TRAINING, QUALITY ASSURANCE FACTORS, AND TOOLS INVESTIGATION: A WORK REPORT AND SUGGESTIONS ON SOFTWARE QUALITY ASSURANCE

Final Report

NASA/ASEE Summer Faculty Fellowship Program–1991

Johnson Space Center

Prepared By:	Pen–Nan Lee, Ph.D.
Academic Rank:	Assistant Professor
University & Department:	University of Houston – University Park Department of Computer Science Houston, Texas 77204–3475

NASA/JSC

Directorate:	Safety, Reliability and Quality Assurance Office
Division:	Quality Assurance and Engineering Division
Branch:	Software Product Assurance Office
JSC Colleague:	Vincent D. Watkins
Date Submitted:	August 16, 1991
Contract Number:	NGT–44–001–800

ABSTRACT

In the past Summer, several research tasks have been conducted, some observations were obtained, and several possible suggestions have been contemplated involving software quality assurance engineering in JSC. This report briefly describes these research tasks. Next, it gives a brief discussion on the role of software quality assurance in software engineering, following that are some observations and suggestions. A brief discussion on a training program for software quality assurance engineers is provided. A list of assurance factors as well as quality factors are also included. Finally, a process model which can be used for searching and collecting software quality assurance tools is presented.

1. INTRODUCTION

It has been a very pleasant and enjoyful experience to work with NASA/JSC software quality assurance engineers during this past Summer. The JSC Summer faculty fellow program office has provided us with a great deal of assistance including providing weekly seminars and tours, special events and activities, such as the SOAR'91 conference, and arranging meetings with JSC management. Every Summer faculty fellow also received the necessary help and convinence within each individual branch to make his/her task work done in the best possible way.

In the past 10 weeks, I worked on several tasks for Quality Assurance Engineering Office. One task was to search and collect materials and references related to software quality assurance engineering, and JSC technical library's RECON data base was the primary resource. Several thousand articles on quality and software engineering were saved to diskette. Key words used in the search for the quality articles were determined by searching JSC technical library's ARIN data base for general books on quality. The software life cycle was used to determine key words to use for searching for articles on software engineering. A list of 688 references in 90 pages has been produced. However, there are more references left to be collected.

The other task was to investigate tools which can be utilized by SQA engineers. After weeks of contacting dozens of organizations, I found that the best possible way could be to coordinate with JSC's Software Technology Brance (STB) and MITRE for the matter. They have a great volume of software available and free, unless otherwise noted, to needed organizations. Assistance in identifying, evaluating and developing software technologies and tools can also be attained from them. I accessed the "library X", but found none of the about 180 software programs has merits for direct usages in software quality assurance. However, there is more software, such as Autolib, COSTMODL, COMPASS, REAP, etc., which deserve a chance to be appraised and determined based on SQA needs.

I also tried to observe the role of SQA in JSC's software engineering and to give suggestions. A preliminary impression is obtained and included in this report. The rest of this report gives some details of my observations and suggestions.

2. THE ROLE OF QUALITY ASSURANCE IN SOFTWARE ENGINEERING

As mentioned in the introduction, only an impression was obtained due to the short period of time. Before expressing my observation, some issues regarding SQA in software engineering in general deserve a brief discussion. Within the context of "role", two issues are briefly stated in terms of quality assurance personnel.

2a. WITHIN THE ORGANIZATION

There are controversial opinions regarding the necessity of having an independent software quality assurance (SQA) group. Arguments for opposing an independent SQA include wasting resources or else other key problem areas could have made the endeavor a greater success, increasing costs, to nurture counterproductive attitudes such as they-against-us, not-our-responsibility, and the total quality management is not possible.

On the other hand, however, a good independent SQA group can shake that complacency of the development organization. The SQA group should become involved before a project begins and work with the managers and developers throughout the life of a project. As a team player, a good independent SQA and total quality management are complementary. In order to promote SQA staff as part of the team and achieve their trustworthiness, effective methods and tools have to be in place and necessary training must be available.

Managers and developers may also be provided with these methods and tools if necessary, hence, the total quality management is emplaced and high quality software is guaranteed.

2b. BETWEEN THE CONTRACTING ORGANIZATION AND ITS CONTRACTORS

Three different roles could be adopted, and each role has its own importance. The role of an "approver" is usually to make, or to make jointly, the decision of accepting and thus allowing the software to be used, and hence, has the decision authority as well as responsibility. The other role is the "evaluator" who's task is to attain a high level of understanding of the system, and to make a judgement as to the reliability based on criteria indicating the assurance level required. The criteria and assurance levels are set by the "policy authority", the third possible role, who is able to issue a certificate based on evaluation. For statutory integrity requirements, these three roles should not be mixed.

The software procurement process should include a set of assurance metrics or measures which would have to be applicable throughout the entire procurement process:

1. Procurement of a major system shall start with a definition of requirement and a risk assessment for the system. Based on the risk assessment, the requisite assurance level for each aspect of the system software should be specified.

2. The measures must be applicable during software development in order that the development process can be geared to produce an engineered level of assurance for the software.

3. The measures should be capable of use by an evaluation team. Specifically, they should form the basis of the assurance judgements made in evaluaton.

4. If a component is to be used in a number of applications, it is necessary to have a "certificate" indicating the assurance level of the component independent of its application. One would expect the assurance measures to be usable in certification.

5. The measures must help in making the decision whether or not to accept a given software in a particular environment. One should be able to compare the achieved assurance levels with the required assurance levels and determine whether or not the implemented software meets or exceeds its requirements.

2c. OBSERVATION

My impression is that SQA engineers in JSC are basically overloaded with work and duty, i.e. under-staffed. If the situation continues, two obvious things will happen: 1) a SQA staff member can not perform the best way he/she could due to lack of time, and 2) he/she must assume more duty and that may blur his/her role. One way to resolve this is to recruit more SQA engineers and provide them with quality training. The concept, which is under development in JSC, of attaching a software quality expert to each project throughout its life-cycle is an effective idea. As mentioned earlier, the expert will be part of the team, and he/she can handle the quality issues in details and in daily bases.

3. DIRECTIONS AND ADVANCEMENTS

Three discussions are given in the areas of training, assurance and quality factors, and investigation of tools. In section 3a, developing and delivering a joint training program with a major local university and industrial software developers is discussed. Section 3b discusses the assurance and quality factors, requirements of assurance measures, and the need of metrics. In section 3c, a process model for investigating SQA tools is presented

3a. TRAINING PROGRAM

It is essential to have skilled software engineering and quality assurance engineering work forces within NASA/JSC as well as its contractor companies. However, highly qualified software engineers are normally attained through training and extensive experience. The main reasons for the lack of these engineers are that most universities do not prepare students to develop industrial software and little is available in the way of continuing professional development.

It is important to develop and deliver software engineering and quality engineering training. By cooperating with a major local university and collaborating with key software developers, a professional training center could be established. Both dedicated consultant staff as well as full time professional training staff could work on developing the training program, to supply the knowledge and to teach the detailed techniques to the trainee.

An organization should establish a policy which requires every employee to receive a minimum number of hours of training per year. These hours, for examples 30 or 40, should be determined based on the organization's scope of needs.

The training center should provide various levels of training programs for a variety of participants, such as senior software engineers, project managers, etc. A basic program may contain a list of fundamental courses, such as Concepts in Programming, Ada, Operating Systems, etc. An advanced training program may contain courses in areas such as:

Project Management
Software Design
Configuration Management

Testing
Metrics
Risk Assessment
Software Process Development and Improvemen
Contracting Issues
etc.

3b. ASSURANCE FACTORS AND QUALITY FACTORS

Technical staff should be concerned with assurance throughout the entire procurement process as well as the installed system in the operational environment. They should be able to and must attain a set of "sound" assurance metrics for some given application domain. The soundness of the measures sould be assessed through the normal scientific proven process, or, in practice, employ the measures, after peer review and reasonable confidence are attained, in real world situations in order to provide evidence of how they work in practice.

The requirements of measures are concerned with the basis which is essentially a comparative analysis of the capabilities of different approaches to software development, and which identifies a range of factors which contribute to assurance. The following is a preliminary set of requirements for measures which should be helpful in any review process and which drive the selection of the technical factors which are believed contribute to assurance.

1. Scale: The measures should be applicable across the spectrum of computer systems.

2. Assurance Levels: The range spans from that gained in well tried commercial software to the highest level achievable with current software engineering techniques.

3. Uses: Applicable measures in specifying requirements, to assist in performing evaluation, to be usable as guidelines in development, and to form a basis for certification.

4. Users of the Measures: Meaningful measures to both "laymen" and technical staff in developing and assessing systems.

5. Consistency in Application: Insurance that consistent results are produced by different evaluation teams.

6. Profiles: A derivable assurance profile for a system, i.e. to assess the assurance in individual functions and properties of a system, rather than giving a blanket assurance level for the whole system:
   a) cost effective to have an assurance profile where there are high assurance defenses against the most serious threats, but where the rest of the system is developed to a lower level of assurance, than to develop a complete system to a uniformly high assurance level,

b) not desirable to have to give a low assurance level to a complete system if a single flaw is found which only affects one minor function. Thus, it is desirable both to specify an assurance profile in requirements, and to produce a profile in evaluation.

Factors contributing to assurance are essential to the applications. The following is a preliminary list of assurance factors which cover facets of the procurement process from initial elicitation of requirements through to the procedures for installing and updating the software in the operational environment.

1. Development: The concern with the technology deployed in the production of the software components.

2. Requirements: The confidence in the accuracy of the model of the operational environment for the software, and the normal software engineering issues of confidence that the requirements elicitation process has correctly identified the specific requirements for the system.

3. Architecture: A high-level design description which identifies the trusted and untrusted components and the interdependencies between the components.

4. Evaluation: The proper checking of the development process, and analyzing and testing the product to look for flaws.

5. Configuration Control: The concern with the ability to identify all components of the system, through all stages of development and evauation, and to control how they are changed.

6. Complexity: A major factor which impacts comprehensibility; the more complex the less assurance.

7. Human Interface: The confidence that the users would be able to comprehend the information from the computer system even during a safety-critical event.

8. Staff Issues: Related to the skills of the staff employed in development and evaluation.

9. Tools: Assurance in the tools which are used in support of development and evaluation.

Another key issue is to attain a set of quality measures. As mentioned above, the set of measures must be sound, that is fewer failures occur in systems judged to have high assurance than those judged to have low assurance. The difficulty is that there is no agreed objective measure of relevant aspects of software such as

reliability. The following is a preliminary list of quality factors. The need is to design measurable requirements.

1. Correctness: deals with the extent to which the software design and implementation conform to the stated requirements.

2. Efficiency: deals with the resources needed to provide the required funtionality.

3. Expandability: deals with the perfective aspects of software maintenance, i.e. increasing the software's functionality or performance to meet new needs.

4. Flexibility: deals with the adaptive aspects of software mantenance, i.e. modifying the software to work in different environments.

5. Integrity: deals with security against either overt or covert access to the programs or data bases.

6. Interoperability: deals with how easy it is to couple the software with software in other systems or applications.

7. Maintainability: deals with the ease of finding and fixing errors.

8. Manageability: deals with the administrative aspects of modification to the software.

9. Portability: deals with transporting the software to execute on a host processor or operating system different from the one for which it was designed.

10. Usability: deals with the initial effort required to learn, and the recurring effort to use, the functionality of the software.

11. Reliability: deals with the rate of failures in the software that render it unusable.

12. Reusability: deals with the use of portions of the software for other applications.

13. Safety: deals with the absence of unsafe software conditions.

14. Survivability: deals with the continuity of reliable software execution in the presence of a system failure.

15. Verifiability: deals with how easy it is to verify that the software is working correctly.

3c. A PROCESS MODEL FOR SQA TOOLS INVESTIGATION

The following is a process model for searching SQA tools. The model can also be used as a basis, i.e. some modifications may be needed, for coordinating with STB and MITRE for searching SQA tools. As mentioned earlier, both STB and MITRE can assist in performing the search task. This model contains 9 phases. The first three phases may not be necessary for someone who has been involved in JSC's SQA activites for long time, while the other six phases are needed in general:

PHASE

1. perform preliminary needs analysis
2. characterize the existing culture
3. analyze and summarize results of the characterizations: preliminary problem identification
4. identify candidate tools/environments that can meet requirements or improvement needs
5. present findings and obtain feedback
6. present final recommendations and plan for evaluation
7. arrange for evaluators from JSC/contractors to participate in decision
8. develop transition and insertion plan
9. develop comprehensive training plan

4. CONCLUSION AND ACKNOWLEDGEMENTS

It has been a very pleasant and enjoyful experience to work with NASA/JSC software quality assurance engineers. In the past Summer, several research tasks have been conducted, some observations were obtained, and several possible suggestions have been contemplated. This report describes these tasks, observations and suggestions.

I would like to thank E. Joseph Ripma and Vincent D. Watkins of the Software Product Assurance Office for their support and patience, George Neil of UNISYS for his enthusiastic attitude on software procurement, Ernest Fridge of STB and Dona Erb of MITRE for their assistance in searching for SQA tools and help in forming the process model mentioned in section 3c, and finally, Robert Youngblood and Bernice Mays of LORAL for sharing their office with me.

5. REFERENCES

BUCK89 Buckley, F., *Implementating Software Engineering Practices* John Wiley & Sons, New York, 1989.

CHAR89 Charette, R. N., *Software Engineering Risk Analysis and Management*, McGraw–Hill Book Company, New York, 1989.

DEIM90 Deimel, L. E., editor, *Software Engineering Education, SEI Conference 1990*, Springer–Verlag, New York, 1990.

DEUT88 Deutsch, M. S. and R. R. Willis, *Software Quality Engineering*, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

DUNN90 Dunn, R. H., *Software Quality: Concepts and Plans*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

EVAN89 Evans, M. W., *The Software Factory*, John Wiley & Sons, New York, 1989.

GERA86 Gerard, M. and P. W. Edwards, *Strategies for Revitalizing Organizations, Proceedings of the Second NASA Symposium on Quality and Productivity*, Washington, D.C., December 2-3, 1986.

HUMP88 Humphrey, W. S., "Characterizing the Software Process", *IEEE Software*, Vol. 5, No. 2, March 1988, pp. 73-79.

NASA91 NASA, *SMAP DIDS - Information System Life-Cycle and and Documentation Standards, Version 4.3., 1991*

PERR91 Perry, D. E., and G. E. Kaiser, "Models of Software Development Environments, *IEEE Transactions on Software Engineering*, Vol. 17, No. 3, March 1991, PP. 283–295.

PRES88 Pressman, R. S., *Making Software Engineering Happen,* Prentice Hall, Englewood Cliffs, New Jersey, 1989.

SENN89 Sennett, C. T., editor, *High–Integrity Software*, Plenum Press, New Tork, 1989.

VINC88 Vincent, J., Waters, A. and J. Sinclair, *Software Quality Assurance*, Volume I, & II, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

WALT91 Walters, N. L., "An Ada Object-Based Analysis and Design Approach", *ACM Ada Letters*, Vol. 11, No. 5, SIGAda, ACM, July 1991.