Quantifying Software Reliability and Readiness

Abhaya Asthana
1 Robbins Road
Westford, MA 01886
Alcatel-Lucent
Ph 978-952-7526
abhaya@alcatel-lucent.com

Jack Olivieri
202 Burlington Road
Bedford, MA 01730
Mitre Corporation
Ph 781-271-3490
jolivieri@mitre.org

ABSTRACT

As the industry moves to more mature software processes (e.g., CMMI) there is increased need to adopt more rigorous, sophisticated (i.e., quantitative) metrics. While quantitative product readiness criteria are often used for business cases and related areas, software readiness is often assessed more subjectively & qualitatively. Quite often there is no explicit linkage to original performance and reliability requirements for the software. The criteria are primarily process-oriented (versus product oriented) and/or subjective. Such an approach to deciding software readiness increases the risk of poor field performance and unhappy customers. Unfortunately, creating meaningful and useful quantitative in-process metrics for software development has been notoriously difficult.

This paper describes novel and quantitative software readiness criteria to support objective and effective decision-making at product shipment. The method organizes and streamlines existing quality and reliability data into a simple metric and visualizations that are applicable across products and releases. The methodology amalgamates two schools of thoughts in quantitative terms: product and process parameters that have been adequately represented to formalize the software readiness index. Parameters from all aspects of software development life cycle (e.g., requirements, project management & resources, development & testing, audits & assessments, stability and reliability, and technical documentation) that could impact the readiness index are considered.

Keywords-component; Software Reliability, Software Readiness

I. INTRODUCTION

Current state of the art approaches to software readiness assessment include the following: I) Checklists, II) Industry Standards, and, III) A Methodology Developed by an Academic Institute.

II. CHECK LIST APPROACH

A typical checklist approach to product readiness includes the following criteria:

- Product feature content is complete and is approved and represented in the product plan
- Customer features committed at Program Launch are in the release
- Interoperability and serviceability tests performed
- 100% test cases complete; 95% test cases passed
- Customer impacting Modification Requests (MRs) are documented for customer use (known problems list)
- No open severity 1 or 2 MRs. If any exist, provide reasoning/issue closure plan
- Overall reliability assessment acceptable for both HW and SW
- Overall risk assessed and acceptable

As can be seen, almost all metrics are subjective and qualitative.

III. INDUSTRY STANDARDS APPROACH

The software industry has been guided by two strategies when evaluating software product quality:

A. Telcordia Standards [1]

Telcordia focuses more on the Release Readiness parameters. The General Availability (GA) criteria recommended by Telcordia similarly lacks the quantitative rigor. Consider the GR 282 "Software Reliability and Quality Acceptance Criteria" Dec 1999 that states mostly qualitative requirements such as:

- R3-27 Regression test shall be performed on every complete software load generated during system test.
- R3-28 The system test plan shall be completed. Quality/performance criteria established by the supplier shall be achieved, and the results shall be documented by the supplier.

- R3-33 100% of all planned system test cases shall be executed.
- R3-34 95% or more of all test cases planned for system test shall pass.

B. ISO/IEC 9126-1:2001 Standard [4]

This standard quite explicitly identifies the product parameters which can be monitored and tracked during the softwar0e product development lifecycle. For example:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

The standard provides definitions for each of the above mentioned quality characteristic and the sub-characteristics of the software which influence the quality characteristic. However, it does not provide a means for quantifying the items or criteria which could be used towards the end of the lifecycle to validate the readiness of a release.

1) ISO 9000 – the current ISO 9000:2000 is an improved standard versus its 1994 version but the approach remains similar: i.e. to have framework of process and procedures and a very well organized quality management system in place. The assumption is that given such a framework problems would automatically be resolved.

2) TL 9000 goes a step further and tries to put some agreed upon measurements in place that are shared by the members of the QuEst forum.

However, even TL 9000 [3] measurements are end product measures primarily related to field found issues, after the software and/or hardware is out of the production gate. The measures do not validate the product readiness before it is shipped to the field or provide any in-process measures and triggers for early corrections.

IV. ACADEMIA

CMM/CMMI [2]- Carnegie-Mellon University's Software Engineering Institute's CMMI model is the recognized standard for this evolution from (immature) level 1 enterprises to (very mature) level 5 enterprises. It is a sincere approach for a step-by-step growth in the maturity of the organization.

There are multiple models to choose from. More than 20 process areas identified (e.g. 22 PA's in CMMI SE/SW Staged v1.1) and none of them directly discuss product quality parameters except one process area (Measurement and Analysis at Level 2 – primarily deals with the framework of a Measurement program), which talks directly about product quality parameters. The CMMI model provides many triggers to improve the process itself and not the end product.

Our conclusion from a survey of the existing approaches was that there is a need for consistent, quantitative, requirements driven criteria to decide software readiness.

V. NEW APPROACH

We have developed novel and quantitative software readiness criteria to support objective and effective decision-making at product shipment. The method organizes and streamlines existing quality and reliability data into simple metrics and visualizations that are applicable across products and releases.

This method amalgamates two ideas in quantitative terms: product and process parameters which have been adequately represented to formalize the software readiness index. Parameters from all aspects of the software development life cycle (e.g., requirements, project management & resources, development & testing, audits & assessments, stability and reliability, and technical documentation) that could impact the readiness index are considered.

The highlights for our methodology (Software Readiness Index) are:

- A composite index that is easy-to-understand
- Enables quantitative "pass" criteria to be set from product requirements
- Easy to calculate from existing data
- Offers a meaningful, insightful visualization
- Enables release-to-release comparisons
- Enables product-to-product comparisons
- Provides a complete solution incorporating almost all aspects of software development activities

The industry trend towards CMMI level's 4 and 5 suggests that this type of metric may be very valuable to integrate into common software development tools, such as MR databases (e.g., ClearCase/DDTS, SABLIME) and/or other project data repositories (e.g. DOORS); these tools would automatically compute a software readiness index, as well as simple trajectories/extrapolation for when readiness should go "green".

The fundamental reason for measuring software readiness is to support business decisions, including:

- GA release readiness
- Setting customer expectations
- Better control of the schedule, cost, and quality of software products

- Estimating risk of liquidated damages.

The design for the quantitative software readiness criteria was driven by the following requirements:

A. Quantitatively characterize a product's software quality and reliability at General Availability

B. Support quantitative "green (go)/yellow (go w/ condition)/red (no go)" criteria, derived from product requirements

C. Be computable from SQA (Software Quality Assurance) & second-pass system test interval through complete system test

D. Apply to most (or all) software-centric products

E. Support easy release-to-release comparison

F. Support meaningful and easy product-to-product comparison

G. Ability to validate criteria with field data, and hence close the loop

These requirements allow the generation of indicators and features useful to "business leaders" or "decision makers", project managers, program managers and system testers.

VI. Software Readiness Index

The Software Readiness Index is implemented in the form of an Excel spreadsheet that defines and measures the growth of the software system or product along the following five dimensions (vectors):

A. Software Functionality

B. Operational Quality

C. Known Remaining Defects

D. Testing Scope and Stability

E. Reliability

The fishbone diagram in Fig. 1 shows the five vectors and the variables that contribute to each vector.

- Each vector is computed as a weighted sum of the constituent variables; magnitude of each vector is normalized to 1.0
- The output is a graphical representation of the integrated result of software readiness based on the five dimensions or vectors.
- The framework relates discovery, reporting, and measurement of software problems and defects.

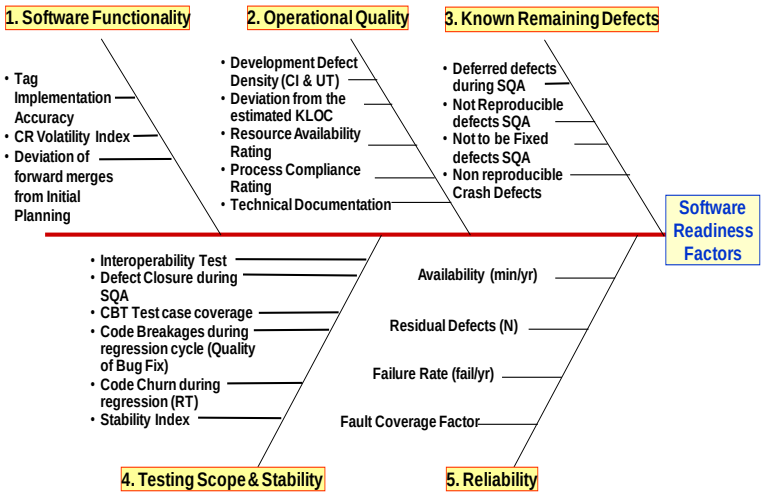

Figure 1. Fishbone Diagram for SRI

- Rules for creating unambiguous and explicit definitions or specifications of software problem and defect measurements are defined.

VII. Implementation

With a systematic approach, software data is collected based on known or anticipated development issues, concerns, questions, or needs.

The data are analyzed with respect to the characteristics of the software development process and products, and used to assess progress, quality, and performance throughout the development.

There are four key components to an effective measurement process:

A. Defining the software development issues clearly and the software measures (data elements) that support insight to the issues.

B. Processing the software data into graphs and tabular reports (indicators) that support the analysis of issues.

C. Analyzing the indicators to provide insight into the issues.

D. Using the results to implement improvements and identify new issues and questions.

Each component of this process is driven by the issues and characteristics that are inherent in the program. For example, the decision of what specific data to collect is based on the list of issues to be addressed by measurement.

1) The indicators developed from the measurement data are flexible and tailored by the development issues and related questions.

2) Analysis techniques are chosen based on what information is desired and what question(s) need to be answered.

Since we concentrate on problems and defects, we define their meaning and detail the relationships they have to other terms and entities. Analysis can be directed at:

- Assessing the feasibility of development plans,
- Identifying new issues,
- Tracking issue improvement trends,
- Projecting schedules based on performance to date,
- Defining possible development tradeoffs, and
- Evaluating the consistency and quality of development activities and products.

An example of a single vector (Software Functionality) data collection template is shown in Fig. 2.

While individual vectors serve as an indicator of software goodness (& risk) in each dimension, they do not convey a sense of the overall software goodness (and risk). To achieve this, the factors to consider are:

- Each vector is computed as a weighted sum of the constituent variables; magnitude of each vector is normalized to 1.0
- The weights for each variable is decided by the relative significance of that variable to the vector (indirectly to the product & program)

Product Quality Parameters	Acronym	Metrics	Green	Yellow	Red
Software Functionality					
Tag Implementation Accuracy	TIA	Tags Correctly Implemented/Planned	>= 98%	<98% to 90%	< 90%
CR Volatility Index	CRR	Number of confirmed major CRs accepted /Total number of CRs	<= 5%	>5% to 10%	> 10 %
Deviation of forward merges from Initial Planning	UFM	(Total forward merges - Initial planned forward merge) / Initial planned forward merge	<= 5%	>5% to 10%	> 10%

Figure 2. *Example of SW Functionality Vector Comprehensive Index*

- The vectors are similarly weighted and combined into a composite index
- The equation for the composite index should be simple, insightful and easy to understand, i.e. a linear algebraic model

Three different linear representations of the SRI were implemented:

- Additive model – each vector contributes a fraction to the overall SRI and can affect the SRI up to its allocation worth.
- Multiplicative model – each vector is independent and affects the entire range of SRI directly; SRI very sensitive to each vector. A geometric weighted average using integer weights may be used.
- Hybrid model – vectors combined into normalized additive groups that are then multiplied to form the SRI; the significant variables affect the SRI more than the variables that are deemed less important; all variables contribute to the SRI.

The final SRI value is compared against three thresholds: Green, Yellow and Red. The green, red, yellow ranges for the SRI scale with the weights for the various vectors; similarly the green, red, yellow thresholds for each individual vector also scale with the weights and the ranges for the individual variables. Thresholds are set for each variable. Thresholds for the vectors and for the final index are computed from threshold for the variables and the weights.

"Go" (green) values are based on:

- Competitive product differentiators
- Previous history; field data
- Similar products
- Industry standards; Best practices
- Customer Expectations

"No Go" (red) values are based on:

- Industry standards
- Field data; customer problem reports
- Below best practices

Threshold values may get refined periodically within a product family and across products

Even though it is desirable that the threshold values have significant commonality, exceptions will be evident from release to release, or from product to product. Exceptions have

to be agreed upon with product management, project management, development and quality group (SQA).

It is important to evaluate the test plan against the readiness index in the early part of the product development cycle. Specific sets of attributes and metrics can be tailored for product lines. A typical plot of a Software Readiness Index is shown in Fig. 3. The vertical bars for each index (vector) along with their threshold ranges are shown. The line graph shows the SRI with the actual value for each vector. The last column has the comprehensive SRI value. In this example the SRI is 0.78 that indicates a conditional "go" decision.

VIII. Application of SRI

Case Study

The tool was used with real data for several recent software product releases in the telecom industry during 2006-2008. The process highlighted areas where testing was insufficient (e.g. fault insertion) and was enhanced to correct this deficiency. (For this reason, it is recommended that the raw numbers be included in the spreadsheet to understand the composition of the metrics.) Also, the tool showed that the overall metric was marginal at release. Using the tool with real data, it was determined that the Hybrid Model was the most realistic approach. Costs associated with the collection and entering of the data were minimal, since the bulk of the data was already being recorded by SQA. Further work that needs to be done includes field data correlation and confirmation of the weighting used in the model.

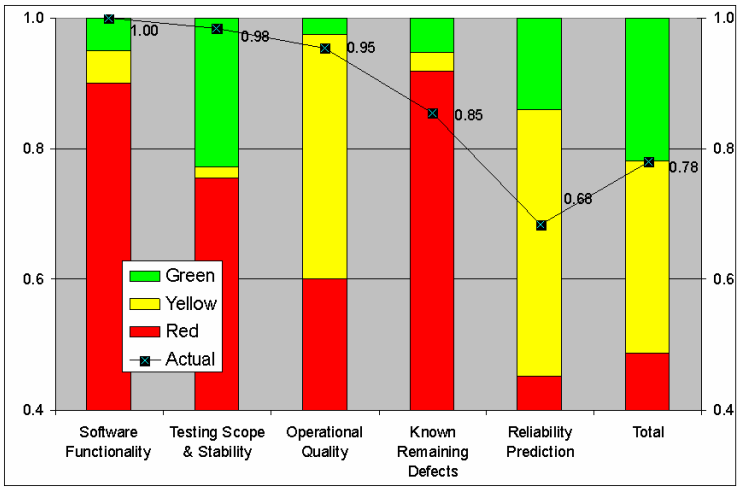

Figure 3. *SRI Graphical Output*

The SRI serves as a useful tool during the entire Software Quality Assurance (SQA) cycle as a dashboard:

- It can be monitored periodically throughout the development phase and can be used to:
- Track vectors and the overall index during the SQA phase
- Assess risks and gaps
- Perform gap closure analysis; schedule, resources, deferrals etc.

It can be used to assess General Availability (GA) – Final visual readout and readiness decision can be based on the index

- Provides the relative contribution of each vector to the composite index
- Final value of the index that is used to make GA decision

The SRI can be used for trending:

- Within a single release
- Historically across releases (same product) or a product family
- Across multiple product families

Finally, the SRI template can be customized for:

- Controlled Introduction (CI) versus General Availability
- Major and Minor releases (including New Product Introduction)
- Validation with field data as the framework gets used across releases and products
- Integration with CMMI Lev 4/Lev 5

IX. Conclusion

We presented a novel and quantitative software readiness criteria to support objective and effective decision-making at product shipment. The readiness index organizes and streamlines existing quality and reliability data into a simple metric and provides visualizations. The benefits of SRI are:

1) Systematic approach to quantifying software goodness:

- Setting measurable targets for each metric along a vector and tracking the status of each metric and the overall index in a dashboard.
- During the Development phase, the dashboard provides a summary of the state of the software, the risk areas, and the size of the gap in each area to aid the prioritization process.
- At product release the Index drives the GA decision.
- Reduces subjectivity in the decision process by quantifying the individual areas in a single composite index.

2) Organizes & streamlines existing quality and reliability data and processes into a simple tool and a methodology: Agree – Track – Verify.

- The index and tool simplify and systematize the process of probing, monitoring and managing the software development and SQA/testing. Each metric is defined, its source is identified, and the method for computing the metric is clearly spelled out.
- Complements and supplement other tools/techniques (e.g., DfS, DfT, DfM) since it is focused primarily on software prior to CI and GA.
- Creates a framework that is applicable across products and across releases; the calibration improves with each release within this framework..

REFERENCES

[1] Telcordia GR 282 "Software Reliability and Quality Acceptance Criteria" Dec 1999

[2] CMMI - CMMI® for Development, Version 1.2 CMMI-DEV, V1.2, CMU/SEI-2006-TR-008, ESC-TR-2006-008

[3] TL 9000 "TL 9000 Quality Management System

[4] Measurements Handbook", Release 4.0 -TL 9000 Quality Management System Requirements Handbook, Release 4.0

[5] ISO 9000 2001 - ISO/IEC 9126-1:2001 Standard.