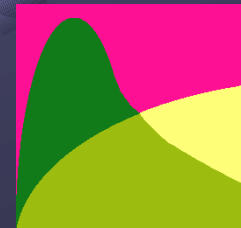
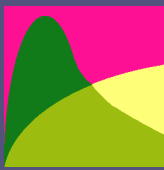


Why Should My Company Invest in Software Reliability?

Copyright,
Ann Marie Neufelder, SoftRel, LLC



Every page of this presentation is copy protected. You may not reproduce without written permission from Ann Marie Neufelder of SoftRel.



These things were measured on 28 real organizations developing real time software [1]

● **Whether defects are predicted**

Before code is written or
During testing or
Never

● **These performance measurements**

Normalized fielded defects (defect density)

Probability of late delivery

Magnitude of late deliveries as a percentage of original schedule

Existence of development practices, organization philosophy, methods, tools, process

Type of application, industry, duty cycle

Product characteristics related to requirements, design, and code

Why should my company invest in software reliability?

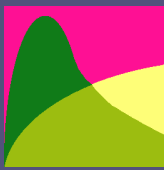


The Facts

Performance measurements when software reliability/defects are predicted as early as these project milestones	Prior to coding*	During testing**	Never
Actual normalized defect density	0.122	0.329	1.033
Probability of late delivery (%)	25	69	80
Magnitude of late delivery as a percentage of original schedule	10	100	123
Number in each group	8	10	10

*Using methods such as the SoftRel Full-scale, Shortcut, SEI CMM lookup, industry lookup, Rome Labs Tr-92-52 or other predictive models

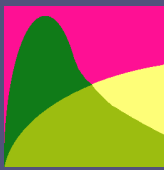
** Using estimation/extrapolation models



These are some of the common traps that cost defects AND time [1]

Trap	% increased defects when this trap exists
The software manager codes (he/she should be managing)	224
Regularly makes enhancements after testing phase has ended (when features should be frozen)	302
Regularly makes enhancements after the coding phases have ended (when features should be frozen)	276
Software engineers are Interrupted to work on unplanned defects from the last project	357
Percentage of copied and pasted code exceeds 50% (copy and paste is the opposite of object oriented)	248
Failing to protect archaic code	365

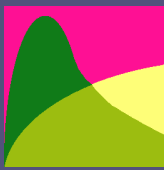
None of the software organizations that were predicting defects in advance fell into the above traps



Fact and Fiction

- Fiction: Predicting and removing defects will make the project late
 - Fact: In study of 46 real software projects
 - Correlation between fielded defect density and probability of being late = .83
 - nearly perfect linear relationship
- In plain English
- Whether a project will be late can be sufficiently determined by latent defects alone
 - The larger the number, the larger the probability of a late delivery

Why should my
company invest in
software reliability?



The most effective development practices executed by MOST/ALL of companies predicting defects [1]

Development practice	Relative cost	% decreased defects when this practice exists
Unit tests are formally reviewed	\$	502
Formal means by which to filter, assign priority and schedule customer requests	\$	371
System test plan started during the requirements phase	\$	344
Traces requirements to code	\$	270
Requirements mapped to unit tests	\$	252
Does path testing	\$\$	248
Changed paths are retested during corrective action	\$	220
Error handling is tested at the unit level	\$	316
Requirements is longest phase of lifecycle	\$\$	181

\$ - < \$500, \$\$ - < \$\$2000, \$\$\$ - < \$10,000, \$\$\$\$ >= \$10,000



Practices that have less correlation to defects

Development practice	Relative Cost	Correlation to fielded defect density
Systems testing tools	\$\$\$	-.00062
Design tools	\$\$\$	-.0197
Design is Object Oriented	\$\$\$	-0.01942

§ **Tools can be useful but not when they are used before the software engineers understand the task that's being automated**



Facts and Fiction

- Fiction: Defects can't be accurately predicted in advance of development
- Fact: If the models are used correctly (i.e. no guessing or wishful thinking) the models can be accurate as shown on next slide

Why should my company invest in software reliability?



Observed relative error on prediction model

Technique	Relative error
Educated guessing at the beginning of the project*	400%
Simple one parameter prediction using application type average	154%
Simple one parameter prediction using SEICMMi level < 2	83%
Simple one parameter prediction using SEI CMMi level >= 2	19%
SoftRel surveys	See below chart
Using similar historical recent data from your organization	Better than or = SoftRel models

Percentile group	Relative error when percentile is not predicted accurately	Relative error when percentile is accurately predicted
World Class	n/a	49%
Very Good	n/a	26%
Good	38%	12%
Average	49%	19%
Fair	27%	26%
Poor	15%	5%
Ugly	n/a	19%

*Software Engineering Economics, Barry Boehm, Prentice Hall, 1981
2004 Applied Reliability Symposium



Myth

- We don't have the resources to predict defects in advance of development



Fact

These are time estimates for doing a prediction

Task	Estimated time
Predict size of code	This task must be done to schedule/staff the project anyhow so no additional time is required for this task for predicting defects
Answer the survey	About 1 day – evenly divided by 3 people who will answer about 1/3 of the survey
Review and validate answers to survey	4 hours by a person knowledgeable of the survey
Collect general inputs and compute results	If the Frestimate software is used this can be done in 1 hour per software component
Review alternatives for reducing defects	The Frestimate software computes the practices that are most effective given the responses to the survey. If the software is used, the review time is 1 hour.



Summary

- This presentation reveals only a few of the quantitative results of the 13 year long benchmarking study done by SoftRel, LLC.
- [1] “The Naked Truth About Software Engineering”, 5th Edition, AM Neufelder, 2006. To get a copy of the complete survey results see <http://www.softrel.com/price1.htm> or email sales@softrel.com.