Software Development Bug Tracking:

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

Leah Goldin[1], Lilach Rochell[2]

[1] GoLden Solutions, P.O.B 6017
44641 Kfar-Saba, Israel,
L_goldin@computer.org
[2] NICE Systems Ltd., CEM, P.O.B 690,
43107 Ra'anana, Israel
Lilach.rochell@nice.com

Abstract. 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. Often times, the implementer 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?

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, permissions, etc.
By using the implemented tools for Bug Tracking (BT); 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.

1.1 Scope

About two years ago, the company decided to strive for development process improvement, and set a goal of achieving CMM level 2 [1]. Specifically, software development BT 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 ver-

sions. Continuus + WebPT [2] tools were purchased to support the company's SCM process along with BT, and a CM Manager was hired.

1.2 Motivation

The importance of BT is obvious. Once a software product is released to customers, all organizations adhere to some process of managing customers problems for the ultimate purpose of customer satisfaction.
However, it is more challenging to start tracking bugs during the software development phases, since Software Engineers are not so cooperative in tracking down their faults. It threatens both their creativity and professional ego.
The company is a product-based company, meaning that it produces modular qualified products that are sold to different customers with different needs assembled as different solutions. Since the products are generic, and declared as general available – GA, a great emphasis is given to their quality. Thus, Intensive testing and controlled BT come into play starting at the early software development phases, much before the product is released as GA.

1.3 Brief Review of Bug Tracking Methods and Tools

Bug reporting and tracking is an integral process in software development and maintenance. When performed correctly, it facilitates follow-up to completion of all software corrections, and accurate reporting on the status of the software product quality
A software bug or defect report is a *perceived problem* with the software product. Each problem report must be evaluated to determine 1) if there is an underlying software defect, and if so, 2) if, how and when to correct that defect. There are a variety of commercial tools that support software BT process, including ClearQuest [3], WebPT [2], MetaQuest [4].
Early work reported on bug or defect tracking mainly focus on measuring defects and collecting defect statistics as a criterion for software quality. Many organizations want to have an estimation of the number of defects in software systems, before they are deployed, so the delivered quality and maintenance effort are predictable. Fenton and Neil [5] provide a critical review of the large literature and the state-of-the-art of the numerous software metrics and statistical models have been developed.
More recently, the software engineering community has been paying attention to the development process [1], and specifically the BT process as a driver of software process improvement [6]. Monterio et al [7] describe an empirical software engineering pilot project using a software defect reporting and tracking system SofTrack, that help software managers to better understand, control and ultimately improve the software process. Wohlin and Wesslen [8] present a study of software defect detection that provides valuable insight into software defect detection in relation to the Personal Software Process (PSP). The results are interesting both for the PSP, and for understanding software defect detection itself for the sake of improving the reliability of software. Thomas et al [9] advocates more research into the human side of soft-

ware quality management. They draw together two separate threads of research: software quality management, and the motivation of software developers.

1.4 Outline of the Paper

Once the software development BT scope, motivation and review have been presented above, the rest of the paper describes the BT process that was chosen, and analyses experience gained from implementing both the BT process and tool.
Section 2 gives the definition of the company's BT process, including the life cycle flow and roles granting at each transition within the "bug's life". Section 3 describes the BT process establishment at the company. Section 4 considers the evaluation of the BT process improvement and progress in various dimensions, i.e., tool usage issues, role's granting complexity, organizational hierarchy involvement, etc. Section 5 analyses the success factors of the BT process implementation and improvement, i.e., organization expectation, managerial culture, accountability, etc. Section 6 provides results obtained from this BT process implementation along with some lessons learned. Finally Section 7 and 8 summarize and describe future work respectively.

2 Bug Tracking Process

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

WebPT BT tool uses a role-based security system, in which transitions (promotion) of bugs are allowed according to a user's role. In a classic bug life cycle, a tester identifies a bug and reports it through WebPT (*entered* status). The Testing Team Leader (TL) verifies the bug, and transits it into an in_review status. It is important that the verifier is highly qualified for few reasons: bugs' 'quality of details and depth is increased, "false-alarm" bugs do not arrive to developers, and it promotes accumulating knowledge within the testing group.
The Development TL assigns the bug to one of the developers, and transits it to *assigned* status. 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. 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. (Figure 1).

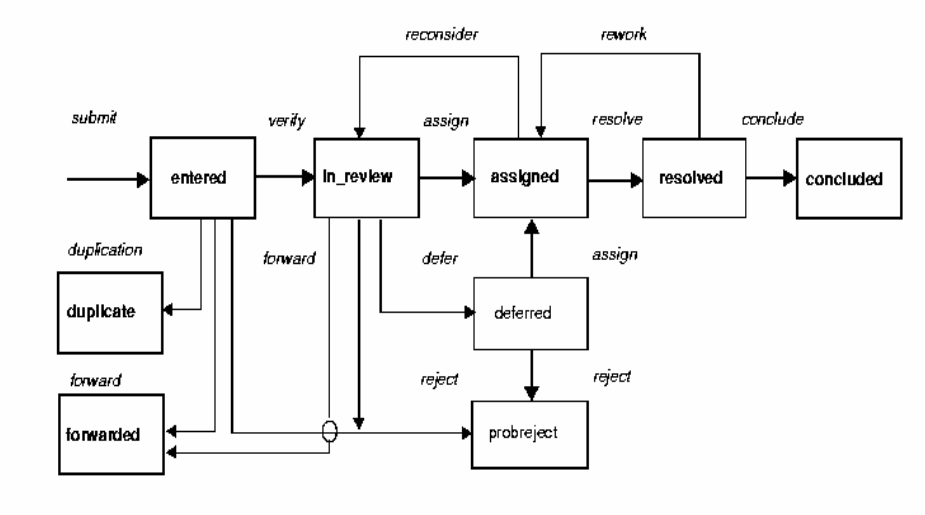

Figure 1: Bug life cycle

2.2 Bug Tracking Roles Granting

The consideration of 'whom to grant which roles' raises two issues. First, it is 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.

3 Bug Tracking Process Establishment

The definition of the company's BT 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 BT 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.

4 Process Improvement

The most interesting observation of the BT 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 BT process visibility and decision-making. It was an interesting lesson to track progress stage by stage:

4.1 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 the company, 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.

4.2 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. People who at that time were involved in promoting bugs to their next status achieved the most progress.

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 (TLs) 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". Those groups only later appreciated the benefit of the verification "filter" when it helped them to prevent "false alarm" bug from arriving to the developers and waste their time.

c) Similarly, Development TLs 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.

4.3 Progression in the Vertical Axis - Hierarchal Organization

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 TLs, 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).

4.4 Progression in the Depth Axis – Emotionally Dealing with the Process

In the first stage of software BT 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.

5 Analysis & Discussion

5.1 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.

5.2 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.

5.3 Accountability

Managing the BT 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 BT 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, TLs had to assume responsibility both for planning the work and (by verifying it) to its quality.

5.4 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.

6 Results and Lessons Learned

A year after the BT process and tool were used, significant quality improvement both in software and process was proved. Comparing version X of software that was re-

leased before the BT process was implemented, and version Y that was released a year after the controlled BT process and tool were used revealed the following data:

1. 'Known bugs' at release time were reduced by 60%!
2. 'Known bugs' severities profile has improved; 80% less 'Severe' bugs'.
3. The amount of bugs arrived from customers after release decreased by 50%.

Other lessons learned about quality factors were observed, but not measured:

1. Mangers benefited from the ability to monitor and follow up work plan and bug statuses, on 'real time' basis.
2. In order to be declared as 'concluded', bug fixes had had to be tested systematically on all operating systems, languages, configurations, etc.
3. Response time to customers' reported problem reduced significantly.
4. R&D department involvement in customers' bug situation, strengthen the relation between R&D personnel and customers.
5. Due to increased awareness of bugs, reported and approved bugs were 'copied' to future releases development streams, and therefore were reported, fixed and tested before products arrived to customers.

7 Summary

In this article, we try to present new angle of looking at the known fact– the human factor is #1, which delineates a triangle with the vertices held by Procedures, People, and Tools. Each side of the triangle represents a bi-directional interaction. Conflicts might be well disguised, i.e., People-Procedure sourced conflict may convincingly looks like a Tool-People disagreement, etc. Aware implementers might choose to solve the problem by a work around (such as technical solution to a Human-Procedure difficulty), but true identification might lead to better conflicts managing.

For example, if People-Tool friction is revealed, it might be wiser not to attempt to judge who is right, i.e., 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 tool can effectively implement the solution, once found.

8 Future Work

Implementing the software development BT process via the WebPT tool in the software R&D department was a huge step forward, which enabled clear visibility of the software quality, and improved the planning and estimating the R&D maintenance effort internally.

Future work is planned to improve the interfaces of the software development BT to the other departments, such as Customer Services (CS) and Operations. CS have their own infrastructure for accumulating customer cases, i.e., problems, complaints, enhancements, etc. These customer cases have to be analyzed and mapped, many to many, to the software development BT process and repository. So, future work will focus on first defining the process interfaces between CS and R&D wrt problems and bugs management, and then integrate the relevant supporting tools. Operations department is responsible for installing and testing (ATP) the software before shipping the product to the customer. Thus, it identifies problems that have to be reported to the R&D department, no matter if the product is corrected before shipment or not. Again, future work will initially focus on defining the process interfaces between Operations and R&D wrt problems management, and then move to integrating relevant supporting tools.

Another work is planned to improve the fault analysis process, in order to help both CS and R&D to better identify the software problem source for each of the reported customer cases. This can tremendously improve the response to customer cases as to known software bugs, and enable more efficient bug fixes, which will provide better product quality.

9 References

1. Paulk M.C., Weber C.V, Garcia S.M., Chrissis M., Bush M.: Capability Maturity Model (CMM), CMU/SEI-93-TR-25, Carnegie Mellon University, Feb. 1993.
2. Continuus/CM: Problem tracking and Task Reference, PTTR-041-011, 1999
3. Rational: ClearQuest, http://www.rational.com/products/clearquest/index.jsp
4. MetaQuest : http://www.metaquest.com
5. Fenton, N.E., Neil, M.: A critique of software defect prediction models, IEEE Transactions on Software Engineering, Vol.25 Issue 5 (1999) 675 - 689
6. McConnell, S: Gauging software readiness with defect tracking, IEEE Software, Vol. 14 Issue 3 (1997) 136, 135
7. Monteiro, A., Almeida, A.B., Goulao, M., Abreu, F.B., Sousa, P.: A software defect report and tracking system in an intranet, The Third European Conference on Software Maintenance and Reengineering Proceedings, Amsterdam Netherlands (1999) 198 – 201
8. Wohlin, C., Wesslen, A.: Understanding software defect detection in the Personal Software Process, The Ninth International Symposium on Software Reliability Engineering Proceedings, 1998. Paderborn Germany (1998) 49 – 58

9. Thomas, S.A., Hurley, S.F., Barnes, D.J.: Looking for the human factors in software quality management, International Conference on Software Engineering: Education and Practice Proceedings (1996) 474 – 480