http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 11 (7): 751-759, 2012
ISSN 1812-5638 / DOI: 10.3923/itj.2012.751.759

Scheduling Quality Related Activities in Incremental Software Development Projects Based on Monte Carlo Simulation

Bin Xu, Jie Chen, Yujia Ge, Zhixian Chen and Yun Ling
College of Computer and Information Engineering,
Zhejiang Gongshang University, Hangzhou 310018, China

Abstract: Quality is very important in software development project especially for those related to medical treatment, aerospace, finance and public infrastructure. Quality related activities including feasibility analysis, requirement review, architecture inspection, code review, software testing and auditing run through the entire software development life cycle. Quality related activities are a set of umbrella task corresponding to the defects contained inside the requirement specification, documents and applications. While quality related activity may account for nearly one third or even a half effort and duration in typical software projects, it has been considered as the bottleneck to deliver the software applications in timely model. Based on the effort analysis on the defect finding in incremental software development projects, this paper introduces the effort model of the quality related activities, the calibration based on local historic incremental software projects and detail scheduling method including the efficiency matrix, capability matrix and the definition of the constraints and the goal and suggests scheduling quality related activities while balancing time-to-market and high quality based on Monte Carlo simulation. Pilot in some incremental financial software projects shows that the suggested method will benefit the scheduling of quality related activities in the incremental software development projects.

Key words: Software testing solution, effort optimization, defect removal efficiency, metric calibration

INTRODUCTION

While software applications become large and complex, software quality turns out to be extremely important in most systems especially those related to medical treatment, aerospace, finance and public infrastructure (Xu, 2010; Xu *et al*., 2004; Vinayagasundaram and Srivatsa, 2007). Bad quality software applications frequently results in revenue loss, hurts the user confidence, unable to provide competitive advantage to business and may create additional workload (Original Software, 2010). Quality related activity can be regarded as a set of umbrella task corresponding to the defects inside the requirement specification, documents and applications. Quality related activity includes the software quality assurance and software quality control which are defined by CMM/CMMI (ISO/IEC, 1998; ISO/IEC 9126-1, 2001; Rose, 2005). It includes feasibility analysis, requirement review, architecture inspection, code review, software testing, auditing and etc., throughout the entire software development life cycle (Shu *et al*., 2009).

Typically, the scope, deadline, manpower and effort can be negotiated most of the time but software quality is a primary criterion which cannot be jeopardized (Bedi and Gaur, 2007; Bulbul and Sahin, 2007; Etaati *et al*., 2011). However, quality related activity may account for nearly one third to a half effort and duration in typical software projects, it has been considered as the bottleneck to quickly deliver the software applications. A survey conducted by Original Software (2010) revealed that the importance of quality management has risen while most managers are not satisfied with their current quality management solutions.

Our previous research has indicated the most significant defect categories and suggested the priority to improve the quality continuously. When scheduling the quality related activities at the beginning of software projects, we should not only care about the most significant defect categories indicated in the historic projects but also need to deal with several uncertain factors including the staffing, duration and cost. This paper identifies the effort model of the software quality related activities, suggests approach to calibrate the metrics and benchmarks based on local historic software project and balances the time-to-mark to quality with Monte Carlo simulation. The efficiency matrix, capability matrix and the definition of the constraints and the goal

Corresponding Author: Bin Xu, College of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

are defined and several case studies are demonstrated and analyzed. Some increments in a financial software project are chosen as the piloted releases. The Function Point is suggested to size the release on the basis of pilots. The results show that the suggested method will benefit the scheduling of quality related activities in incremental software development projects.

Though there are lots researches on software quality management, including the total quality management (Deming, 1986; Juran and Gryna, 1993; Crosby, 1979; Chung *et al*., 2010) unified process (Jacobson *et al*., 1999; Norbjerg, 2002; Kroll and Kruchten, 2003; Shuja and Krebs, 2007) six sigma, lean quality management and value-based quality management, few research work is related to scheduling the quality related activities (Xu and Pan, 2006) in incremental software development (Wohlin, 1994).

The authors have done some research on how to continuously improvement the quality in the incremental software development projects. In that research, the authors suggested using Pareto analysis to assist the quality improvement process decision by indicating the significant defect categories and root cause (Yu, 1998). When scheduling the quality related activities for a new release, the staff, timeline and budget limitation of quality tasks are non-determined. Such uncertainty results in uncertainty of effort and final quality status. For example, if we arrange too less effort in quality related activities, we could arrange much more effort in development so as to reduce the defect injection but the injected defects may have less chance to be detected. Further, quality related effort distribution in different order will result in different defect removal efficiency. While the project management has budget limitation of the entire project and the request of quality of service, Monte Carlo simulation can be useful for the management to distribute the effort among development and quality tasks and then to distribute both effort at the operational (or detail) level.

Monte Carlo simulations has been widespread used in managing the risks related to different domains, including the construction projects (McCabe, 2003), real-life systems (Kadry, 2007; Ahortor and Olopoenia, 2010) investment management (Demir and Bostanci, 2010) wireless network reliability (Peiravi and Hossein, 2008; Rezaii and Tinati, 2009) operational risk access and management (Ergashev, 2009; Xu *et al*., 2009) Value at Risk computing (Suhobokov, 2007).

In software project management domain, Monte Carlo simulation has been used to assess sensitivity in the COCOMO II model for software cost estimation (Musilek *et al*., 2002; Hari and Reddy, 2011) to assess staffing needs in software maintenance projects (Antoniol *et al*., 2004), to generate the project management solution (Putnam and Myers, 1992).

In this study, the authors will identify the relationship between different activities, calibrate with the local historic quality status and former performance, define the constraints and suggest using Monte Carlo simulation to satisfy the defect removal efficiency request with the constraints of budget limitation based on the activity relationship identification.

EFFORT MODEL IN SOFTWARE DEVELOPMENT PROJECT

Effort dependence between development and quality tasks: Though the tasks of a software development project cannot be determined in detail in the early stage, the effort dependence is naturally exists between development tasks and quality tasks. A large scale software project requires more effort for both development and quality tasks. Quality tasks require more ratio of effort in a quality-critical software project while development tasks require more ratio of effort in a technique-critical software project. To be mentioned, insufficient effort in previous stage results in more defects remained and more rework risk in following stages.

Definitions of development and quality tasks relationship: In order to identify the effort model between development and quality tasks, is better to be defined the project, release (or increment), the development tasks, quality tasks and the constraints.

Definition 1. Project: A project is defined as PRJ:: = <pid, name>, where pid is the identification number of the project and name refers to the name of the project.

Definition 2. Release (or Increment): A release (or increment) is defined as REL:: = <pid, rid, name, size, effort, DRE> where pid is the identification number of the project, rid is the identification number of the release, name, size and effort are the name, functional size and budgeted effort of the release and DRE refers to the defect removal efficiency request.

Definition 3. Development tasks: Development tasks refer to all the tasks except the quality related tasks. Here in this research, development task is an abstract concept which refers to all the development other than quality tasks in requirement phase, architecture phase, coding phase and testing phase. A development task can be defined as DT:: = <pid, rid, ph, effort, sdate, edate>, where pid is the identification number of the project, rid is the identification number of the increment, ph is the phase number, effort, sdate and edate are the effort, start date and end date of the development task.

Definition 4. Quality tasks: Quality tasks refer to the quality related tasks. Here in this research, the definition of quality tasks is different than development tasks. A quality task can be defined as QT:: = <id, pid, rid, ph, tid, effort, sdate, edate>, where id is the identification number of qt, pid is the identification number of the project, rid is the identification number of the increment, ph is the phase number, tid is identification number of the quality task, effort, sdate and edate are the effort, start date and end date of the quality task. The quality task is categorized as QTC:: = <tid, tname>, where tid and tname are the identification number and the name of the quality task.

Definition 5. Defect list: Defect can be fetched from the defect management system, such as BugZilla, Jira or Clear Quest. A unified definition of defect list can be DL:: = <id, pid, rid, sdate, rc, pri>, where id is the identification number of the defect, pid is the identification number of the project, rid is the identification number of the increment, sdate, rc and pri are submit date, root cause and priority of the defect. The root cause is defined as RC:: = <id, name, labor> where id, name and labor refer to the identification number, name and the labor rate of the root cause.

EFFORT SCHEDULING USING MONTE CARLO SIMULATION

The entire effort scheduling using Monte Carlo simulation can be divided into calibration, constraints definition, sampling and simulation.

Calibration: Effort is scheduled according to previous project performance and the historic defect status. The effort to detect the defects is an important factor to determine the composition of the quality related tasks and the quality effort distribution between different phases. The capability of the quality related tasks which indicated by the defect removal efficiency is another important factor to forecast the limitation of each quality related tasks composition and the entire defect removal efficiency of the project. The labor cost of the quality related tasks and the cost for the remained defects are the essential impact factors to evaluate the ROI for the effort scheduling.

Most available researches and product (such as COCOMO II and SLIM) towards the software defect estimation suggest that there is linear relationship between defect number and the project size. In this research, the defect number is estimated in linear function on its size. To be more close to the actual value we calculated the optimistic, most likely and pessimistic from the minimal, average and maximal defect density from the historic project. Both Source Lines of Code (SLOC) and Function Point (FP) are used to size the historic projects.

Definition 6: Defect Status Benchmark. Defect status benchmark should be calibrated from found defect list. It can be defined as DSB::=<pid, rid, DSL>, where pid is the identification number of the project, rid is the identification number of the increment and DSL is the defect status list which is queue of <rc, n, wn> where rc is the identification number of the root cause, n and wn are the number and weighted number of the defects in the release rid in the category of rc. The defects are weighted by the priority, typically the priority equal to 1 will be weighted as 3, priority equal to 2 or 3 will be weight as 1 and other defects will be weighted as 1/3. Wn is calculated as the summary of the defect weight and n is the count of the defects.

Definition 7. Reference project group: Similar projects can be grouped together to serve as the reference in estimating new project. Reference project group is used to distinguish the incremental projects with different profile, performance, or quality status. Reference project group can be defined as RPG :: = <gid, relLst > where gid is the identification number of the reference project group and relLst is the release list which is queue of <pid, rid>, where pid is the identification number of the project, rid is the identification number of the increment.

Definition 8. Quality task performance: In the execution of each quality related task, defects in different categories may be detected. The performance of quality tasks is defined as QTP :: = <Id, pid, rid, ph, Task, TPL> where Id is the task identification of qtp, pid is the identification number of the project, rid is the identification number of the increment, ph is the phase id, Task is the task category for quality work and TPL is the queue of <RC, DRE, Effort, Defect Found> where RC is the category of root cause, DRE is the defect removal efficiency of Task, Effort is the effort to finish task qtp and Defect found is the number of defects in the RC category found in the execution of qtp.

Algorithm 1: Quality tasks performance calculation

```
Input: Projects PRJ, releases REL, quality tasks QT, defect list DL, project id pid and release id rid.
Output: quality tasks performance matrix QTP.
sort DL in the descending order of pid, rid, sdate
FOR (each dl in DL)
IF (dl.pid = pid AND dl.rid = rid) break;
submit Date = dl.sdate; defectNumber = 0; defectTotal = 0; effort = 0;
FOR (each dl in DL) {
IF (dl.pid <> pid OR dl.rid <> rid) break;
IF (submitDate = sdate)
defectNumber[rc] ++; defectTotal[rc] ++;
ELSE {
updateEffort(effort, defectNumber, QT);
generateQTP(pid, rid, effort, defectNumber, QT);
submitDate = dl.sdate; defectNumber = 0; effort = 0;
}
}
updateDREinQTP(QTP, pid, rid, defectTotal);
```

Table 1: Effort distribution for each root cause category

Root cause	Defect Number	Effort distribution (man-h)
System design defect	7	= 4.2* 7 = 29.4
Detail design defect	10	= 4.2* 10 = 42
Coding defect	33	= 4.2* 33 = 138.6
Sum	50	210

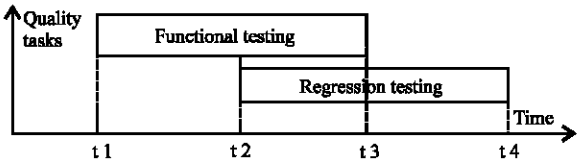

Fig. 1: Overlapped quality tasks at the timeline

Algorithm 1 walks through defect list of one release, counts the defect for each root cause category and calculates the related effort according to the ratio of defect found, as what is shown in Table 1. The procedure update effort at Line 10 is use to distribute the related effort according to the ratio. When there is overlap of Quality tasks at the timeline, such distribution should consider the ratio of effort of each quality task, as shown in Fig. 1.

The effort is assumed to be level loading, that is, the resource is average staffing for each quality tasks. Therefore, the effort between t1, t2, t3 and t4 will be as formula (1):

$$\begin{aligned}
\text{Effort}_{(t1-t2)} &= \frac{t2-t1}{t3-t1} * \text{Effort}_{\text{FunctionalTesting}} \\
\text{Effort}_{(t3-t4)} &= \frac{t4-t3}{t4-t2} * \text{Effort}_{\text{RegressionTesting}} \\
\text{Effort}_{(t2-t3)} &= \frac{t3-t2}{t4-t2} * \text{Effort}_{\text{RegressionTesting}} + \frac{t3-t2}{t3-t1} * \text{Effort}_{\text{FunctionalTesting}}
\end{aligned} \tag{1}$$

After that, the effort can be distributed according to the defect number found as what is demonstrated in Table 1. Since some defects maybe found after the application has been delivered to the clients, such defects will be submitted to the defect management system after the regular quality tasks. Therefore, the final defect removal efficiency could only be calculated after the algorithm 1 walk through all the related defects for the release. The procedure update DRE in QTP at Line 15 in the algorithm 1 is used to update the DRE accordingly:

$$\forall rc, DRE_{rc} = \frac{\text{DefectFound}_{rc}}{\text{DefectTotal}_{rc}} \tag{2}$$

Most time, the quality tasks are not executed individually. For example, there will be unit testing, code review, functional testing, regression testing and etc. after the code has been developed. It's hard to evaluate the performance of a single quality task. For example, the unit testing may detect 3 coding defects in 5 h, while at the same time it facilitate the code review and following system testing. Basically, the foregoing tasks brings more benefit than just calculating from the defect found list and the posterior tasks has less benefit than which is calculated conversely. Before we could find a good model to refine the performance evaluation, quality task group is defined so as to evaluate the quality tasks within a certain phase as a whole.

Definition 9. Quality task group: Quality task group is defined as QTG :: = <tgId, pid, rid, ph, TL, TPL> where tgId is the task identification of qtg, pid is the identification number of the project, rid is the identification number of the increment, ph is the phase id, TL is the queue of task and TPL is the queue of <RC, DRE, Effort, DefectFound> which is similarly defined as Quality Task but here they are the contributes belong to the group of quality tasks qtg. Quality task group performance QTG is calculated on the base of quality tasks performance matrix QTP using Algorithm 2.

Algorithm 2: Quality task group performance Calculation

```
Input: quality tasks performance matrix QTP, project id pid and release id rid.
Output: quality task group performance matrix QTG.
1 sort QTP in the descending order of pid, rid, ph
2 FOR (each qtp in QTP)
3 IF (qtp.pid = pid AND qtp.rid = rid) break;
4 currentPhase = qtp.ph; defectNumber = 0; effort = 0; DRE= 0; TaskList = EmptyList;
5 FOR (each qtp in QTP) {
6 IF (qtp.pid <> pid AND qtp.rid <> rid) break;
7 IF (currentPhase = qtp.ph) {
8 TaskList.add(qtp.Task);
9 FOR (each tpl in qtp.TPL) {
10 defectNumber[tpl.rc] += tpl.DefectFound;
11 DRE [tpl.rc] += tpl. DRE;
12 Effort [tpl.rc] += tpl. Effort; }
13 }
14 ELSE {
15 generateQTG(pid, rid, ph, TaskList, DRE, Effort, defectNumber);
16 currentPhase = qtp.ph; defectNumber = 0; effort = 0; DRE= 0; TaskList = EmptyList;
17 }
18 }
```

Quality task group performance matrix QTG can be considered as the summary of the quality task performance matrix QTP. Since the defect removal efficiency is an absolute value, here in Algorithm 2, the DRE can be directly summed together to generate the group value:

$$\forall pid, rid, ph, QTG_{pid,rid,ph} = \sum_{task} QTG_{pid,rid,ph,task} \tag{3}$$

In this research, quality task group performance matrix is actually used in the Monte Carlo simulation but

Table 2: Quality related task in category

ID	Quality related task
FT	Functional testing
SIT	Integration testing
RT	Regression testing
SMT	Smoke testing
BR	BSD review
AR	Architecture review
CR	Code review
...	...

the quality task performance matrix is still waiting for the future exposure.

Goal and constraints definitions: The goal is to achieve the minimal total labor cost for the quality tasks and the cost for remained defects.

$$\text{Goal :Minimal} \sum_{rc,ph} \text{Effort}_{rc,ph} * \text{LaborRate} + \sum_{rc} \text{Defect Remained}_{rc} * \text{penaltyCost} \tag{4}$$

There are several constraints in the quality effort scheduling, including the budget constraint and capability constraint. Because there is project budget limitation, the quality related activities are limited within the entire budget boundary. The summary of the quality related activities could not beyond a certain part of the budget:

$$\forall \text{ph}, \sum_{rc} \text{Effort}_{rc,ph} <= \text{EffortAvailable}_{ph} \tag{5}$$

We assume that the tasks couldn't detect more defects than the estimated number:

$$\forall \text{rc}, \sum_{ph} \text{DRE}_{rc,ph} <= 100\% \tag{6}$$

The capability of the quality related tasks shows that each task has its technique limitation in detecting the defects. In other words, the tasks don't require more effort than its request:

$$\forall \text{rc, ph}, \text{Effort}_{rc,ph} <= \text{DRE}_{rc,ph} * \text{EstimateDefect}_{rc,ph} \ \& \ \text{Effort}_{rc,ph} >= 0 \tag{7}$$

Besides, we suggest an assumption that the quality related tasks don't have tendentiousness to detect only one kind of root cause. Therefore, kinds of defects will be detected by each quality task as it has done in the previous projects:

$$\forall \text{ph}, \text{DRE}_{rc1,ph} = \text{DRE}_{rc2,ph}, \text{if}(\text{DRE}_{rc1,ph} > 0 \ \& \ \& \text{DRE}_{rc2,ph} > 0) \tag{8}$$

Sampling and simulation: The effort for each phase and each root cause category is the schedule results and is sampling in this research. Considering that there will be multiple quality task groups can be chosen for different phases, we tried to choose different quality task group for each phase in the simulation.

Algorithm 3: Simulation

Input: Reference project group RPG, estimated project size psize and available effort for each phase AvailableEffort[ph].
Sampling: Effort[ph,rc] and the quality task group for each phase, QTP[ph].
Output: required effort for each phase and each root cause Effort[ph,rc].
1 Estimate the defects for each root cause category as defectEstimated[rc];
2 FOR (each ph in phases) //phases from 1 to 3. Requirement, Architecture and Coding
3 Calculate the capability of the QTP towards different rc from RPG;//DRE[ph,rc]
4 Set the goal in the simulation with totalCost(Effort, DefectRemained) ;
5 FOR(each ph) set the budget constraints with budgetConstraints(Effort, AvailableEffort);
6 For (each rc) set the defect number constraints with defectConstraints(DRE);
7 For (each ph) FOR (each rc) set the effort constrains with effortConstraints(Effort,DRE, defectEstimated);
8 For (each ph) set the nature constraints with natureConstraints(DRE);
9 Execute the simulator;

The constraints function total cost at Line 4 is calculated with the formula 4. The function input with the effort matrix and defectRemained matrix and output the total cost to serve as the goal of simulation. The procedure budget constraints at Line 5 defines the constraints of budget according to formula 5 which ensures that the result effort will be limited within the boundary of available effort per phase. The procedure defect constraints at Line 6 defines the constraints according to formula 6 to avoid the unreasonable result which has impossible defect number. The procedure effort constraints at Line 7 defines the constraints according to formula 7 to avoid the impossible effort result. The procedure nature constraints at Line 8 defines the constraints according to formula 8 to avoid the tendentiousness.

CASE STUDY IN GLOBAL IT CORP

The suggested approach has been piloted in a financial software company. A finished increment has been chosen from an incremental software development project. The team leader showed the pilot team with the delivered size in SLOC but the quality related effort is kept unknown to the pilot team during the pilot. The metrics and benchmark were calibrated from historic projects at first. Table 2 shows the most available quality related tasks in the project in piloting.

Calibration according to the historic data: Table 3 shows the root cause category used in the pilot team. The optimistic, most likely and pessimistic defect density are calibrated with the minimal, average and maximal defect

Table 3: Quality related task in category

ID	Root cause	Optimistic defect density density (Defect/KIU)	Most likely defect density (Defect/KIU)	Pessimistic defect (Defect/KSLOC)
RQ1	Requirement misunderstanding/unclear	0.71	2.75	4.44
RQ2	Requirement change/enhancement	0.83	2.88	5.04
SD	System design	0.33	0.33	0.33
DD	Detail design	0.36	3.17	6.45
CD	Coding defect	2.48	5.06	6.94
HR	Human resource: Technology, communication	2.87	2.87	2.87
SW	Software: System/tools	0.23	0.23	0.23
TC	Test cycle	0.26	1.03	2.42
OM	Operational mistake	0.18	0.18	0.18
OT	Cause unknown	0.15	0.63	1.48

Table 4: Historic quality tasks in the previous increments

ID	Task	Description	Start date	End date	Phase	Effort (PHR)	Release
1110	FT	Functional Testing	21-Jan-11	14-Feb-11	4-build	477.5	RP004
1111	RT	Regression testing	15-Feb-11	18-Feb-11	4-build	233.0	RP004
1210	FT	Functional Testing	28-Feb-11	23-Mar-11	4-build	2115.0	RP005
1211	RT	Regression testing	24-Mar-11	20-Apr-11	4-build	598.0	RP005

Table 5: Quality task performance matrix

Task	Task description	Root cause	Root cause description	DRE (%)	Effort (H.)	Defect number	Phase
FT	Functional test	RQ1	Requirement misunderstanding/unclear	72.09	1648.77	124	4-Build
FT	Functional test	RQ2	Requirement change/enhancement	50.57	1313.98	89	4-Build
FT	Functional test	SD	System design	100.00	68.69	5	4-Build
FT	Functional test	DD	Detail design	78.54	2012.45	194	4-Build
FT	Functional test	CD	Coding defect	65.49	1383.82	167	4-Build
FT	Functional test	SW	Software: System/tools	33.33	22.73	2	4-Build
FT	Functional test	TC	Test cycle	75.36	409.84	52	4-Build
FT	Functional test	CU	Cause unknown	50.00	71.71	5	4-Build
RT	Regression test	RQ1	Requirement misunderstanding/unclear	18.60	493.08	32	4-Build
RT	Regression test	RQ2	Requirement change/enhancement	15.34	384.84	27	4-Build
RT	Regression test	DD	Detail design	11.34	281.76	28	4-Build
RT	Regression test	CD	Coding defect	20.39	502.46	52	4-Build
RT	Regression test	SW	Software: System/tools	66.67	44.08	4	4-Build
RT	Regression test	TC	Test cycle	21.74	133.59	15	4-Build
RT	Regression test	CU	Cause unknown	30.00	69.19	3	4-Build

density of the pervious increments. The unit is in defect per kilo-SLOC or defect per FP.

Table 4 shows the quality tasks have been performed in the previous increments. Two increments are chosen and they are grouped together into a reference project group to forecast the "new" project (to the pilot team, as they don't know the actual effort in the pilot). Though the regression testing is determined by the size of application which includes the baseline part and the delivered part, here in this research we don't distinguish the regression testing from other tests. No special characteristic will be considered for any quality task. Since the business analyst didn't have their working hours recorded in detail, the business solution document review and architecture review are not mentioned in the first case.

Both releases have two quality tasks, functional testing is to ensure the accurate implementation of the new functionality and regression testing is to ensure there is no defect has been brought for the available functionality.

Table 5 shows the performance matrix for individual quality task. The functional test is more effective and more efficient than the regression test in finding the different root causes. The effectiveness is indicated by the high defect removal efficiency and can be owned to the order of implementation. The efficiency is indicated by the effort cost for each defect which can be owned to the capability and performance of the individual quality related task.

Quality task group performance matrix is the summary of quality task performance matrix, Table 6 summarized the data in Table 5 which will be used as the capability and performance benchmark used for the quality effort estimation.

Scheduling the quality effort sizing with SLOC and FP: The pilot team sized the delivered increment both in SLOC and FP (using IFPUG 4.1), the quality related effort is unknown to the pilot team before the sizing has been finished. Table 7 and 8 show the deviation analysis on effort, DRE and defect number. From the result, we may found that Function Point sizing is good at estimate the effort, SLOC sizing is good at estimate the defect number and both are similar when estimating the defect removal efficiency.

Table 6: Quality task group performance matrix

Group ID	Task composition	Root cause	Root cause description	DRE (%)	Effort (H.)	Defect number	Phase
Build testing	Functional test, Regression test	RQ1	Requirement misunderstanding/unclear	90.69	2141.85	156	4-Build
Build testing	Functional test, Regression test	RQ2	Requirement change/enhancement	65.91	1698.82	116	4-Build
Build testing	Functional test, Regression test	SD	System design	100.00	68.69	5	4-Build
Build testing	Functional test, Regression test	DD	Detail design	89.88	2294.21	222	4-Build
Build testing	Functional test, Regression test	CD	Coding defect	85.88	1886.28	219	4-Build
Build testing	Functional test, Regression test	SW	Software: System/tools	100.00	66.81	6	4-Build
Build testing	Functional test, Regression test	TC	Test cycle	97.10	543.43	67	4-Build
Build testing	Functional test, Regression test	CU	Cause unknown	80.00	140.90	8	4-Build

Table 7: Deviation analysis on effort and DRE

	Total effort needed			Total DRE achieved		
Pilot	Estimated	Actual	Error (%)	Estimated (%)	Actual (%)	Error (%)
SLOC Pilot	1406 (1094 ~ 1721)	1708	21.48	91.92	82.03	10.76
FP Pilot	1728 (965 ~ 2374)	1708	1.16	91.93	82.03	10.77

Table 8: Deviation analysis on defect number

	Total defects			Found defects		
Pilot	Estimated	Actual	Error (%)	Estimated	Actual	Error (%)
SLOC Pilot	143 (109 ~ 177)	126	11.89	132 (99 ~ 164)	105	20.45
FP Pilot	178 (112 ~ 242)	126	29.21	164 (103 ~ 222)	105	35.98

DISCUSSION

While many software metrics have been developed to verify the quality traceability (Singh and Saha, 2010; Xu, 2005), the efficiency of object-oriented technique (Parthasarathy and Anbazhagan, 2006; Boroni and Clausse, 2011; Changchien *et al.*, 2002) and to validate special applications (Pan and Xu, 2010; Vinayagasundaram and Srivatsa, 2007; Bedi and Gaur, 2007) individual metrics makes less sense to determine the status of quality. In our previous research we suggested using Pareto analysis method (Barro, 2009; Zhihuan *et al.*, 2010; Xu *et al.*, 2012) to identify the most significant defect which enables the managers to improve the quality by deal with the most significant root causes. When there is relationship between different defect category, or the related implementation cost to deal with the root cause is not linear with the amount of defect in number, Monte Carlo simulation is better to be adopted so as to generate an nearly-best solution for the quality improvement. It's very important to use such approach in the planning of quality tasks when there are many uncertain factors (such as the resource, cost and staff) with different relationship between them.

In this research, we used Monte Carlo simulation (Mooney, 1997; Elafify *et al.*, 2004) in the pilot and found that the result is acceptable. Besides, we found that the Function Point (ISO/IEC 20926, 2003) sizing method is better than SLOC sizing method in estimating the quality related effort. This is really good news for the software practice as the final SLOC is rather difficult for us to estimate at the beginning of a software project but we could get the Function Point sizing data if there is sufficient requirement material. The results showed that the gap between estimated effort and actual effort is small but the number of defects in different categories has different gap. The interview with the project team showed that the root cause may be defined in different criteria which results in different ratio in the releases. Such deviation can be reduced when more historic data is calibrated and put into the benchmark.

CONCLUSION

Considering the difficulty in making the best decision for quality improvement especially when scheduling the quality effort at the very beginning, here in this paper we suggested using Monte Carlo simulation based approach to identify the uncertainty and relationship, calibrate the historic project data and generate the nearly-best solution for the quality effort scheduling. It has been piloted in an incremental development project. In the pilots, we found that the results may indicate some better alternative effort scheduling solutions than manually scheduling. Though the quality of historic data will result in some deviation in the scheduling, such as the gap for defect estimation in root cause category such deviation will be reduced when there is sufficient historic data to serve as the benchmark.

ACKNOWLEDGMENT

This study was financial supported by The Science and Technology Department of Zhejiang Province, China with No. 2008C11009 to Dr. Bin Xu and Education Department of Zhejiang Province with No. yb07035 to Prof. Yun Ling and No. 20061085 to Dr. Bin Xu.

This study was also supported by Zhejiang Gongshang University, China with No. Xgz1102 to Dr. Weigang Chen.

REFERENCES

Ahortor, C.R.K. and R.A. Olopoenia, 2010. Determinants of portfolio flows to Ghana: A dynamic stochastic general equilibrium analysis. J. Applied Sci., 10: 1-19.

Antoniol, G., A. Cimitile, G.A. Di Lucca and M. Di Penta, 2004. Assessing staffing needs for a software maintenance project through queuing simulation. IEEE Trans. Software Eng., 30: 43-58.

Barro, D., 2009. Conditional dependence of trivariate generalized Pareto distributions. Asian J. Math. Stat., 2: 20-32.

Bedi, P. and V. Gaur, 2007. Trust based quantification of quality in multi-agent systems. Inform. Technol. J., 6: 414-423.

Boroni, G. and A. Clausse, 2011. Object-oriented programming strategies for numerical solvers applied to continuous simulation. J. Applied Sci., 11: 2723-2733.

Bulbul, H.I. and Y.G. Sahin, 2006. How to select the best method for database applications, database independent platform or not? J. Applied Sci., 7: 127-131.

Changchien, S.W., J.J. Shen and T.Y. Lin, 2002. A preliminary correctness evaluation model of object-oriented software based on UML. J. Applied Sci., 2: 356-365.

Chung, Y.C., Y.W. Hsu and C.H. Tsai, 2010. Research on the correlation between implementation strategies of TQM, organizational culture, TQM activities and operational performance in high-tech firms. Inform. Technol. J., 9: 1696-1705.

Crosby, P.B., 1979. Quality Is Free: The Art of Making Quality Certain. McGraw Hill, New York, USA., ISBN-10: 0-451-62585-4, Pages: 270.

Deming, W.E., 1986. Out of the Crisis, 1st Edn., MIT Press, Cambridge, ISBN: 0911379010. pp: 98.

Demir, H. and B. Bostanci, 2010. Decision-support analysis for risk management. Afr. J. Bus. Manage., 4: 1586-1604.

Elafify, M.M., K.M. Elawadly and S.G. Elsharkawy, 2004. Prediction of the thermal conductivity of gas mixtures using direct simulation Monte Carlo method. Inform. Technol. J., 3: 23-35.

Ergashev, B., 2009. Estimating the lognormal-gamma model of operational risk using the Markov chain Monte Carlo method. J. Oper. Risk, 4: 35-57.

Etaati, L., S. Sadi-Nezhad and A. Makue, 2011. Using fuzzy group analytical network process and ISO 9126 quality model in software selection: A case study in E-learning systems. J. Applied Sci., 11: 96-103.

Hari, CH.V.M.K. and P.V.G.D.P. Reddy, 2011. A fine parameter tuning for COCOMO 81 software effort estimation using particle swarm optimization. J. Software Eng.,5 5: 38-48.

ISO/IEC, 1998. International standard, information technology: Software product evaluation, Part 5: Process for evaluators. International Standards Organization.

ISO/IEC 9126-1, 2001. Software Engineering-Product Quality Part 1: Quality Model. International Organization for Standardization, Geneva.

ISO/IEC 20926, 2003. Software engineering-IFPUG 4.1 Unadjusted functional size measurement method: Counting practices manual. International Standards Organization, Geneva.

Jacobson, I., G. Booch and J. Rambaugh, 1999. The Unified Software Development Process. 1st Edn., Addison-Wesley Longman, MA., USA., ISBN-10: 0-201-57169, Pages: 512.

Juran, J.M. and F.M. Gyrna, 1993. Quality Planning and Analysis: From Product Development through Use. 2nd Edn., McGraw-Hill, New York, USA., ISBN-10: 0-07-033178-2, Pages: 629.

Kadry, S., 2007. Probabilistic analysis of eigenvalue of stochastic systems. J. Applied Sci., 7: 565-569.

Kroll, P. and P. Kruchten, 2003. The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP, Addison-Wesley, MA, USA., ISBN-10: 0321166094, Pages: 416.

McCabe, B., 2003. Monte Carlo simulation for schedule risks. Proc. Winter Simul. Conf., 2: 1561-1565.

Mooney, C.Z., 1997. Monte Carlo Simulation. Quantitative Applications in the Social Sciences Thousand, Sage Publications, Oaks, CA.

Musilek, P., W. Pedrycz, N. Sun and G. Succi, 2002. On the sensitivity of the COCOMO II Software cost estimation model. Proceedings of the 8th Symposium on Software Metrics, August 07, 2002, IEEE Computer Society, pp: 13-20.

Norbjerg, J., 2002. Managing incremental development: Combining flexibility and control. Proceedings of the European Conference on Information Systems, June 6-8, 2002, Gdansk, Poland, pp: 229-239.

Original Software, 2010. Application quality management survey result. http://www. origsoft.com/products/qualify/docs/aqm_survey_results.pdf

Pan, L. and B. Xu, 2010. Towards collaborative master student talent development with E-CARGO model. Inform. Technol. J., 9: 1031-1037.

Parthasarathy, S. and N. Anbazhagan, 2006. Analyzing the software quality metrics for object oriented technology. Inform. Technol. J., 5: 1053-1057.

Peiravi, A. and T.K. Hossein, 2008. Fast estimation of network reliability using modified Manhattan distance in mobile wireless networks. J. Applied Sci., 8: 4303-4311.

Putnam, L.H. and W. Myers, 1992. Measures for Excellence: Reliable Software on Time, within Budget. Computing Series. Yourdon Press, Englewood Cliffs, ISBN: 9780135676943, Pages: 378.

Rezaii, T.Y. and M.A. Tinati, 2009. Improved joint probabilistic data association filter for multi-target tracking in wireless sensor networks. J. Applied Sci., 9: 924-930.

Rose, K.H., 2005. Project Quality Management: Why, What and How. J. Ross Publishing, Fort Lauderdale, Florida, USA.

Shu, Y., H. Liu, Z. Wu and X. Yang, 2009. Modeling of software fault detection and correction processes based on the correction lag. Inform. Technol. J., 8: 735-742.

Shuja, A. and J. Krebs, 2007. IBM Rational Unified Process Reference and Certification Guide: Solution Designer, IBM Press, UK.

Singh, Y. and A. Saha, 2010. Predicting testability of eclipse: A case study. J. Software Eng., 4: 122-136.

Suhobokov, A., 2007. Application of Monte Carlo simulation methods in risk management. J. Bus. Econ. Manage., 8: 165-168.

Vinayagasundaram, B. and S.K. Srivatsa, 2007. Software quality in artificial intelligence system. Inform. Technol. J., 6: 835-842.

Wohlin, C., 1994. Managing Software Quality through Incremental Development and Certification. In: Building Quality into Software, Ross, M., C.A. Brebbia, G. Staples and J. Stapleton (Eds.). Computational Mechanics Publications, Southampton, United Kingdom, pp: 187-202.

Xu, B. and X.P. Pan, 2006. Optimizing dual-shore SQA resource and activities in offshore outsourced software projects. Proceedings of the 19th Annual Canadian Conference on Electrical and Computer Engineering, May 2006, Ottawa, Ont, pp: 2405-2409.

Xu, B., 2005. Extreme programming for distributed legacy system reengineering. Proc. 29th Ann. IEEE Int. Comput. Software Applic. Conf., 1: 160-165.

Xu, B., 2010. Cost efficient software review in an E-business Software Development Project. Proceedings of the 2010 International Conference on E-Business and E-Government, May 7-9, 2010, Guangzhou, pp: 2680-2683.

Xu, B., M. Chen, C. Liu, J. Li, Q. Zhu and A.J. Kavs, 2012. Enabling continuous quality improvement with quantitative evaluation in incremental financial software development. Inform. Technol. J., 11: 76-84.

Xu, B., X. Yang, Z. He and S.R. Maddineni, 2004. Achieving high quality in outsourcing reengineering projects throughout extreme programming. Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, October 10-13, 2004, IEEE, pp: 2131-2136.

Xu, Z.S., Z.J. He and P. Song, 2009. The application of risk management to investment of the railway tunnel project. Proceedings of The first International Conference on Information Science and Engineering, December 26-28, 2009, IEEE Computer Society, pp: 4547-4550.

Yu, W.D., 1998. Software fault prevention approach in coding and root cause analysis. Bell Labs Tech. J., 3: 3-21.

Zhihuan, L., L. Yinhong and D. Xianzhong, 2010. Improved strength pareto evolutionary algorithm with local search strategies for optimal reactive power flow. Inform. Technol. J., 9: 749-757.