eQualite:
Quality Assessment of Software Suppliers

Tim Dietz
Nadeem Malik, Ph.D.
IBM Software Procurment Engineering

January 30, 2003

Overview

eQualite methodology is based on software engineering best practices and standards to assess and improve software deliverables from suppliers.

Determines viability of a software supplier to engineer quality software on schedule and support it over it's life-cycle

- **Identifies schedule and quality risks associated with a supplier in terms of being able to reliably take requirements and convert them into a product in a repeatable, efficient and consistent manner**
- **Provides a brief assessment of the SEI Capability Maturity Model (CMM) level, which is used as the basis for profiling software development capabilities such as productivity rates and ratios of system engineering, development, test, service/support and project management needed for a required reliability**
- **Models engineering/management effort and development processes practiced for a given product development to determine the impact on schedule and quality**
- **Provides a predictive measure of the software product quality in terms of expected defects to the field for a given criticality and complexity**
- **Assesses long-term robustness of the enterprise**
- **Recommends actions for reducing cost//warranty exposure and risk abatement**
- **Identifies weaknesses and shortcomings towards instituting improvements**

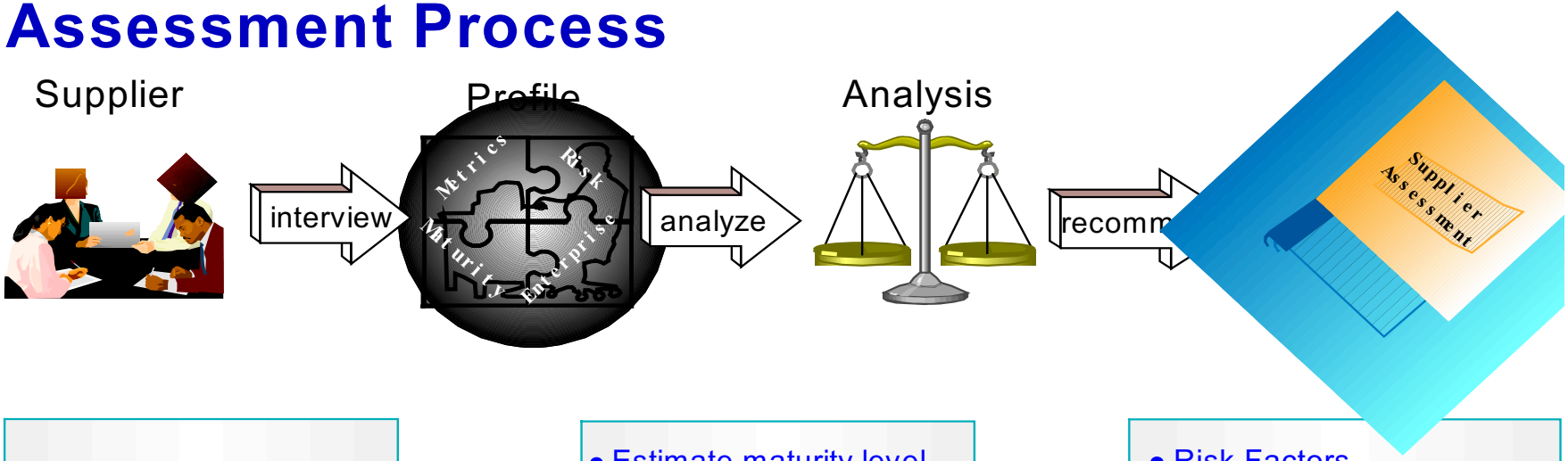
Methodology

A compendium of models and methods:

- Maturity Model
  - **Key KPA's are assessed to determine an approximate equivalence to a SEI SW-CMM level**
- Risk Model
  - **Linear model that determines the software life-cycle development capability and operational readiness by assessing the best practices that are implemented and how well the organization performs against it**
- Cost and Quality models
  - **Product development effort and schedule models for system design, programming, test, service/support and project management. Quality models provide product defect rate projections and permit reconciliation of defect data from early discovery through system test, if available.**
- Enterprise model
  - **Linear model that determines the robustness and long-term viability of the enterprise**
- Support Model
  - **provides support requirements (L3-L1) based on product and customer data**

Key Outcome: Assessment of supplier's ability to continue to operate and produce timely, quality software and support for it

Assessment Process



- Maturity questions
  - Process maturity
- Risk questions
  - preparation plans
  - implementation efforts
- Data gathering
  - Quality models
  - Cost model
  - Support model
- Enterprise questions
  - Customers
  - products
  - Skills, facilities, resources, processes
- Non-verbals

- Estimate maturity level (KPAs for SEI SW-CMM)
- Analyze data
  - Productivity
  - Development Effort model
    - Systems engineering
    - Test
    - Project Mgmt
  - Product defect rate
  - support requirements
- Compare actual with historical data
- Determine risk score
- Assess enterprise robustness

- Risk Factors
  - development effort
  - quality
  - schedule
  - enterprise viability
  - effort and schedule impact
  - warranty exposure
  - support resources
- Recommendations
  - corrective actions
  - risk mitigation
  - improvements
  - strengths

Capability Maturity Model

Level	Focus	Key Process Areas
5. Optimizing (1%)	Continuous process improvement	Defect prevention, Technology change management, Process change management
4. Managed (1.5%)	Product and process quality	Quantitative process management, Software quality management
3. Defined (8%)	Engineering process	Organization process focus, Organization process definition, Training program, Integrated software management, Software product engineering, Inter group coordination, Peer review
2. Repeatable (15%)	Project management	Requirements management, Software project planning, Software project tracking, Software subcontract management, Software quality assurance, Software configuration Mgmt.
1. Initial (75%)	Ad hoc	Ad hoc

Software Engineering Capability in terms of Best Practices

The Software Engineering Capability can be determined by a linear model that ranks a development team based on their planned use of the best practices and how they perform against that plan. The operational readiness assessed by this model can be used as the measure of development capability to determine the early defect removal potential of a team.

Such a linear model has been successfully used for the last three years in assessing the capability of IBM internal and external software organizations. The model used for these assessments was originally developed based on data collected (by Nathan Davis, Kyle Rone and Kitty Olson) from the mid seventies to the mid nineties by IBM federal Systems Group for more than 250 projects. These projects include safety critical projects for the space shuttle to mission critical projects for the Olympics to the commercial projects such as the Ford Motor Company, Postal Service, etc. We have now collected data from over 80 projects that will be used to further validate the model with recent trends.

Best practices are examined for project management, systems engineering, software engineering, test and use of tools (engineering, support and management) and standards. These best practices are assessed against the resulting product sizing, cost planning, change management, project scheduling, resource planning, quality and performance plans, defect estimation and risk planning for a given product. Such a measure of capability as a result also implicitly accounts for defect insertion/removal that may arise from possibly incorrect fixes for other defects.

The Quality Equation

Product Defects =

Total Inserted Defects - (Early Discovery Defects + Integration and System Test Defects)

The *Total Inserted Defects* can be determined from past history of a given project team in terms of its proficiency in engineering a product through its entire life cycle. In other words, it can be related to the level of maturity of a project team having a well defined and well managed organization in terms of being able to track projects and apply prior experience effectively towards repeatable success and continuos improvement.

***Early Discovery Defects* are the defects that are found through design inspections, code reviews and unit testing. Therefore, this can be related to the capability of a development team in terms of being able to adequately review and verify requirements, design and code for correctness/conformance to specs.**

Finally, *Integration and System Test Defects* are the defects that are exposed during the formal test phase (also referred to as the *Independent Test* phase) of the project. The target or acceptable defect rate and complexity for a given product determines the effort needed in this phase.

Key Observations

The Rayleigh Defect Curve implicitly assumes a high maturity and capability level of a development organization. The Rayleigh equation defining the curve was validated using defect data from teams mostly developing mission and life critical projects. As such, adjustment should be made for the actual maturity and capability of an organization to use the Rayleigh curve effectively

The number of defects found during integration and system test accounts for 17% of the total area (total inserted defects) under the Rayleigh curve. However, the shape of the curve during integration and system test can be the same as the one for the Rayleigh Defect Curve even if an inadequate test plan is followed.

Achieving 99.9% reliability by extending the test cycle is an exceptional (high development capability) outcome. Increase in schedule beyond 50% provides diminishing returns.

Insertion Defect Rates and Maturity/Proficiency

Best estimate of insertion error rate is the demonstrated proficiency of a project team on a prior project. However, it can also be determined from reported historical averages for a given maturity of a team.

B. Boehm, et al. (COCOMO), C. Jones and Davis, Rone and Olson (DRO), have independently analyzed empirical data with the following estimates of Defect Insertion Rates. All three sources have defined it in terms of team's maturity with the latter two using CMM level as the actual maturity index.

Table 1: Defect Insertion Rate per KSLOC unless indicated otherwise. Multiple values delimited by commas are for first, second and n+2 release, respectively. The numbers in parenthesis are ranges of defects per KSLOC for C language. C. Jones also provides insertion defect rates broken down by defect origins, if a more detailed estimation is needed.

Maturity	COCOMO	C. Jones	DRO
CMM 1	60 (Nominal Prof.)	5/FP (30-83)	90, 75, 60
CMM 2	N/A	4/FP (24-66)	60, 50, 40
CMM 3	N/A	3/FP (18-50)	30, 25, 20
CMM 4	N/A	2/FP (12-33)	15, 15, 15
CMM 5	N/A	1/FP (6-17)	15, 15, 15

Early Defect Removal and Development Proficiency

The removal of defects in the earlier stages of development (prior to formal test) depends directly on the software engineering capabilities of the development team in terms of experience level to conduct effective design/code reviews, use of standard practices, configuration management, etc. This can be determined directly by tracking such defects from the start of development and matching against the Rayleigh curve.

However, since all early discovery defects are generally not tracked, historical averages for different levels of development capability have been reported by B. Boehm, et al. (COCOMO) and Davis, Rone and Olson (DRO) can be used instead. Conversely, if the early discovery defects are tracked, the observed rate vs. expected can be used to drive team's capability.

Table 2: Early discovery defects found as a percent of total inserted defects for 5 levels of development proficiency. These are the defects that are found during design reviews, code inspections, unit test, etc., before the code is committed to formal test. Multiple values delimited by commas are for first, second and n+2 release, respectively. The COCOMO numbers are based on a Delphi process.

Software Engineering Capability	COCOMO	DRO
Low/Minimum	53	50, 55, 60
Nominal/Average	76	60, 65, 70
High/Good	88	70, 75, 80
Very High/State-of-the-art	94	80, 85, 90
Extra High	97	N/A

The Rayleigh Defect curve predicts that Early Defect removal rate should be 1-(0.17+(1-0.95)) = 78%, but it implies a certain capability. Based on Table 2, the Rayleigh curve appears to model High to V. High capability.

Development Factors and Product Quality

Historical development factors can be used as a guideline to distribute the effort over the development of a project. There appears to be a correlation between development factors and quality, but the averaged numbers reported below (other than the IBM group) mask the correlation.

Table 6: The test factors are relative to the programming effort. System engineering is relative to the total of test and programming effort and project management is relative to the total of programming, test and system engineering effort. The number of defects in the last row in the table is based on using average C programming language FP to KSLOC ratio. The parenthetic test factor in the first row is for a CMM level 1 organization while all others for the first row are for CMM level 2-5 organizations. MetaGroup and C. Jones factors are for an average maturity organizations, which are at the "bottom half" of CMM level 1.

	Development Factors				Defects/KSLOC
	Test	System Eng.	Project Mgmt.	Total	
IBM Federal Systems Group (DRO)	1.1 (1.2)	1.2	1.3	1.72	1.0
MetaGroup (av. of 1100 worldwide projects)	1.39	1.32	1.23	2.26	1.77
C. Jones (system and commercial apps)	1.61	1.21	1.19	2.32	3.06 (Best Average 2.3)

Programming Productivity and Test Factor/Product Quality

Programming Productivity implied in IBM FSD development factors:

C and other high level languages, low complexity code = 255-650 SLOC/PM. The high end of the range results from increasing maturity of the development environments. Av.. = 450SLOC/PM

Programming Productivity for MetaGroup factors:

Worldwide average productivity measured over 770 projects = 650SLOC/PM.

Programing Productivity for factors reported by C. Jones:

Average productivity for commercial and system software using average FP conversion factor for C language = 960 SLOC/PM

The higher programmer productivity reported by MetaGroup and C. Jones may be because of the lack of software engineering discipline employed by the observed projects towards code reviews, design, unit test, etc., which increases programmer productivity, but pushes defects into system and integration test phase, hence requiring more test resources. Higher product defect rate also supports this conjecture.

Product Defect Ranges

Given the product size and early discovery defects (either actual or estimated), the independent test defects can be tracked against the Rayleigh curve to reach the target product defect rate.

Criticality based product defect ranges that have proved to work for some benchmark applications, such as the space station, shuttle and large commercial systems, that can be used as reasonable product defect goals are shown below.

Table 3: The defect rates that were determined to be reasonable during the NASA and IBM Federal Systems programs for the three criticality levels of software systems.

Criticality	**Defect Rate** (Defects/KSLOC)
Low	1
Medium	0.5
High (mission/life critical)	0.1

Other Product Defect Averages

Table 4: The defect numbers from C. Jones were calculated using average lines per FP for C programming language and average defect rate for a SEI SW-CMM level. The minimum defect rates are given in parenthesis for each level.

CMM Level	Defects/FP	Defects/KSLOC
1	0.75	4.4-12.5 (1.17)
2	0.44	2.5-7.3 (0.94)
3	0.27	1.6-4.5 (0.59)
4	0.14	0.8-2.3 (0.18)
5	0.05	0.3-0.8 (0.02)

Table 5: The defect rates observed by Meta Group over 770 worldwide projects that delivered programming productivity of twice the average for all the projects. These companies used 4GL languages more than the average projects did.

IT Development Organizations	Defects/KSLOC (1999 US/Worldwide)
High Productivity Leaders (4GLs)	2.94/2.83
Average for all IT organizations	1.56/1.77

Development Effort and Quality

Historical Trends

- IBM FSD:
  - Four times additional test effort and twice the project management to reduce the defect rate by half
  - Eight times additional test effort and five times project management effort to reduce product defect rate to 1/10th
- Putnam
  - 25% increase in schedule to reduce product defect rate by half
  - 50% increase in schedule reduces product defect rate to one fourth

References

Nathan Davis, Kyle Rone and Kitty Olson, *A Matrix Method for Software Labor and Quality Estimations*, Proceedings of the 1992 Decision Support Conference, IBM, Thornwood, NY, September, 1992.

L. Boehm et al., *Software Cost Estimation with COCOMO II,* Prentice Hall, 2000

Caper Jones, *Software Assessments, Benchmarks, and Best Practices,* Addison-Wesley, 2000

Lawrence Putnam and Ware Myers, *Measures for Excellence: Reliable Software on Time, Within Budget,* Prentice Hall, 1992

B.G. Kohlkorst and A. J. Macina, Developing Error Free Software, IEEE AES Magazine, pp25-31, 1988

Meta Group reports on IT Performance Engineering & Measurement Strategies, 1999-2001

IBM Software Procurement Engineering Supplier Assessment Reports, 2000-2002