ago, Nice Ltd. decided to strive for development process improvement, and set a goal of achieving CMM level 2 [1]. Specifically, development Bug Tracking was the first initiative as part of the organizational Software Configuration Management process (SCM). Among other things, this was called for by the increasing size of the R&D department (about 250 people); and increasing complexity of product releases caused by combination of generic and customer specific versions. Continuus + WebPT [2] tools were purchased to support the Nice SCM process along with Bug Tracking, and a CM Manager was hired.

3. Bug Tracking Process

3.1 Bug Tracking Flow – "A Bug's Life" ☺

WebPT Bug tracking tool uses a role-based security system, in which transitions (promotion) of bugs are allowed according to a user's role. A specific role is required for each transition. Any number of roles (including none) may be granted to one person.

In a classic bug life cycle, a tester identifies a bug. She reports it through WebPT; it is now in entered status. (Actually, any role is sufficient in order to report a bug.)

The Testing Team Leader (TL) verifies the bug (which is now in entered status), and transits it into an in_review status. Testing TL should first make sure all fields are correctly filled, and the bug description is clear and sufficiently detailed. Moreover, this Testing TL's transition is actually about verifying that the bug is really a bug and not just an operational misunderstanding. Thus, it is important that the verifier is highly qualified testing person that can effectively filter bugs. Still, this bug "filter" has remained within the testing environment, so "false-alarm" bugs will not disturb the developers. Additionally, this way we centralize and mange the knowledge accumulated within the testing group.

The Development TL assigns the bug (which is now in *in_review* status) to one of her developers, and transits it to *assigned* status. The assignment of bug fixing usually incorporates ongoing development

tasks with R&D work that was not planned (we would like to think that our software is bug free). Since the Development TL is the person responsible for planning and allocating tasks, and has a wide view of all factors, she is the one to allocate development resources for the bug fixing.

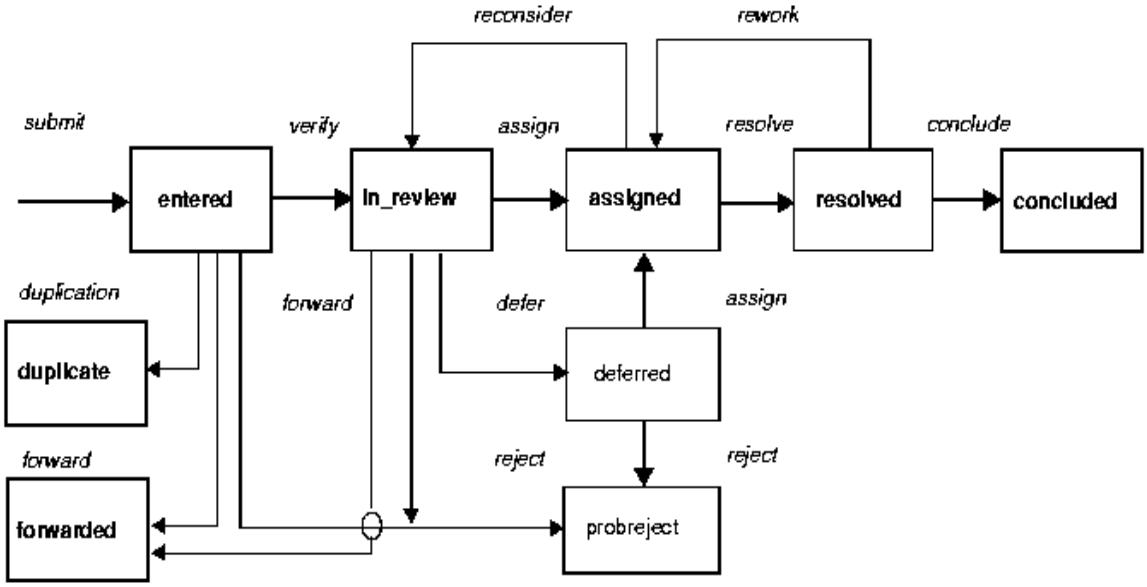

Figure 1: Bug life cycle

Upon fixing the bug and unit-testing it, the developer transits the bug into a *resolved* status.After the original finder-tester re-tests the bug fix, to ensure its complete fix, the Testing TL brings the bug life to its end by transiting it to a *concluded* status.

Of course, many other routes are possible in a bug life, such as rejecting it, from various stages, marking it as duplicate of other bugs, sending it to re-work by the developer, or requesting re-considering its assignments, etc. The whole picture is shown in Figure 1.

3.2 Bug Tracking roles granting

The consideration of 'whom to grant which roles' raises two issues. First, it was subject to many group-specific variants and preferences. Crystallizing these decisions also raised many oppositions and disagreements within the group/teams involved. A second and interesting issue: Identifying clear ownership of bugs in certain statuses revealed situations in which managerial responsibilities of bugs were sometimes vague, ambiguous, or shared between two different position-holders (This will be further discussed later at the process improvement section).

We have tried to both fit the solution to the specific team while keeping 'correct' concepts and methods.

4. Bug Tracking Process Establishment

The definition of the Nice Bug Tracking process went through several stages, aiming towards minimizing resistance while achieving a full consensus.

First stage was setting a 'Testing Forum' that through brainstorming and discussion gathered preliminary requirement definitions of the desired Bug Tracking process and tool. Second stage was "thinking" meetings of each group representatives, to identify specific R&D group needs prior to implementation. Last stage and most important, not to say vital to the process, was achieving a level of involvement from the relevant senior management (R&D VP & R&D Development Director). This also includes ensuring their commitment to implement the system in the R&D department.

A very outstanding contribution to the Bug Tracking process establishment was the decision of the Project Manager: Starting from 1/1/2000, only bugs reported via WebPT will be formally handled and tracked. Looking back, that Project Manager's call set a deadline for starting actual work with the WebPT, and stopped endless discussions regarding required tool customization. Accumulating additional WebPT issues based on real work (this time), led eventually to a very little of tool customization.

5. Process Improvement

The most interesting observation of the Bug Tracking tool implementation was to see the learning curve happening in real-time, both from tool and process views. The main issue was that this was the first Support tool introduced to R&D department for managing development bugs. Thus, the whole organization was taking a big step forward in the Bug Tracking process visibility and decision-making. It was an interesting lesson to track progress stage by stage:

5.1 Technical tool-use issues

In this stage we saw users' common learning issues, problems caused by misconception of the tool, or even more commonly, trying to impose conceptions developed using other tools, on Continuus/ WebPT. This is also the stage in which users reveal the most emotional antagonism. At Nice, specifically, in this stage we had a major upgrade of Continuus software, which was a 'road-bump' in the adoption process. This kind of setback should be expected, and handled with patience.

5.2 Progression in the horizontal axis - bug status transition

In this stage we saw the progression of the tool's penetration into the live texture of the working R&D department. The most progress was achieved by people who at that time were involved in promoting bugs to their next status.

a) After the 1/1/2000 deadline, in which bugs were treated only within Continuus/WebPT, the first roles that had to deal with the WebPT were the testers, who had to either gather all bugs (from isolated Excel sheets, for example) or accumulate all their knowledge of bugs into the WebPT records. After this stage was over (and we learned how to automate it), bugs were 'stacked' at the *entered* stage.
b) The Testing Team Leaders had to first set a policy of verifying bugs and then learn how to do it technically with the WebPT. The bug verification activity was a totally new and was accompanied by some change resistance, mainly in groups with highest number of bugs (!), complaining that it is "time consuming". The benefit of the verification "filter" was only later appreciated by those groups when it helped them to prevent "false alarm" bug from arriving to the developers and waste their time.
c) Similarly, Development Team Leaders needed then to realize their mechanism of assigning bug's tasks (changes) to their staff, and then to learn how to do it via the WebPT tool (transition bugs into *assigned* status). They also needed to audit their ways of monitoring their staff's work, and use Continuus/WebPT.
d) Developers, then, needed to take responsibilities of their bug fixes by transiting bugs into *resolved* status. Developers had to get used to the fact that from now on, there is a 'stamp' of their name (and accountability) on each source change they make, or any bug transition.
e) Finally, the Testing group had to put the final stamp of bug fix approval by transition bugs into *concluded* status. Bugs were "stuck" in *resolved* state, meaning they were not transited to the *concluded* state, if they could not be reproduced. This raised another question concerning the development life cycle: Making sure bugs (and bug fixes) are reproducible.

5.3 Progression in the vertical axis - hierarchal organization (roles)

We could also identify progression in another dimension: the managerial responsibility hierarchy. At first, the quick learners were the 'production workers' (Developers and Testers), which were doing 'the real work', meaning reporting bugs, and producing them. They struggled with the tool, learned it, raised real questions of working policies with the tool, and suggested valuable directions.

Then, the managerial level, Development and Testing TLs, had to deal with the bug content via the WebPT tool reports. They learned to plan (according to bugs found) and supervise the work through the

tool, evaluate its quality, re-plan work plans, etc. In their turn they also revealed other sides of the tools possibilities, like ease of use of certain actions, co-operability between teams, etc.

Group Managers were next in line to realize the tool place in their ~30 workers' group. They discovered the 'polling' concept of the tool; demanded more extended reports; sought for mails triggered automatically (like mail announcing all 'show stopper' bugs to all team leaders, etc).

Other roles' holders, organization-wide, discovered their interests as well: Project Mangers learned to produce sophisticated reports of the release status and bugs overall behavior. The R&D Software Director found the tool useful in evaluating many things, from specific person's performance to inter group productivity comparison. The VP R&D could use Continuus/WebPT to have 'just-in-time' reports produced for him (using pre-defined queries on an intranet site, accessible from anywhere).

5.4 Progression in the depth axis – emotionally dealing with the process

In the first stage of SW bug tracking implementation, can be characterized as the ***Euphoria Stage,*** everybody involved has a fantasy vision of how it can solve practically everything. Then, the tool merely reflects the reality, which is (in some areas) not so glamorous. Reality, that is, obviously, Bugs' situation, but also and even more important, reflects the decisions making mechanism and culture of what, when, by whom and how to fix a bug, for better and for worth. Some people slip into *denial* status. Others use it to improve the process.

As responsibilities vs. activities balance, (i.e. bug verification, work assignment, bug conclusion), becomes transparent, some groups tend to 'roll responsibility down', while others tend to centralize it. Perhaps, the best-fit solution is somewhere in the middle?

As 'tool-is-a-tool-is-a-tool', some groups tend to find creative solutions, which promote them into a more mature stage. Others complain about the tool: "till these flaws are fixed, this tool is unworkable". Other wonder: "How did we ever managed to work before?"

Even after a very successful initial implementation, improvement tends to arrive into a ***stagnated stage***. Does the fact that our quality assurance isn't so sure is due to our process that need improvements, and the tool should be updated accordingly? Or, perhaps, is it because the QA system could be of a better quality? In other words, does the tool limit us, or our process maturity limits the tool usage?

Recognizing and being aware of this emotional factor, can substantially help the implementer to ease and resolve conflicts, and to provide more solid support to the people and the organization throughout the process of coping with the change.

6. Analysis & Discussion

6.1 Conciliating the expectations of company, employee and tools

When implementing a development process-supporting tool in a company, there is a need to walk on a thin line. On one side, there is a need to ease people's minds; to tell them that the tool's purpose is to serve them: To make their work easier, i.e., more automated, less error prone, and richer in functionalities. Additionally, the tool can and should match the company's way of doing things, and should be tailored to the company's policies, procedures, processes and temperament. On the other hand, and contradicting (or not), during implementation we suggested to people, to also examine all those factors, and if there's anything they have always wanted to change, or if they have a 'vision' – now is the time to try to implement it, as this is a chance for house cleanup and renewing.

6.2 Group Managerial Culture

Tool's implementers need to quickly identify different 'styles' of working groups, and to adjust the implementation itself to the group's 'culture'. Some groups are very hierarchal. Usually, there is one (very talented) contact person, who addresses all the issues; sets the guidelines, tests it and then pass the knowledge to the group. In this kind of group, implementation is slow but safe, easy, and invokes less resistance. The weakness, though, is that the solution reflects one person's way of dealing with problems and issues, which might be limited or biased.

On the other hand, other groups are more individualistic in spirit. Here, implementers will find all kinds of resistance/feedbacks from virtually everybody. In such a group, the result may be creative and sophisticated, but there will be numerous deadline-threating obstacles on the way, technical and emotional.

None-the-less, the organizational policy of assuring quality, requires all the R&D group to comply, even if it does not always fits their personal view. Thus, one should keep a delicate balance of fulfilling all the R&D Managers touch & feel "dreams" of bug management, while helping the R&D staff towards fruitful acceptance of the organization policy.

6.3 Accountability

Managing the Bug Tracking via the tool exposed some real issues concerning accountability: identifying precisely whose bug it is, who is responsible for fixing it, and who is holding it, thus "causing" delay of the product release. The managerial visibility via WebPT tool support has sharpened accountability issues, which in turn assisted in improving the Bug Tracking process (and the product quality).

In the past, before the WebPT, Testers were used to talk directly with the developers, "mentioning" the existence of a bug and "agreeing" to fix it. This way of managing bug fixes led to uncertainty in tracking the bug status, the actually fix of the bug, and amorphous planning and tracking of the R&D work for the next product release. Also, by implementing this structural tool, Team Leaders had to assume responsibility both for planning the work and (by verifying it) to its quality.

6.4 Tool vs. Style

It is long known in music that there are two-way influences between the style's idioms and the instruments' evolution. Very briefly, the ideals of the current style had a great effect on the design and progression of the musical instruments. At the same time, instrument limitations or new technological improvements led to tremendous changes in the style's ideals. Making the implied analogy, we, as process change agents, can use this understating to strengthen the implementation progress. In the beginning, the current style of doing things will affect the desired design of the tool. We should be (and almost could not escape) obeying that. But in turn, the tool's technological functionality will eventually lead to change of concept in regard to the ways things should be done. This human-nature-pendulum does happen, and we might as well be aware of it and harness it to our aim.

7. Summary

In this article, we try to present new angle of looking at the fact everybody knows – the human factor is #1. This new angle delineates a triangle with the vertices held by Procedures, People, and Tools. Each side of the triangle represents a bi-directional interaction. Conflicts, as they rise, might be well disguised. (People-Process sourced conflict may convincingly look like a Tool-People disagreement, etc.) Implementers, should be aware of that, and try to identify the real source of the conflict, (although they might choose to solve the problem by a work around such as technical solution to a Human-Procedure difficulty). True identification might lead to better conflicts managing.

For example, if human-tool friction is revealed, it might be wiser not to attempt to judge who is right (the tool or the human). If we are pre-supposing that the problem is in the People-Tools interaction, and is thus mainly technical- such a technical problem can be addressed either by improving the tools, or by training the users. However, this disagreement may not technical at all. In that case we'd really identify the problem on the People-Procedures interaction– what the user is actually saying, "the tool is forcing me to perform a procedure I do not like". If so, technical solutions might not fail, but one must realize that the problem is in the tougher realm of creating human-friendly processes and training process-friendly humans. The good news is that the tool has already delivered a major benefit in unearthing this conflict. Further good news: It's quite likely that the solution, once found, can be effectively implemented by the tool.

8. References

[1] Capability Maturity Model (CMM), CMU/SEI-93-TR-25, Carnegie Mellon University, Feb. 1993.
[2] Problem tracking and Task Reference, Continuus/CM, PTTR-041-011.