Metric-based Tracking Management in Software Maintenance

TANG Li MEI YongGang
Department of Telecommunication Engineening
Xi'an University of Post and Telecommunications
Xi'an, China
tangli75@yeah.net
mygang@xupt.edu.cn

DING JianJie
School of Information Science and Technology of Northwest University
ShaanXi Education Institute
Xi'an, China
dingjianjie@yeah.net

***Abstract*—As the information technology industry gains maturity, the number of software systems having moved into maintenance is rapidly growing. Software maintenance is a costly, yet often neglected part of the development life-cycle. A software product maybe has been modified several times for different reasons, but this process don't be efficient manage, so resulting in the software been discard in advance. This paper is motivated by a desire to develop a more practical model to track and manage the maintenance process, to support the maintenance task. The maintenance request form (MRF)'s submission means the start of maintenance activities. One of important tasks is tracking the state of MRF, to control activities in the current environment of large and complex applications. And the other character is import measure into the model, to offer information to help organization control and processing their activities.**

Keywords-software maintenancet; metric; tracking management

I. INTRODUCTION

Software maintenance has been an important issue since the first computer program was written, although not much emphasis was placed on it. History shows that typically 50-80% of the total budget for a software system is spent on maintenance [1]. Many organizations are faced with the same issues that maintenance activities has occupied the mostly effort of their work especial when increasing complex software be produced. In addition to just as system size and complexity, insufficient or poorly written documentation, a manual or undefined software process, and shrinking budgets make maintenance activity difficulty. As a result, almost 50% of a maintainer's job is spent just trying to understand what a system is really supposed to do ([1] [2]). Too often, time and mission constraints cause a system problem to be remedied by patching (quick fixes) instead of correctly and efficiently solving it. In fact, a hidden trouble is disorder management in maintenance process, it makes more difficulty to understand software and bring worse affect in next modification. Software must be discarding through such a few times of modification because need more time and more cost to understand it.

Based above reasons, we build a practical software maintenance model that will solve some problems and improve maintenance process at a certain extent.

II. PRACTICAL SOFTWARE MAINTENANCE MODEL

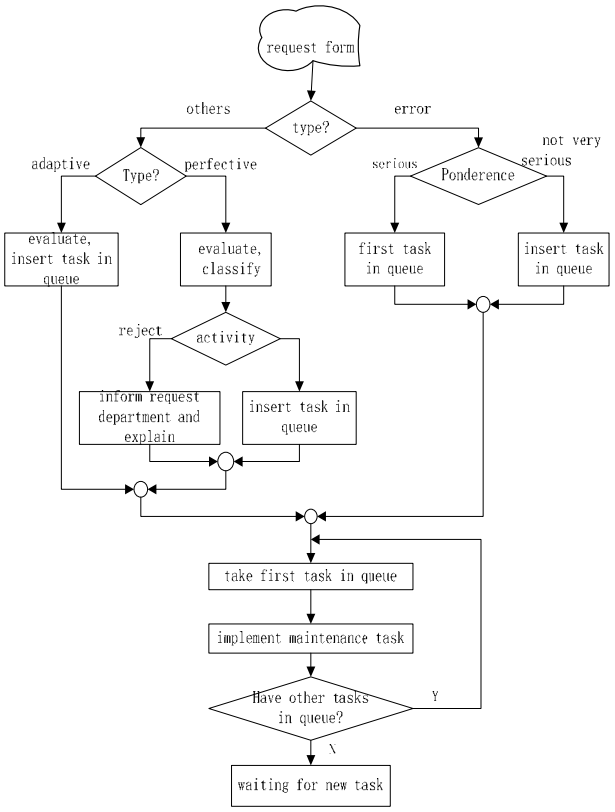

Figure 1 Maintenance Flow

A great deal effort is now being devoted to the study of the maintenance model and maintenance flow. Fig 1 shows the maintenance flow of a maintenance organization. Most maintenance task is been operate according to this flow. But we don't know who take charge analysis, modification and review from Fig 1. As to the request form, its state not been marked. These disadvantageous factors result in tangly

978-0-7695-3987-4/10 $26.00 © 2010 IEEE
DOI 10.1109/ETCS.2010.571

management because it is difficulty to track and monitor maintenance process. On the other hand, we should pay attention to collecting data, analyzing result in process. In a word, software measure and statistical process control (SPC) technology should apply to improve maintenance process. In practical maintenance model, we describe the roles in maintenance team, state of the MRF and some measures in the software maintenance.

A. Roles in Maintenance and MRF's States

For better understanding our model, we define four roles in maintenance organization: user, coordinator, decision-maker and maintenance operator. In fact, in our actual circumstance, roles classify maybe not so clearly. Because MRF is a core of maintenance activity, monitoring the state of MRF is a prime task. Basing on the maintenance flow, we define six states to mark the current state: accepted, analyzing, waiting, maintaining, reviving and finished. Fig 2 shows the MRF's states.

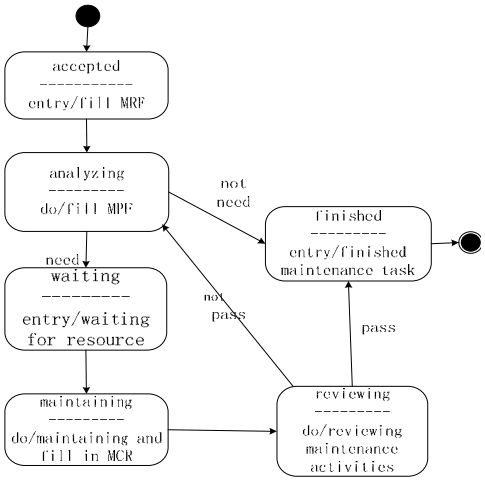

Figure 2 MRF's state chart

First, user should put in a question report to coordinator who answers for user's question. Coordinator try to understand the problem as expressed in the user's language. Then, he make out formal MRF which include a description of how the system work now, how the user wants the system to work, and what modification are needed to produce the changes. MRF been marked "accepted". Usually, in a MRF should contain follow other items: request department, submit time, maintenance type, and so on. Then coordinator offer MRF to the decision-maker.

Decision-maker accepts the MRF indicate that MRF's state has turn to "analyzing". Decision-makers are aided by understanding what happens to systems over time. They are interested in changes in size, complexity, resources, and ease of maintenance. They need estimate the impact of the modification. If request of the user is unreasonable indicate that MRF been rejected, MRF's state turn to "finished". If not, Decision-makers establish maintenance plan form (MPF) base on MRF. Usually, MPF contains follow items: decision-maker, maintenance operator, plan finish time, plan cost, maintenance type etc. All analyzing wok has finished, MPF been offer to maintenance operator.

Maybe maintenance resource don't available, MRF's state is "waiting", it been insert maintenance queue in term of rule.

When maintenance operators have MPF, MRF's state turn to "maintaining" at same time. After finished task, operator must fill in maintenance change report (MCR). MCR should contain software name, decision-maker, finish time, total cost, and modification module. MCR was put in decision-maker.

Decision-maker or review team need review the maintenance activity, then MRF's state turn to "reviving".

MRF's state turn to "finished" if review has passed, if not, MRF's been remark "analyzing" until it turn to "finished".

Although maintenance flow and activities is simple-looking and familiar to us, but our performance is worse. Most of the projects started quite early and were completed with large delay with respect to the final deadline. Clearing obligated distribution and effective control maybe improve process. For instance, we should check maintenance resource if a MRF keeps "waiting" for long time. In a word, control MRF's state help us understand maintenance process and monitor all actor work at a certain context.

B. Metric

Software measure has been applied into many facets, more and more organizations are aware of that it is necessary to usage software measure in their software processes. The Software Engineering Institute has published the capability maturity model integration (CMMI) that can be used to rate an organization process maturity on a five levels scale. In software organization, moving an organization from one level to the next one depends on the capability of the organizations software process to address key practices that accomplish key process areas goals. At level 3, metrics are collected, analyzed, and used to control the process and to make corrections to the predicted costs and schedule, as necessary. The emphasis at level 3 of the CMMI is on well established process and metrics [3]. At level 4 and level 5, metrics work should be more elaborate. In our maintenance process, it's not enough to mange flow and track MRF's states because these work don't make great sense in improving maintenance process. We need measures help us to resolve more questions. First, there are measures that help us to understand what is happening during development and maintenance. We assess the current situation, establishing baselines that help us to set goals for future behavior. In this sense, the measurements make aspects of process and product more visible to us, giving us a better understanding of relationships among activities and entities they affect. Second, the measurement allows us to control what is happening on our projects. Using our baselines, goals and understanding of relationships, we predict what is likely to happen and make changes to processes and products that help us to meet our goals. Third, measurement encourages us to improving our process and product ([4] [5]). It is important to manage the expectations of those who will make measurements-based decisions.

Measurement is a common and necessary practice for understanding, controlling and improving our environment [6]. "You cannot control what you cannot measure." (DeMarco, 1982). In software organization, different roles interesting in different information ([7][8]). Senior management lean to attention Table 1 because it offer information which can help them control overall situation. There are three measure names in the table 1: average time of a MRF, average cost of a MRF and number of MRF in a week. What information these measures can bring to us? Average time of a MRF maybe reflect efficiency of a maintenance team. It's necessary to collect data like longest time turnover of a MRF or highest cost of a MRF because it's important to control process. Number of MRF in a week can reflect the state of the maintenance organization. We use control chart display the measurement result. If trend exceeds our scope, we should explore causation and define next plan. If in a period time, new product delivery maybe leads to high number of MRF, we should add hand in maintenance team. We also consider a perfective maintenance for a software if many MRF from it.

TABLE I. MEASURES INFORMATION WHICH SENIOR MANAGEMENT INTERESTED

Measure name	*Measure item*	*Measure model*	*Show style*
Average time turnover of a MRF	Ti (time turnover of No.i MRF i=1,2,3,…n)	（T1+T2+T3+…+Tn)/n	Numerical value(day/a task)
Average cost of a MRF	Ci(cost of No.i MRF i=1,2,3,…n)	（C1+C2+C3+…Cn)/n	Numerical value(dollars/a task)
Number of MRF in a week(one software)	The number of MRF in a period time	The number of MRF in every week of a period time	Control chart

TABLE II. MEASURES INFORMATION

Measure name	*Measure item*	*Measure model*	*Show style*
Percentage of task which not finished in time	N: total number of task n: number of task which not finished in time	n/N	Numerical value (cake chart)
Percentage of task which exceed budget	N: total number of task n: number of task which exceed budget	n/N	Numerical value (cake chart)
Percentage of every maintenance type	n1:corrective maintenance n2:adaptive maintenance n3:perfective maintenance n4:preventative maintenance	ni/(n1+n2+n3+n4)	Numerical value (cake chart)

As a middle-lever leader, maybe a leader of maintain team, he also interesting in Table2. Percentage of task which not finished in time and percentage of task which exceed budget provide information to adjust our budget and make more accurate prediction. Percentage of every maintenance type helps us in control of software product. Of course, there are many measures in maintenance process but not list here.

Collecting data and analyzing result are critical in measure activities. In the model, most of data are from MRF, MPF and MCR. So the content of the form must true and accurate. Data analyzing can help organization to make decision. SPC is a most important technology because Software organizations have started to appreciate the value of applying SPC techniques ([9] [10]). Late and over budget software procurements are well known as large scale software problems. The use of SPC methods can determine the process capability of sustaining stable levels of variability, so that processes will yield predictable results (Florac and Carleton, 1999). This enables to prepare achievable plans, meet cost estimates and scheduling commitments, and deliver required product functionality and quality with acceptable and reasonable reliability. In a short, SPC techniques can determine sources of variation distinguishing among variations caused by normal process operation and variations caused by anomalies in the process.

III. FUTURE RESEARCH

It's a key that software tool support for maintenance to achieving maintenance productive its gains. In addition to improving productivity, the use of these tools may contribute significantly to improving the quality of the software being maintenance. But by now, most of tools for maintenance are focus on code analysis, instead of management. So the next of the work is developing a tool according of the above model. This tool provides a platform for maintenance organization and facilitates coordination of all maintainer. In addition to it can monitor every maintenance task and provides data such as cost, progress and employer's performance. It will contribute to improving maintain process and increase productivity.

IV. CONCLUSION

In this paper, we have analyzed the states of the software maintenance at first, and indicated that essential problem is inefficient management in the maintenance process. It leads to low maintenance production and make subsequent work more difficulty and come into vicious circle.

Certainly, there are many works of technology and management to do for the improvement of maintenance process. In this paper it emphasizes particularly on management issue, and proposes a practical maintenance model. Hanging together of the tool which is been developing, we define four roles in this model: user, coordinator, decision-maker and maintenance operator. They cooperating each other to accomplish maintenance task. One of the task is we should track the state of the MRF's states at any moment. Basing on the maintenance flow, we assigned six words for mark the MRF's state: accepted, analyzing, waiting, maintaining, reviving and finished.

Anther character is we impart metric to the model. CMMI that can be used to rate an organization software process maturity, and the emphasis at level 3 of the CMMI is on well established process and metrics. In addition to it is not enough to just track MRF's state. We need more information to support our maintenance process, like progress, cost, and staff performance, etc.

The next step of task is to developing a tool that based on practical maintenance model to help maintenance process.

ACKNOWLEDGMENT

It's a pleasure to thank National High-Tech Researched Development Plan of China under grant No.2007AA010305

REFERENCES

[1] C. McClure. The Three Rs of Software Automation. Prentice Hall, Englewood Cli_s, NJ 07632

[2] J. Hagemeister, B. Lowther, P. Oman, X. Yu,and W. Zhu. An annotated bibliography on software maintenance. ACM SIGSOFT Software Engineering Notes, 17(2):79-84, April 1992

[3] M.C. Paulk, B. Curtis, and M.B. Chrissis et.al.Capability maturity model for software. Technical Report CMU/SEI-91-TR-24, Software Engineering Institute, 1991

[4] A. Brown and M. Penedo. An annotated bibliog.raphy on integration in software engineering environments. ACM SIGSOFT Software Engineering Notes, 17(3):47-55, July 1992.

[5] Shari Lawrence Pfleeger, Software Engineering Theory and Practice (Second Editin), 464-506,Higher Education Press,2001

[6] Norman E.Fenton, Shari Lawrence Pfleeger. Software Metrics A Rigorours & Practical Approach. 24-71.Tsinghua Education Press.2003

[7] Andrea De Lucia , Eugenio Pompella ,Silvio Stefanucci, Assessing the maintenance processes of a software organization:an empirical analysis of a large industrial project. The Journal of Systems and Software 65 (2003) 87–103

[8] Aversano, L., Betti, S., DeLucia, A., Stefanucci, S., 2001a. Introducing workflow management in software maintenance processes. In:Proceedings of International Conference on Software Maintenance,Florence, Italy. IEEE Computer Society Press, Los Alamitos,CA, pp. 441–450.

[9] Aversano, L., Canfora, G., Stefanucci, S., 2001b. Understanding and improving the maintenance process: a method and two case studies.In: Proceedings of the 9th International Workshop on Program Comprehension, Toronto, Canada. IEEE Computer Society Press,Los Alamitos, CA, pp. 199–208.

[10] .Basili, V., Briand, L., Condon, S., Kim, Y.M., Melo, W.L., Valen, J.D., 1996. Understanding and predicting the process of software maintenance releases. In: Proceedings of 18th International Conference on Software Engineering. IEEE Computer Society Press, Los Alamitos, CA, pp. 464–474.