ADVANCE:
Implementing a Defect Model for Performance Prediction

Stan Martin, P.E.
Principal Systems Engineer
Modeling & Simulation/Operations Analysis
Greenville, TX
stan.martin@L-3com.com
Overview of “ADVANCE”

- Stochastic model
- Includes Defect Creation, Spread, and Detection
- Includes Effects of Rework
- Based on Historic Company Performance
- Uses Company-specific Key Process Attributes
- Predicts Defects (mean, \( \mu \)) per phase & total
Outline

• Basis and Previous Work
• Implementation
• Calibration
• Deployment
• Conclusions and Future Work
• References
Outline

• Basis and Previous Work
  • Implementation
  • Calibration
  • Deployment
  • Conclusions and Future Work
  • References
**Basis: Requirements**

- OPP-SP 1.5 requires Process Performance Models (PPM) to estimate or predict the value of a process-performance measure from the values of other process, product, and service measurements.

- Chrissis, et.al., list four uses of PPMs:
  1. The organization uses them for estimating, analyzing, and predicting the process performance associated with the processes in the organization’s set of standard processes.
  2. The organization uses them to assess the (potential) return on investment for process improvement activities.
  3. Projects use them for estimating, analyzing, and predicting the process performance for their defined processes.
  4. Projects use them for selecting processes or subprocesses for use.
Basis: Summary of Approach

The technical approach is based on established methodologies for reliability prediction of software defect densities:

- Historical data is used to produce a process performance baseline
- The process performance baseline is characterized by key process attributes
  - Specific attributes are defined for each lifecycle stage
- At the beginning of the project, the model will predict process performance based on the defined attribute values
- At the end of each lifecycle phase, predicted phase attributes are replaced by actual phase attributes, actual phase results are entered, and the remaining lifecycle performance is updated
Basis: Previous Work

- Rout and Abel 1993 "A Model for Defect Insertion and Detection in Software Development"
- Chulani 1999 "Constructive Quality Modeling for Defect Density Prediction: COQUALMO"
- Others (see paper or References)
1. Errors are introduced into a software component at each stage of development, the rate of insertion being dependent on a number of factors.

2. Errors are detected and removed during all stages of development, at a rate which is primarily dependent on the detection technique employed.

3. Errors that are not detected during one stage of development may result in multiple errors in succeeding stages.
4. "When an error which results from an error in a preceding stage is detected, all related errors are not necessarily detected."

5. "When an error is corrected, there is a non-zero probability that new errors will be introduced."
Difficulty: Rout and Abel were unable to represent framework points 4 and 5 in an analytic equation.

Solution: use a discrete-event model to represent processes without an analytic equation.
Chulani utilized the COCOMO production function:

\[ E = a(S)^b \]

where

- \( S \) = size (SLOC),
- \( a \) and \( b \) are empirically derived,
- \( a \) is a product of "Quality Adjustment Factors"
Basis: SW Development Phases

1. Requirements
2. Design
3. Code & Unit Test
4. SW Int & FTV*
5. Sys Integration

* Functional Test Verification
Basis: Defect Model (Per Phase)

Defects created (= FeP)

Defects inherited from previous stage

Multiplicative Drive Factors

Fc = Creation (as function of phase Products P)

Fs = Spread (of inherited defects)

Fd = Detection (as % of phase total defects)
Basis: Three Factors per Phase

Å Defect Creation:
\[ Dc = Fc \times f(P) \]

Å Defect Inheritance:
\[ D_{adj} = D_i \times F_s \]

Å Defect Detection:
\[ D_d = F_d \times (D_c + D_i \times F_s) \]

Å 3 Factors x 5 Phases = 15 Factors
Outline

• Basis and Previous Work
• Implementation
  • Calibration
  • Deployment
• Conclusions and Future Work
• References
Implement: 3 Parts

- Extend Discrete-Event model
- User Interface (Microsoft Excel)
- Calibration Grid (Microsoft Excel)

User Interface \( \Rightarrow \) Settings \( \Rightarrow \) Calibration Grid \( (M_{jn[i]}) \) \( \Rightarrow \) Factors \( \Rightarrow \) Extend™ Model

(trademark of Imagine That, Inc.  http://www.imaginethatinc.com)
Implement: Drive Factors

\[ F_n[i] = F_{nom}[i] \prod_{j=0}^{k} M_{jn}[i] \quad n = \{C, I, D\} \]

Where:

\( M_{jn}[i] \) is the Model Drive Factor for attribute j as applied to factor n for phase i.

\( F_{nom}[i] \) is a nominal value for \( F_n[i] \)

Similar to use by Chulani in COQUALMO
Implement: Extend Model

- Commercial process modeling tool
- Graphical, Hierarchical
- Discrete Event
Implement: e.g. Defect Creation

- Items flow through process network
- Queuing, time delay, decisions
- Numeric calculations, statistical distributions
Implement: User Interface

- VBA controls for user inputs
- Receives Extend model outputs
Implement: Calibration Grid

- Converts UI settings to Model Drive Factors
- Isolates Extend model from calibration
- Does not contain actual historical data
Implement: Results (per-run)

Å In-Phase Defects
Å Out-of-Phase Defects
Å Cumulative per Phase
Å Estimated Remaining
Implement: Results (statistical)

- Mean, Variance, Std. Deviation
  - Per phase
  - Total

- Lower & Upper Confidence limits
  - Per phase
  - Total
Outline

- Basis and Previous Work
- Implementation
- Calibration
- Deployment
- Conclusions and Future Work
- References
Cal: Mapping the Problem

\[ F_n[i] = F_{\text{nom}}[i] \prod_{j=0}^{k} M_{jn}[i] \quad n = \{C, I, D\} \]

- Determine the values of \( M_{jn}[i], F_{\text{nom}}[i] \)
- From 53 User Interface inputs, 10 best were selected for initial calibration
- 10 inputs mapped into 21 drive factors
- Multiple settings per input factored into 55 values requiring calibration
Cal: Fitting the Curve

Curve fitting was used to determine the $f(P)$ ($F_{\text{nom}}[i]$) for the Defect Creation function.

Several different curve functions were evaluated.

COCOMO-type power equation was the best fit:

$$E = a(S)^b$$
Cal: Running the Calibration

Å Linear regression and iteration were used to optimize $M_{jn}[i]$
Å 55 values calibrated to 72 (later, 66) historical datasets
Å Approximately 500,000 model runs required
Å Final correlation 0.984, improvement of .109
Outline

- Basis and Previous Work
- Implementation
- Calibration
- Deployment
- Conclusions and Future Work
- References
Deployment: Pilot

A Pilot deployment to SW Engineering Organization

A Feedback positive

- minor changes to UI
- Identified and removed 11 outliers and recalibrated
Deployment: Application

Â SW Engineering runs model
Â Initial recalibration each year with new data
Â Organization use:
  ï Evaluate organization process performance
  ï Assess quantitative return-on-investment for potential process improvement impact
  ï Establish organizational objectives
Deployment: Application

Â Project use:
    ï Run to establish realistic and achievable project objectives
        Â Establishes quantitative basis for negotiations
    ï Run to estimate or predict the project’s performance of selected subprocesses
        Â Ensure project’s success by predicting future outcomes based on current performance
        Â Establish corrective actions today to alter the future course of the project
        Â Proactive process performance risk identification
    ï Run to assess progress and evaluate corrective action
    ï Compare predicted vs. actual
    ï Update calibration only if project rebaselines
Outline

• Basis and Previous Work
• Implementation
• Calibration
• Deployment

Conclusions and Future Work

References
Conclusions

- Significant Effort to Develop
  - But, completely fitted to our process
- Development Effort helped refine metric collection
- Development helped organizational buy-in
Future Work

Å SW Model:

• Calibrate additional UI attributes
• Calibrate internal distribution spreads
• Add defect type categories

Å Others:

• Family of models to include other Engineering disciplines: Aero, Systems, HW

Å Integrated Product Model

Technique is adaptable to all disciplines
References


Â Devnani-Chulani S., (1997), Modeling Software Defect Introduction, USC î Center for Software Engineering


Contact

Stan Martin, P.E.
903.408.8756
Stan.Martin@L-3com.com

Dawn Littrell
903.457.6011
Dawn.H.Littrell@L-3com.com

L-3 Communications/Integrated Systems
P.O. Box 6056 CBN-108
Greenville, TX 75403-6056