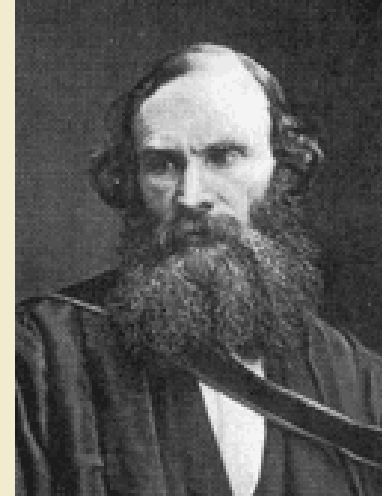


Measuring the Requirements Process

Ellen George
Steve Janiszewski
PS&J Software Six Sigma

Measurement

"To measure is to know."



Lord Kelvin
(Sir William Thomson)

"If you can not measure it, you can not improve it."

Measurement Framework

- A software process can be fully characterized with only three measurements
 - Effort: the effort required to perform an activity
 - Defects: the number & type of defects, fix time, point of injection, point of removal, and a description adequate for analysis
 - Size: the size of the work product produced
- Effort & defect metrics should be collected in-process
- Size metrics can be collected in post-mortem whenever a work product is completed

DefectDefinition

- A defect is an unexpected non-conformance to any program requirement or applicable standard that **must be corrected before the program can be used**
 - Defects usually occur in requirements, design, code or user documentation
 - Typos in comments or documentation should not be considered defects unless they could cause a misunderstanding of requirements, program functionality, or meaning of a program message

SizeMeasurements

- **Measuring productivity [unit product size/hr] and product quality [defects/unit product size] requires a product size metric**
 - **proportional to effort expended to produce product**
 - **number of defects injected proportional to the size**
- **Should be easy to count via automation**
- **Different metrics for different products**
- **The “best” size metrics have the highest degree of linearity**
- **If there are multiple size metrics with comparable characteristics, it is a matter of convention to pick one and use it**

Defining the Product

- **A good product standard is the first requirement for measuring product size**
 - **Should control product content and format**
 - Consistent from product to product**
 - Consistent from author to author**
 - **Products resulting from the same activity should not be too dissimilar or it will be impossible to pick a useful size metric**
- **A loose product standard with many optional elements will allow so much variability that it will be impossible to find a useful size metric**

Creating Measurable Requirements

- **Template based standards are frequently a good way to control requirements content and format**
 - **Template fields control content**
 - **Template directly controls format**
- **Another common approach is to write each requirement as a *single* sentence of the form “subject *shall* perform an action on an object when a condition is true”.**
 - **Individually numbered**
 - **Hierarchical organization**
 - **Document, traceability tool, database**

CaseStudy A

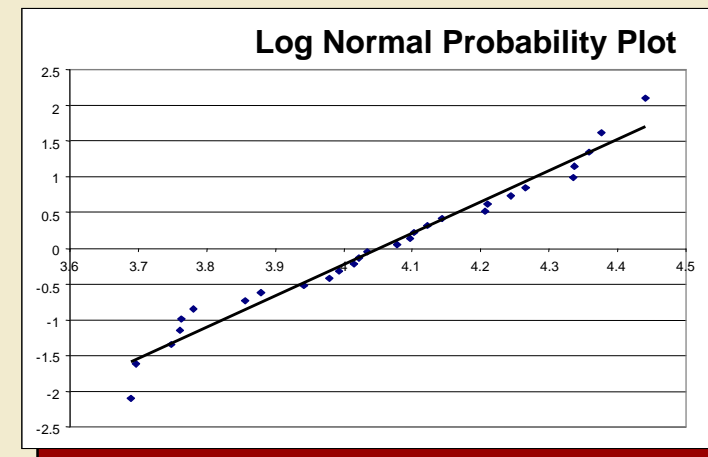
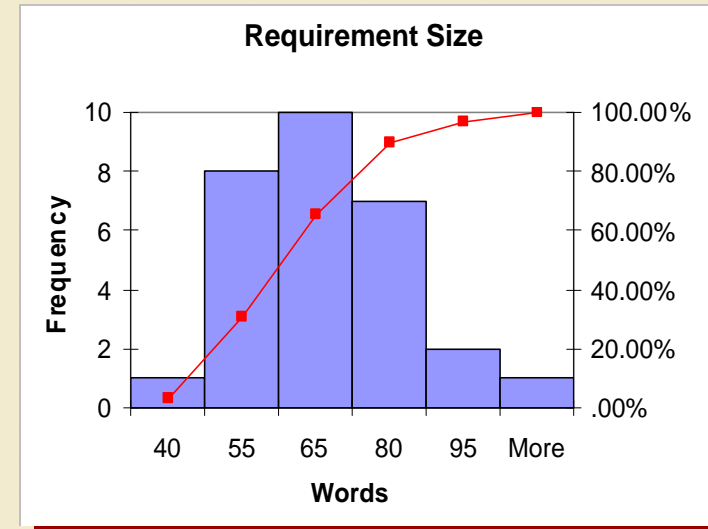
- **Template based requirements specification used for enhancement to a large distributed commercial system**
- **Brainstorming identified candidate size metrics: requirements, pages, paragraphs, words**
- **Size-Effort correlation used to screen potential size metrics**
- **Preliminary data indicated that any of the proposed size metrics could produce a good correlation with effort**
- **“Words” was tentatively selected as the size metric based on ease of automated counting**
- **Algorithms to estimate the size of and effort to write a section of a requirements spec were developed & validated**

Estimating Size

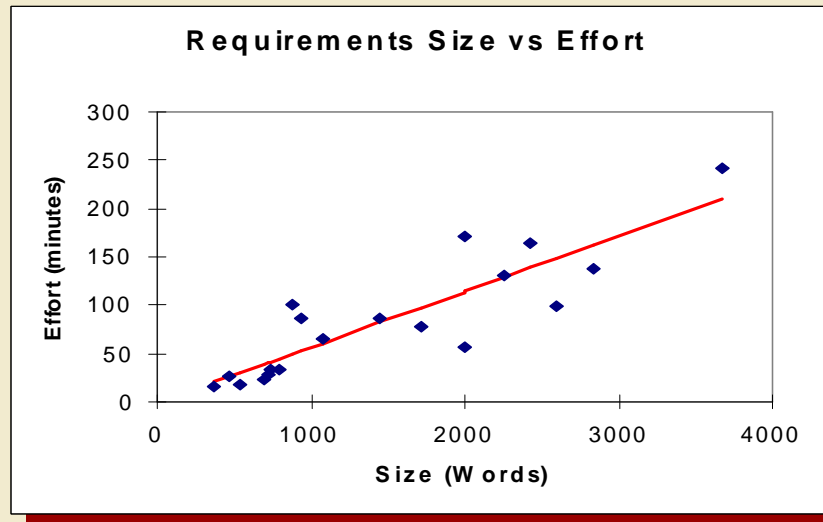
- Histogram of requirements size distribution suggested trying log-normal distribution
- Log-normal distribution provided adequate fit to data
- Proxy Based Size Estimates based on relative size of requirements

Very small	37
Small	46
Medium	58
Large	72
Very Large	90

- Example: 10 small + 30 medium + 15 large → $10 \cdot 46 + 30 \cdot 58 + 15 \cdot 72 = 3280$ words



Estimating Effort & Defects



- Productivity data segmented by engineer
- Good correlation ($r^2 = 0.77$) between size and effort for data resulted in straight forward application of linear regression to estimate effort based on size estimates
- Planned to estimate defects based on actual size and measured defect density

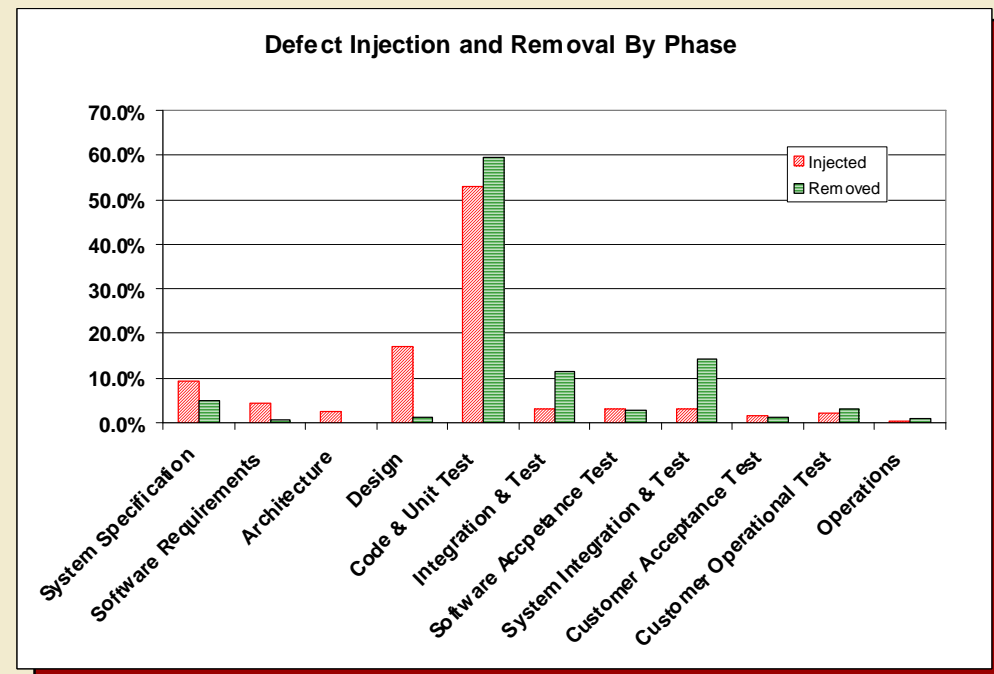
Requirements Defects

- A commonly held point of view is that most software defects, and, in particular, most “important” software defects are requirements errors
 - In some cases, this is probably correct
 - In other cases, it only appears to be the case because the organization doesn’t track defects until a product is in acceptance test where most requirements defects are found (hidden factory)
 - Labeling a defect a requirements error often place the responsibility for the problem elsewhere: “It’s not a failure - it’s a correct implementation of an incorrect requirement*.”
- What does the data say?

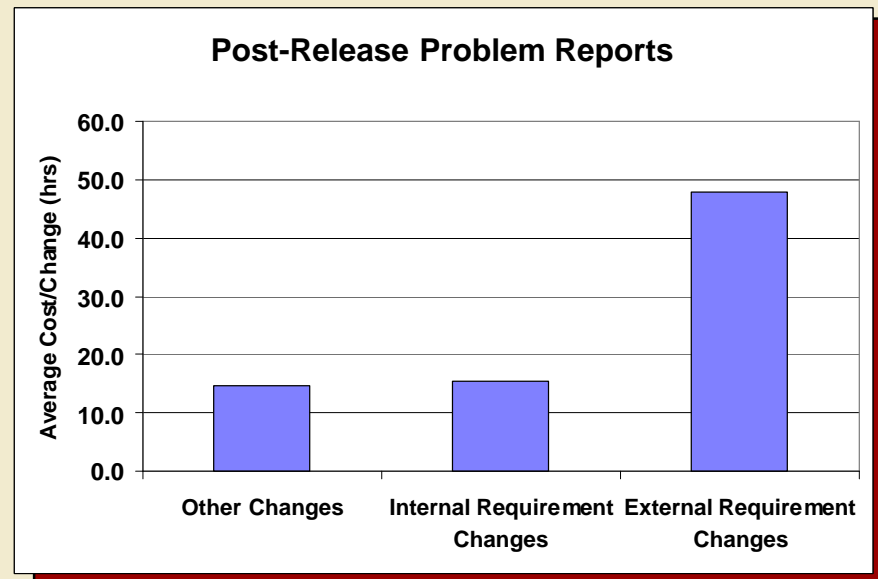
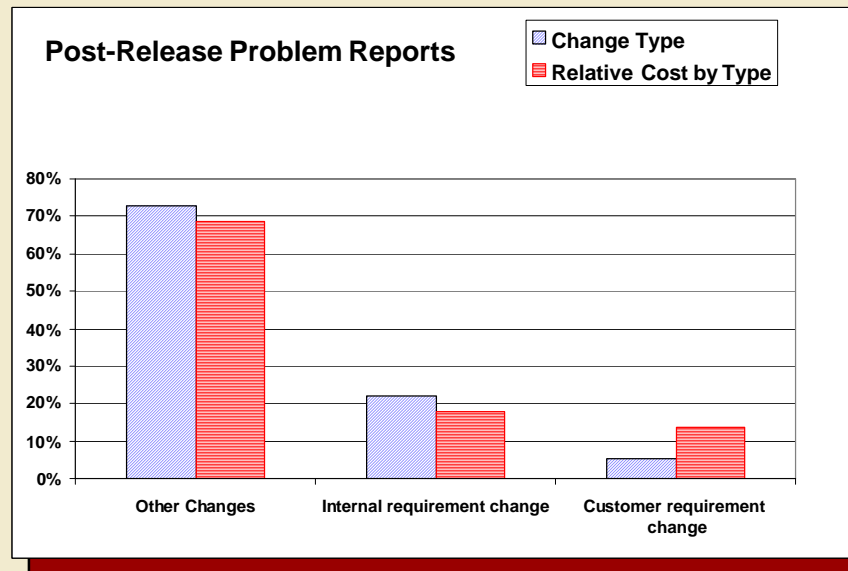
*provided by somebody else

CaseStudy B

- Defect data from multiple projects, typically real time embedded avionics systems, developed under DoD contract
- Approximately 9000 defects collected over a period of about 5 years
- About 14% of defects were requirements related
- Barry Boehm's COCOMO II model indicates 16% injected in requirements, 34% in design, and 50% in code



Post-Release Problem Reports



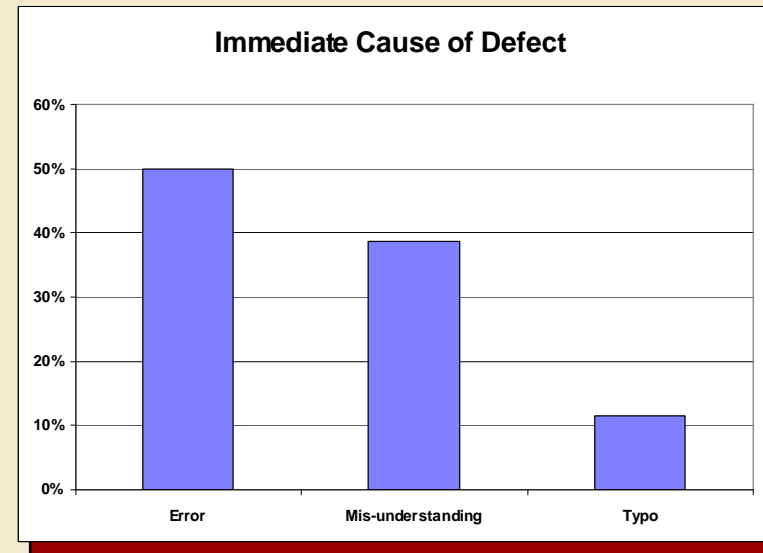
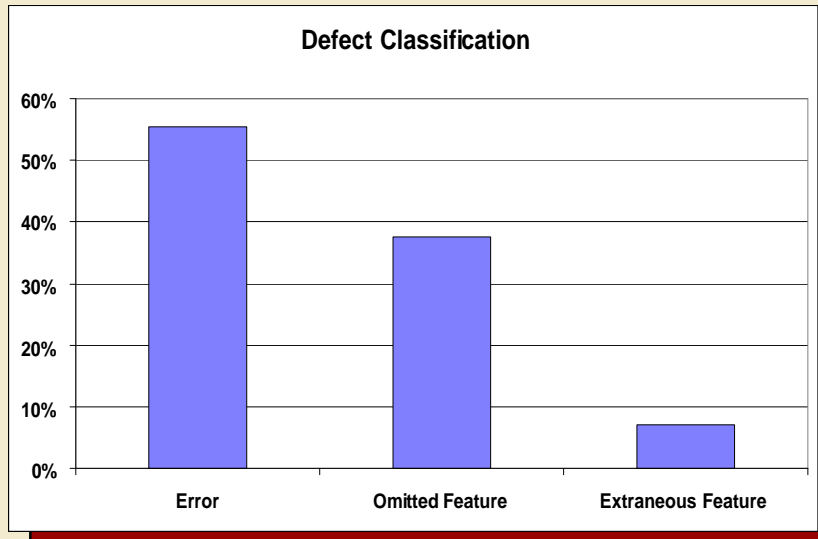
- Includes Integration & Test and subsequent phases
- 4250 Problem Reports over 35 months
- Requirements changes accounted for 27% of change activity and 31% of change related costs
- Customer directed changes cost more, but internal requirement changes cost the same as other changes

TrackingProcess Performance

	Removed At							
Injected At	Requirements	Design	Code & Unit Test	Integration & Test	Formal Test	System Integration & Test	Flight Test	Grand Total
Requirements	0	7	20	157	37	10	94	325
Design		495	213	109	35	38	0	890
Code & Unit Test			1921	490	107	28	25	2571
Integration & Test				177	5	3	0	185
Formal Test					36	10	0	46
System Integration & Test						2	0	2
Grand Total	0	502	2154	933	220	91	119	4019

- Leakage Matrix related point of injection and point of removal
- Used to assess process performance
- Diagonal terms are found in-phase
- Off diagonals are escapes

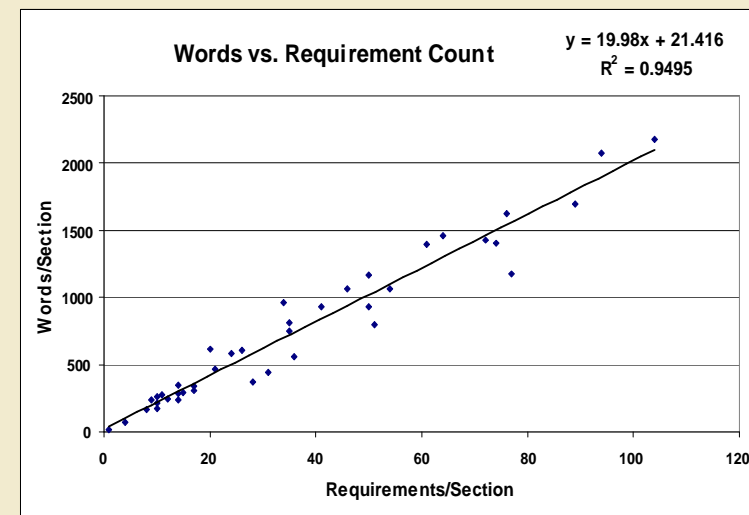
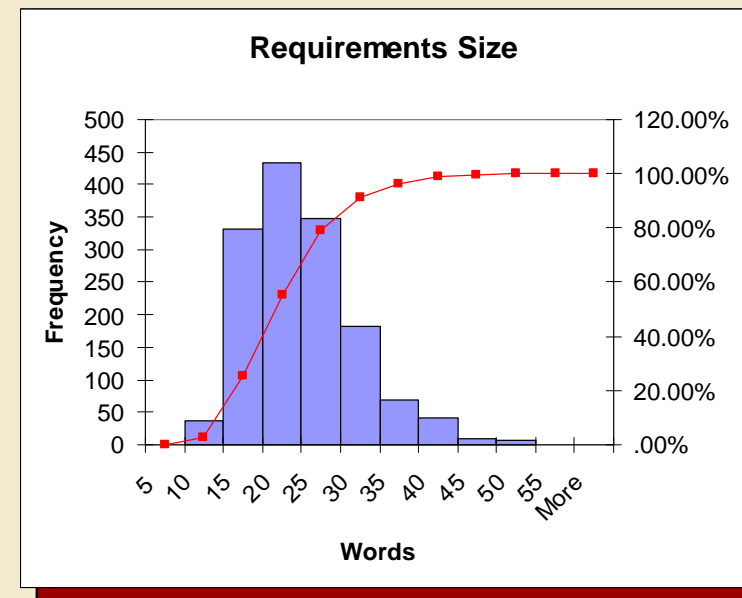
Looking at the Cause of Defects



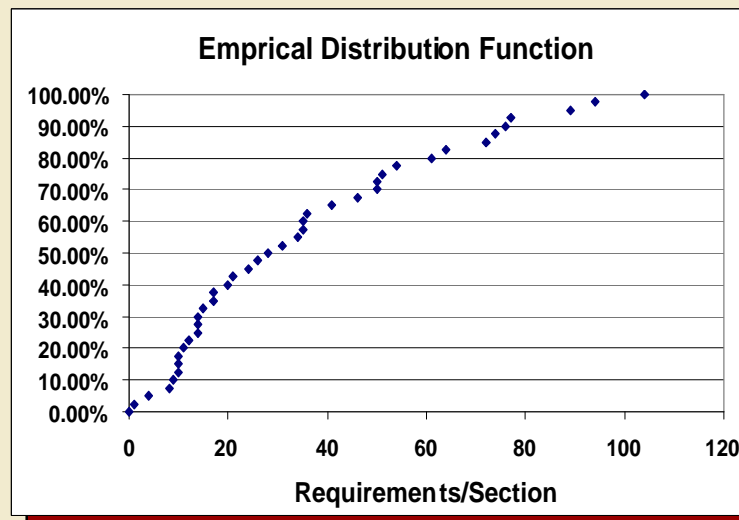
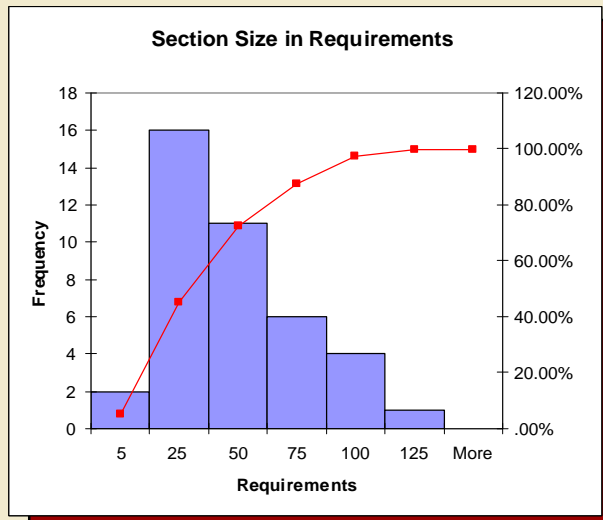
- Team members were asked to classify defects and make a subjective assessment of the immediate cause when the defects were recorded.
- The 37% classified as omitted features tracks the 39% attributed to misunderstandings
- The numbers suggest that the clarity of and/or communication of requirements may be an issue even when they are “correct”

CaseStudy C

- Data drawn from upgrades to several major avionics systems
- Textual requirements stored in a commercial traceability tool
- Requirements size measured in words looks similar to Case Study A, but has narrower range due to single sentence standard
- Strong correlation between requirements count & words indicates requirements count is also a good size metric



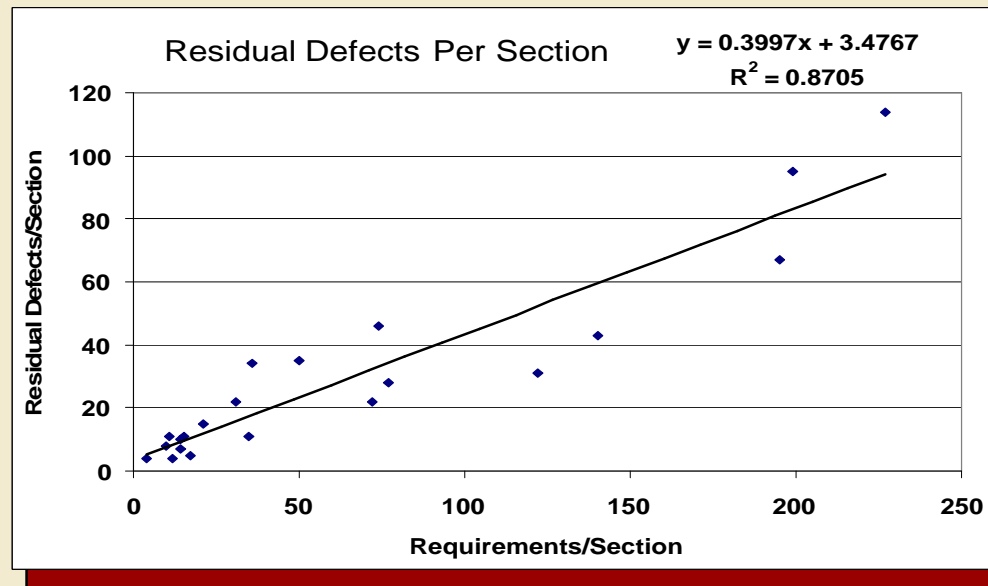
Estimating Algorithm



Estimating Scheme	
very small (5%)	4
small (15%)	10
medium (50%)	28
large (85%)	72
very large (95%)	89

- Goal is to estimate number of requirements per section of specification
- Algorithm based on statistics of historical section sizes
- Section sizes don't follow log normal distribution, but algorithm can be based on empirical distribution function
- Outline spec can be translated directly into size estimate via algorithm

Specification Quality



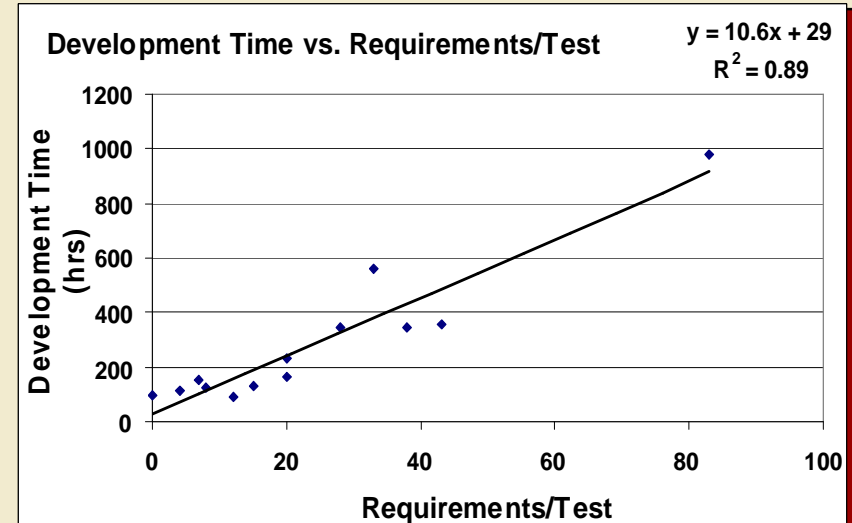
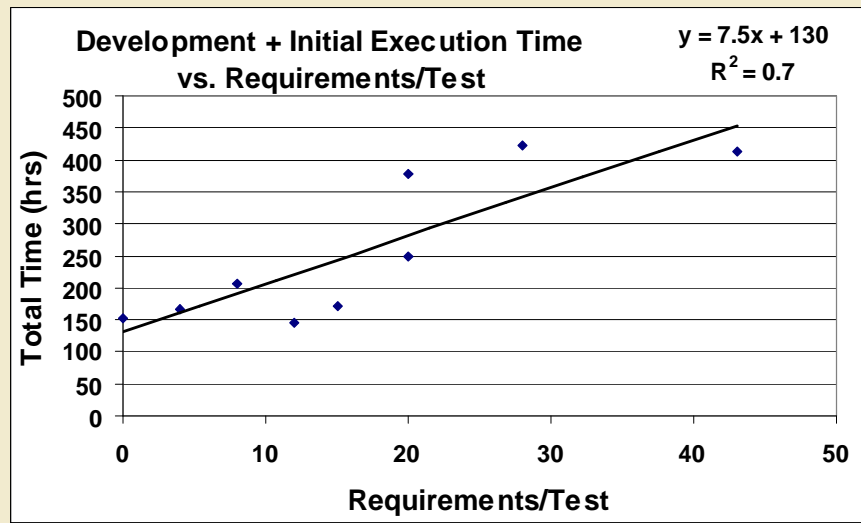
- What is the quality of a requirements spec after completion of peer review and a formal requirements review?
- Independent audit of a released specification indicated a residual defect density = 0.4 defects/requirement

Residual Defects

	Issue										
Priority	Ambiguous	Conflicting	Erroneous	Incomplete	Negative Statement	Passive Voice	Redundant	Undefined Term	Unnecessary	Vague	Total
High	1	2	3	9	0	0	0	12	5	5	37
Med.	93	9	7	230	1	5	5	50	2	75	477
Low	26	11	2	3	9	4	11	19	2	29	116
Total	120	22	12	242	10	9	16	81	9	109	630

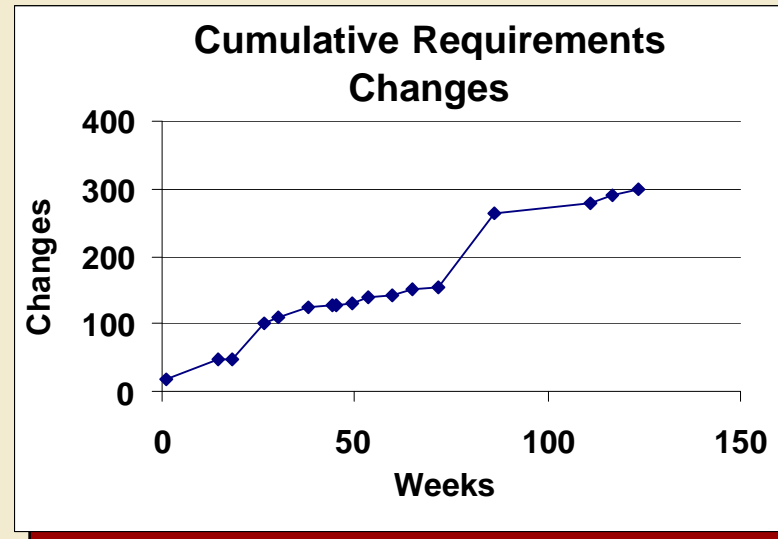
- Many residual defects have to do with requirements that use undefined terms, are incomplete in that they don't fully specify actors or conditions, are ambiguously stated, or are just plain vague
- The meaning of most would be clear to someone experienced in the application
- Is that enough given the Case Study B data that indicates omitted functionality and misunderstanding as common sources of defects?

Estimating Formal Testing Effort



- It may be possible to use the number of requirements per test case to estimate test case development, test case debug, and test case execution times
- Depends on code having a reasonably consistent quality level prior to start of formal testing

Tracking Requirements Changes



- Typically it is useful to track requirements changes (total number of additions, deletions, and modification per time period) over time relative to start of project or some intermediate baseline
- Rate of growth usually indicates maturity of project/product

References

**For additional information visit our
web site or contact us at:**

**Ellen George 201- 358-8828
EllenGeorge@SoftwareSixSigma.com**

**Steve Janiszewski 201- 947-0150
SteveJaniszewski@SoftwareSixSigma.com**

www.SoftwareSixSigma.com

Abstract

The CMMI's placement of the Measurement and Analysis Process Area at maturity level 2 emphasizes the importance of a structured approach measurement early on in an organization's quality journey. This emphasis on measurement falls nicely into line with another prominent approach to process improvement, Six Sigma, where a fundamental tenet is "What gets measured gets managed."

Many organizations perceive that the requirements development sub-process is the weak link in their overall software development process. This presentation provides a road map for setting up a measurement framework for requirements. Three fundamental measures are required for any measurement framework: size, effort, and defects. Many organizations are comfortable with measuring effort and defects but have difficulties picking an appropriate size metrics. We illustrate the process of picking an appropriate size metric in some detail and explore its use in estimating. We also address quality measurements, stability, and the relative importance of requirement defects vs. implementation, design, and test case defects.

We identify common pitfalls and mechanisms for avoiding them. The presentation includes sample data and results from several organizations.