Presentation
Paper
Bio
Return to Main Menu

P R E S E N T A T I O N

W4

Wednesday, March 8, 2000
11:00AM

MEASURING THE COMPLEXITY AND IMPACT OF DESIGN CHANGES

Mike Libassi
Intel Corporation

International Conference On
Software Management and Applications of Software Measurement
March 6 – 10, 2000
San Jose, CA

Measuring the Complexity and Impact of Design Changes

using the Weighted Stability Index (WSI) Metric Model

Mike Libassi - Intel Corporation

WSI Presentation Objectives

- ✓Purpose of complexity measurement
- ✓The WSI Model
- ✓Use and Process of the WSI
  - Weighing Design Changes
  - Process and Calculation
  - Results and Interpretation

Purpose and Scope

- The WSI Metrics is designed as a tool to “plug-in” projected systems changes and graph the effect to the application stability and the overall design process.
- This acts as a prediction models to help gauge testing and development efforts.
- Stability measures can be correlated to defects and test time

The WSI Model

$$S = \left[M - \left(wFa + wFc + wFd \right) \right] / M$$

$$DP = M / T$$

- S = Stability “The estimated effect to an application related to the amount of change.”
- DP = Design Process “The maturity of an application throughout its life, measuring the amount of added and deleted functionality”

Use and Process

- Both Stability (S) and Design Process (DP) are measured.
  - There are three types of functionality changes that are weighed:
    - Added functions (Fa)
    - Deleted functions (Fd)
    - Changed Functions (Fc)

Use and Process

- All function changes are weighted to the estimated impact. An "impact scale" is used.
  - The "impact scale" can be customized to meet development goals and needs.
- The weighted scale is typically from 0.1 (lest impact) to 1.0 (greatest).

Example Impact Scale

Impact	Example
0.1 → 0.3	Text box, non-critical dialog, cosmetic
0.4 → 0.5	Report/screen using existing logic and data
0.6 → 0.8	Report/screen using existing data but new calculations
0.8 → 1.0	Report/screen using new data and logic, Data model platform change

Weighing Design Changes

- Bucket application changes into the (Change, Add, of Delete category)
- Weight each change.
- Sum weighted changes
- Record current and projected functionality count:
  (M = current ,T = projected)

Calculate Stability

$$S = \left[M - \left(wFa + wFc + wFd \right) \right] / M$$

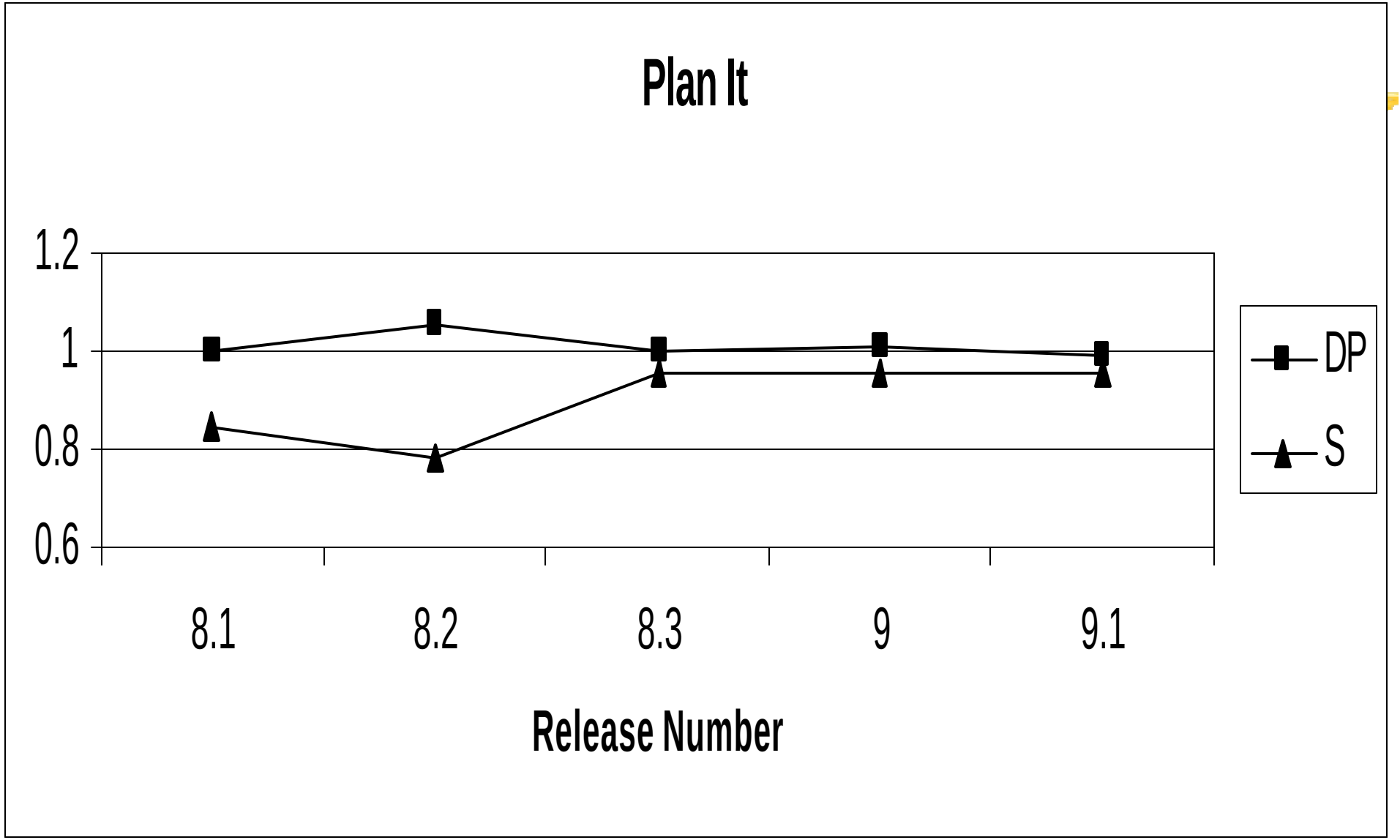
- "Plug-in" M, *w*fa, *w*Fc, and *w*Fd to get Stability (S) for the projected release

Calculate Design Process

$$DP = M / T$$

- Design Process is calculated by a simple division of current system functions (M) by projected system functions (T)

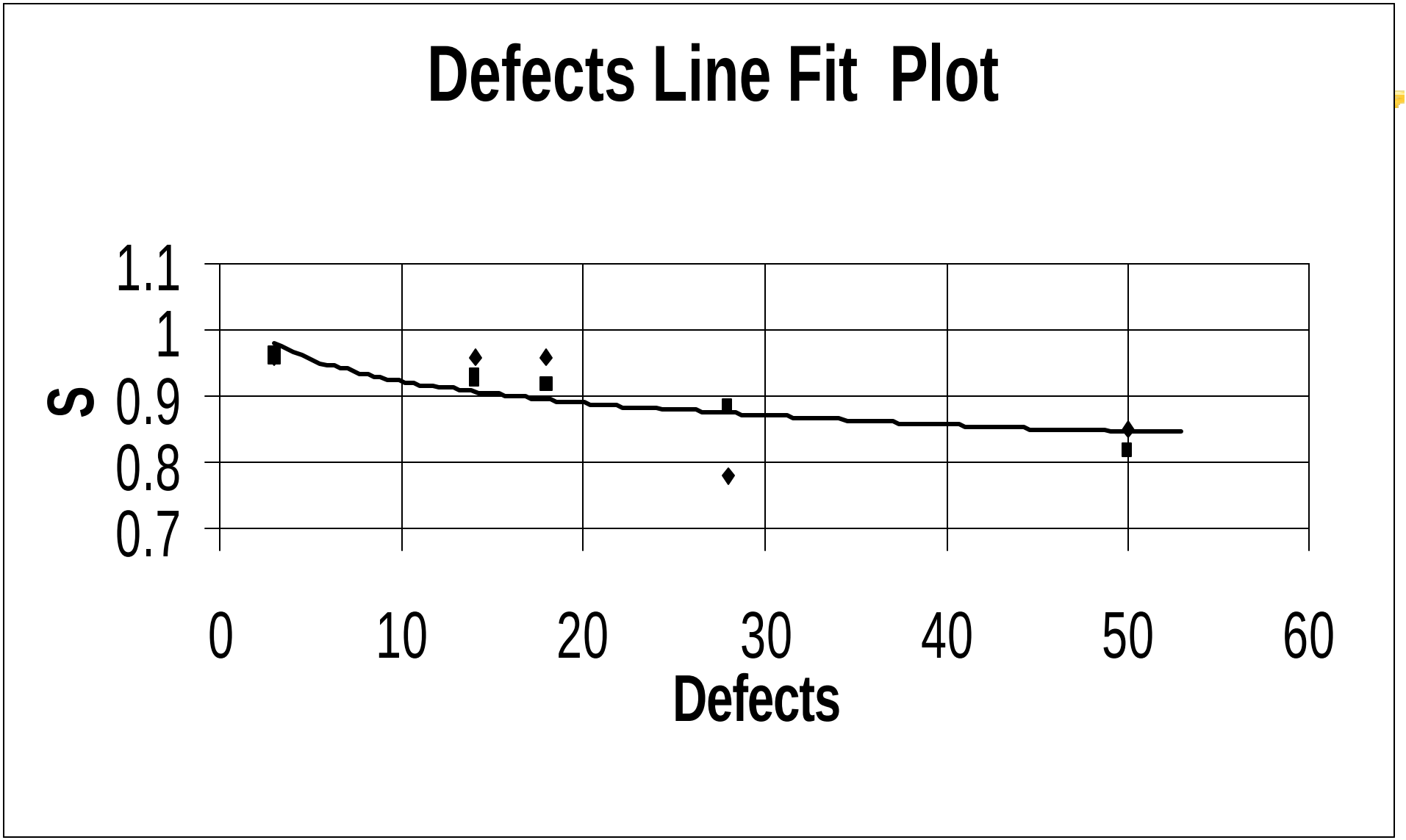
Results



Interpretation

- As DP approaches 1, the system is showing maturity.
- The closer S is to 1, the less impact to the next release
- S can be correlated to defect counts from prior releases

Defect Correlation



Process Review

- Sum of all weighted Function Adds as (*w*Fa) Changes as (*w*Fc) and Deleted as (*w*Fd)
- Total current functions as (M)
- Total projected functions as (T)
- Plug into the WSI Model & Chart Results

Summary

- Complexity is a good predictor of defects.
- WSI helps project development effort and effects of change.
- Easy to use, and customize.
- Though it is not an exact science, WSI is a good tool to help estimate.

“Manage with measurement.”

Weighted Stability Index (WSI) Metric Model

Mike Libassi

Intel Corp

8/11/99

Abstract

Methods, such as McCabe's Cyclomatic Complexity, have proven that complexity is a reliable predictor of defects. Although several methods exist to measure current system complexity, by using the Weighted Stability Index (WSI) Metric Model the potential impact of design changes can be weighted and measured. This provides a method to judge, and plan, for the potential stability impact from system changes.

The WSI method is a variation of the US Army's Design Stability Metric (Dept. of the Army Pamphlet 73-7, 31 July 1996). Both metrics are comprised of two measures on a 1.0 scale. Stability (S) measures changes made to the software design. Design Process (DP) measures the design completeness over time. This dynamic model provides a context for surveying complexity of projected design changes and the impact to system stability.

Analyzing functionality changes to the application vs. module changes is one way WSI Metric Model differs from the original Army method. However, the ability to weight each change to allow for a more precise measurement is the most significant difference. Weighing changes allows for a more realistic measurement of design change impacts. Development teams can customize the weighting process. This allows flexibility in the model, while not losing accuracy. The metric contributes to the development effort buy allowing development or quality managers to plan testing efforts for the predicted stability impact.

Granted, no metrics model is an exact view of reality. This metric is no different, but as stated in many publications and papers, "if we don't measure, how can we manage."

Weighted Stability Index (WSI) Metric Model

Mike Libassi, Intel Corp.

8/9/99

mike.j.libassi@intel.com

Introduction

Where we can measure, we can manage. Measurement of current processes and history release data help us see what we have done, right and wrong. Measuring future impact to current systems is the next step in software management.

Scope

The scope the WSI metric is to measure projected system changes and system maturity by release.

Definitions

Function / Functionality - A functional part of an application: ex. A report, login screen, data entry screen. May include one of more modules.

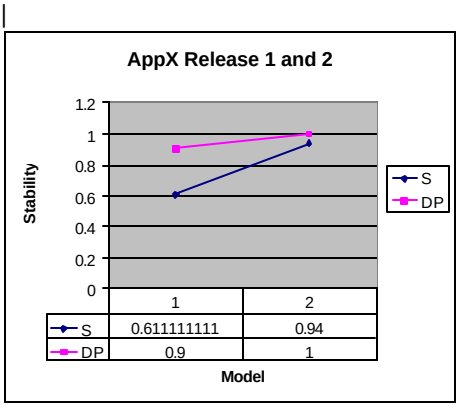
Module - A component, code module, class module, sub procedure, SQL query or stored procedure, that is used in the software.

Stability - The estimated effect to an application related to the amount of change to the system.

Design Process - The maturity of an application throughout it's life. Measured by the amount of functionality added and/or deleted from the system.

Methods

Plotting stability (S) and design progress (DP) over time, or each release, is the recommended display of this metric. Example 1 shows the original release (release 1) as introducing more instability (points S and DP are further apart and DP is < 1.0). However, release 2 shows greater stability in the design process (DP) and very minor impact



Example 1

from the changes (S and DP are almost 1.0). Although not indicated in example 1, it is possible for DP to be a negative value. This may indicate that every system function has been changed and more added. In addition, it is possible for DP to be greater that one if some functionality is only deleted form the original system.

The design progress (DP) measure portrays the complete system stability over time. The more the system is changes the less stable it becomes. The closer this value is to one, the higher stability. Stability (S) assess the actual impact of the next release. A 1.0 indicated little to no stability impact from the proposed changes. Both DP and S can be charted separately, but when used together, the metrics becomes a tool to indicate overall design progress and the stability impact for past and future releases.

Original Method (Design Stability Metric)

$$S = \left[M - \left(Fa + Fc + Fd\right)\right] / M$$

Equation 1 "Stability Metric"

$$DP = M / T$$

Equation 2 "Design Process"

S = Stability Measure

DP = Design Progress

T = Total number of projected design functions

M - Total number of current system functions

Fa = Number of functions to be added

Fc = Number of functions to be changed

Fd = Number of functions to be deleted

The Weighted Stability Index (WSI) Method

As in the original method, Design Process (DP) calculation has not changed (Equation 4) but in the Stability calculation (Equation 3) we now sum weighted function changes (*wFx*). Weighing functional changes brings more flexibility and precision to the original method. Otherwise, the calculation process has not changed.

$$S = \left[M - \left(wFa + wFc + wFd\right)\right] / M$$

Equation 3 "WSI Metric"

$$DP = M / T$$

Equation 4 "Design Process"

S = Stability Measure

DP = Design Progress

T = Total number of projected design functions

M - Total number of current system functions

Weighing Functional Changes

The new weighted factors (Equation 5) is a summation of each weighted change. The weight factor is scaled from 0.1 to 1.0 (0.1 being minor and 1.0 major). The process of weighing is discussed later in *Customization*.

Example:
Two added functions to the system:
1. Adding some new text to an existing screen (impact 0.2)
2. Adding a new calculation on existing data (impact 0.9)

wFa = 1.1 (or 0.2 + 0.9)

$wFa = \sum Fa^n$ Sum of all weighted added functions

$wFc = \sum Fc^n$ Sum of all weighted changed functions

$wFd = \sum Fd^n$ Sum of all weighted deleted functions

Equation 5 "Weighted Change Factors"

Example

Application X has undergone a major release (Release 1) and is to have a projected minor "maintenance" release next month (release 2).

Both of the above examples were weighted as, non-weighted method you just sum the changes.

Example Release 1

Large release, that had new and changed functionality

Two new functions

- New, non-calculated report (weight 0.5)
- New calculated report from several data tables (weight 0.8)

Five changes:

- Added text to an existing screen (weight 0.1)
- Changed existing dialog on one screen (weight 0.2)
- Expanded one data filed in the database (weight 0.5)
- Changed login screen (weight 0.3)
- Changed an existing filter and combo box (weight 0.2)

Example Projected Release 2

Projected small maintenance release.

Two changed functions

- Changed report sort logic (weight 0.4)
- Changed menu from "Past" to Past As…" functionality (weight 0.2)

Running the two above examples through the standard model

In the standard model the total changes and addition are summed (Fa = 2, Fc = 5 for example one)

Std Model	1	2
S	0.611111	0.94
DP	0.9	1
T	20	20
M	18	20
Fa	2	0
Fc	5	3
Fd	0	0

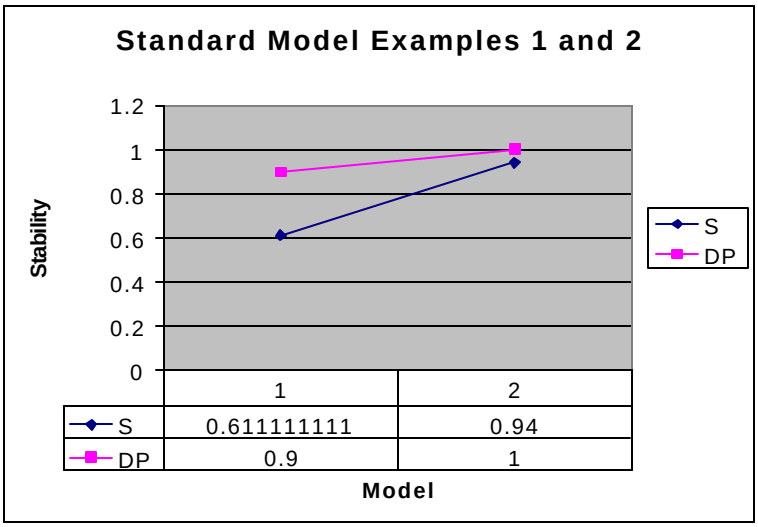


Figure 1

The standard method shows how the application is reaching a stable state. Release 1 was a larger impact. DP was slightly under the 1.0 (stable) value and S was considerably below the DP point, showing the introduction of instability into that release.

Running through the WSI Metric Model

The *weighted* items are summed (Equation 5)

Weighted	1	2
S	0.861111	0.988
DP	0.9	1
T	20	20
M	18	20
Fa	1.2	0
Fc	1.3	0.6
Fd	0	0

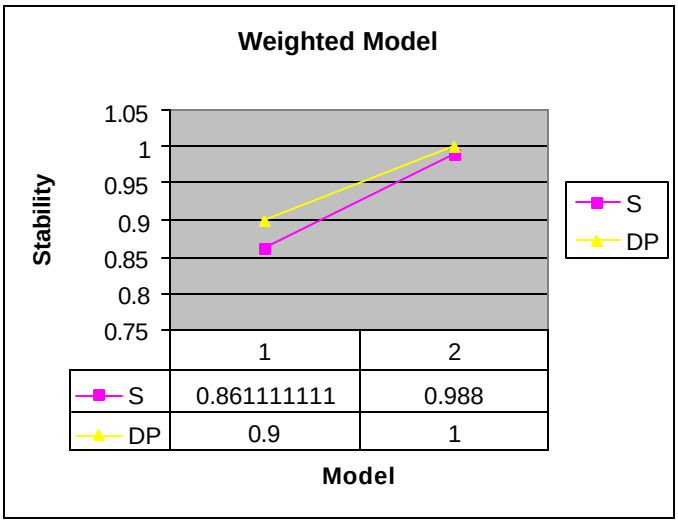


Figure 2

The WSI Metric Model also shows the application is moving to stable (1.0). However, since the changes were weighted as to their impact, Release 1 actually did not introduce the amount of instability as stated in the standard method. The ability to assign a weight to a functionality change allowed the greater accuracy.

Differences between using standard and weighted in the two examples are:

- In release 1 there was a -25% difference in S compared to the standard model
- In release 2 there was a -4.8% difference in S compared to the standard model

Precise calculations hale when correlating S to defects from past software releases, thus making the WSI model more precise than the standard model.

The Process

A Standard process to use the WSI Metric Model is:

1. Bucket application changes into three (Change, Add, of Delete category)
2. Weight each change to an agreed impact scale.
3. Sum weighted changes
4. Record current and projected functionality count (M = current , T = projected)

In Excel, or another tool:

5. Sum of all weighted Function Adds as (*w*Fa)
6. Sum of all weighted Function Changes as (*w*Fc)
7. Sum of all weighted Function Delete as (*w*Fd)
8. Total number of current system functions as (M)
9. Total number of projected functions as (T)
10. Calculate S and DP as show in equation 3 and 4
11. Chart on a 1.0 scale

Customization the "Impact Scale" and Implementation

Each system change can be measured with the WSI Metric Model. Standardization of the impact scale can be ranged between 0.1 to 1.0. The lower the weight the less of an impact.

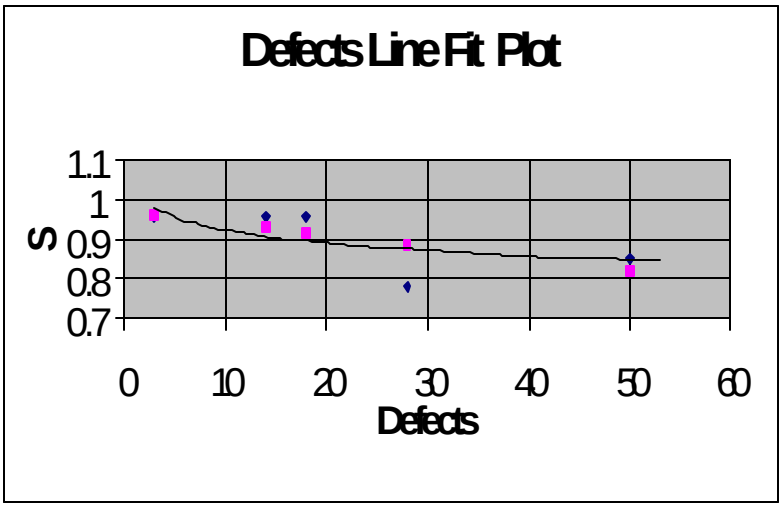
Reason to weigh a system change; not all changes add the same complexity to the existing system. Addition of new text label to a window has less impact that new server side calculations on data. The actual impact should be assessed and designed by the development team. The impact scale allows for greater consistency when each change is weighed. This weight value can be added to current development and design documentation and can be reviewed and changed upon team discretion.

The ability to create a custom impact scale allows flexibility within the WSI Metric Model.

Example Impact Scale

Impact	**Example System Changes**
0.1 → 0.3	Text box, non-critical dialog, cosmetic
0.4 → 0.5	Report/screen using existing logic and data
0.6 → 0.8	Report/screen using existing data but new calculations
0.8 → 1.0	Report/screen using new data and logic, Data model platform change

Correlating Stability Data



Stability data can be correlated to the amount of defects. Over time the WSI model can be used to predict defects. The sample graph was created from seven previous releases. The scatter plot does show some degree of coloration. The lower S is the more defects were produced. This is just a way to judge on the theory that complexity is the greatest predictor of defects. This same data could be correlated against development time or cost.

Summary

Measuring complexity in software development is a sound method to judge quality and predict defects. Thought the art/science of software measurement is not exact. It is better to manage with measurements, then not measure at all.

Reference

Software Test and Evaluation Guidelines, Department of the Army, Pamphlet 73-7 (DA PAM 73-7), 31 July 1996

Mike Libassi

After serving eight years in the U.S. Navy, Mike Libassi was employed by Intel Corp., in 1995, starting on the manufacturing floor. He also worked in metrology for a year. After gathering enough experience, he then moved into a quality engineering group.

His next move was into a software testing position in Chandler, Arizona. After working one year in quality engineering and attending two years of night school, in 1998, Mike received a bachelors degree in information systems from the University of Phoenix. After graduation, Mike worked as a software tester for two years, performing roles with automated testing and metrics.

During the past four years at Intel, quality and metrics have been a part of Mike's job function. Currently, Mike's position at Intel Corp. involves driving software metrics for the Capacity and Execution Systems Department.