

Improving Project Planning and Control: A 10-Step Process Within CMMI or other Process Orientations

Daniel D. Galorath – galorath@galorath.com

Copyright Galorath Incorporated 2008

Agenda



- Introduction
- Definitions
- How estimation fits into larger process initiatives
- The 10 step process
- Questions - Answers

The Key to project planning and control are



Viable estimates as the basis of achievable plans

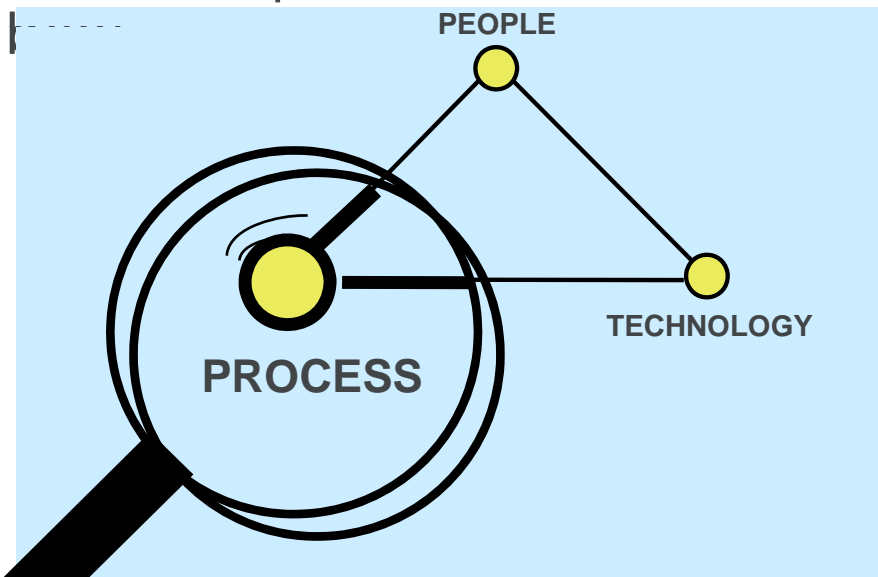
Repeatable Processes

Measurement and adjustment during the process

Refined estimates & plans if the project changes

People, Process, Technology Are Keys Source CMMI Tutorial

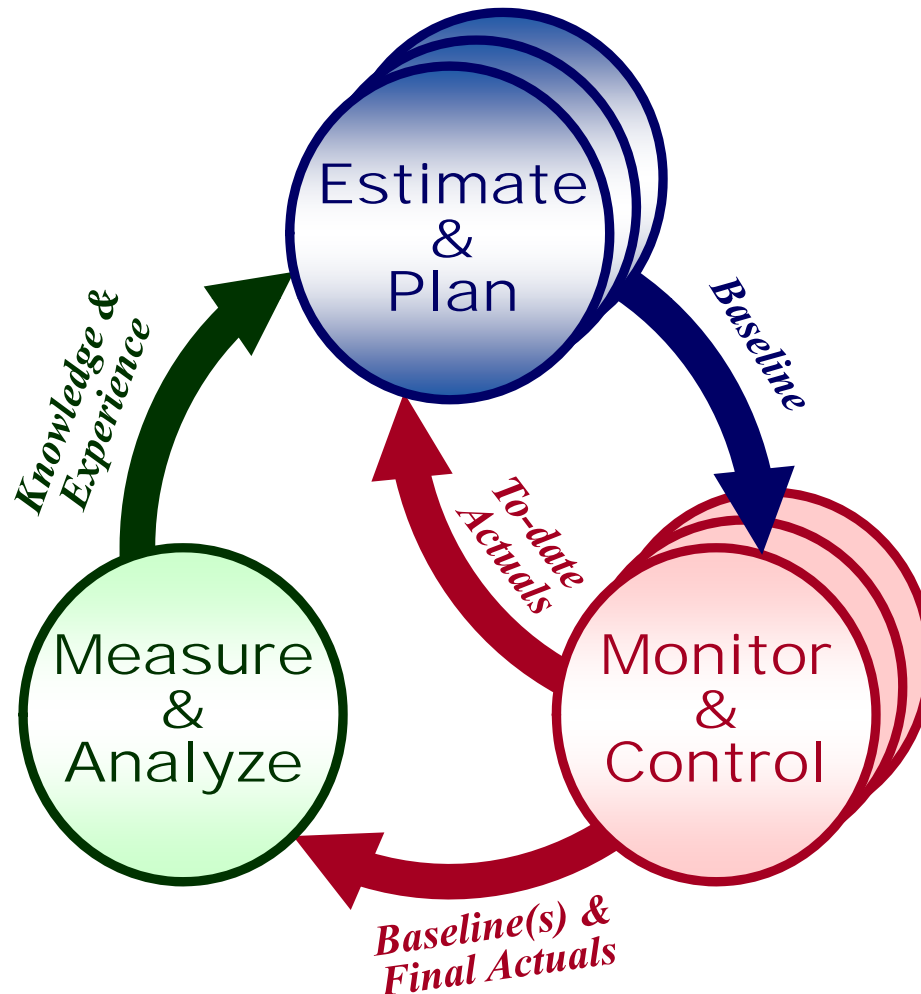
- Everyone realizes the importance of having a motivated, quality work force but...
- ...even our finest people can't perform at their best when the process is not understood or operating "at its



**Major determinants of
product cost, schedule, and
quality**

CMMI Repeatable Process Areas of Interest

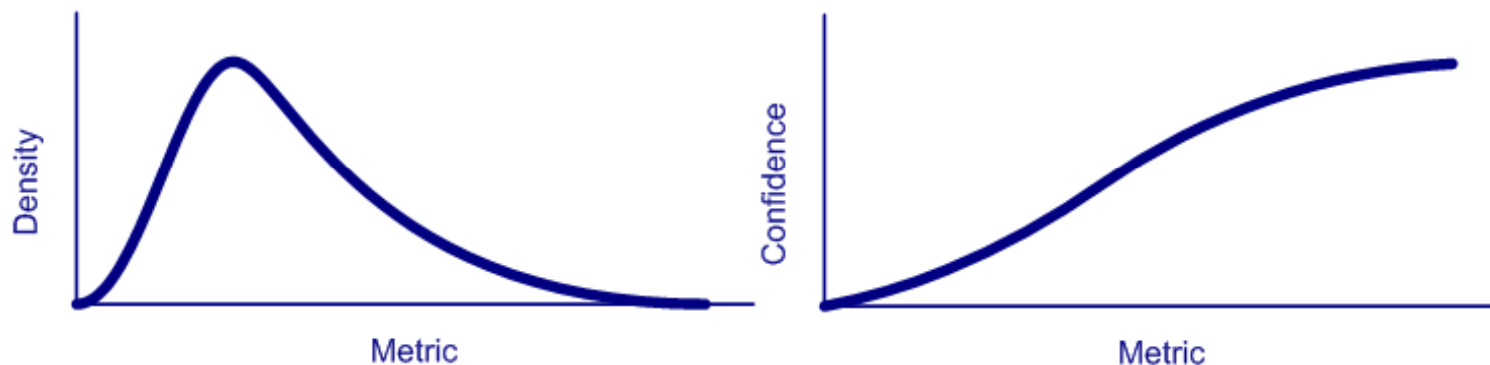
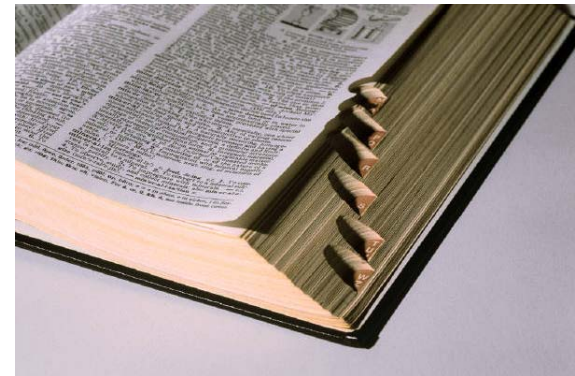
Quantitative Project Management



A Foundation of Risk Management

ESTIMATION & PLANNING: An Estimate Defined

- An **estimate** is the most knowledgeable statement you can make **at a particular point in time** regarding:
 - Effort / Cost
 - Schedule
 - Staffing
 - Risk
 - Reliability
- Estimates more precise with progress
- ***A WELL FORMED ESTIMATE IS A DISTRIBUTION***



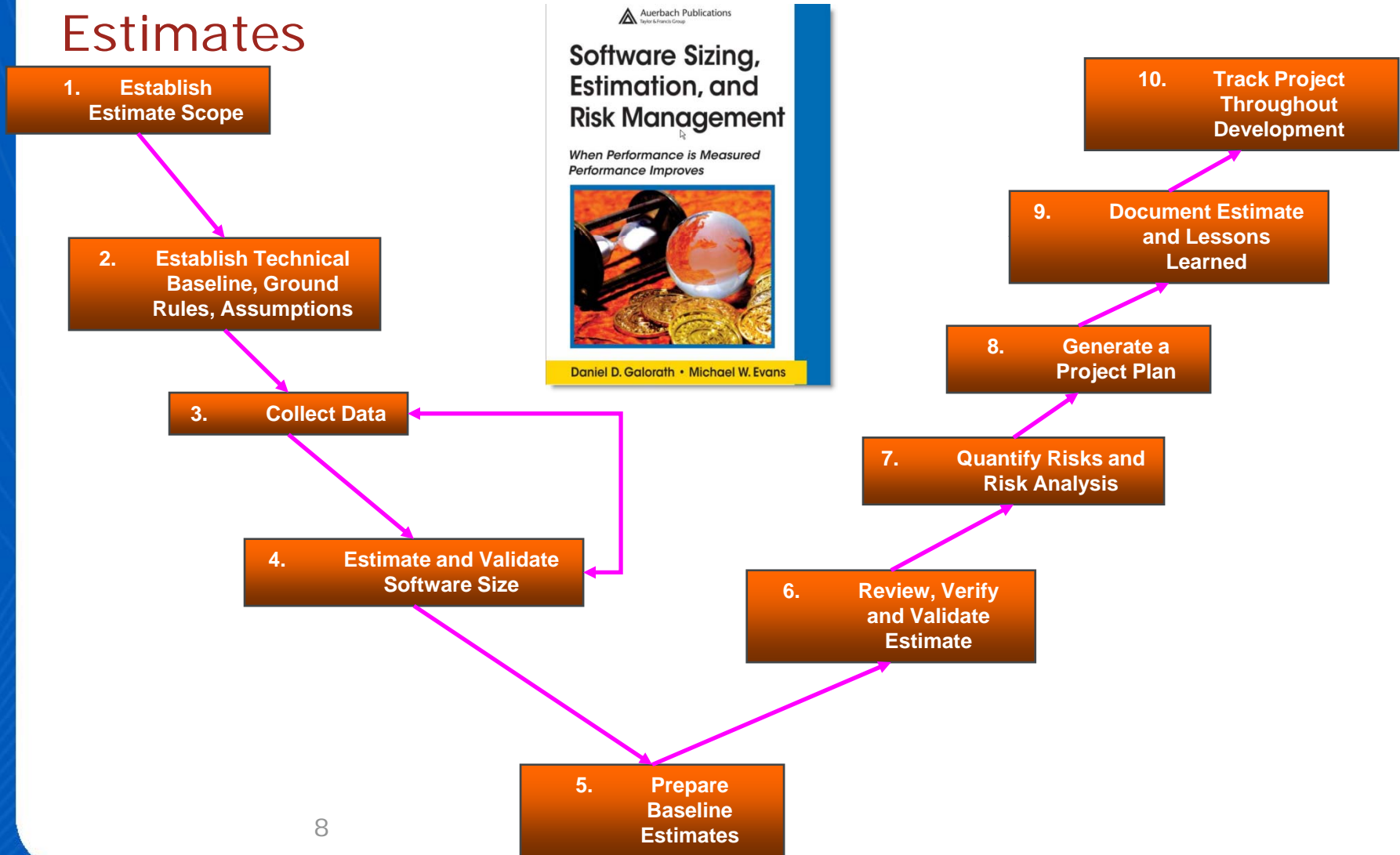
Poor Estimates Effects on Projects



- Inaccurate estimates significant impact on project success:
 - Poor implementations
 - Critical processes don't scale
 - Emergency staffing
 - Cost overruns caused by underestimating project needs
- Lack of well defined objectives, requirements, & specifications, results in creeping scope resulting in:
 - Forever changing project goals
 - Frustration: Death Marches
 - Customer dissatisfaction
 - Cost overruns and missed schedules
 - Project Failures
- Incorrect estimates / bad plans are a root cause of subsequent program risk

Estimating & Planning are key to software project success

10 Step Software Estimation Process: Consistent Processes = Reliable Estimates



Estimation Methods 1 of 2



Model Category	Description	Advantages	Limitations
Guessing	Off the cuff estimates	Quick Can obtain any answer desired	No Basis or substantiation No Process Usually Wrong
Analogy	Compare project with past similar projects.	Estimates are based on actual experience.	Truly similar projects must exist.
Expert Judgment	Consult with one or more experts.	Little or no historical data is needed; good for new or unique projects.	Experts tend to be biased; knowledge level is sometimes questionable; may not be consistent.
Top Down Estimation	A hierarchical decomposition of the system into progressively smaller components is used to estimate the size of a software component.	Provides an estimate linked to requirements and allows common libraries to size lower level components.	Need valid requirements. Difficult to track architecture; engineering bias may lead to underestimation.

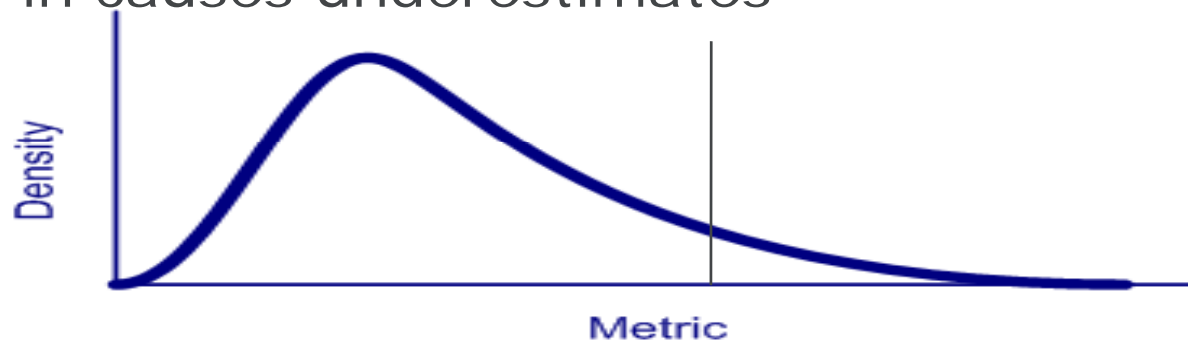
Estimation Methods 2 of 2



Model Category	Description	Advantages	Limitations
Design To Cost	Uses expert judgment to determine how much functionality can be provided for given budget.	Easy to get under stakeholder number	Little or no engineering basis.
Simple CER's	Equation with one or more unknowns that provides cost / schedule estimate	Some basis in data	Simple relationships may not tell the whole story Historical data may not tell the whole story
Comprehensive Parametric Models	Perform overall estimate using design parameters and mathematical algorithms.	Models are usually fast and easy to use, and useful early in a program; they are also objective and repeatable.	Models can be inaccurate if not properly calibrated and validated; historical data may not be relevant to new programs; optimism in parameters may lead to underestimation.

Manual Estimates Human Reasons For Error

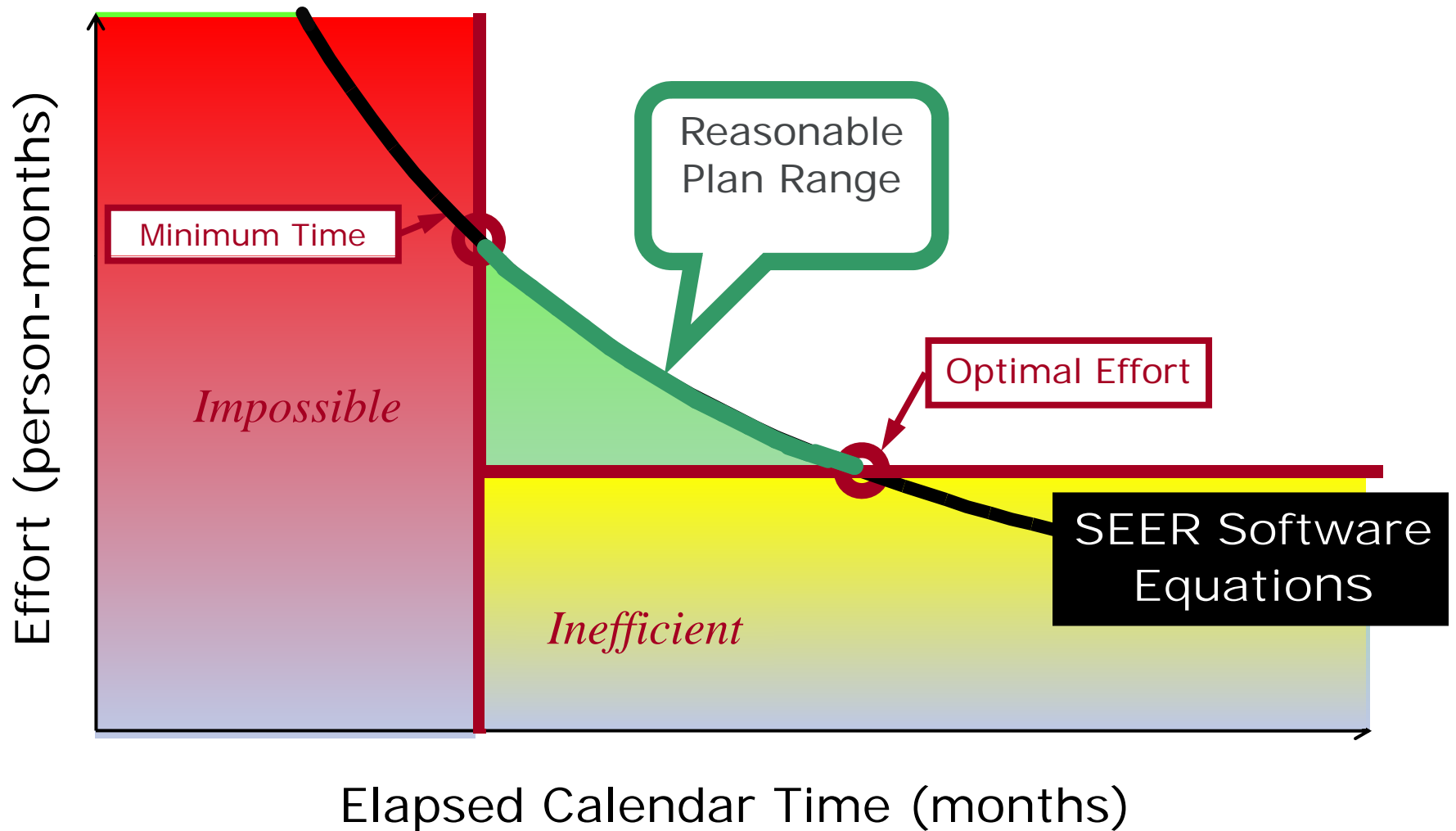
- Desire for “credibility” motivates overestimate behavior (80% probability?)
 - So must spend all the time to be “reliable”
 - Better approach force 50% probability & have “buffer” for overruns
- Technical pride causes underestimates
- Buy-in causes underestimates



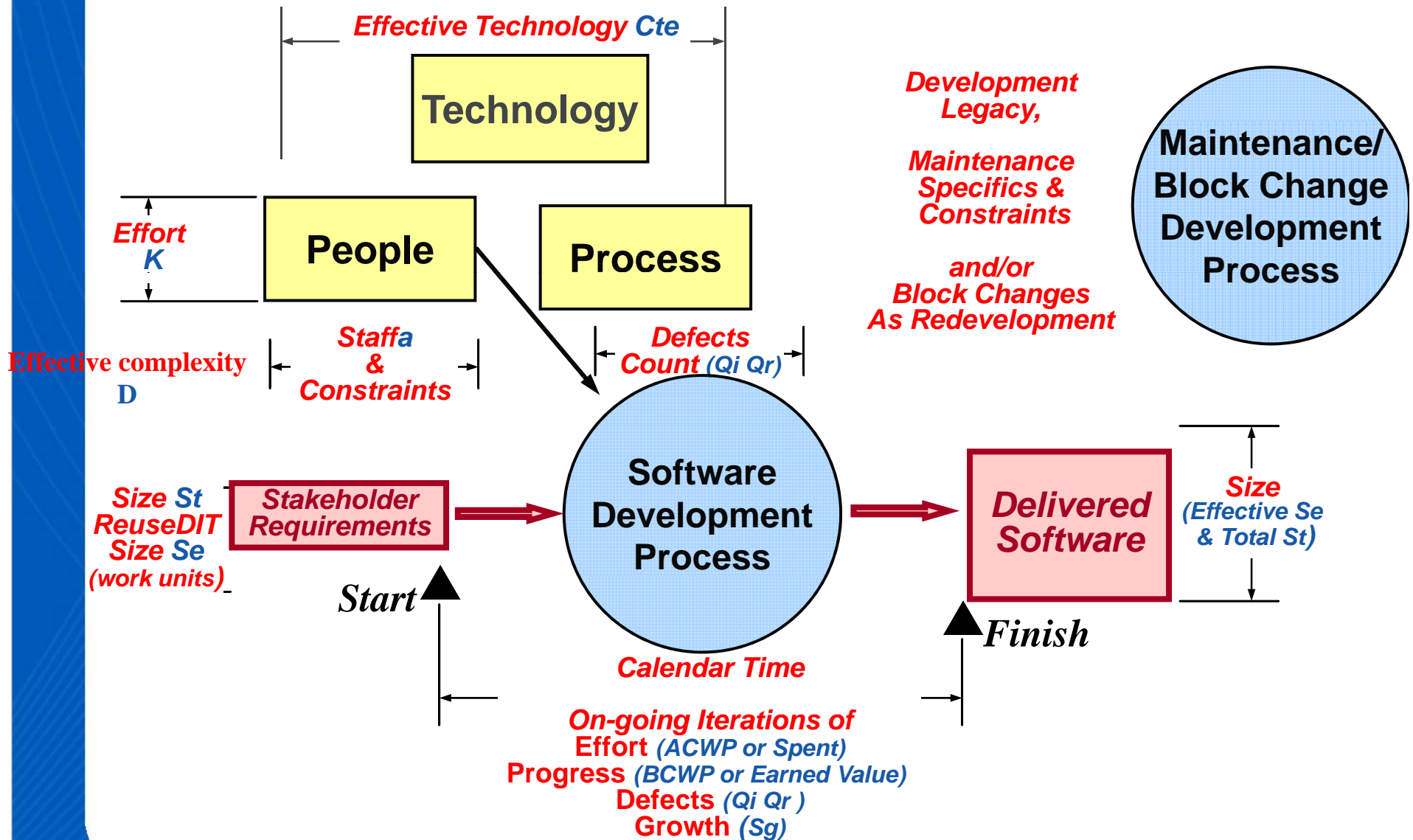
Knowing Software Planning Possibilities Is Critical To Success



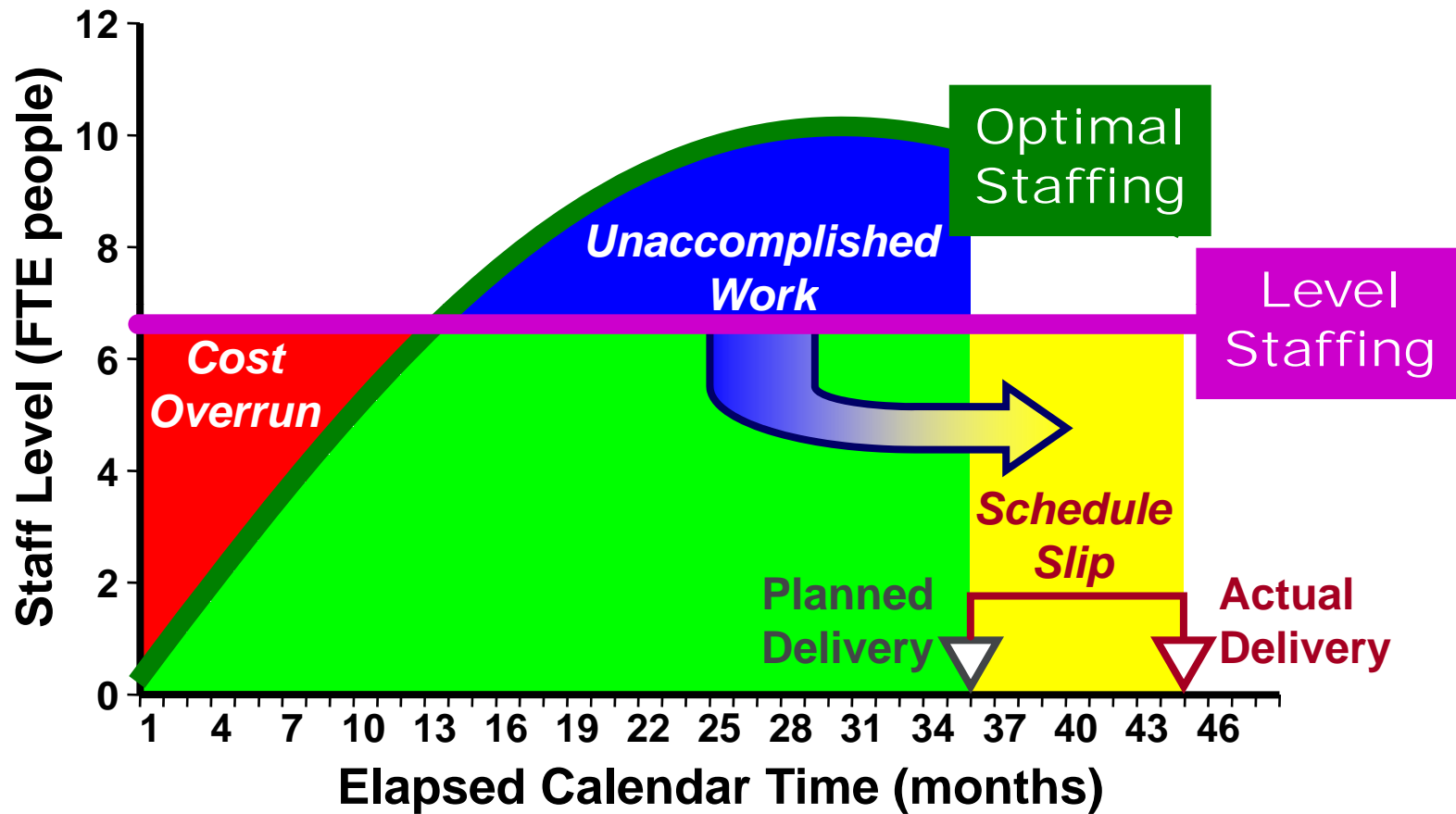
For a given Size, Technology, Complexity & Probability



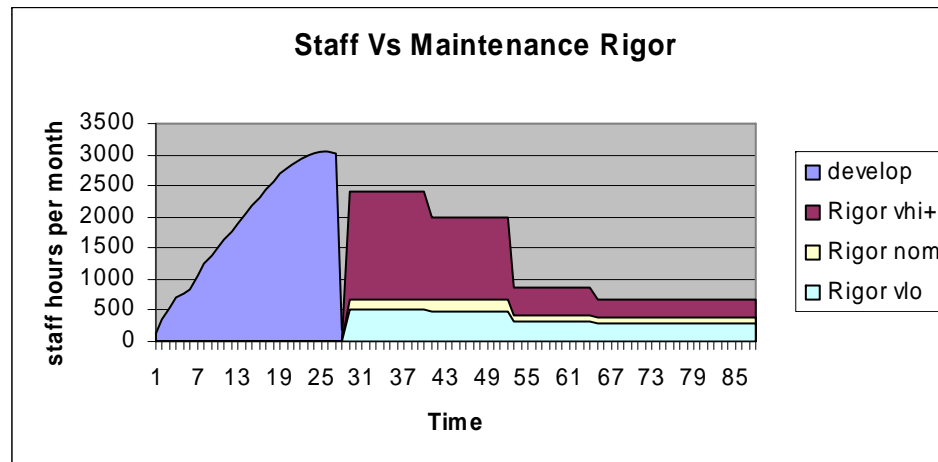
Software Estimation Basic Model & Associated Metrics



Avoid "Death Marches" and Failed Projects By Applying "Brooks Law"

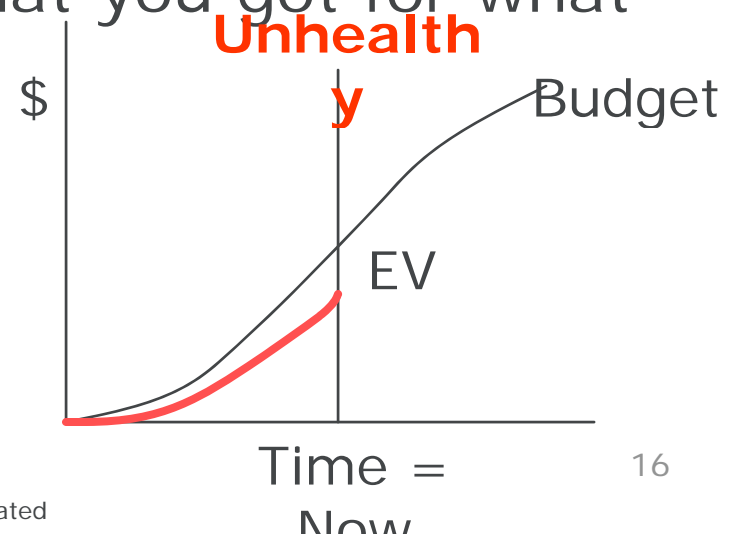
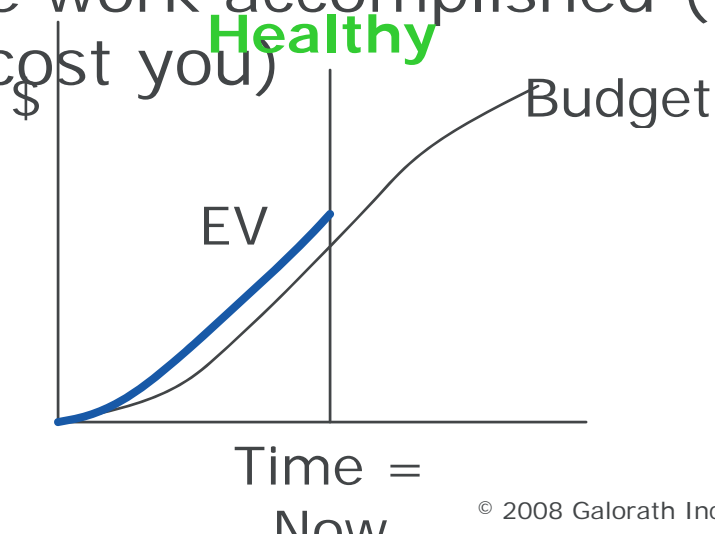


Understand Project Total Ownership Costs Up Front Most Projects Spend Low During Maintenance



Use Earned Value TO Quantify

- The main concern of EVM is what has been accomplished in a given time and budget, versus what was planned for the same time and budget
 - A project is generally deemed healthy if what has been accomplished is what was planned, or more
 - A project is deemed unhealthy if accomplishment lags expectations
- Definition: Earned value = budgeted value for the work accomplished (what you got for what it cost you)



What To Measure: Multiplicity of Metrics



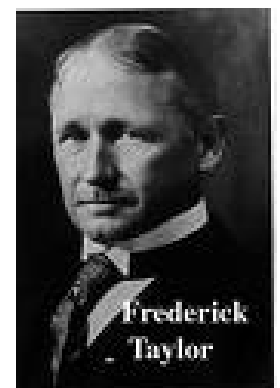
1. Obvious: Status / Trend Metrics: e.g. productivity, defects removal rate, cost, schedule
2. Most important for improvement: Effectiveness (5 max)
 - “What we are doing that we should not do” e.g. number of delivered critical defects
 - “What we are not doing that we should do” e.g. number of defects that got past inspections
 - These metrics may change over time as we improve

The Hawthorne Effect:

People Respond To Being

Measured

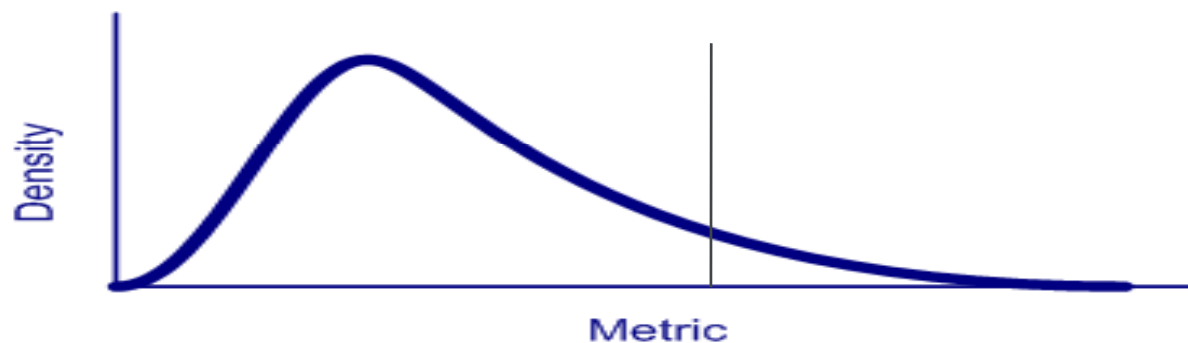
- Goal: Find optimum for productivity 1924 to 1927
- **Increase, No Control Group**; Three departments; all showed an increase of productivity, whether illumination increased or decreased.
- **Increase, Control group** = change in lighting; experimental group got sequence of increasing light. Both groups substantially increased production, no difference between groups
- **Decrease, Control group got stable lights**; other sequence of decreasing levels. Both groups steadily increased production until the light in experimental group got so low they protested and production fell off
- All back to original: Productivity went up



Manual Estimates: Human Reasons For Error (Metrics Can Help)

- Desire for “credibility” motivates overestimate behavior (80% probability?)
 - So must spend all the time to be “reliable”
 - Better approach force 50% probability & have “buffer” for overruns
- Technical pride causes underestimates

• Buy



Theory of Constraints

Questions Regarding Value

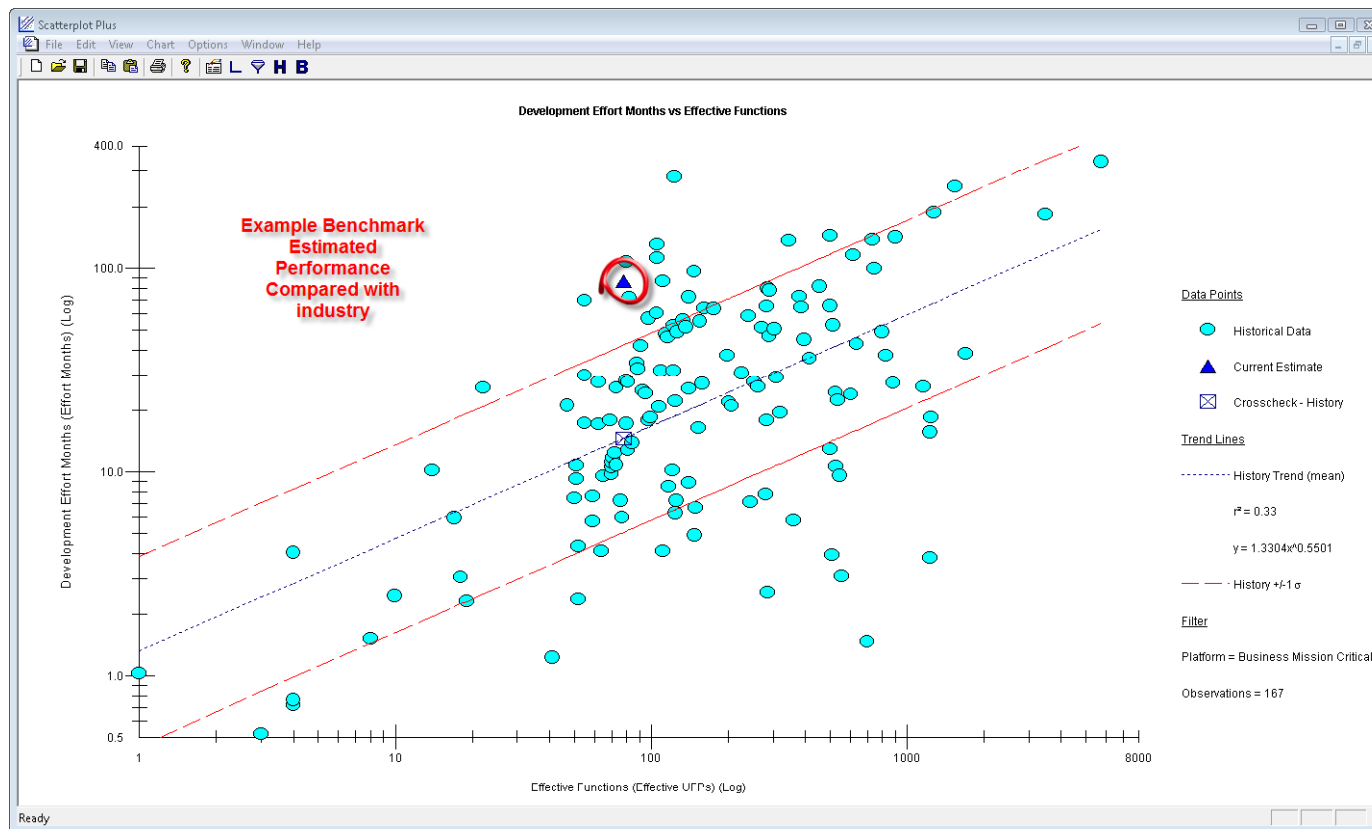
1. What is the main power of the technology?
(Source Goldratt)

2. What limitation does it diminish?

3. What rules helped us to accommodate the limitation?

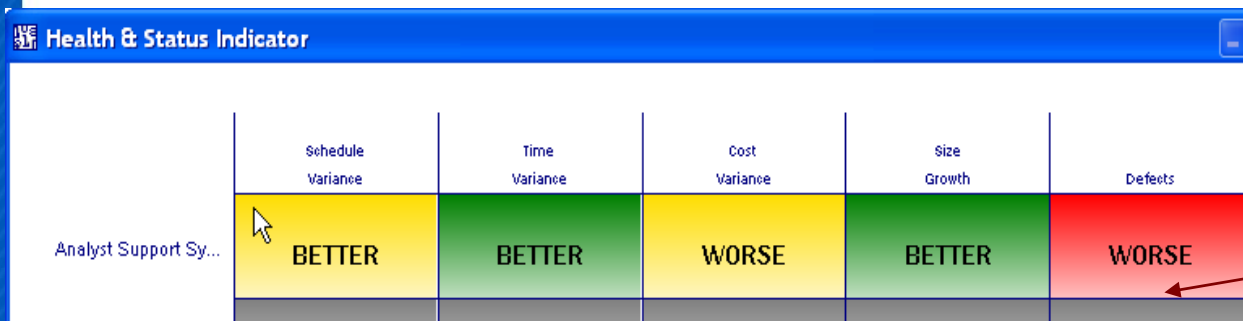
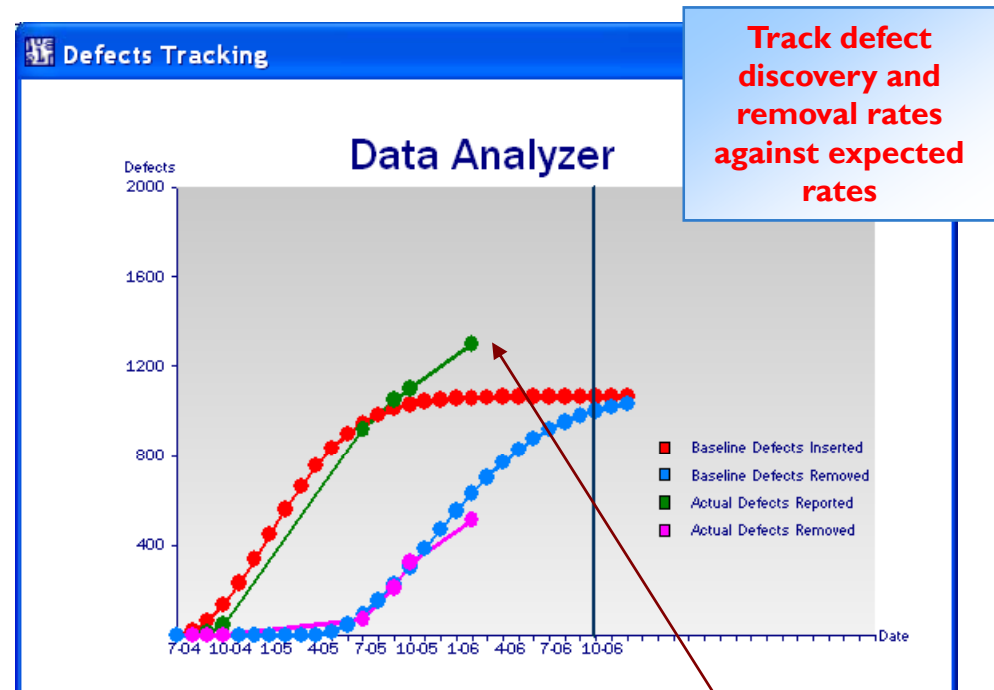
4. What rules should we use now?

Example Benchmark Versus an Estimate.. Why Are We So Expensive?



Defects and Growth Impact Software Process

Health and Status Indicator shows status and trends from the previous snapshot
Thresholds are user definable

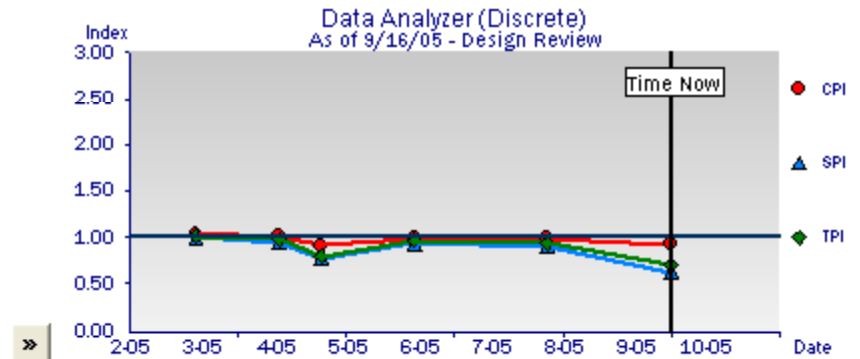


Increased defect reporting rate shows a worsening trend

Parametric Project Monitoring & Control

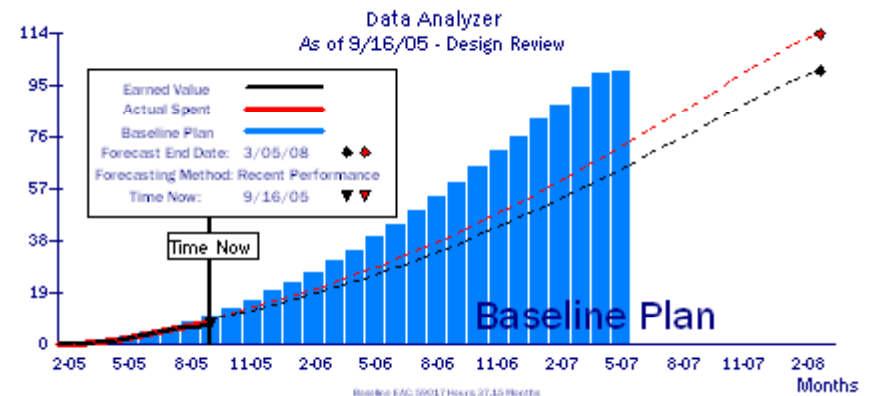
Provides Performance Measurement aspects of ANSI/EIA-STD-748

- Adds Performance Measurement (*Earned Value*) methods to parametric estimation model
- Accepts progress & expenditure inputs
- Provides cost, schedule, and time variances
- Provides cost, schedule, & time indices
- Performance-based cost & schedule Estimate at Completion

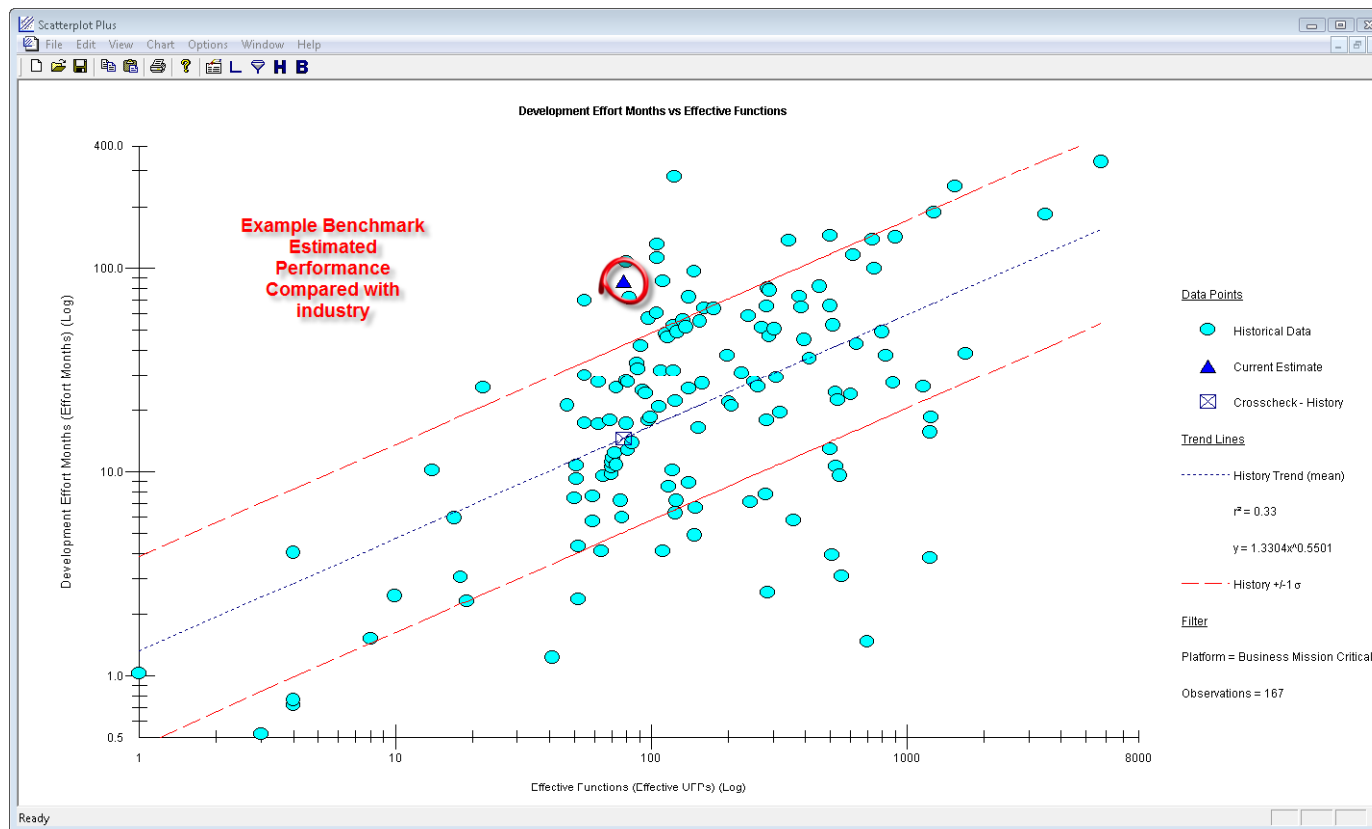


- Displays health and status indicators

Health & Status Indicator					
	Earned Value	Schedule	Cost Variance	Size Growth	Defects Outstanding
CPU 1	WORSE	WORSE	BETTER	NO CHANGE	NO CHANGE
	N/A	N/A	N/A	N/A	N/A
	N/A	N/A	N/A	N/A	N/A
	N/A	N/A	N/A	N/A	N/A

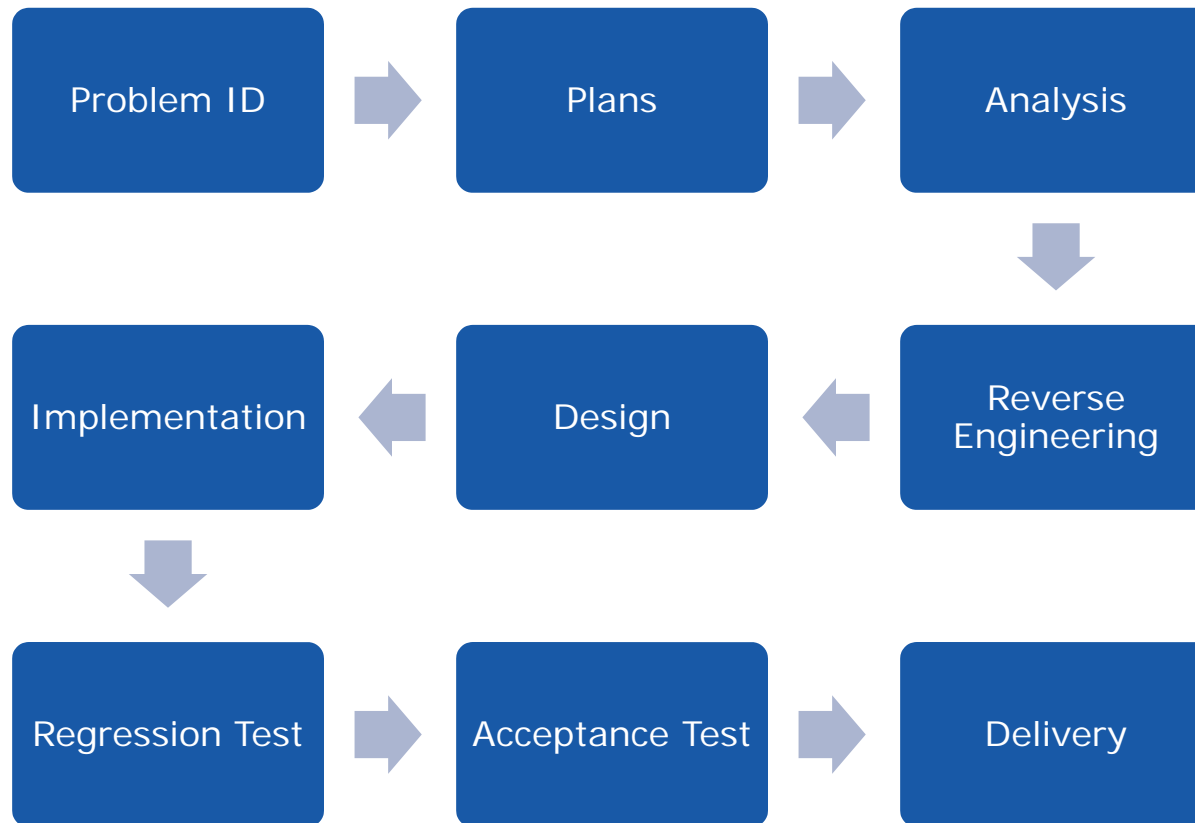


Example Benchmark Versus an Estimate.. Why Are We So Expensive?



Maintenance Phases / Activities

IEEE Et Al...



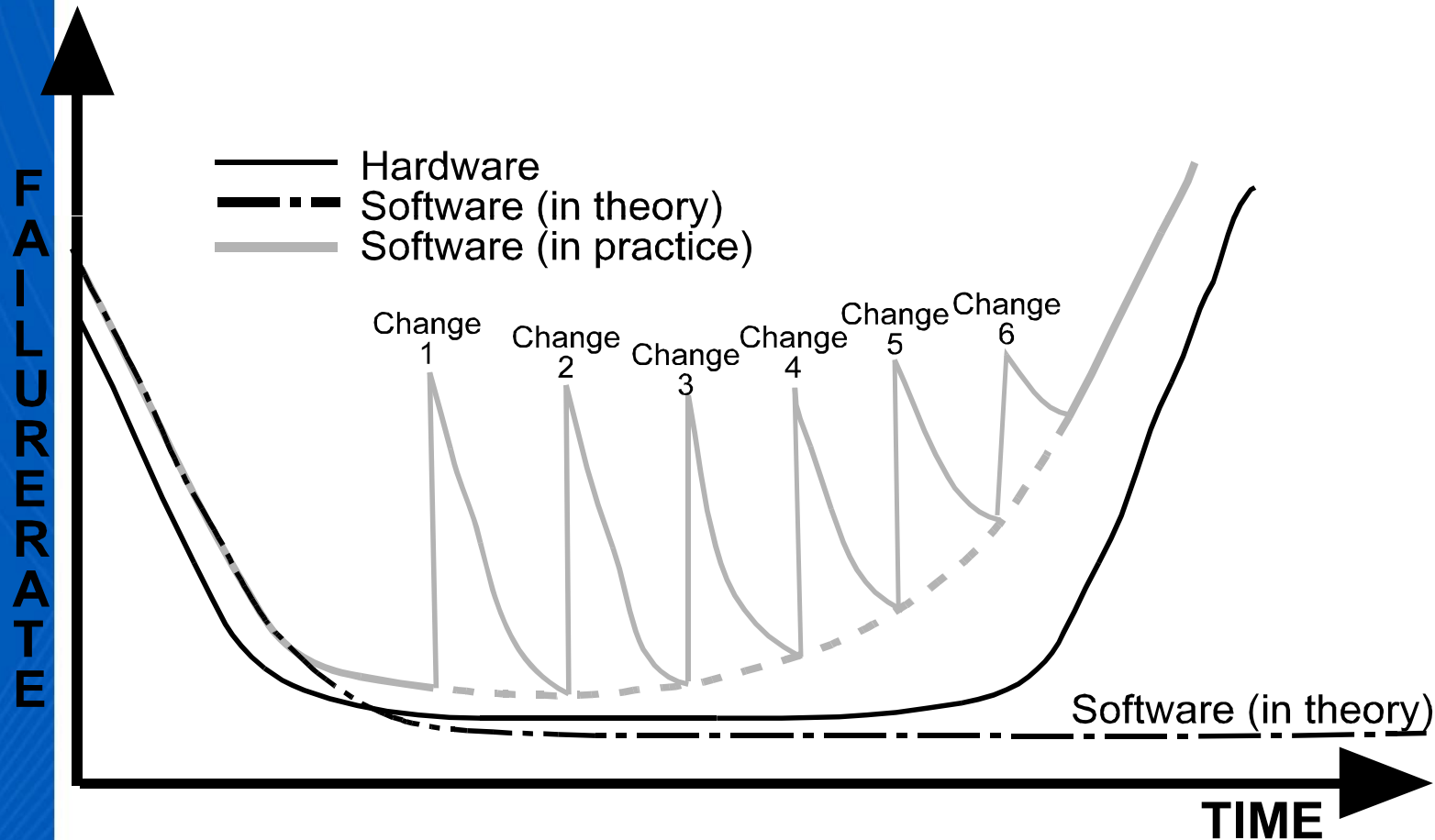
Development Metrics Are Applicable In Addition To Maintenance Metrics

Why Total Lifecycle Measurement matters



- NIST Study
 - Software defects cost U.S. almost \$60 billion annually
 - 80% of development costs software developers identifying and correcting defects
- CHAOS Report (Standish Group)
 - Canceled projects cost \$55 billion dollars

Software Maintenance Is Often A Series of Block Changes



Software Maintenance Critical Success Factors (Source IEEE)



Functionality: Preserve or enhance functionality



Quality: Preserve or increase quality of system



Complexity: Should not increase product complexity relative to the size



Volatility: should not lead to increase in product volatility



Costs: Relative costs per maintenance task should not increase for similarly scoped tasks



Deadlines: Agreed upon release deadlines should be kept and delays should not increase



User Satisfaction: Increase or at least not decrease



Profitability: Be profitable or at least cover its costs

Why Maintenance Is Hard



- May not have had maintenance as a goal
- System may not have been fully tested
- Documentation may be inadequate
- Maintenance staff may be inexperienced
- The tendency to produce quick & dirty fixes
- Process or language experience may have left a mess
- The "but I only changed 1 line syndrome"

Why Software Maintenance Metrics Are Harder

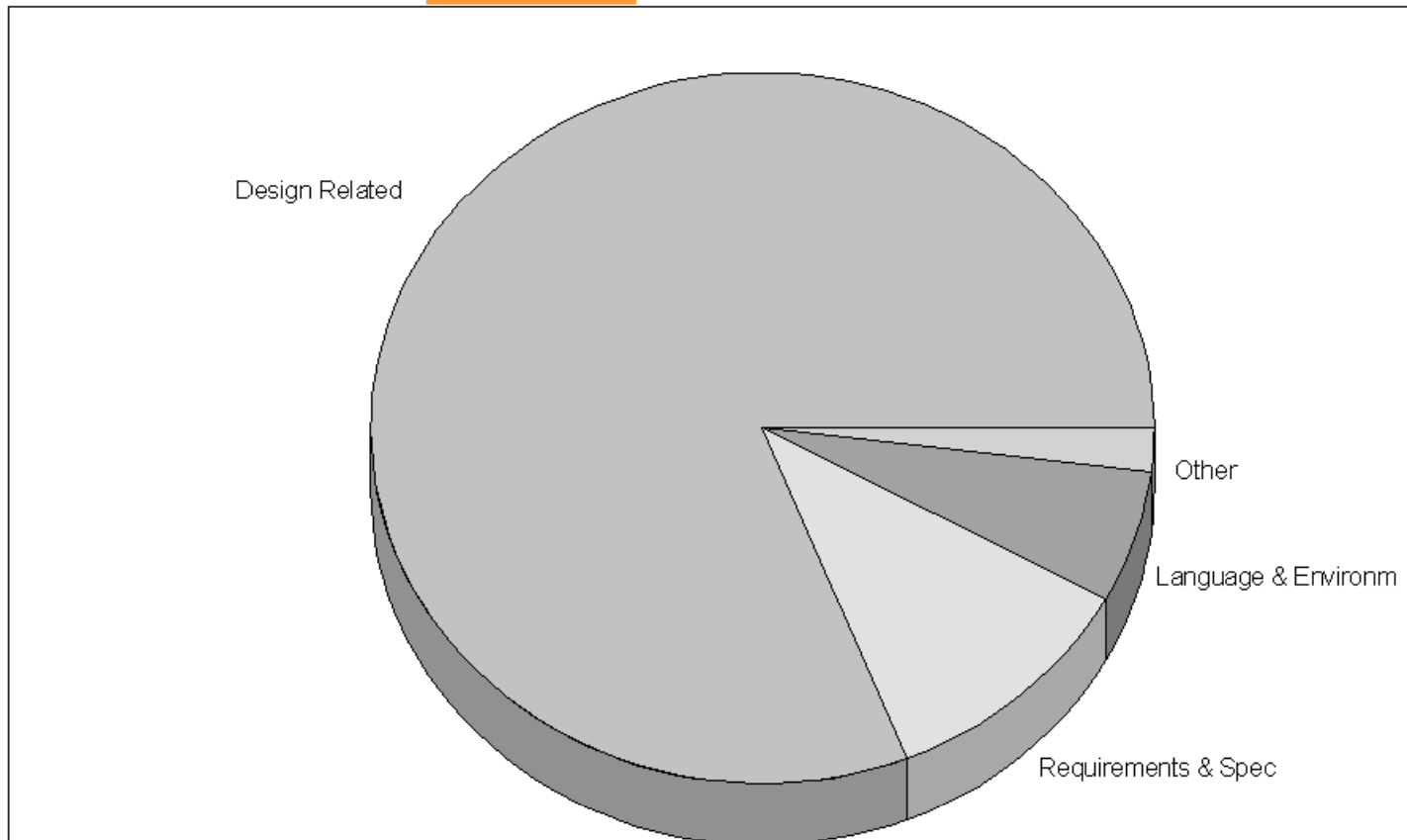


- Software Maintenance treated as A Level Of Effort Activity
- This Means You Can Maintain Software With A Larger Or Smaller Staff Depending On Your Desires / Budget

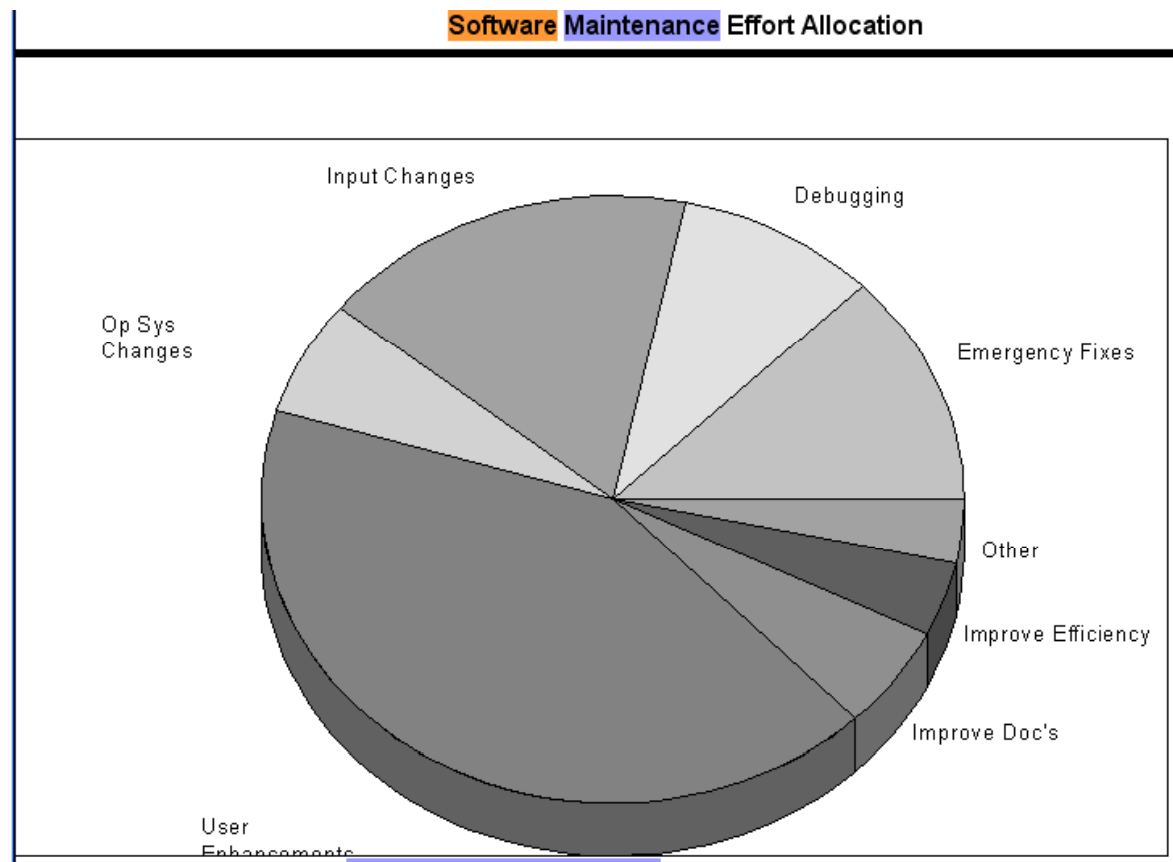
Maintaining A Car	Maintaining Software
High Maintenance: Go By The Book (Regular Oil Changes, Etc.)	<ul style="list-style-type: none">• Fix emergencies• Provide new functionality as needed• Adapt as necessary• Software may not degenerate over time
Nominal Maintenance: Go Partially By The Book (Less Frequent Oil Changes, Etc.)	<ul style="list-style-type: none">• Fix emergencies• Provide some required new functionality• Adapt when there is time
Low Maintenance: Go Slightly By The Book (Add Oil When The Low Oil Light Goes On)	<ul style="list-style-type: none">• Fix only emergencies and small adaptations• Software will degenerate over time

Sources of Software Errors

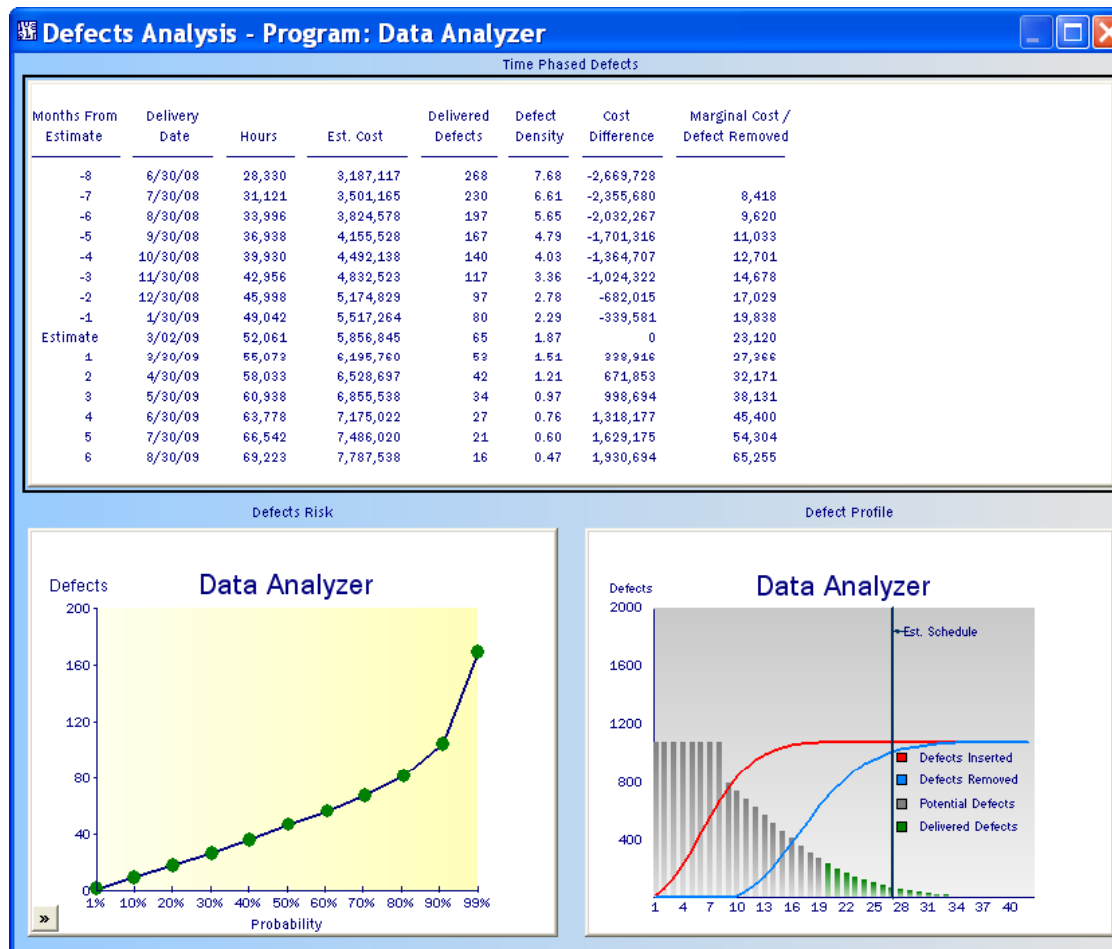
sources of **software errors** (source IEEE transactions)



Allocation of Software Effort (Source IEEE)



Development Defects Analysis Is a Clue to Maintenance Issues



Example Maintenance Metrics



- Defects Inserted per correction
- Defects removed per unit time
- Productivity for block changes
- Maintainability
- Mean time to find the next k faults
- Maintenance backlog
- Increases / decrease on maintenance backlog
- Number of trouble reports opened and closed

More Example Maintenance Metrics



- Mean time until problem closed
- Defects during warranty period
- Mean time to resolution
- Defects by type and severity
- Time to respond to customer reported defects
- **Mccabe & Halstead complexity metrics**

Software Maturity Index

(Example of Metric from IEEE 982 Standard
Dictionary of Measures to Produce Reliable
Software)



M = number of modules in current version

A = number of added modules in current version

C = number of changed modules in current version

D = number of deleted modules in current version
compared to the previous version

$$\text{SMI} = (M - (A + C + D)) / M$$

- when SMI approaches 1.0 the product is stable

Example Effectiveness metrics for Maintenance



- Number of new defects created by fixes
- Number of defect corrections that were not correct
- Number of defects not repaired in promised time (Delinquent)
- **Defect Seepage.. (Customer reported defects during pre-delivery testing)**

Identify the metrics that YOUR organization needs

Product Age / Technology Metrics



- Becomes increasingly difficult to maintain older technology
- Would you recommend a student study COBOL, Ada or PASCAL
- People become less available
- Tools and practices become obsolete

Estimation Lessons Learned



- Measure early
 - Before trouble
 - Early under-budget milestones not necessarily a good sign
 - Programs that skimp on the early, upfront planning; requirements and design work, will most likely be in trouble later.
- Follow known, proven development processes
- Estimation, planning, tracking, controlling – then using the information to do better next time
- EVM shouldn't be used alone
 - Other metrics are necessary to be kept in concert with the EVM metrics in order to keep a project on track
- Recognize that you have a problem

7 Characteristics of a Dysfunctional Software Projects

• (Source: Mike Evans, et al.)
Based on 350 projects:

- Failure to Apply Essential Project Management Practices
- Unwarranted Optimism and Unrealistic Management Expectations
- Failure to Implement Effective Software Processes
- Premature Victory Declarations
- Lack of Program Management Leadership
- Untimely Decision-Making
- Lack of Proactive Risk Management

Summary



- Software projects are manageable
- Basis of management is a viable estimate
- The 10 step process provides a repeatable approach to estimation process
- You can help ensure useful results by adopting a process that is standardized and repeatable
- Measurement is a key part of the estimation process
- Beware of using simple measurements to drive estimates