Based on Quantification Software Quality Assessment Method

Yang Aimin , Zhang Wenxiang
Computer and Information Technology College, Zhejiang Wanli University , Ningbo, CHINA
yhyang@sina.com.cn, zwx1962@zwu.edu.cn

Abstract—**The influence which the software quality exerts on software industry is more deeply in recent years.From the beginning of software development to the present software development, the assessing of the software quality and the assurance of the software quality has already become an important and indispensable factor in software development. A lot of experts, scholars in the world begin to study how to assess and guarantee the quality of the software. Some assessing methods come from the author's understandings to the software quality are discussed and an effective quantitative assessing model is proposed.**

Index Terms—**software quality,quality assessing, algorithm,hiberarchy model,**

I. Introduction

With continuous deepening of computer application, the quality of software will directly affect depth and scope of application. Obviously, PC can't be widely used today without steady and powerful operation system software. So do other application fields such as aeronautical & astronautic industry and national defense industry, etc. It is necessary for both developers and users to assure software quality.

Software quality evaluation refers to a series of tasks including end product evaluation, development process evaluation and evaluation on comparison among software with similar function. Currently, it is difficult to effectively evaluate software quality in software engineering field, which arouses high attention in US, Japan and Europe in recent years. For example, some relevant effective evaluation methods such as software quality method have been extensively used in complicated fields including aeronautical & astronautic industry, communication engineering and bank credit, which have brought substantial effectiveness. This development trend will play an important role in the development of software industry.

II. Software Quality Overview

A. Software Quality

Despite of numerous definitions of software quality, through examining each definition and combining with my own understanding on software quality, I think reasonable definition should be as follows: software quality characterizes all attributes on the excellence of computer system such as reliability, maintainability and usability. In terms of practical application, software quality can be defined with three points on consistency: consistency with determined function and performance; consistency with documented development standard; consistency with the anticipated implied characteristics of all software specially developed. The above two definitions are based on two different perspectives, but they share the same essence: the satisfaction of customers' demands, that is, the satisfaction by software products of operating requirements.

B Software Quality Evaluation

Software quality directly affects the application and maintenance of software, so how to objectively and scientifically evaluate software quality becomes the hot spot in software engineering field. Software quality evaluation involves the following tasks throughout software life cycle and based on software quality evaluation standard, which is implemented during software development process: continuously measure software quality throughout software development process, reveal current status of software, predict follow-up development trend of software quality, and provide effective means for buyer, developer and evaluator

A set of evaluation activities may generally include review, appraisal, test, analysis and examination, etc. Performance of such activities is aimed to determine whether software products and process is consistent with technical demands, and finally determine products quality. Such activities will change the phase of development, and may be performed by several organizations. A set of evaluation activities may be generally defined in the software quality specifications of project plan, special project, as well as related software quality specifications. The developer may have software quality evaluation on the finished products before delivery of semi-finished products at every phase of development, identify the difference between current quality level of products and the required quality level of products, and take timely corrective measures, to ensure the required quality is incorporated in every stage of manufacturing, so as to ensure the final quality level meets the requirements.

This paper is based on "Quantification Software Quality Assessment Method," by Yang aimin and Zhang wenxiang which appeared in the Proceedings of the 9th International Conference for Young Computer Scientists(ICYCS 2008) Zhangjiajie, China, November18-21, 2008. °@ 2008 IEEE.

This work was supported by National Natural Science Foundation of NinBo under Grant No. 2009A610107.

While software evaluation method may provide independent third party software quality evaluation, and present impersonal, authoritative, and just results of evaluation[1].

The evaluation plan also includes the evaluation and measure of software development process and the activities and methods forming such process, so such evaluation is actually a verification of initially chosen methodology. However, it is naturally notable that the precondition for software quality evaluation by the software developer is that the development process shall comply with software engineering standards.

III. SOFTWARE QUALITY EVALUATION METHODS

A Factors Affecting Software Quality

There are any factors affecting software quality, generally include: human factor, software demand, testing limitation, difficulty in quality management, traditional custom of software personnel, development specifications, insufficient support of development tools and others, the foregoing of which may be considered as factors that may affect software quality. Generally speaking, such factors may be considered in either respect of the developer and the administrator [2].

From the administrator's perspective, the factors affecting software quality may include:

(1) The administrator is unaware of quality, without overall plan or effective measures that could ensure the quality, and is not paying enough attention to the quality, nor stressing on the quality from the beginning.

(2) The administrator has not set up a good incentive mechanism. Developer's personal proceeds (whether physical or mental) are not related directly to its working performance. There is not any good personal performance evaluation mechanism, so the fact is it causes the developer to feel doing not well is fault of the entirety of us, well its own interest is not affected, however, doing well will not result in timely and obvious reward. Delay for one month more will be paid for one month more, advance for one month cannot help the next project. All in all, doing well is almost the same as doing badly, so every one is not active, no one will try the best to finish task with high quality.

From the developer's perspective, the factors affecting software quality may include:

(1) the developer cannot put quality assurance as the priority that is material and required to be completed, unfortunately, product quality assurance is deemed to be responsibility of quality inspector. Lack of the idea of overall quality management and that every one is the quality assurer and person liable.

(2) Everybody lacks this idea: Must no do unqualified work in each product development stage, must not bring any unqualified intermediate product to next stage, avoiding resort to specialized quality inspector for examination and product quality assurance at the last stage of product. This requires design of examination standard for each stage of work explicitly, letting everybody know what work is qualified [3].

(3) No one can see how important the increased quality is to the existence and development of company, in general lack of the sense of ownership.

Obviously, not only either of the two has problems affecting the quality of software products, but also the relation between the two affects the quality. For example, inconsistence of versions arises because the administrator's direction is not implemented as far as practicable. For another example, the measures take by the administrator to emphasize quality and maintain quality will arouse the developer's revulsion. If everyone can better communicate and cooperate, this kind of problems will become far less. Additionally, we are unfamiliar with customer's quality requirement, don't understand customer's mentality and lack the idea of rendering customers satisfactory [4].

B. Software Quality Quantitative Evaluation Methods

The above sets out factors affecting software quality, mainly from the perspective of inside factors of software development enterprises, and in fact the factors affecting software quality shall also include the problems that occur when users are communicating with software developers. Such problems can be found in the software quality evaluation method.

When we say some software is good, some software has complete function, reasonable structure, and clear arrangement. These expressions are ambiguity, not exact for evaluation of software. To users, when the developer, based on its own demand, develops a application system, completes it on time and delivers it for use, and the system correctly perform the functions required by the user, it is far from enough with satisfaction of the above. Since the users will encounter many problems during introduction of a set of software, for example, it is hard to understand the customized software, or modify it, and during the maintenance period, the maintenance costs of users increase substantially, so the users become to skeptical about the quality of outsourced software, however, the users have no suitable indicator for evaluating software quality, and the developer usually lacks productivity indicator for developing software, so the users are unable to accurately evaluate the working quality of the developer. This kind of evaluation method for software will directly result in shrinkage of life cycle and further development by the developer [5] [6] [7].

For this purpose, it is necessary to discuss a complete quality evaluation system, which evaluation model is 3-tiered: The 1st tier is software quality elements which can be divided into 6 elements which are fundamental features of software, including:

(1) Functionality is the degree to which the functions realized by the software can satisfy user's demand. It reflects the degree to which the developed software can satisfy the alleged demand or implied demand of users, that is, whether the functions required by users are fully realized.

(2) Reliability is the degree to which the software can maintain its performance level in specified time and conditions. Reliability is an important quality requirement for some software, it reflects the degree to which the software can continue to operate in case of failure, in addition to the degree to which the software can satisfy the normal operation required by users.

(3) Facility to use is the degree to which the users spend efforts in learning, operating, preparing input and understanding output of some software. It reflects the user-friendliness, that is, the facility of this software in use.

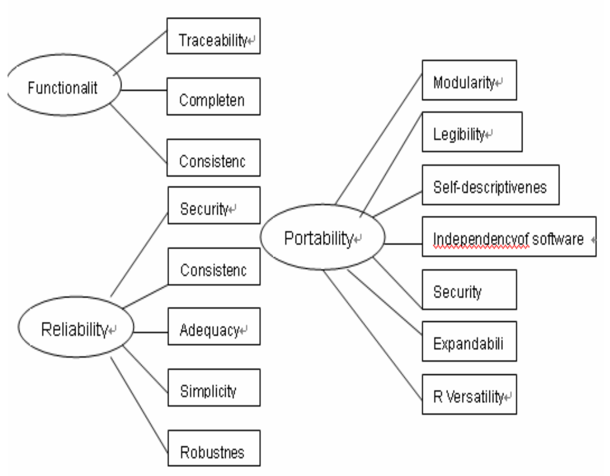
(4) Efficiency is the effectiveness of computer resources (including time) required for some function realized by software in specified conditions. It reflects whether there is waste of resources or so when the required functions are completed.

(5) Maintainability is the facility of modification in case of change in environment or software error in some operable software, in order to satisfy user's demand.

(6) Portability is the facility of transplantation from one computer system or environment to another computer system or environment.

The 2nd tier is evaluation standard, which can be divided into 22 points, including accuracy (software property of accuracy required in computer and output), robustness (software property of continuous performance and system restoration in case of accident), security (software property of prevention from accidental or willful access, use, change, destruction or disclosure), and communication validity, treatment validity, equipment validity, operability, trainability, completeness, consistence, traceability, visibility, independence of hardware system, independence of software system, expandability, utility, modularity, legibility, self-descriptiveness, simplicity, structuredness, and completeness of product files. A certain combination of evaluation standard will reflect the quality elements of some software, the relation between which and evaluation standard is summarized into a dendrogram as shown in Figure 1.

The 3rd tier is metrics: Design a questionnaire for each stage of the seven stages including software demand analysis, preliminary design, detailed design, realization, assembly testing, identification testing and maintenance & use, to realize the quality control over software development process all at once. To an enterprise, no matter for customization or for secondary development



Figue1. Relations between Software Quality Elements and Evaluation Standards

after software outsourcing, it is crucial to know and monitor the progress and product level at every sector of software development process, since the level of software quality substantially depends on user's participation. It is necessary to explain the following points [9]:

(1) To different types of software, system software, control software, management software, CAD software, educational software, Internet software and different scales of software will have different emphases on quality requirements, evaluation standards, quality problem, so they should be distinguished, as shown in TABLE Ⅰ.

Software quality assurance and evaluation activity have different emphases. In the stages of demand analysis, preliminary design, detailed design and its realization, the evaluations are mainly on whether the software demands are complete, whether the design has fully reflected the demand or whether the coding is concise or legible. And each stage has a specific metric work sheet, composed of specific metric units whose score is the basis for score of metric standard and elements, and further evaluation. This point is very applicable to enterprises developed in

TABLE Ⅰ

Different Factors to be Considered by Different Software

Software application features	Factors to be considered
Software demanding longer lifetime	Portability, maintainability
Real time system	Reliability, efficiency
Software to be used in different environments	Portability
Bank related system	Reliability, functionality

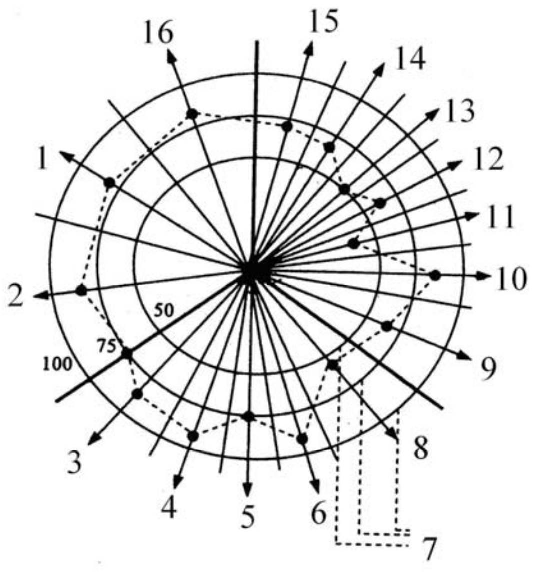
cooperation with software developers.

(2) The fundamental purpose for metrics at every stage of software quality is to control cost, progress and improve efficiency and quality of software development, however, currently there are not many large-scaled software companies in China; this requires continuous improvement of a majority of software development

enterprises, for establishing their own department dedicated to software quality assurance and software quality evaluation, and alternatively, they can also entrust professional agencies to participate in or help software quality control and assurance.

(3) The users, when selecting software suppliers, developers, needs to inspect whether that company has established its own software quality metric and evaluation data, whether the database has kept any software related to proper industry, and whether it has related experience of development.

In combination with foregoing software 3-tiered evaluation model, a quantitative software quality model is given here. This model, combined with evaluation chart (as shown in Figure 2), applies following rating method.



Figue2 Quantitative Software Quality Evaluation Standards

(1-16 as shown in TBALEⅡ)

TABLE Ⅱ

Cross Reference Table

1	Completeness
2	Consistency
3	Security
4	Consistence
5	Adequacy
6	Simplicity
7	Performance intensity scale line
8	Robustness
9	Versatility
10	Legibility
11	Modularity
12	Self-descriptiveness
13	Expandability
14	Independency of hardware
15	Independency of software system
16	Traceability

(1) Weight rating of principal characteristic factors: The evaluator, in combination with current software, will rate the weight of principal characteristic factors in evaluated model. There are 3 kinds of principal characteristic factors, namely functionality, reliability and portability. Weight is represented by the size of area taken by the factor in the evaluation chart.

(2) Weight rating of sub-characteristic factors: To compute the weights of sub-characteristic factors included in each principal characteristic factors. The number of sub-characteristic factors depends on the principal characteristic factor. The weight of sub-characteristic factor is represented by the percentage of such factor in the area of principal characteristic factor. It is easily found that the sum of every sub-characteristic factor's percentage in the total area is a fixed value 1, so the percentage of each sub-characteristic factor can be construed as such factor's weight in the system.

(3) Sub-characteristic factor performance rating: To estimate the performance score of each sub-characteristic factor, according to the evolution chart in the performance intensity scale division line divided with circles, and spot on the internal bisector of the sector zone occupied by each sub-characteristic factor according to estimate value.

(4) Connect all points to form a line, the surrounded area is the quality evaluation of such software.

The quantification formula is induced as follows: Supposing the evaluated software has n sub-characteristic factors, the performance score of which is represented by Vi, the internal angle of surrounded sector is represented by αi, where 1≤i≤n，0≤i≤MAX (performance intensity scale line), and the characteristic factor of continual marking in the evolution chart is neighboring.

By assumption

$$\sum_{i=1}^{n}\alpha_i = 1 \qquad (1)$$

Software quality evaluation:

$$Q = \sum_{i=1}^{n} V_i \times \qquad (2)$$

$$V_{(i+1)\bmod n} \sin\ (\alpha_i/2 + \alpha_{(i+1)\bmod n}/2) \qquad (3)$$

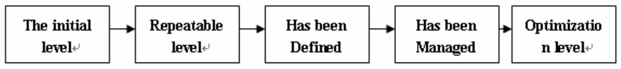
where mod represents modulo operation, that is, the residue of acquisition dividing operation [10].

IV. Capability Maturity Model(CMM)

CMM (Capability Maturity Model) is an early research results of non-profit organizations ----Software Engineering Institute (Software Engineering Institute, SEI) .SEI was federally funded and founded by the United States Congress and major U.S. companies cooperate with The Research Centers in 1984.The model provides the results of software engineering and management framework, has been made since the 90's, has been successfully applied in North America, Europe and Japan. The model has become the industry standards of software process improvement. Any development of software development, maintenance and software organizations can not be separated from the software process and software process experienced the process of development that immature to mature, imperfect to perfect. It need continuous improvement of the software process to obtain the final results. CMM is designed based on this guiding ideology. In order to guide the software process activities correctly and orderly, the model establish an effective description and expressed in the framework of software process improvement to enable it guide the various stages of the software process and management. The model is based on the concept of product quality and software engineering experiences and lessons to guide enterprises how to control the development, stick up for software production process and how to set a software process and management system.

（1）classification criteria

CMM model describe and analyse the software development process level of software process capability, establish a grading standards level of software process maturity, show in Figure 3. On the one hand, software organizations can make use of it to assess their current process maturity and advance a strict software quality standards and process improvement methods and strategies, through continuous efforts to achieve a higher level of maturity. On the other hand, the standard can also be used as a evaluation criteria of the organization for software user so that the user no longer be a blind and uncertain when in the choice of software developers.



Figue3 Level of the software process maturity

Hierarchical structure of the CMM can be described as:

The initial level: the character of software process is disorderly and sometimes disordered. The definition of software process almost in the state of following no rules and the steps , software product success is often highly dependent on the efforts of individuals and opportunities.

Reusable level: It has established a basic project management process and also can be used to track the cost, schedule and function features. When it comes to the similar application projects, can rule-based and repeat the success of the past.

Has been Defined level: The software process which is used to manage and engineer have been documented, standardized, and formed a standard software process for entire software organization. All projects use the normal software process which is in line with the actual situation and with appropriated modifications to operate.

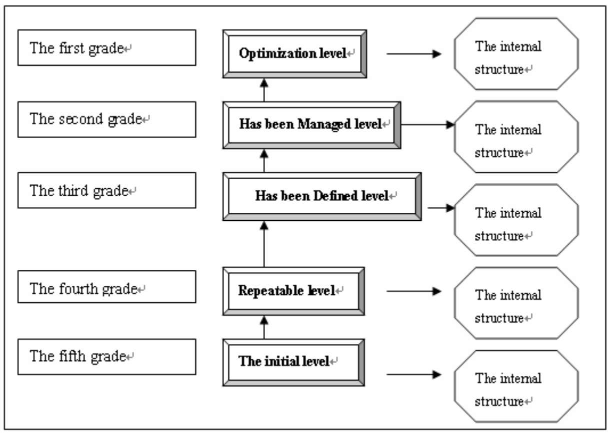
Has been Managed level: software process and product quality has measurement standard in detail. Software process and product quality has the quantitative understanding and control.

optimization level: can be constantly and continuously improve the promotion process via the process , new concepts and new technologies, such as the various aspects of the of useful information quantitative analysis . In addition to the first level, each level set a batch of objectives, if achieved the goal of this group indicating the maturity level has reached and can move to the next level. CMM systems are not in favor of inter-level evolution. Because each low-level, since the beginning of the second level, realization are the elements of the high-level realization.

(2) Main content of CMM

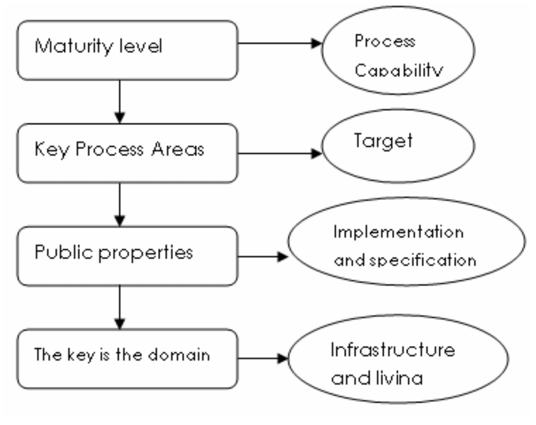
CMM provide a step-by-step evolution of the framework for process capability of software companies which use hierarchical way to explain the starting component as show in Figure 4. In the second to the fifth maturity level, each level contains a concept of the internal structure. Detail description of the internal structure of CMM in the following internal structure for a column. The process of every level move to higher level has its own specific plan to improve.

(3) The internal structure of CMM



Figue4 The five maturity levels of CMM

CMM provide a improvement means for software process capabilities. CMM Maturity comprises 5 levels and each maturity level have their own functions. Except the first level, every level of CMM has exactly the same internal structure. Show as Figure5. The top-level is Maturity level, different maturity levels reflect the software process capability of software organizations and the expected results extent of the organization may be achieved[8].



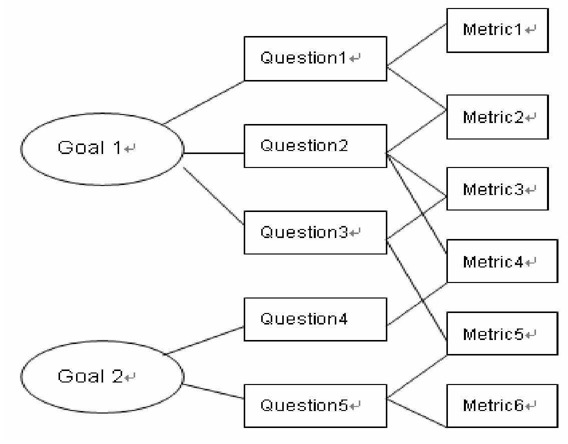
Figue5 Internal structure of CMM

V. Process Performance Metric and Analysis Model Study in Software Quality Management

Process performance metric in software quality management is the key to quantitative process management which relies on quantitative technique. In software production, all jobs are composed of a series of interrelated processes which are changing, so the establishment of effective process performance metric is the key to knowledge and understanding of the processes. The understanding of process data and changes is an important feature of high mature organization, establishment of effective process performance metric model is for one purpose of effective understanding of and control over processes, and for the other purpose of real time decision-making scheme based on process data.

A. GQM Rationale

Implementation of Goal-Question-Metric (GQM) method [11] will result in a specification of metric system,



Figue6 GQM Model

including a set of specific problems and a set of rules to interpreting observed data. The resulting metric model includes 3 tiers as shown in Figure6:

(1) Conceptual Tier (Goal): to designate a goal for metric object which is generally any or all of those software-related activities involving time, such as specification, design, test and review.

(2) Operating Tier (Questions): to describe those methods used to evaluate/realize a specific goal with a group of questions which describe the metric object based on the chosen quality points, and measure its quality from selected perspective.

(3) Quantitative Tier (Metric): To reply each question in quantitative manner based on a group of data acquired for such question. These data may be objective, or may be subjective.

The GQM method is based upon such a supposition: purposeful metric of a software organization requires clarification of the goal of its organization and the goal of each project, then it is required to define operable data for such goals and collect them, and finally it needs to provide a kind of framework to interpret such data based on determined goals [12]. The GQM model is a tiered structure, starting from a high level goal, for example, "to improve the timelines of demand changing process from a project manager's view". This goal designates the measure intent (improvement), measure object (demand changing process), measure focus (timeliness) and from which perspective to measure (from a project manager's view). These goals are segmented into many questions, usually segmenting measure focus as its key component. Then each question is segmented into metrics to answer such questions, and follow up the conformity of product and process with goals. Such metrics may include objective ones, or may include subjective ones. One metric may be used to answer different questions under the same goal. More than one GQM model may also share some questions and metrics. However, it should be noted that the measure should be performed correctly from defined perspective, that is, the measure may have different values from different perspective.

Benefit of this methods [13]:

(1) It can ensure the adequacy, consistency and completeness of measure plan and data collection. The designer of metric procedure (i.e. metric analyst) should have huge information and the interdependency between them. In order to ensure the measure set is adequate, consistent and complete, the analyst is required to accurately know any reason for measuring these properties, any implied premises, any model to be used in measuring data.

(2) It can also help manage the complexity of measure plan. When there are numerous measurable properties and the number of measures to be taken for each property is rising, the complexity of measure plan is undoubtedly increasing. In addition, the means chosen for full measure of a certain property will also rely on the measure goal. Without a goal-driven structure, the measure plan will go out of control soon. Without a mechanism capturing the interdependence between all properties, any change of measure plan may easily introduce inconsistency.

(3) In addition, it also helps us discuss the measure and improvement goals on the structural basis of common understanding, and finally form an agreement. Vice versa, this also enables us to define the widely accepted measure

and model in the organization. And such is premise of successful measure.

B. Problem and Development Existing in GQM Method

GQM technique is proven to have offered great help to the definition of reasonable measure, though it also has some limitations.

GQM method results in definition of measure through breaking down the goal and problem. But such process is not strictly defined, and its quality depends on executor's experience. David N. Card once pointed out following limitations existing in the GQM method [14]:

(1) It cannot ensure repetition: two different teams in the same organization may have different problems and measures, even if starting from the same goal. The same team will have some difference in defined problem and measure after segmenting the goal again after several months.

(2) It is unable to determine the time of termination: It is not explained in GQM when to stop raising problem and defining corresponding measure. Therefore, user may have raised too many questions for developing a "complete" measure plan, thus the final measure amount becomes very huge.

(3) It produces impractical results: Some questions raised by GQM method may be the ones that could not be answered by the organization, unless it transforms its operating style to enable the performance of necessary measure. If our main purpose is data collection (i.e. test carried bout by researchers), there is no problem. But in typical industrial environment, that is not suitable. The implementation of GQM producing a great deal of problems and measures with priority level will trap the entire organization into mud.

(4) In addition, our practice also indicates that because there is no clear explanation of size segmented by the "goal" in GQM method, we cannot make it clear when to stop such segmentation as we segment the initial goal into sub-goals. This segmentation process also has foregoing three limitations. What is more, there is no explanation about how to select measure goal in the GQM.

(5) In GQM, it also lacks the analysis on measure results, and the instruction on explanation work. Without reasonable analysis and explanation, we cannot make full use of the measure results, or even possibly mislead others.

At the same time, Basili's and Weiss' work takes consideration of the measure process and its validity, but does not combine the measure process with the software process where it is located [15]. Thus it can be seen that GQM can be regarded as one kind of guiding principle, used to direct the definition of measure, in stead of a strict engineering method in the design of measure system. The method often needs to be supplemented by project judgment and common sense. However, one kind of effective supplement to GQM is to model measure objects, through which, we can effectively select measure according to the validity, instead of merely based on our wills. To define the measurable properties and express the model which explains the relations between them is also very important for explanation of measure data. The analysis of measure results, in turn, also possibly deepens our understanding of the model, and helps us improve these models. To some degree, our modeling ability decides our measure ability.

Due to the recognition of various limitations in early versions of GQM, Victor R. Basili et al. are also constantly improving the GQM method. For example, they introduce the modeling thought in GQM, increase the modeling of measure object, and provide support for defining measure goal, but they have not given operable definition of modeling.

The GQM method obtains widespread application in the software industry, and many companies have published their experience in applying GQM. In addition, many people have made improvement or supplement, based on their practical experience, to the GQM method, like Rini van Solingen and Egon Berghout [16], also some people have developed measure tools to support GQM implementation [17]. Although it is largely improved, GQM has not completely solved the limitations above-mentioned, and measure plan maker needs to have a profound understanding of the organization's software in order to have meaningful segmentation of measure goal, in implementing the GQM method.

Although the GQM/GQIM method model provides a feasible method for the selection and definition of process performance in software quality management, it is quite abstract after all. But software organizations, when carrying on actual measure, are often not very clear about what measure goal and question to propose, or don't know which process performance to measure, but the CMM/CMMI model happens to provide possible solutions for these problems. CAS software, on the basis of GQM/GQIM, proposes the P-GQIM measure model [18], where P represents "the Process", GQIM represents "Goal-Question-Indication-Measure". This model increases process modeling content in view of actual situation of process measure, and increases data analysis and other parts, simultaneously has provided a more explicit instruction to the original part of GQM method. The main extension includes:

(1) Increases process modeling. We can model organization process in light of the organization's business goal, forming process model base. Every actual process performed by the organization is example of process model in the model base. In the process model including with the process correspondence's GQIM plan, such in the future when will work out the GQIM plan to the similar process's identical measure goal, may entrust with heavy responsibility the existing process wealth. Process model includes the GQIM plan specific to such process, so we can reuse the used process wealth in designing the GQIM plan for the same measure goal of the same process.

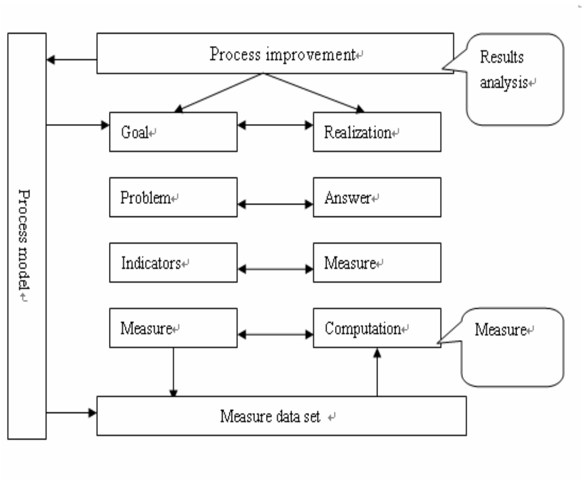
(2) Limit the options scope of measure goal. In the GQM method, there is no limitation to the options of measure goal, and moreover, the introduced process/product model is mainly used to break down problems according to the goal. In the P-GQIM measure model, the measure goal is limited to have options only

according to organization's business goal and corresponding process model. Such limit is to guarantee that the measure serves for the business goal, namely guarantee its rationality. Therefore, we establish the organization's process model according to its business goal at first, and then determine measure goal according to its business goal and process model, instead of opting for measure goal at first and then opting for process model.

(3) Break down the 3-tier structure of GQM (Goal-Question-Metric) into 4-tier structure of GQIM (Goal-Question-Indicator-Measure), where Indicator and Measure replace Measure section in GQM, enhance definition of indicators and make the tiers of model clearer.

(4) Enhance the technique of analysis and explanation on measure result, and add the improvement section of process model.

This model is shown in Figure 7:



Figue7 P-GQIM Model

C. Process Performance Measure & Analysis based on P-GQIM Model

(1) Starting from Business Goal

Although we call the P-GQIM Model as goal-driven measure model, its apparent difference with traditional goal-driven measure method is that it is starting from business goal. Traditional goal-driven measure method starts from measure goal, but in P-GQIM, the first to do is determine the business goal of the organization, and then we can build the model for the organization process based on business goal. Its measure goal will be exported from the business goal and process model ensuring the entire measure program to serve the business goal of the organization.

(2) Reusability of Measure Scheme

The P-GQIM Model records the GQIM plan of a certain measure goal specific to such process through the “measure” dimension in process model, in such a way, we can use existing GQIM plan directly in developing measure procedure under the same process and the same measure goal. In doing so, we can solve the problem “unable to ensure repetition” existing in GQM method to some degree.

(3) Consistency of Measure Scheme with Process

In P-GQIM Model, the measure goal, together with following problem, indicator, measure in the GQIM plan are all designed on the basis of existing process model, thus their consistency with process model is ensured. All measure data can be collected from examples of existing process models, therefore implementing the measure plan designed by the P-GQIM Model will avoid the limitation “producing impractical results” existing in the GQM method.

(4) Determination of Operability of Measure Goal

Through defined process model, it can also guide the breakdown of measure goal. If we select such goal as a higher level process, including sub-process, the goal size defined specifically for it will be probably larger. In this way, we need to break it down into several sub-goals step by step at first, and then design GQIM plan based on the final sub-goals. In breaking down the measure goal, we can make reference to the structure of process model. According to the sub-process composition of the process, we can beak down the measure goal into sub-goals specific to sub-processes, step by step until we break it down to the lowest level process unit.

However, due to close relationship between the problem and specific goal or and environment, there is not any good solution to the limitation “unable to determine when to stop raising the question and defining the corresponding measure” existing in the GQM method.

VI. CONCLUSION

Software quality evaluation is another important approach to further drive quality forward by great steps after software quality management and independent software test. It is fair to say that software quality evaluation provides an important guarantee for quantitative evaluation of software quality, and plays an irreplaceable role. This article gives a model for quantitative evaluation of software quality. Such a model is visualized, easy to measure, and it is easy to outline the effect curve of software characteristic factors on software quality according to this model, and better evaluate the quality level of the software, and accurately control, manage and improve software quality.

REFERENCES

[1] Roger S Pressman, “Softwear engineering, a practitioners’s approach”, [M]. *Fourth Edition. McGraw-Hill Press* , 1997.
[2] Witold Pedrycz and Giancarlo Succi, “ Genetic granular classifiers in Modeling software quality”,*.Journal of Systems and Software*,2005,76(3):277-285
[3] Anders Henriksson, Uwe Aβman,and ames Hunt, “Improving software quality in safety-critical applications by model-driven verification”, [J]*Electronic Notes in Theoretical Computer Science*,2005,133(31):101-117.
[4] Fan Dongping and Liu Youcheng, “Structure Modeling and System Building of Self-adaptation Application Software System”, [J] *Computer Engineering and Application*, 2001, 37(12).

[5] Fan Dongping, "Enterprise MIS System Component Developing Methodology Study" [D]. *Beijing: Beihang University*, 2001.
[6] Tang Ying, "Component Reuse Technique Research and Realization in the Development of Commercial Management Automation System", [D]. *Beihang University*, 2000.
[7] Li Jia, "MIS System Development Method Study and Component Library Realization", [D]. *Beijing: Beihang University*, 2001.
[8] Wang hao, "Based on J2EE technology software quality monitoring model research and application" *Dissertation Chinese full-text database*,2005
[9] Zhou Bosheng, Xu Hong and Zhang Li, "Introduction to Process Engineering Principle and Process Engineering Environment",[J].*SoftwareJournal,*1997.8, Additional: 519-534.
[10] Karl E and Wiegers. "Software Metrics: Ten Traps To Avoid",[J]. *Software Development*. 1997, October. pp. 111-125.
[11] Victor R. Basili and M. W. Weiss, "A methodology for collecting valid software engineering data",[J]. *IEEE Transactions on Software Engineering*. 1984, Nov, Vol. 10(NO. 6). pp. 36-49.
[12] Victor R. Basili, Gianluigi Caldiera and H. Dieter Rombach. "The Goal Question Metric Paradigm"[J], *Encyclopedia of Software Engineering-2* Volume. 1994, pp.528-532.
[13] Lionel C. Briand, Christiane M. Differding and H. Dieter Rombach,"Practical Guidelines for Measurement-Based Process Improvement",[J]. *Software ProcessImprovement and Practice Journal*. 1997, Vol. 2(4). pp. 231-238.
[14] David N. Card., "What makes for effective measurement",[J]. *IEEE Software*. 1993,Nov, vol. 10,pp. 94-95.
[15] Maurizio Morisio, "A methodology to measure the software process",[C].*Proceedings of the 7th Annual Oregon Workshop on Software Metrics*. 1995,pp.216-221.
[16] Rini van Solingen and Egon Berghout, "Integrating Goal-Oriented Measurement in Industrial Software Engineering: Industrial Experiences with and Additions to the Goal/Question/Metric Method(GQM)", [C]. *Proceedings of the 7th International Software Metrics Symposium.* 2001. pp. 178-186.
[17] Luigi Lavazza, "Providing Automated Support for the GQM Measurement Process",[J]. *IEEE SOFTWARE.* 2000, May/June. pp. 32-36.
[18] Yi Haifeng, "Software Quality Management Process Performance Measurement and Analysis" , *Dissertation Chinese full-text database*,2006

Yang AiMin

associate professor，（1967.8 ,born in HeBei) ,Holder of a master's degree in Tianjin Universyty in 1997.6 , specialty:Computer Organization and System Architecture.
major field of study : Computer Application
Current job:
Zhejiang Wanli University /Computer Science and Information Technology College,(2000-) Ningbo, CHINA
Major articles:

[1] Yang Ai-Min,Zhang Wen-Xiang ，A Fast Multiplier Design over Composite Fields 2006 , WSEAS TRANSACTIONS ON SYSTENS ,Issue 3,Volume 6,March 2007, (EI).
[2] Yang Aimin WuJunping, THE DIGITAL SYSTEM VIRTUAL LAB BASED ON EMBEDDED STRUCTURE, International Sysmposium on Computer Science and Technology,2007.5 (ISTP).
[3]Aimin Yang, Junping Wu，Lixia Wang ，Research and Design of Test Question Database Management System Based on the Three-Tier Structur ， WSEAS TRANSACTIONS on SYSTEMS Volume 7, 2008ISSN: 1109-2777

Zhang Wenxiang

professor,（1963.1 ,born in HeBei) ,Holder of a master's degree in BeiHang Universyty in 1999.3,specialty:Computer Software;

Zhejiang Wanli University /Computer Science and Information Technology College,(2001-) Ningbo, CHINA
major field of study : Computer Application
Current job:
Zhejiang Wanli University /Computer Science and Information Technology College,(2000-) Ningbo, CHINA