DOWNLOAD PDF SOFTWARE QUALITY ASSURANCE AND EVALUATION

Chapter 1 : ASQ - A Testing Maturity Model for Software Test Process Assessment and Improvement

In spite of that, it gives an excellent foundation is using software metrics, shows dependence of metrics on proper process, and shows the importance of how metric information is presented to the information that can be gleaned from the metric data.

The internal structure of the TMM is described, as well as the model framework of maturity goals, subgoals, and activities tasks and responsibilities that support the incremental growth of test process maturity. This article also addresses the TMM Assessment Model, which allows organizations to determine the current state of their testing processes and provides guidance for implementing actions to support improvements. Results from a trial evaluation of the TMM questionnaire in industry are discussed, and feedback received from the software industry regarding the TMM and maturity model integration issues is presented. They have a strong impact on vital operations in domains such as the military, finance, and telecommunications. For this reason, it is imperative to address quality issues that relate to both the software development process and the software product. The authors are developing a testing maturity model TMM designed to assist software development organizations in evaluating and improving their testing processes Burnstein, Suwanassart, and Carlson a,b, c. The TMM complements the Capability Maturity Model CMM by specifically addressing those issues important to test managers, test specialists, and software quality professionals. Testing as defined in the TMM is applied in its broadest sense to encompass all software quality-related activities. The authors believe that applying the TMM maturity criteria will improve the testing process and have a positive impact on software quality, software engineering productivity, and cycle-time reduction efforts. It is to be used by: None of these models, however, focuses primarily on the testing process. The concept of testing maturity is not addressed. Inadequate attention is paid to the role of high-quality testing as a process improvement mechanism. Testing issues are not adequately addressed in the key process areas. Quality-related issues such as testability, test adequacy criteria, test planning, and software certification are not satisfactorily addressed. Because of the important role of testing in software process and product quality and the limitations of existing process assessment models, the authors have focused their research on developing a TMM. The following components support their objectives: A set of levels that define a testing maturity hierarchy. Each level represents a stage in the evolution to a mature testing process. Movement to a higher level implies that lower-level practices continue to be in place. A set of maturity goals and subgoals for each level except level 1. The maturity goals identify testing improvement goals that must be addressed to achieve maturity at that level. The subgoals define the scope, boundaries, and needed accomplishments for a particular level. There are also activities, tasks, and responsibilities ATRs associated with each maturity goal that are needed to support it. An assessment model consisting of three components. The general requirements for TMM development are as follows. The model must be acceptable to the software development community and be based on agreed-upon software engineering principles and practices. At the higher maturity levels it should be flexible enough to accommodate future best practices. The model must also allow for the development of testing process maturity in structured step-wise phases that follow natural process evolution. There must also be a support mechanism for test process assessment and improvement. To satisfy these requirements, the following four sources served as the principal inputs to TMM development: The SW-CMM is a comprehensive process evaluation and improvement model developed by the Software Engineering Institute that has been widely accepted and applied by the software industry Paulk et al. Both models require that all of the capabilities at each lower level be included in succeeding levels. This view is essential, since a mature testing process is dependent on general process maturity, and organizational investment in assessments can be optimized if assessments in several process areas can be carried out in parallel. The TMM reflects the evolutionary pattern of testing process maturity growth documented over the last several decades. This model design approach will expedite movement to higher levels of the TMM as it will allow organizations to achieve incremental test process improvement in a way that follows natural process evolution. Designers of the SW-CMM also considered historical evolution an important factor in process improvement model development. For example, concepts

from Philip B. The authors used the historical model provided in a paper by Gelperin and Hetzel as the foundation for historical-level differentiation in the TMM. The Gelperin and Hetzel model describes phases and test goals for the s through the s. Testing was an ad hoc activity associated with debugging to remove bugs from programs. Current industrial testing practices. A survey of industrial practices also provided important input to TMM level definition Durant It illustrated the best and worst testing environments in the software industry at that time, and has allowed the authors to extract realistic benchmarks by which to evaluate and improve testing practices. Its influence on TMM development is based on the premise that a mature testing organization is built on the skills, abilities, and attitudes of the individuals who work within it. At the time the TMM was being developed, two other models that support testing process assessment and improvement were reported. Three maturity levels are described, along with six key support areas that are reported to be analogous to key process areas in the SW-CMM. The three levels are loosely defined as weak, basic, and strong. The internal level structure is not described in detail in the report, nor is it clear where the six key support areas fit into the three-level hierarchy. A simple score card that covers 20 test-process-related issues is provided to help an organization determine its Testability Maturity Model level Gelperin No formal assessment process is reported. Their model contains 20 key areas, each with different maturity levels. Each level contains several checkpoints that are helpful for determining maturity. In addition, improvement suggestions for reaching a target level are provided with the model, which are helpful for generating action plans. In contrast to these researchers, the authors have used a systematic approach to developing their TMM based on the four sources described, allowing them to satisfy the requirements for TMM development. The authors believe that their developmental approach has resulted in a TMM that is: More comprehensive and fine-grained in its level structure Supported by a well-defined assessment model Well defined and easier to understand and use Able to provide greater coverage of test-related issues Better suited to support incremental test process maturity growth. The model framework is shown in Figure 1. Burnstein, Suwanassart, and Carlson a,b, c. Each level implies a specific testing maturity. With the exception of level 1, several maturity goals, which identify key process areas, are indicated at each level. Each maturity goal is supported by one or more maturity subgoals, which specify less-abstract objectives and define the scope, boundaries, and needed accomplishments for a particular level. The maturity subgoals are achieved through a set of ATRs. The ATRs address implementation and organizational adaptation issues at a specific level. Activities and tasks are defined in terms of actions that must be performed at a given level to improve testing capability; they are linked to organizational commitments. Responsibility for these ATRs is assigned to the three groups that the authors believe are the key participants in the testing process: The hierarchy of testing maturity goals is shown in Figure 2. Burnstein, Suwanassart, and Carlson a, b, c. Following is a brief description of the maturity goals for all levels except level 1, which has no maturity goals. Phase Definition At TMM level 2 an organization begins to address both the technical and managerial aspects of testing in order to mature. A testing phase is defined in the software life cycle. Testing is planned, is supported by basic testing techniques and tools, and is repeatable over all software projects. It is separated from debugging, the latter of which is difficult to plan. Following are the level-2 maturity goals: Develop testing and debugging goals This calls for a clear distinction between testing and debugging. The goals, tasks, activities, and tools for each must be identified and responsibilities must be assigned. Management must accommodate and institutionalize both processes. Separating these two processes is essential for testing maturity growth since they are different in goals, methods, and psychology. Testing at TMM level 2 is now a planned activity and therefore can be managed. Managing debugging, however, is more complex, because it is difficult to predict the nature of the defects that will occur and how long it will take to repair them. To reduce the process unpredictability often caused by large-scale debugging-related activities, the project manager must allocate time and resources for defect localization, repair, and retest. At higher levels of the TMM this will be facilitated by the availability of detailed defect and repair data from past projects. Initiate a test planning process Planning is essential for a process that is to be repeated, defined, and managed. Test planning requires stating objectives, analyzing risks, outlining strategies, and developing test design specifications and test cases. Test planning also involves documenting test-completion criteria to determine when the testing is complete. In addition, the test plan must

address the allocation of resources, the scheduling of test activities, and the responsibilities for testing on the unit, integration, system, and acceptance levels. Institutionalize basic testing techniques and methods To improve testing process maturity, basic testing techniques and methods must be applied across the organization. How and when these techniques and methods are to be applied and any basic tool support for them should be clearly specified. Examples of basic techniques and methods include black-box and white-box glass-box testing strategies; use of a requirements validation matrix; and the division of execution-based testing into subphases such as unit, integration, system, and acceptance testing. Integration Testing at TMM level 3 is expanded into a set of well-defined activities that are integrated into all phases of the software life cycle. At this level management also supports the formation and training of a software test group. Following are the level-3 maturity goals: Establish a software test organization Since testing in its fullest sense has a great influence on product quality and consists of complex activities that are usually done under tight schedules and high pressure, it is necessary to have a well-trained and dedicated group in charge of this process. The test group formed at TMM level 3 oversees test planning, test execution and recording, defect tracking, test metrics, the test database, test reuse, test tracking, and evaluation. Establish a technical training program A technical training program ensures that a skilled staff is available to the testing group. At level 3, the staff is trained in test planning, testing methods, standards, techniques, and tools. The training program also prepares the staff for the review process and for supporting user participation in testing and review activities. Integrate testing into the software life cycle Management and technical staff now realize that carrying out testing activities in parallel with all life-cycle phases is critical for test process maturity and software product quality. Support for this integration may come from application of a development model that supports integration of test-related activities into the life cycle.

Chapter 2 : Software Testing QA, QC & Testing

Quantitative Evaluation of Software Quality B. W. Boehm J. R. Brown M. Lipow TRW Systems and Energy Group Keywords software engineering quality assurance.

The initial exploring can then be based on feature analysis. Since the information systems management of Banco Central de Venezuela BCV expects to improve on the quality of the development process, the nature of the impact in this case is qualitative. The nature of the evaluation object is clearly a tool, meaning a specific approach within a generic paradigm. BCV would like to select a software quality management tool. The scope dimension of the impact is the extent of it. It identifies how the effect of the tool is likely to be felt by the product life cycle. In this case, it is limited to all stages of software development, aiming to improve the quality software process. According to the maturity of the tools, they become very relevant to organizations that aim to improve their software products or process. The learning time aspect involves two issues: The tools studied here are available in the market. Finally, the authors assume that the evaluation maturity of the organization is a qualitative evaluation capability, because they have well-defined standards for software development and the adherence to these standards can be monitored. DESMET recommends case study quantitative and feature analysis and feature analysis survey for a long or medium timescale. According to the comments given for each method, the authors observe that feature analysis case study FACS is the favored method. Both methods could be applied with a high-risk ranking. However, FACS seems to be better, since quantitative case study requires more than one project. Finally, the cost criterion will be considered. According to Kitchenham and Jones , there are certain considerations when presenting and analyzing the results of an evaluation based on the FACS method. The analysis should be based on the differences among the values obtained for each evaluated tool when there is an explicit level of acceptance. There is a group of activities, specific for the evaluation, that should be performed. To identify the tools to evaluate To identify a group of characteristic to evaluate To evaluate the tools against the identified characteristics To select a project pilot To test each tool in the project pilot To assign value to each characteristic for each tool To analyze the resulting values and carry out an evaluation report The deliveries of these activities are described in the next sections. Each tool supports certain quality management tasks estimation, project management, so on. This will enable one to determine the meeting percentage of functionality presented by each one vs. The features proposed, which are broad and generic for any tool that tends to support this discipline, will represent this. The comparison between the tools makes it possible to determine the scope of each tool with respect to minimal requirements established that is, the weight assigned to every feature. In this way, the more complete tool will be the one that has more and better features. The characteristics used to evaluate each tool are reflected in the set of features proposed to select tools that support software quality management, which constitutes the objective of this research. These features are inspired by an extensive review of innovation and diffusion literature on software quality, quality management, quality assurance, quality planning, quality control, and technology management. The set of proposed features has been classified in technological and organizational types. These features support the selection process of a quality management tool. The technological features refer to the tool directly, such as design, use, and its atmosphere. The organizational features are related to the use of this type of tool in organizations. Based on Rojas et al. Internal features are related to the evaluated item tool or organization issue. External features are related to the context issue. Since features and metrics proposed are generic and represent the requirements for a quality management tool, some will not be applied depending on the particular scope of every tool. However, a whole application enables one to evaluate the meeting level of organizational requirements that is, the weight assigned to every feature. It means that the best tool will be the one that shows the highest values for the features evaluated. Metrics for Technological Features The metrics for technological features, either internal or external, are classified based on 12 approaches: Figure 1 shows internal and external technological features. Metrics for Organizational Features The metrics for organizational features, either internal or external, are classified based on four approaches: Figure 2 shows the internal and external organizational features. There are seven types of answers

that can be obtained see Figure 3: To carry out any mathematical and logic operation on the answers to the questions associated with each feature, the values of the answers should be standardized using the domain types. In this sense, the answers were given on a scale from 1 to 5 where 1 is the minimum and 5 is the maximum value. The facilitator assigns a weight to each question. The weight corresponds to a real value between 1 and 5, where 1 represents less importance and 5 represents more importance to the evaluator organization. Once all answers are standardized in the 1 to 5 scale, each must be multiplied by its associated weight. In this way, the final value of the answer is obtained. Now, each feature has an associated series of questions, and their answers have values in one domain. To assign a value to the metric, an algorithm should be applied to take into account the final value of the answers. If at least half of the answers have values greater than or equal to 3, then the metric value is the average of the answers; Otherwise, the metric value is 1 Applying the same algorithm, the value for each approach can be calculated: The value of each approach is: If at least a half of the metrics have a value greater than or equal to 3 points, then the approach value is the metrics average; Otherwise, the approach value is 1 The value of each category is: If at least a half of the approaches have a value greater than or equal to 3 points, then the category value is the approaches average; Otherwise, the category value is 1 The value of each feature type is: Based on the concepts presented related to FACS evaluation method, it should be noted that, for evaluating the selected tools, the manager of information systems of BCV carried out the role of sponsor, and the authors of this article carried out the evaluator and advisor roles. A questionnaire was developed containing the questions associated with each feature. Then, a repository was created for each tool to collect the questions, standardized answers associated to each feature, and the organization weight assigned to each question. This repository also contained the calculations according to the algorithm presented previously to obtain the value of the metrics for approach, category, and type. A second repository was used to collect the values of the metrics and their precedent hierarchies approach, category, and type. Construct Validation of Questionnaire Content validity, which assesses the completeness and soundness of the measurement, was established through the careful selection of items that had been validated in prior studies. To further reduce the possibility of any nonrandom error, five academic experts from different universities and senior information systems executives were asked to review the questionnaire with respect to its validity, completeness, and readability. The analysis showed that the reliability of variables was significantly higher than the value of 0. Principal component factor analysis was used to test this validity property. Data Collection To obtain the answer to each question, an evaluation of each tool was made by using the product and analyzing its whole documentation, considering the characteristics and constraints of the case study as the pilot project. For the questions whose answers were not observable directly, a questionnaire was sent to both suppliers or dealers and clients or users to obtain more information. When there is an explicit level of acceptance, the analysis should be based on the differences among the values obtained for each tool evaluated. The final value of the evaluation was considered in order to make a decision regarding the tool to be acquired. A tool with a value smaller than 3 explicit level of acceptance, according to the FACS evaluation method is not advisable and needs a bigger analysis on the part of the evaluator organization. G was the one that obtained an evaluation with more value 4. The difference between the first and the second is quite big 0. This suggests that for evaluator organizations, it would not be very complicated to select G as the best tool. Figure 4 shows the obtained results. Since the value obtained in the organizational features oscillates between 4. If some doubt is presented, the final decision can be based on the organizational metrics. The results of the technological features are shown in Figure 5. It is important to note that only seven of the nine tools have obtained a value greater than 3: The other two tools not mentioned in this group have their strength in aspects that the evaluator organization did not consider as high of a priority. The higher tools still are the same, but the fourth became E. The difference between the first and the second is even bigger 0. Again, it suggests that it would not be complicated to select G as the most appropriate tool. On the other hand, the values can also be analyzed starting from the internal and external technological features. Only four tools obtained both values higher than 3 points: The differences among the values of the first two tools are again big 0. These results are shown in Figure 6. The D, F, I, C, and E tools do not figure in the first places because of reasons similar to those outlined when the analysis of the tools evaluation results, according to the proposed

technological and organizational features, was carried out. Figure 6 shows the five tools whose values of internal and external technological features are lower than 3. Four C, E, F, and I show external feature values bigger than internal ones. This indicates that the final value of the technological features for these tools is being driven by the external technological metrics. This aspect has been considered lower than the external ones for the decision-making process. The evaluation of these tools was carried out according to BCV needs. Because of this, the results can vary from one organization to another. The remaining five tools obtained good results in aspects that the evaluator company did not consider important according to their current needs. Although the tools D, F, I, C, and E were not inside the first four positions mentioned previously, it is important to highlight the following: The processes documentation feature shows that D is an excellent tool to manage the documentation activities. The quality planning and costs estimate features place F as a good tool to carry out the inherent activities to the quality planning process and to reduce the defects correction costs. The development processes quality analysis, processes documentation, quality planning, and software quality evaluation features demonstrate that I is a good tool to introduce quality to the software processes of conformity to ISO and The quality control and software quality evaluation features place C as an excellent tool that serves from support to the planning phase of the measuring of programs based on the goal question metrics GQM paradigm. The software product quality analysis, software quality evaluation, and measuring the product and the software development process quality metrics show that E is a good tool to implement the metric derived from the GQM paradigm. The tools that support software quality management have a great utility, since they make it possible to plan and track down the quality activities characteristic for each of the software development process phases.

Chapter 3 : Software quality - Wikipedia

This is a methodology and process for repeatable assessment of software life cycle quality risks, such as maintainability, evolvability, and portability. The no-cost license includes some training materials and a software toolkit.

It is extracted from their Reusable Software Management Plan. Each Provider shall include in the required software management plan see section 1. SQA includes the process of assuring that standards and procedures are established and are followed throughout the software acquisition life cycle. Compliance with agreed-upon standards and procedures is evaluated through process monitoring, product evaluation, and audits. Software development and control processes shall include quality assurance approval points, where an SQA evaluation of the product shall be done in relation to applicable standards. The Project SAM is assigned this oversight responsibility. As part of the oversight role, the Project will perform both scheduled and unscheduled audits of providers to establish the degree of conformance to the standards and procedures and to reported status. The provider shall conduct software quality assurance activities throughout the software life cycle in accordance with the following requirements: During these development cycles, the software quality assurance activity shall conduct the appropriate phase-specific activities described above. The provider shall explain, in the required SMP, the methods and techniques to be used. The results of the audit will be conveyed to the provider so that appropriate action can be taken to correct any deficiencies found. Copies of the reports shall be furnished to the Project, as required in section 4. The reviews shall encompass the items to be included in the configuration management baselines to be established after the successful completion of the review. After each formal review, the Project Software Manager will decide upon the readiness of the provider to begin the next development life cycle phase. The Project Software Assurance Manager will make a readiness recommendation to the Software Manager based on an assessment of status and readiness of processes, procedures and standards needed in the next phase. After completion of rework for problems found during the review and correction of any readiness problems, permission to begin the next phase will be given. They will be conducted by the Office of Flight Assurance. These reviews will be at the system level, and software will be among the items reviewed. The provider shall support these reviews as required above for Project formal reviews. The provider shall conduct an inspection and internal technical review program as follows: The provider shall conduct three levels of testing: Unit, and integration testing shall be informal testing conducted by the provider. Acceptance readiness testing shall be formal testing conducted by the provider and witnessed by the Project. The purpose of acceptance readiness testing shall be to show that the software is ready for acceptance testing by the Project. Test planning shall be done for all levels of testing. The provider shall submit to the Project for review and approval test plans for the formal testing, such as the acceptance readiness testing. Once a test plan is approved, the provider shall prepare test procedures according to DID A The procedures shall be used for the tests, and shall be available for Project review and comment. The Project will conduct Formal Acceptance testing on delivered software, following a Test Readiness Review of the results of Acceptance readiness testing. Formal Acceptance Testing will include the generation of the system from source code, using installation procedures provided. The Project will prepare test plans, based on requirements and operations manuals, and will develop test procedures according to DID A The provider shall correct all discrepancies found. The provider shall conduct verification and validation activities throughout the software life cycle in accordance with the following: Copies of visual aids and other supporting material that are pertinent to the review shall be submitted to the Project at least 3 working days before the review. Walkthroughs shall be conducted according to provider developed and Project approved procedures. The provider shall conduct informal testing in accordance with Project-approved provider standards and procedures. The provider shall conduct, and the Project will witness, formal testing in accordance with the Project-approved test plan and provider developed procedures. The results of informal reviews and walkthroughs shall be documented in the appropriate Software Development Folder See Section 5. The provider shall summarize the results of informal reviews and walkthroughs in the Monthly Progress Report. The results of informal testing shall be recorded in the appropriate software development folders. Software

quality is often defined as the degree to which software meets requirements for reliability, maintainability, transportability, etc. In addition, the program shall satisfy the requirements in the remainder of 8. These categories, the software assigned to each category, and the activities to be conducted are described in section 5. The provider shall collect data, analyze metrics, and use them to guide quality engineering activities. Metrics and associated requirements are described in section 5. The Project will compute metrics and trends using PC based tools. Identified safety risks will be tracked by the Project as technical risks, and risk mitigation actions will be the responsibility of the SRMB. In addition, the software safety program shall satisfy the requirements in the remainder of section 8. For identified safety critical software components, software safety activities shall be initiated to include requirements, design, and code analyses and special testing. The provider shall include the following in the SMP: The information, including both programs and data, will be categorized according to its sensitivity. The categorization will meet the requirements contained in NMI The security requirements shall encompass system access control, including network access and physical access; data management and data access; environmental controls power, air conditioning, etc. A minimum security assurance program shall ensure that: The provider shall describe in the SMP the planned approach to meeting the security and privacy requirements.

Chapter 4 : Selecting Tools for Software Quality Management - ASQ

The Quality Assurance (QA) approach to addressing quality of care issues incorporates three core quality assurance functions: defining quality, measuring quality, and improving quality (QAP/URC, a.) The QA triangle effectively illustrates the synergy between these three QA functions.

Measuring software quality is motivated by at least two reasons: Software failure has caused more than inconvenience. Software errors have caused human fatalities. The causes have ranged from poorly designed user interfaces to direct programming errors. An example of a programming error that led to multiple deaths is discussed in Dr. As in any other fields of engineering, an application with good structural software quality costs less to maintain and is easier to understand and change in response to pressing business needs. Moreover, poor structural quality is strongly correlated with high-impact business disruptions due to corrupted data, application outages, security breaches, and performance problems. However, the distinction between measuring and improving software quality in an embedded system with emphasis on risk management and software quality in business software with emphasis on cost and maintainability management is becoming somewhat irrelevant. Embedded systems now often include a user interface and their designers are as much concerned with issues affecting usability and user productivity as their counterparts who focus on business applications. The latter are in turn looking at ERP or CRM system as a corporate nervous system whose uptime and performance are vital to the well-being of the enterprise. This convergence is most visible in mobile computing: In both cases, engineers and management need to be able to make rational decisions based on measurement and fact-based analysis in adherence to the precept "In God we trust. All others bring data". Edwards Deming and others. Definitions[edit] There are many different definitions of quality. For some it is the "capability of a software product to conform to requirements. The first definition of quality History remembers is from Shewhart in the beginning of 20th century: There are two common aspects of quality: The other has to do with what we think, feel or sense as a result of the objective reality. In other words, there is a subjective side of quality. The transcendental perspective deals with the metaphysical aspect of quality. In this view of quality, it is "something toward which we strive as an ideal, but may never implement completely". The product perspective implies that quality can be appreciated by measuring the inherent characteristics of the product. The final perspective of quality is value-based. This perspective recognises that the different perspectives of quality may have different importance, or value, to various stakeholders. Software quality according to Deming[edit] The problem inherent in attempts to define the quality of a product, almost any product, were stated by the master Walter A. The difficulty in defining quality is to translate future needs of the user into measurable characteristics, so that a product can be designed and turned out to give satisfaction at a price that the user will pay. This is not easy, and as soon as one feels fairly successful in the endeavor, he finds that the needs of the consumer have changed, competitors have moved in, etc. Two of these meanings dominate the use of the word: Quality consists of those product features which meet the need of customers and thereby provide product satisfaction. Quality consists of freedom from deficiencies. Nevertheless, in a handbook such as this it is convenient to standardize on a short definition of the word quality as "fitness for use". Reliability An attribute of resiliency and structural solidity. Reliability measures the level of risk and the likelihood of potential application failures. It also measures the defects injected due to modifications made to the software its "stability" as termed by ISO. Efficiency The source code and software architecture attributes are the elements that ensure high performance once the application is in run-time mode. Efficiency is especially important for applications in high execution speed environments such as algorithmic or transactional processing where performance and scalability are paramount. An analysis of source code efficiency and scalability provides a clear picture of the latent business risks and the harm they can cause to customer satisfaction due to response-time degradation.

DOWNLOAD PDF SOFTWARE QUALITY ASSURANCE AND EVALUATION

Chapter 5 : Software Quality Assurance

Note: Citations are based on reference standards. However, formatting rules can vary widely between applications and fields of interest or study. The specific requirements or preferences of your reviewing publisher, classroom teacher, institution or organization should be applied.

Eventide has developed Quality Factor software to help you answer the following questions and more: How are your dispatchers and call takers performing? What training should you offer them? Who are your stars? Who needs to improve specific skills in order to get that raise or promotion? Quality Factor software is a tool to help communications center managers evaluate and quantify the performance of dispatchers and call takers. With Quality Factor software, you can easily measure performance trends and identify the skills that need improvement. Evaluation questions and forms can be quickly adapted for special incidents, changing protocols, and new requirements. Simply select a call or calls , right-click, select Evaluate, and choose the desired form. The evaluation form shown below appears in a new tab. Quality Factor Reports Quality Factor reports help identify key performance results on an individual, shift, and center basis so that training can be focused on areas that need improvement. Quality Factor software includes the following standard reports: Evaluation Score Trends Report: Provides graphical displays of evaluation scores over time for all agents, per agent, and per evaluator. Also displays flags per agent. Intragroup Score Trends Report: Provides graphical displays of evaluation scores over time for all agents in a single agent group, and per agent within the group. Interform Score Trends Report: Provides a graphical display of evaluation scores over time, between forms. Provides a graphical comparison of evaluation scores by skill.

Chapter 6 : Quality Monitoring | Quality Assurance | Call Center Management

The scope of application of the quality models includes supporting specification and evaluation of software and software-intensive computer systems from different perspectives by those associated with their acquisition, requirements, development, use, evaluation, support, maintenance, quality assurance and control, and audit.

Chapter 7 : NASA SQA Plan Template

Software Quality Assurance & Testing Stack Exchange is a question and answer site for software quality control experts, automation engineers, and software testers.

Chapter 8 : Quality Factor QA Software | Eventide Communications

The purpose of this Software Quality Assurance Plan (SQAP) is to define the techniques, procedures, and methodologies that will be used at the Center for Space Research (CSR) to assure timely delivery of the software that meets specified requirements within project resources.

Chapter 9 : What is Software Quality Assurance (SQA)? - Definition from Techopedia

The NIST SAMATE (Software Assurance Metrics And Tool Evaluation) project is dedicated to improving software assurance by developing methods to enable software tool evaluations, measuring the effectiveness of tools and techniques, and identifying gaps in tools and methods.