IMC is a Service-Disabled Veteran-Owned Small Business (SDVOSB)

A QUANTITATIVE APPROACH TO SOFTWARE QUALITY EVALUATION

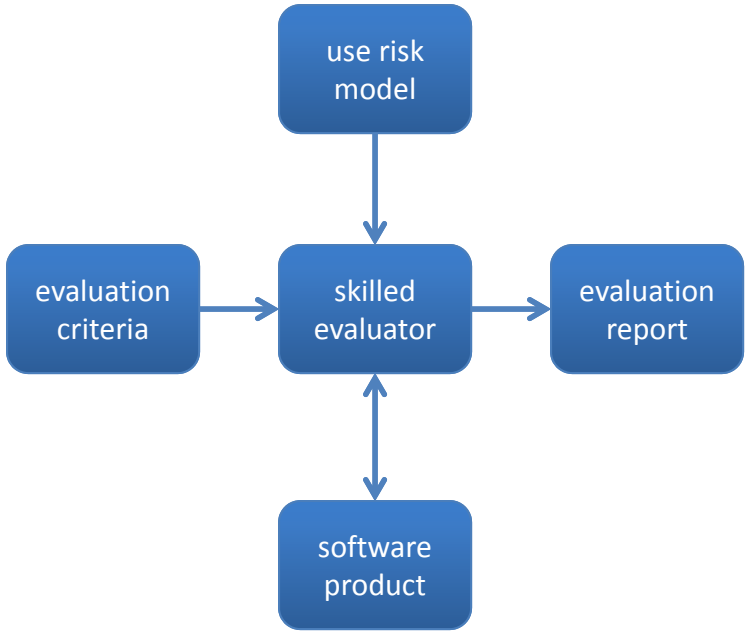
S.Y. Harmon & M.L. Metz
Innovative Management Concepts
2014 Software Technology Conference

Technical Problem - Challenges that Face Software Quality Evaluation

- The value of SQE products depends directly upon the truth of the information in those products but … truth is hard to define much less measure.
- Measurement repeatability and reproducibility are necessary conditions for truth and these can be measured through repeated experiments. As information approaches being absolutely repeatable and reproducible (i.e., zero variances), it approaches truth.
- All information gained through observation can only approximate truth. Information consumers would find useful measures of how faithfully an information product represents truth.
- These observations suggest that credible evaluations of software quality should
  - apply the entire range of possible products from the software development process;
  - be repeatable and reproducible;
  - be accompanied by estimates of their uncertainties (i.e., measures of how well they approximate truth); and
  - indicate something meaningful about the experience of using or maintaining the software product.
- Actual SQE situations vary considerably so practical solutions also need to be easily tailored and adapted in real time.

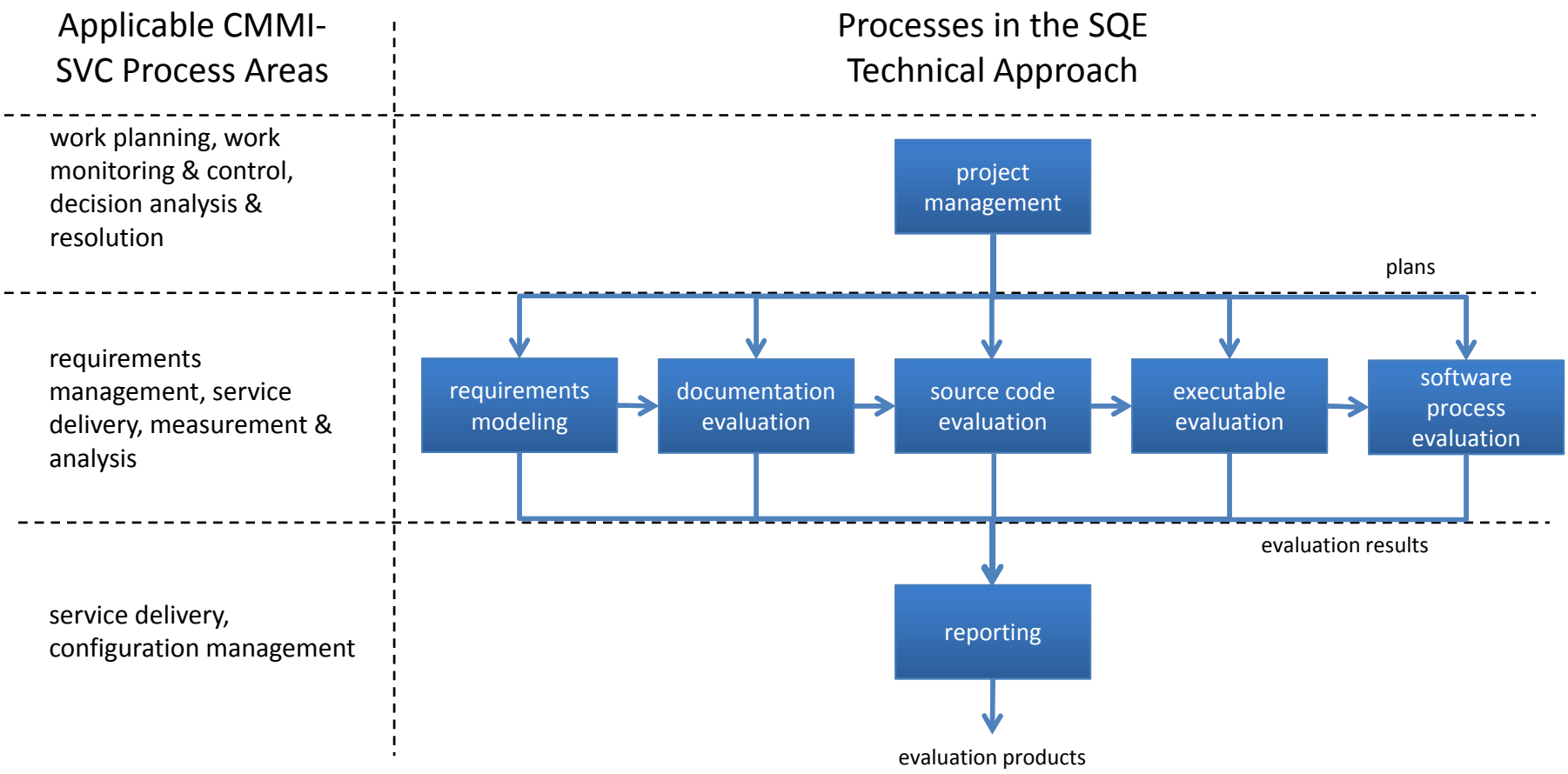
Technical Approach to SQE of Any Software Product – Implement a Criteria-Based Process

- Represent the SQE process as an instrument observing the characteristics of a software product.
  - This instrument consists of a
    - Skilled evaluator and
    - Evaluation criteria that the evaluator is trained to use.
  - The evaluator examines the software product and identifies locations where that product contradicts one or more evaluation criteria. Each detected contradiction produces a separate anomaly observation.
  - The evaluator then applies a use risk model to the collected anomaly observations to estimate the magnitudes of different components of use risk.
- Maintain a database of the collected evaluation results to define baselines from which to estimate observation uncertainties.
- Tailor SQE projects to meet user quality needs by changing evaluation criteria, number of evaluators and evaluation scope.
- Employ closed-loop project management that monitors and controls evaluator assignments to optimize product quality and cost from observations of actual product quality and evaluator effort.



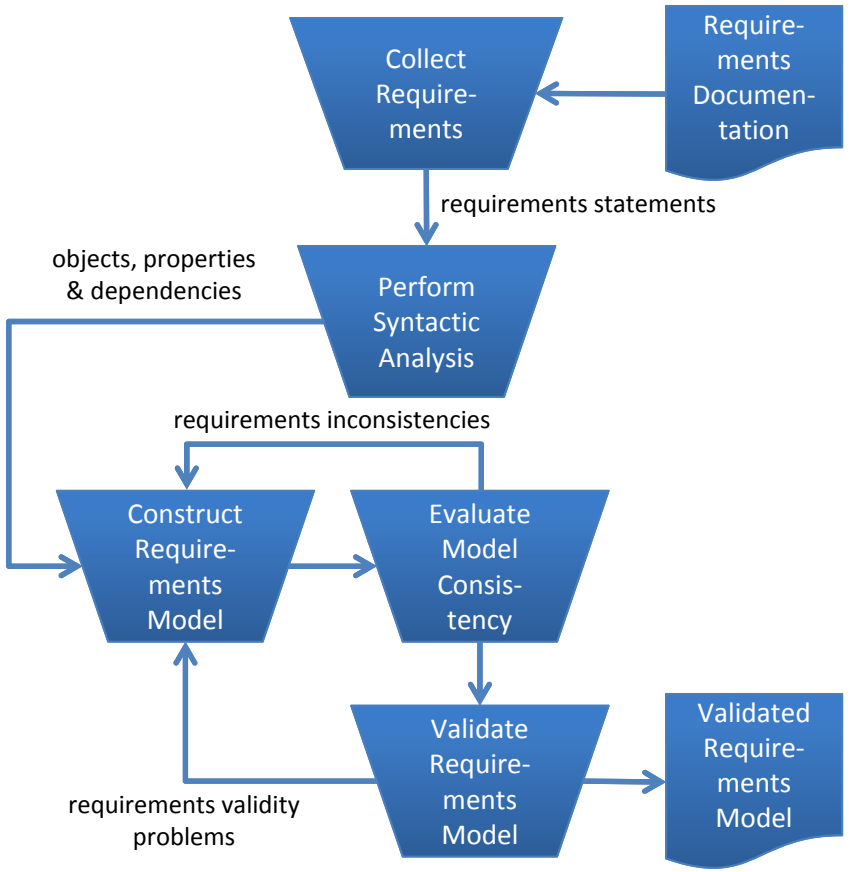
Software products include documentation, source code and executable packages. Each of these product types requires different evaluator skills and evaluation criteria.

Processes in this SQE Approach – Leverage Guidance from CMMI Services



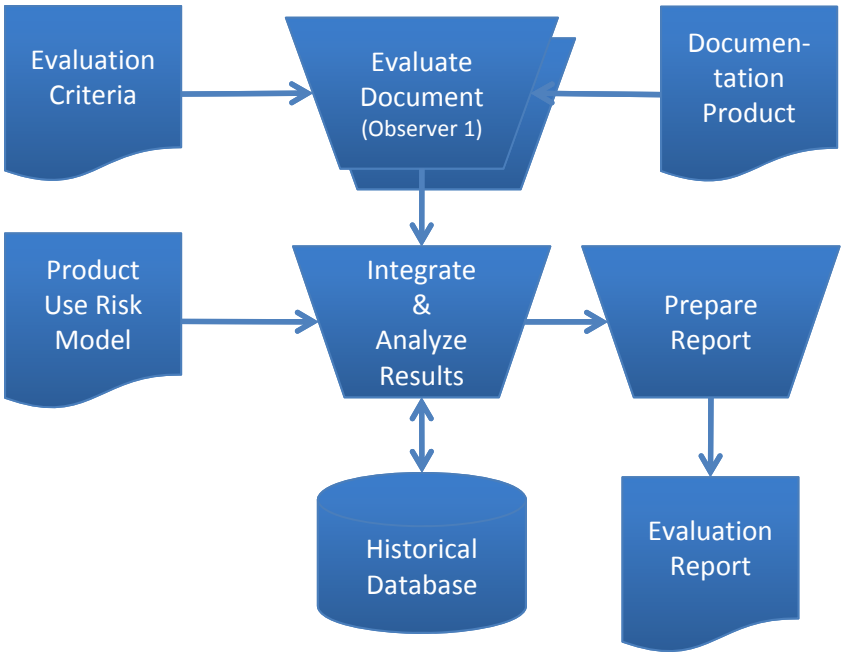
Requirements Modeling – An Essential Part of SQE

- Well-formed requirements are essential to evaluating the ability of software to meet the needs of its users.
- This process prepares a concise but formal description of the known requirements to drive risk-based evaluation, estimate evaluation coverage and reduce evaluation uncertainty.
- Requirements come in a variety of forms; having a consistent representation reduces the subjectivity introduced when evaluators interpret requirements statements.
- The final requirements model consists of two graphs:
  - Object graph – vertices represent objects & edges represent the interactions between objects and
  - Dependency graph – a bipartite graph with vertices that represent both object properties and dependencies & edges that represent the mappings between the object properties and the dependency independent and dependent variables.



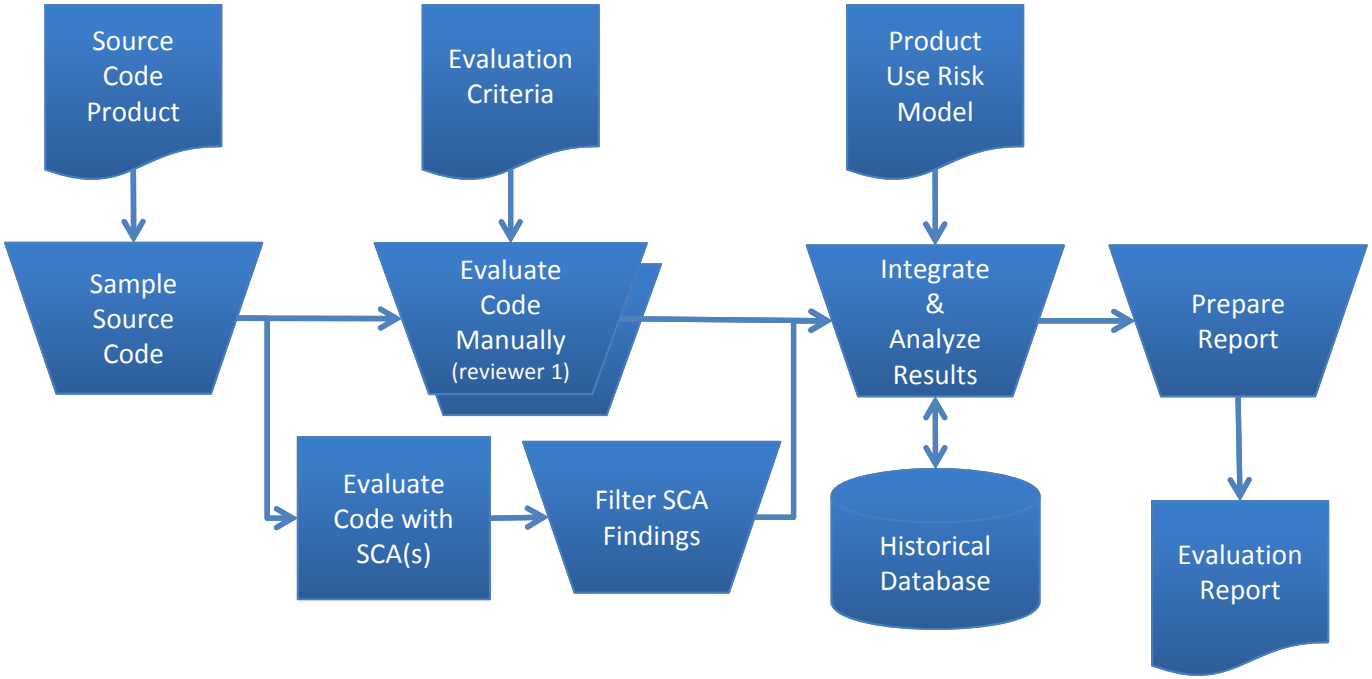
Documentation Evaluation – Determining the Quality of Document Products

- Documentation (corresponds loosely with software development phases)
  - Process documentation (plans, reports)
  - Design documentation (e.g., ADD, SDD, IDD)
  - Release documentation (e.g., user manuals, user training, technical manuals, compliance documentation)
- Evaluate document internal consistency as well as consistency with evaluation criteria
- Apply multiple evaluators then integrate their anomaly observations
- Use the variances between evaluators' observations to estimate the uncertainties associated with the evaluation results
- Build baselines from historical data that can then be used as standards for comparison



Source Code Evaluation – Determining the Quality of Source Code Products

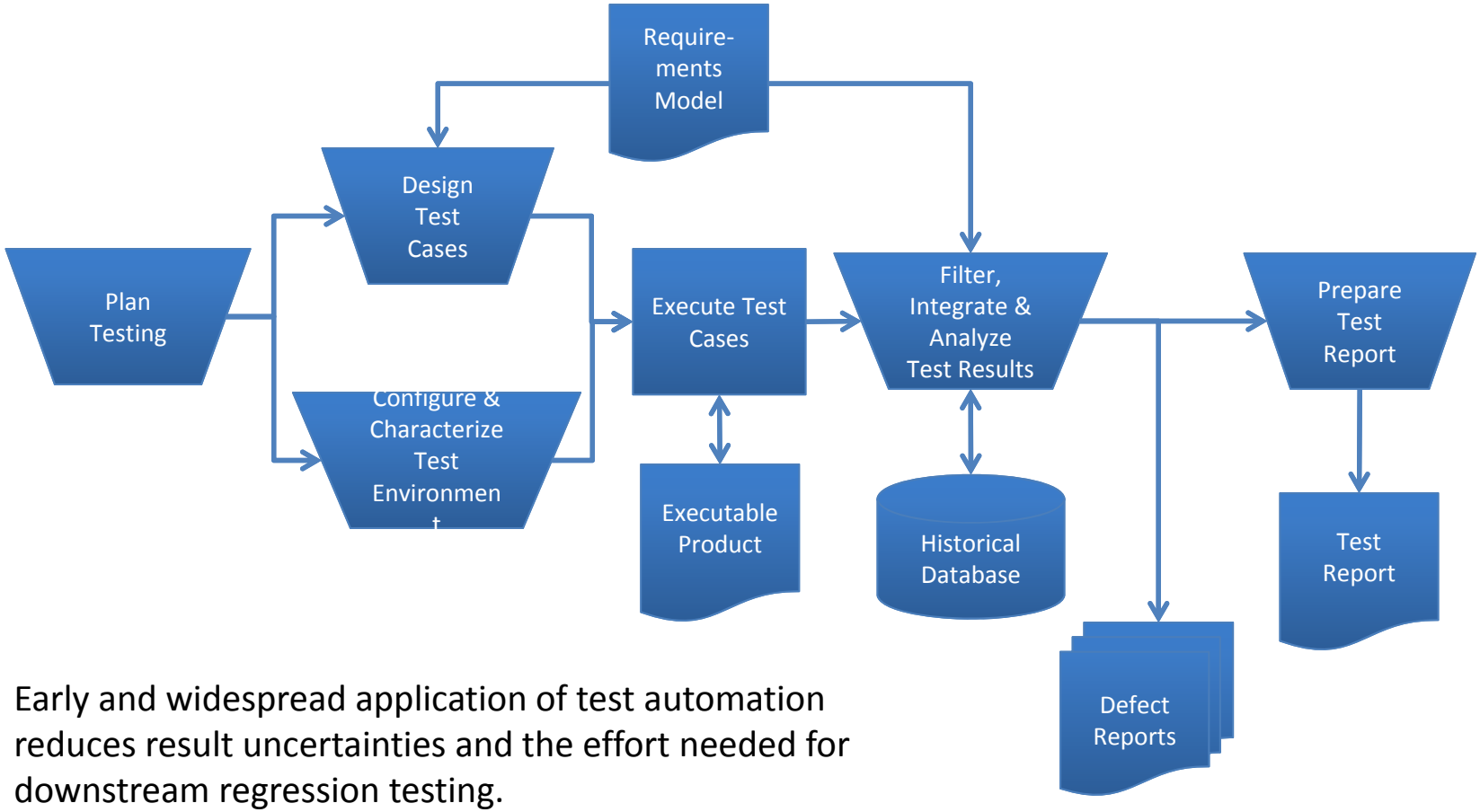
The source code evaluation process integrates rigorous manual examination with automated source code analysis to reduce the uncertainties of the combined results.



Source Code Evaluation – Controlling and Estimating Evaluation Uncertainties

- Evaluation begins by sampling the source code submitted for consideration; different sampling techniques produce
  - Different views of the source code and
  - Results with different uncertainties.
- Multiple evaluators can decrease the uncertainties of evaluation results and increase the accuracy of uncertainty estimates.
- One or more source code analyzers complement manual evaluation results but their output must currently be manually filtered to
  - Reduce the noise level of the output and
  - Decrease the uncertainties in the evaluation results.
- Storing source code evaluation results in a historical database enables building statistical baselines for comparison from that data.
- We currently have collected statistics and and constructed preliminary baselines for the Java and C++ source languages.

Software Executable Evaluation – Also Known as Software Testing



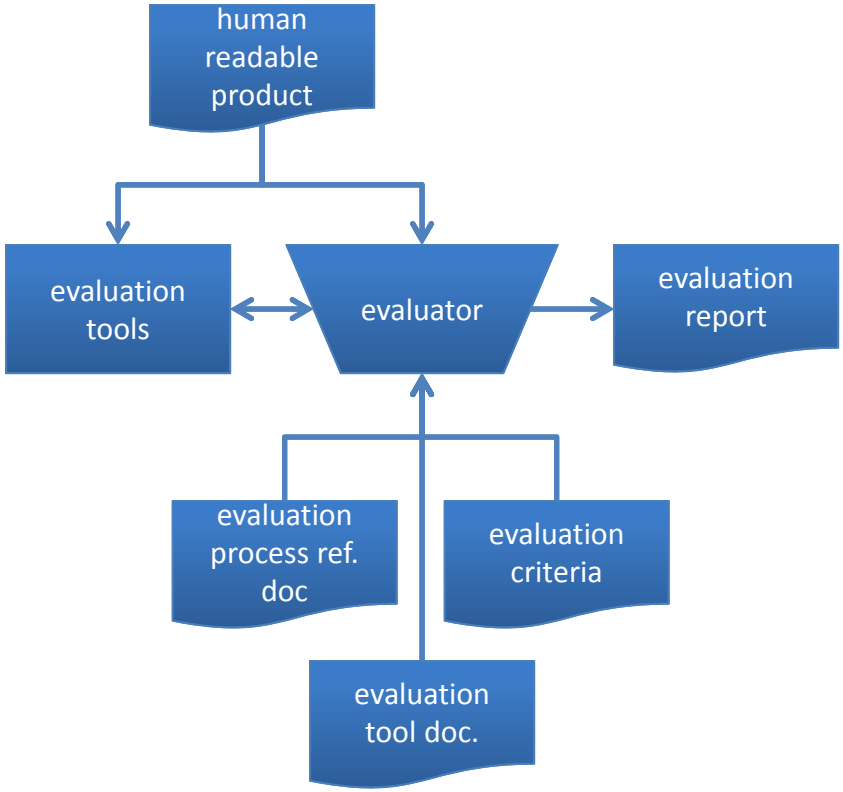
Early and widespread application of test automation reduces result uncertainties and the effort needed for downstream regression testing.

Software Executable Evaluation – Implement Testing Processes that Incorporate Modern Testing Technology

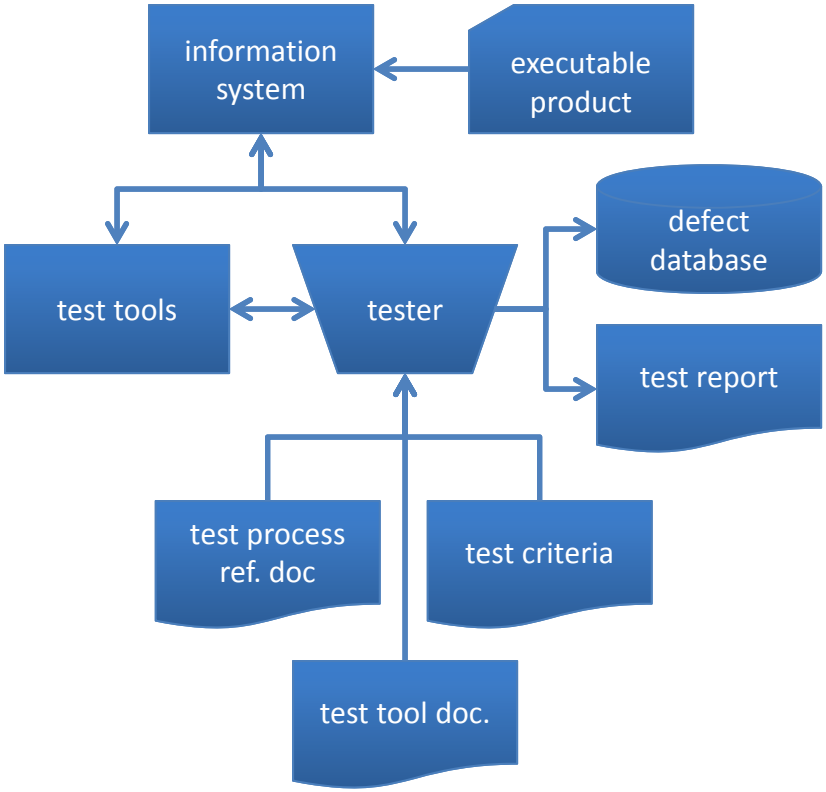
- The requirements model defines the test criteria (a.k.a., evaluation criteria) for software testing. Industry standards can define additional specialized criteria (e.g., security from the NIST SPs).
- Modern software testing guidance (e.g., automation, design of experiments, test environment characterization) defines most of the techniques applied in the software testing process:
  - Design test cases through design of experiments techniques [1, 2] to optimize coverage/$ (and not for defect hunting);
  - Build automated test scripts and data pools from these test case designs;
  - Apply test automation to execute the test scripts; and
  - Estimate test coverage of requirements graph and use those estimates to estimate results uncertainty.

This approach to software testing strives to aggressively minimize manual testing to improve testing coverage, repeatability and reproducibility.

Technical Approach – Two Basic Variants to Software Product Evaluation



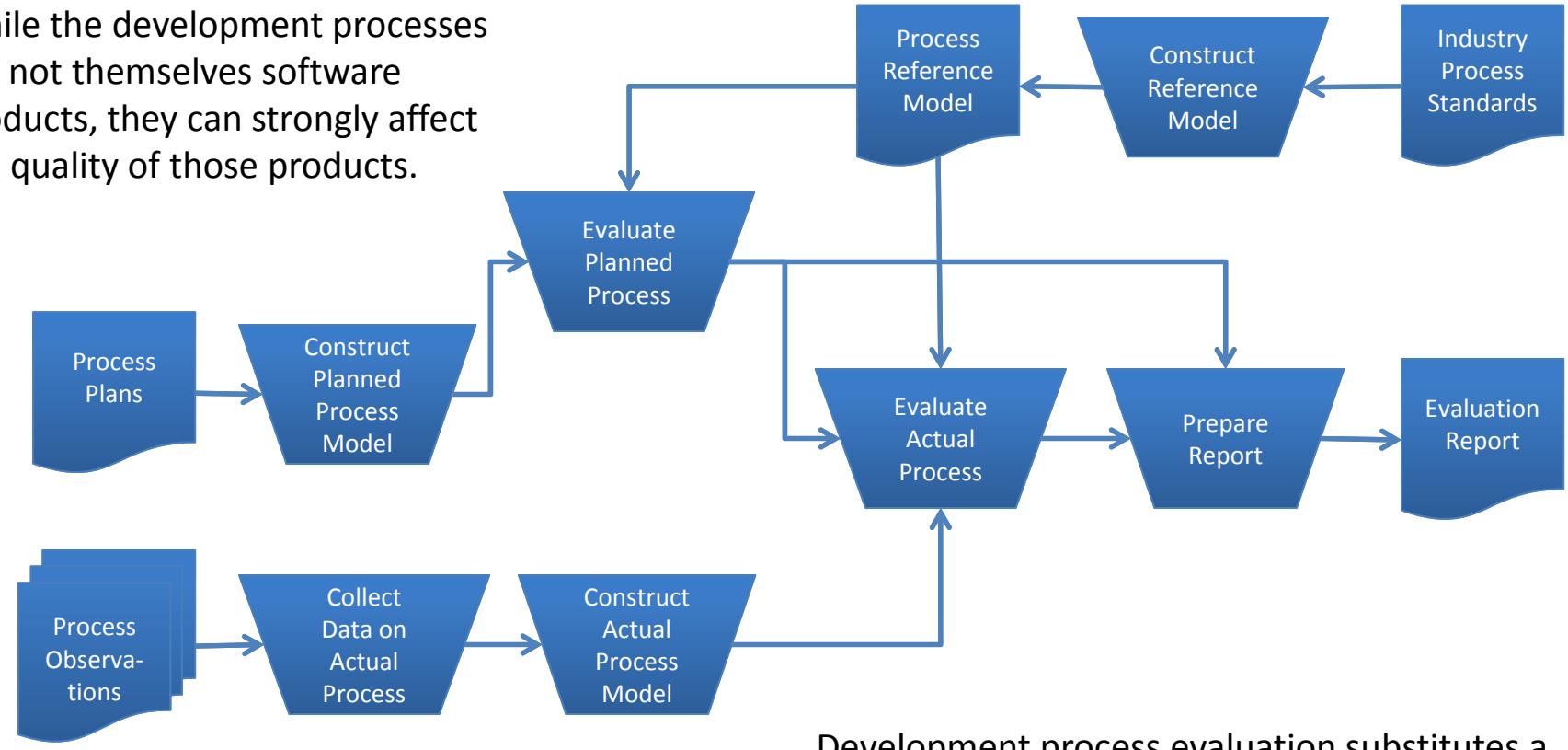
Human-Readable Products



Machine-Executable Products

Process Analysis – Determining the Quality of How Software Products Are Produced

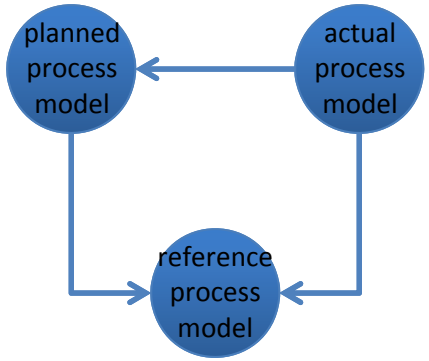
While the development processes are not themselves software products, they can strongly affect the quality of those products.



Development process evaluation substitutes a process reference model for the evaluation criteria used in the product evaluation processes.

Process Evaluation – Compare Planned and Actual Processes against Best Practices

- Collect process observations from
  - Documents (e.g., plans, document products)
  - Surveys of the personnel executing the process
  - Direct observations of process execution by independent evaluators
- Construct models of planned and actual processes from the collected data
- Construct a process reference model from a survey of industry best practices and standards for the particular process
- Compare planned and actual process models against each other and against the process reference model; each disparity produces an anomaly observation

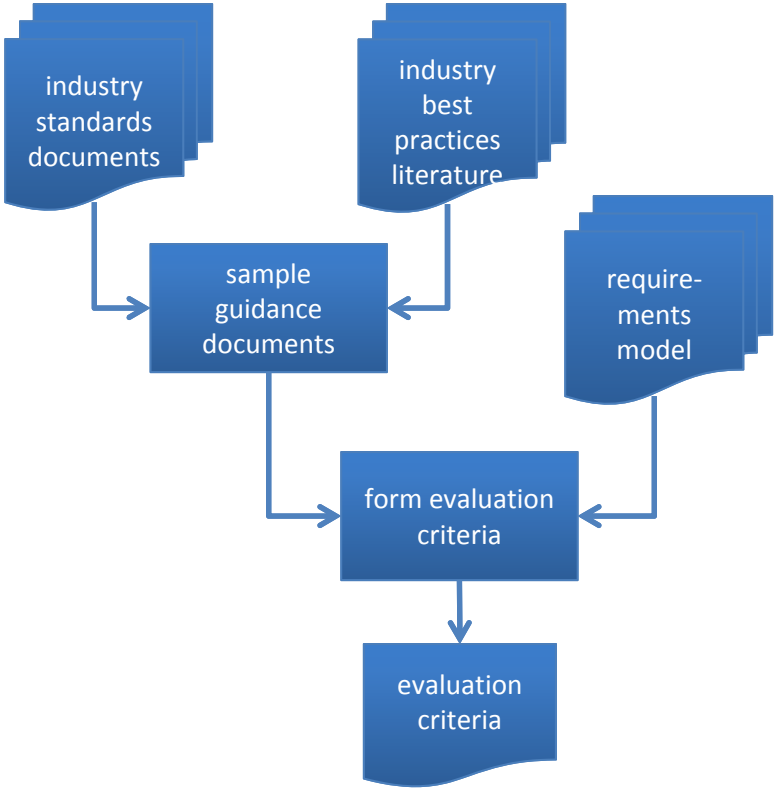


This process can evaluate any process in the system or software development lifecycle described in IEEE Std. 12207-1997.

Evaluation Criteria – Keys to Reducing Evaluation Subjectivity

The value of software quality evaluation products depends directly upon the credibility of the evaluation criteria to the users of those products. Thus, all evaluation criteria must be derived from authoritative sources to minimize the subjectivity of the evaluation process and the uncertainty introduced into the evaluation product by that subjectivity.

Software Product Type	Evaluation Criteria Sources
Process documentation	**Internal consistency, industry standards (e.g., IEEE, ISO, NIST, ITIL, PMI, organizational), industry best practices described in technical literature**
Design & release documentation	**Internal consistency, functional requirements (e.g., BRD, RSD, use cases, user stories), industry standards (e.g., IEEE, ISO, NIST)**
Source code	**Industry & organization coding standards & guidance (e.g., secure & reliable computing communities) [3-5]**
Executable code	**Functional & technical requirements (e.g., BRD, RSD, use cases, user stories, requirements databases, applicable legislation), specialty standards (e.g. NIST SPs)**



Software Anomaly Observations – Raw Data from the Quality Evaluation

- Each anomaly observation identifies the type of anomaly observed (noting the specific evaluation criterion contradicted) and the anomaly location.
- The total number of anomalies observed in a software product or process is then normalized to produce a more meaningful measure of software quality. The best normalization seems to be a measure of the product size which produces one estimate of anomaly density.
- Different product types have different measures of product size and, thus, different normalization factors.

Software Development Product	Definition of Anomaly Location	Measure of Product Size
documentation	document title, document version, document release date, starting page number, section name, section number, paragraph number, statement number	number of statements
source code	application name, application version number, module file name, module version number, source file line number	number source lines of code
executable software	application name, application version number, design element or user story number, test case designator, test script designator, test script statement designator	number of bytes of executable

anomaly observation – an instance where the information in a software product or process contradicts one of the evaluation criteria applied in the evaluation

Use Risk Modeling – Turning Anomaly Observations into a Use Risk Assessment

- Use risk modeling takes the accumulated anomaly observations as input and produces quantitative assessments of the risks of using and maintaining a software product as output.
- Software use risks are assessed in six quality dimensions: functionality, vulnerability, reliability, usability/accessibility, efficiency and maintainability [6-8].
- The model maps each anomaly type (determined by the associated evaluation criterion) to the quality dimensions based upon the wisdom from industry best practices and standards.
- Risk is assumed to be proportional to the density of the anomalies associated with the quality dimension where

  anomaly density = Σ anomalies detected / size of the product reviewed

- This risk model also estimates evaluation uncertainties from variability of reviewers (from historical data) and scope of evaluation criteria.

Evaluation Reporting - Meet the Standards for Reporting of Scientific Experiments

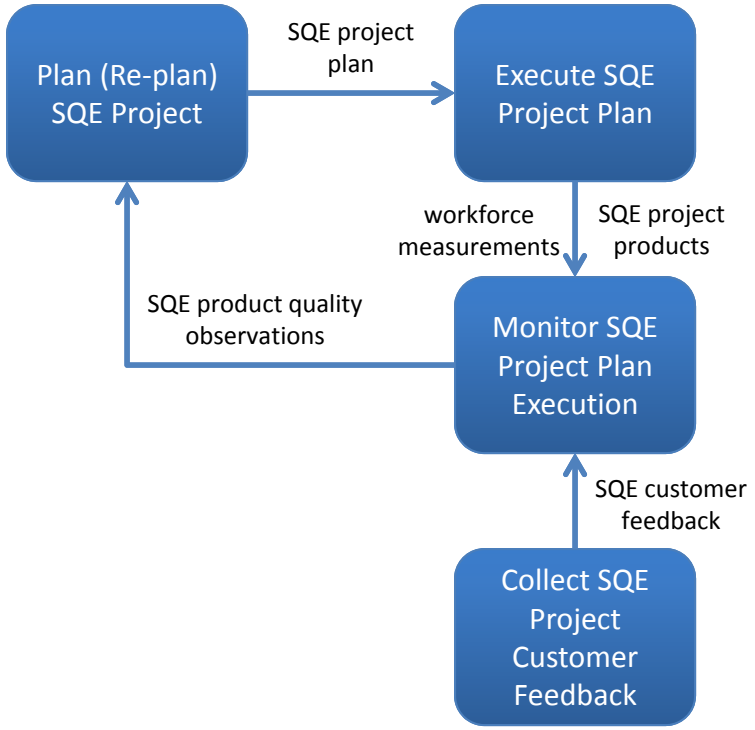
- Our evaluation reports meet a scientific standard of reporting:
  - An independent observer can reproduce the reported results (within the reported limits of uncertainty) from the report contents and the supplementary information cited in the report's references.
  - We have adopted a report structure commonly found in scientific experiment reports.
- Our reporting format builds upon and extends the reporting guidance from the IEEE software test documentation standard (IEEE Std. 829-2008).

- Purpose and scope of the evaluation
  - Evaluation purpose
  - Evaluation scope
  - Assumptions associated with the evaluation process
- Data collection process and summary of collected data
  - Product sampling process
  - Manual evaluation process
  - Automated evaluation process
- Data analysis process and results from analyzing the collected data
  - Manual data analysis
  - Automated data filtering process
  - Process for combining manual and automated data
  - Process for estimating use risks
- Uncertainties in the evaluation process and the impact of those uncertainties
- Conclusions, problems encountered and lessons learned
- References

SQE Process Quality Control – Optimize SQE Product Quality from Quantitative Process Observations

The availability of meaningful quality measurements improves monitoring and controlling the quality of the SQE products:

- Workforce measurements that include measurements of the evaluator effort required to
  - Evaluate the software product and produce anomaly observations
  - Analyze the collected anomaly observations to estimate product use risks
  - Prepare the evaluation reports
- Measurements of product characteristics
  - Size (e.g., SLOC, function points)
  - Complexity (e.g., cyclomatic complexity, edges in dependency graph)
  - Coupling (e.g., input bandwidths, output bandwidths)
- Anomaly observations & analysis results
- Measurements of process repeatability & reproducibility from baseline information



A closed-loop SQE project management approach leverages the collected empirical data to improve responsiveness to dynamic user needs and product availability.

SQE Process Tailoring – Suit a Customer's Unique Quality Needs and Cost Constraints

- Development project individualities create diverse SQE requirements that only process tailoring can successfully address.
- We can systematically tailor our SQE processes by changing
  - Number of evaluators assigned,
  - Scope of evaluation criteria, and
  - Evaluation coverage (scope & sampling density).
- These tailoring actions change the uncertainties in and costs of the SQE products.
- Uncertainty in the SQE products is the primary measure of their quality.
- These relationships enable us to tailor an SQE approach to best suit the cost constraints and quality needs of the users of the SQE products.

The risk that a decision maker assumes, when using the information from an SQE product in their decision making process, is directly proportional to the uncertainty of that information.

Recent Experience in Applying this SQE Process

- Documentation evaluations
  - Evaluated plans, requirements descriptions (use cases & user stories), design documents, and release documentation for VA CH33, VBMS, CWINRS, Virtual VA, VETSNET, and VONAPP II projects
  - Evaluated requirements descriptions and design documents for the VA eBenefits Portal project
- Source code reviews
  - Evaluated C++ source code for the VA VETSNET project (> 5000 SLOC)
  - Evaluated Java source code for the VA CH33, VETSNET and BGS projects (> 20000 SLOC)
- Software Testing
  - Implemented test case automation on the VA CH33, VETSNET and eBenefits projects
  - Implemented DoE-driven test case design on the VA eBenefits project
- Process analyses
  - Evaluated configuration management processes for the VA VBMS, VETSNET and CH33 projects
  - Evaluated the help desk processes for the VA CH33 project

Example: Source Code Baselines

Baseline	Number of Reviews	Sample Size (in SLOC)	Total Anomaly Density (anomalies / SLOC)	Vulner-ability Density (anomalies / SLOC)	Reliability Density (anomalies / SLOC)	Efficiency Density (anomalies / SLOC)	Maintain-ability Density anomalies / SLOC)
Java	17	20095	0.338 ± 0.114	0.097 ± 0.065	0.148 ± 0.083	0.028 ± 0.014	0.214 ± 0.077
C++	4	6774	0.287 ± 0.041	0.098 ± 0.037	0.206 ± 0.016	0.020 ± 0.009	0.178 ± 0.065
PL/SQL	3	3095	0.467 ± 0.146	0.112 ± 0.036	0.112 ± 0.019	0.016 ± 0.012	0.402 ± 0.131

This table summarizes data collected from 24 reviews of source code in three programming languages. These baselines measure the aggregate quality of current programming practices and define the references against which to assess the quality of individual source code samples undergoing review.

Conclusions & Summary

- Our approach to SQE explicitly manages the uncertainties in the evaluation processes and produces evaluation products that are credible to the management and development teams.
- This approach requires more discipline than is usually applied in SQA endeavors but produces considerably more credible, defensible and valuable results.
- Every software product development effort is different; we can tailor our SQE processes to efficiently meet a wide range of customer evaluation needs and cost constraints.
- Every software project is dynamic and involves uncertainty; we monitor the quality and costs of our products and control our evaluation process we apply to maintain the desired quality at the required costs in dynamic development situations.
- IMC has successfully applied components of this approach to seven large scale software development programs that use both traditional and Agile development disciplines. This experience has matured our approach and reduces the risk of our employing it.

References

1. T.J. Santner, B.J. Williams & W.I. Notz, The Design and Analysis of Computer Experiments, Springer-Verlag, New York, NY, 2003.
2. K-T. Fang, R. Li & A. Sudjianto, Design and Modeling for Computer Experiments, Chapman & Hall/CRC, Boca Raton, FL, 2006.
3. R. Shiralkar & B. Grove, Guidelines for Secure Coding, Atsec Information Security Corp., Austin, TX, January 2009.
4. Secure Coding Guidelines for the Java Programming Language, Version 4.0, Oracle, at < http://www.oracle.com/technetwork/java/seccodeguide-139067.html>
5. S.M. Christey, J.E. Kenderdine, J.M. Mazella & B. Miles, eds., Common Weakness Enumeration, Version 2.0, Security and Information Operations Division, The MITRE Corp., Bedford, MA, 2011.
6. Software engineering Technical Committee, Product quality – part1-4, ISO/IEC 9126 1-4:2003-2004, International Standards Organization, Geneva, Switzerland, 2004.
7. Systems and software engineering – Systems and software Quality Requirements and Evaluation – System and software quality models, ISO/IEC 25010-2011, International Standards Organization, Geneva, Switzerland, 2011.
8. CISQ Workgroups for Reliability, Performance Efficiency, Security and Maintainability, CISQ Specifications for Automated Quality Characteristic Measures, CISQ-TR-2012-01, Consortium for IT Software Quality, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2012.

Acronyms

ADD	architectural design description
BGS	Benefits Gateway System
BRD	business requirements description
CH33	Chapter 33, Automate GI Bill
CMMI-SVC	Capability Maturity Model Integrated – Services
DoE	design of experiments
IDD	interface design description
ISO	International Organization for Standardization
ITIL	Information Technology Infrastructure Library
IEEE	Institute of Electrical and Electronic Engineers
NIST	National Institute of Standards and Technology
PMI	Project Management Institute
RSD	requirements specification document
SDD	system design description
SLOC	source lines of code
SP	special publication
SQA	software quality assurance
SQE	software quality evaluation
VA	Department of Veterans Affairs
VBMS	Veterans Benefit Management System
VETSNET	Veteran Service Network
VONAPP	Veterans On-Line Application