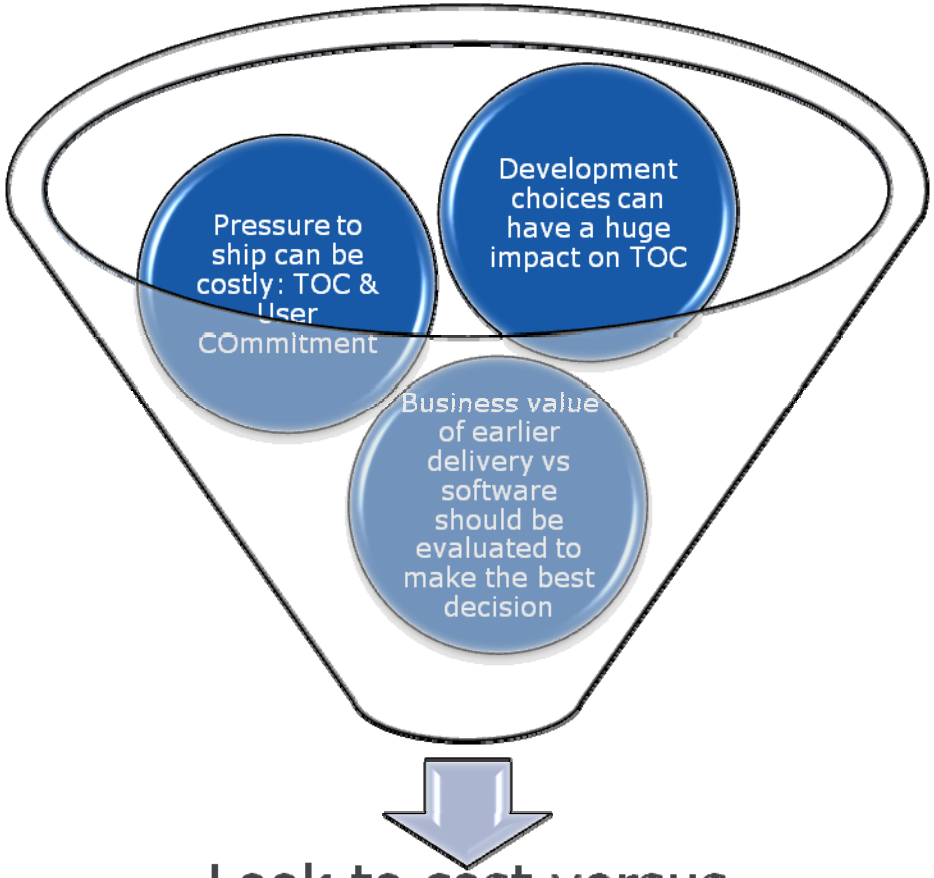
CMMI's Role in Reducing Total Cost of Ownership: Measuring and Managing New and Legacy Software

SEER® by GALORATH

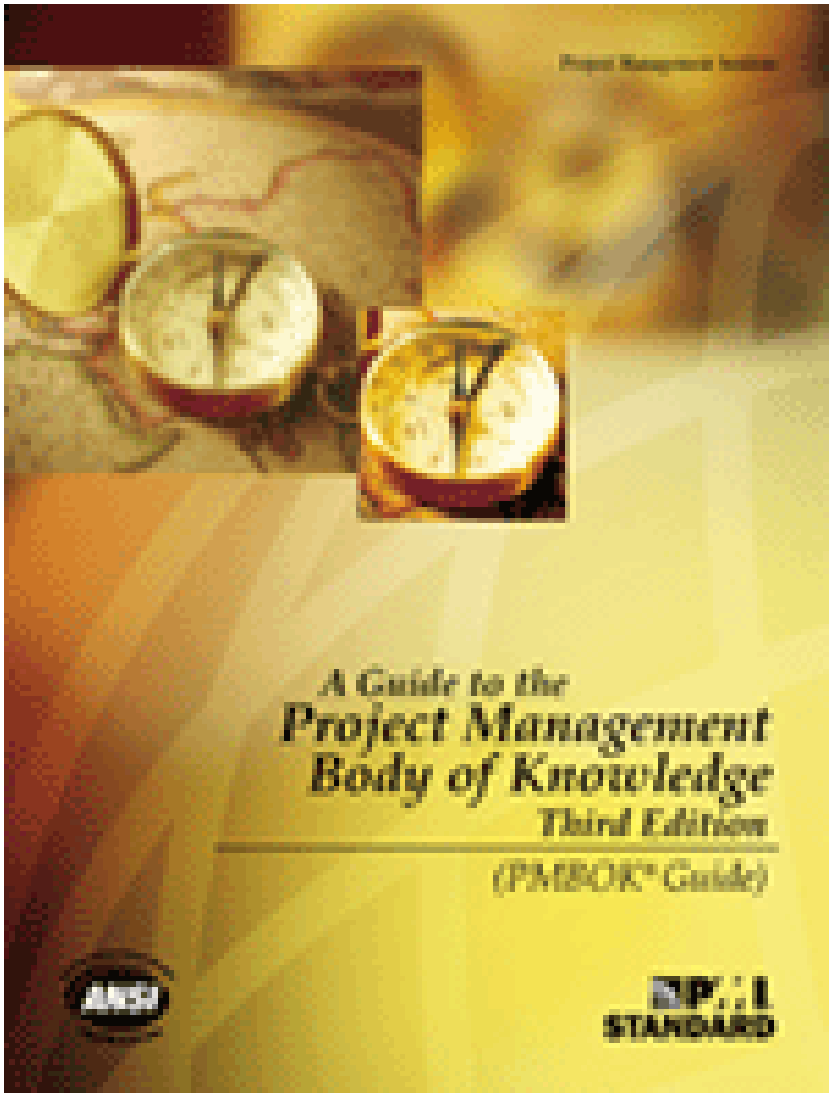
Total Ownership Cost: The Tradeoffs In Summary



Look to cost versus business value to make viable ROI decisions

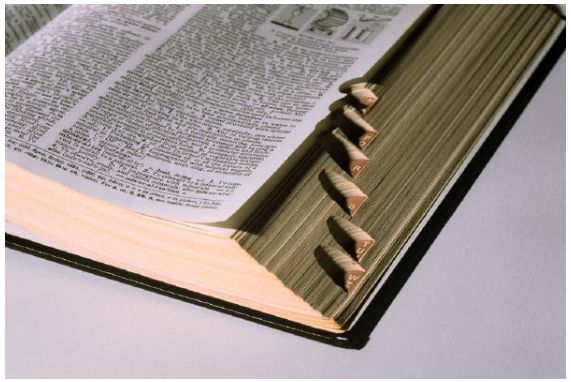
Project Management Defined

- The application of knowledge, skills, tools, and techniques to project activities in order to meet or exceed stakeholder needs and expectations from a project

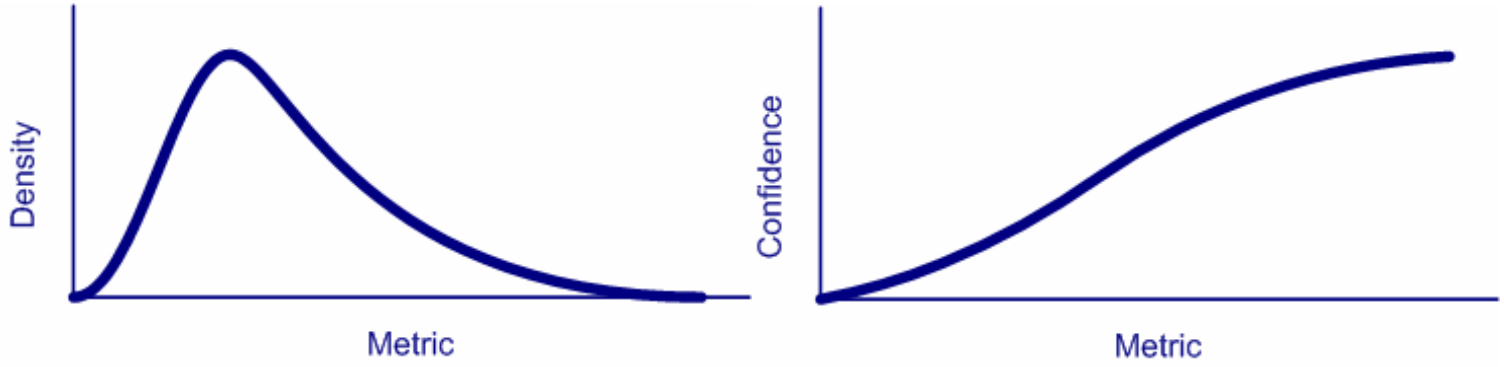


An Estimate Defined

- An *estimate* is the most knowledgeable statement you can make *at a particular point in time* regarding:
  - Effort / Cost
  - Schedule
  - Staffing
  - Risk
  - Reliability



- *A well formed estimate is a distribution*
- *A well structured plan defines probability*



Poor Estimates Effects on Projects

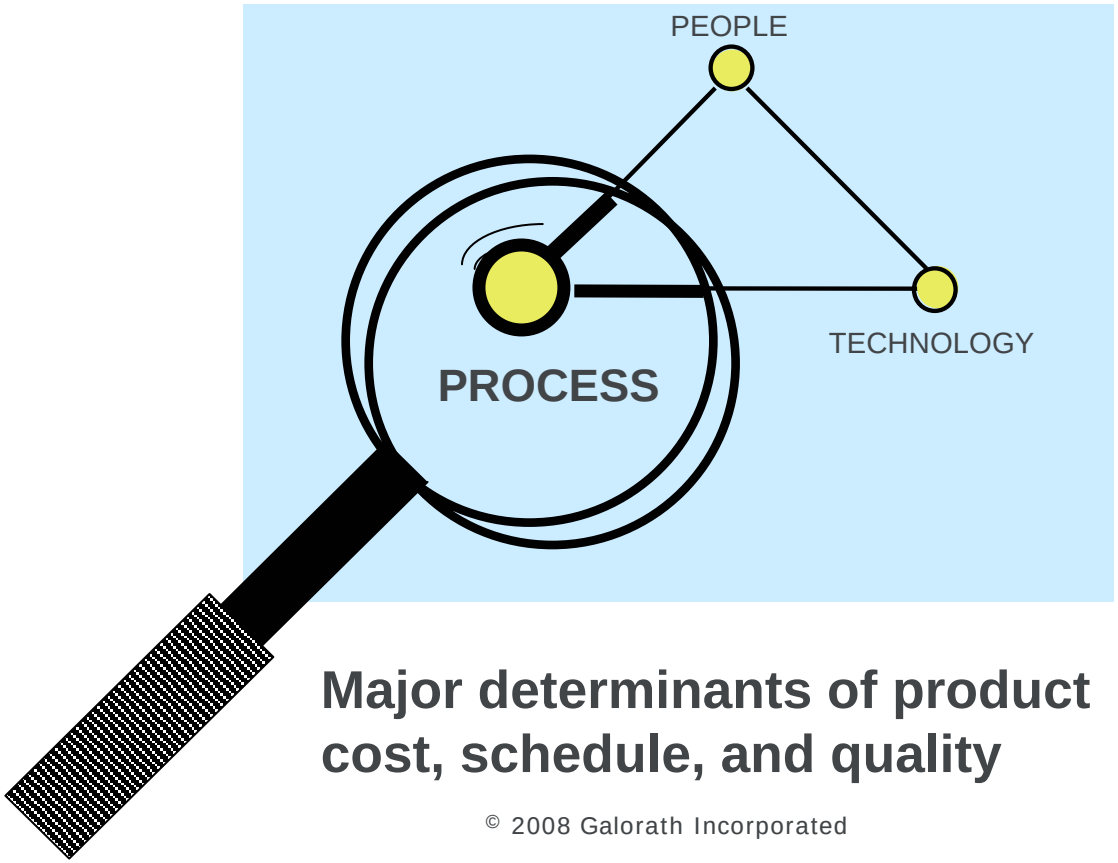
- Inaccurate estimates can reduce project success:
  - Poor implementations
  - Critical processes don't scale
  - Emergency staffing
  - Cost overruns caused by underestimating project needs
- Scope creep from lack of well defined objectives, requirements, & specifications
  - Forever changing project goals
  - Frustration
  - Customer dissatisfaction
  - Cost overruns and missed schedules
  - Project Failures
- Poor estimates & plans are root cause of program risk

However, the most important business decisions about a software project are made at the time of *minimum knowledge* and *maximum uncertainty*

Development, CMMI & Estimation Process

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 best."



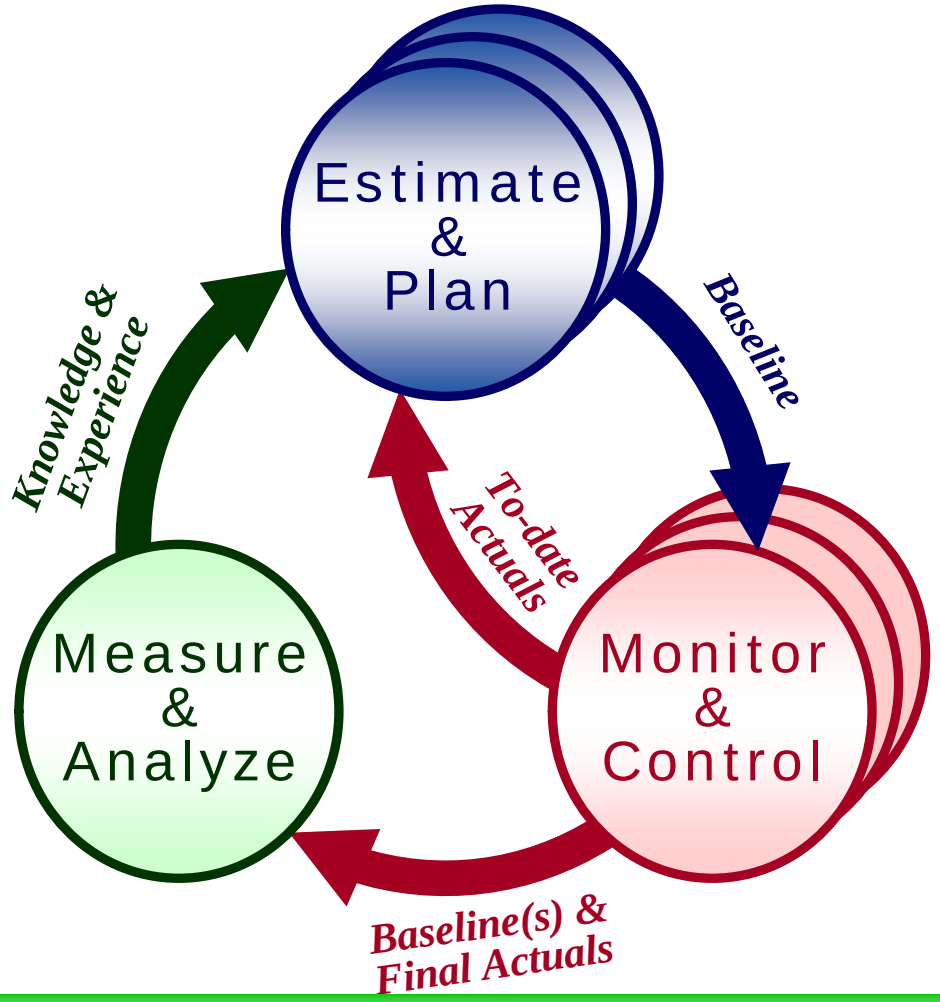
Major determinants of product cost, schedule, and quality

SEER: Software Analysis Tools

A Complete Software Project Management Solution

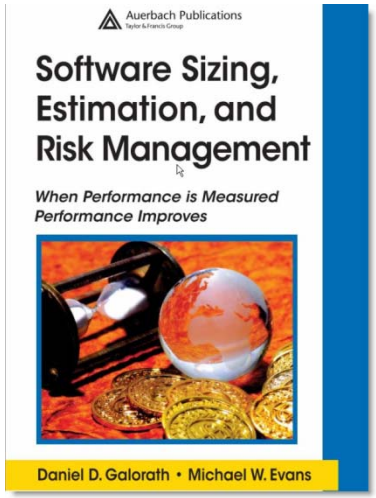
SEER by GALORATH

Quantitative Project Management



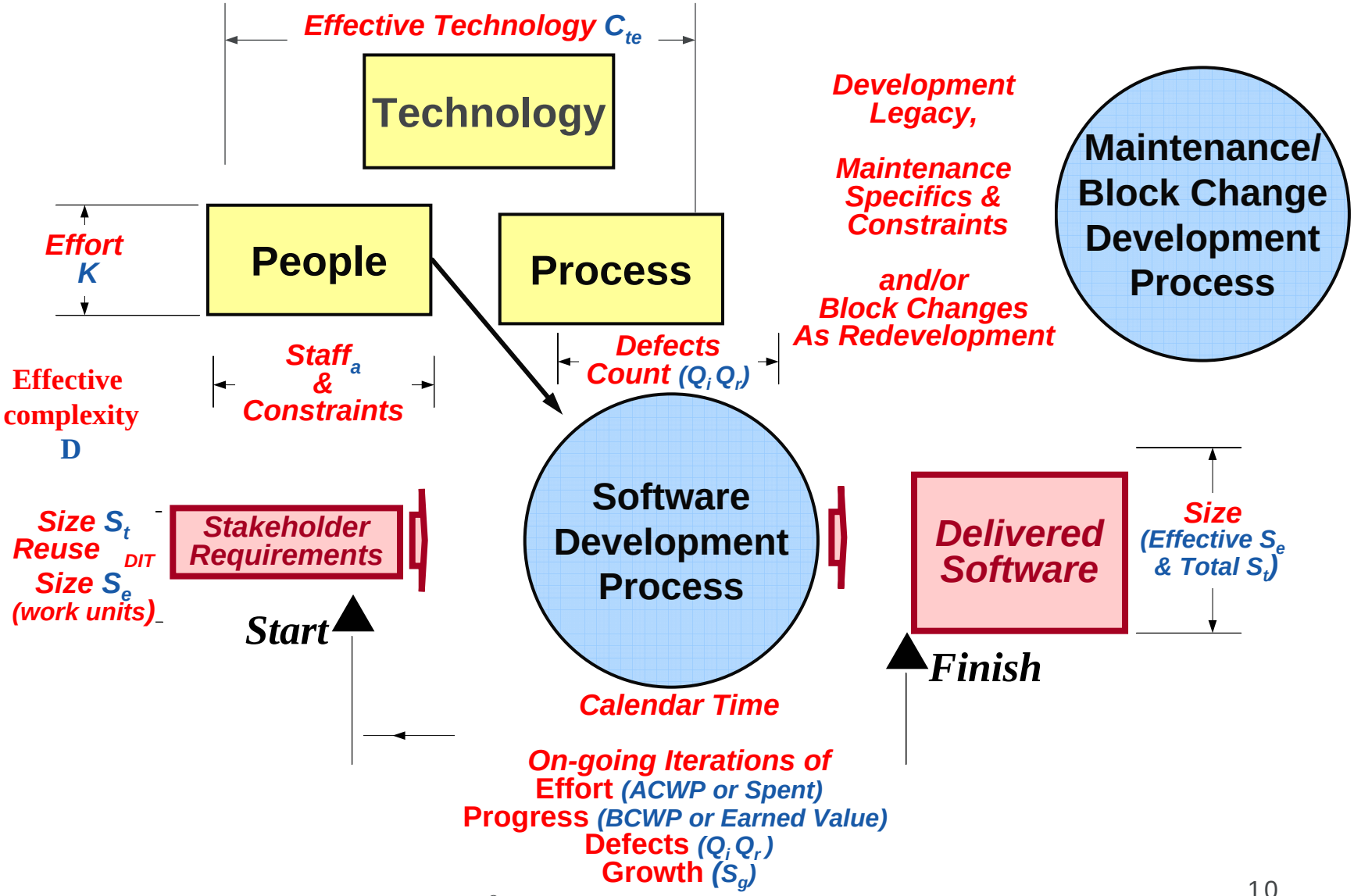
A Foundation of Risk Management

10 Step Software Estimation Process: Consistent Processes Help Reliable Estimates

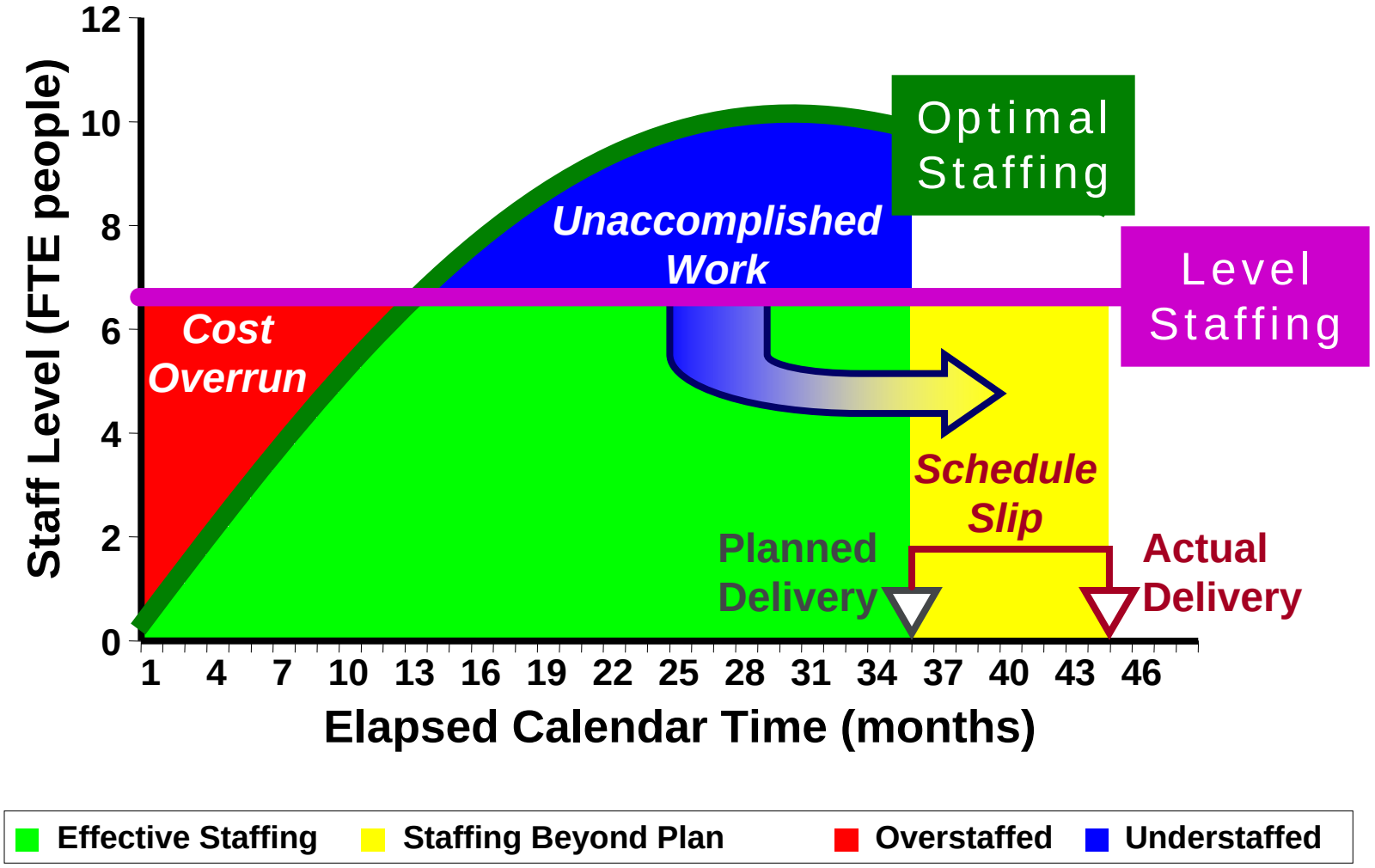


1. Establish Estimate Scope
2. Establish Technical Baseline, Ground Rules, Assumptions
3. Collect Data
4. Estimate and Validate Software Size
5. Prepare Baseline Estimates
6. Review, Verify and Validate Estimate
7. Quantify Risks and Risk Analysis
8. Generate a Project Plan
9. Document Estimate and Lessons Learned
10. Track Project Throughout Development

Software Estimation Basic Model & Associated Metrics

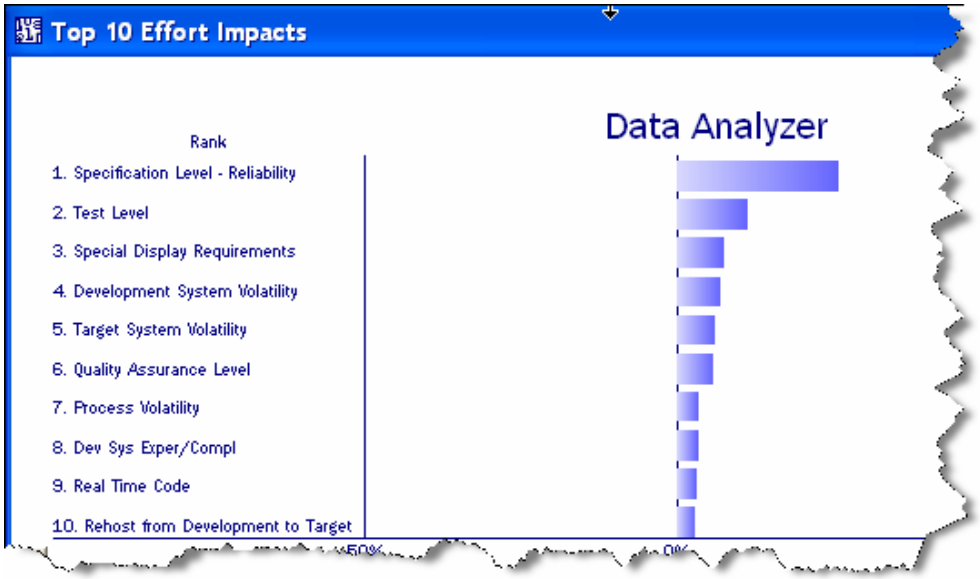


Avoid "Death Marches" and Failed Projects By Applying "Brooks Law"



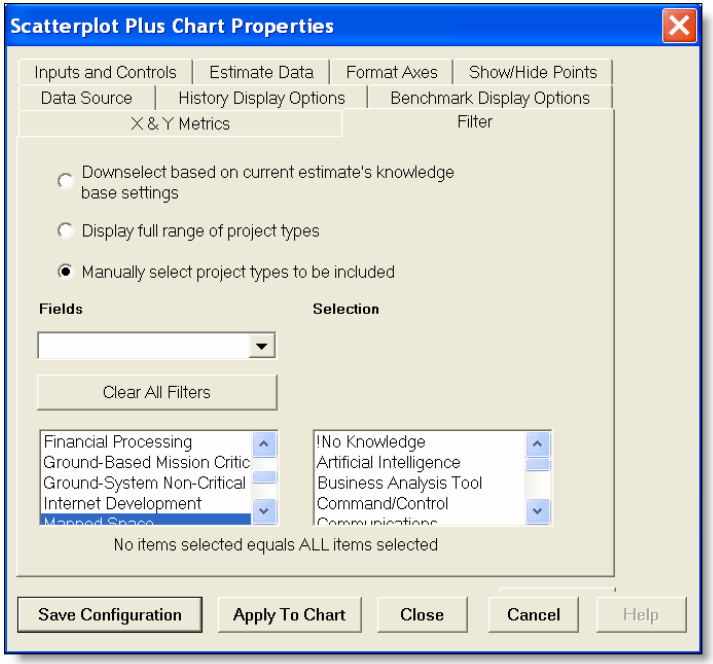
Generate the Estimate

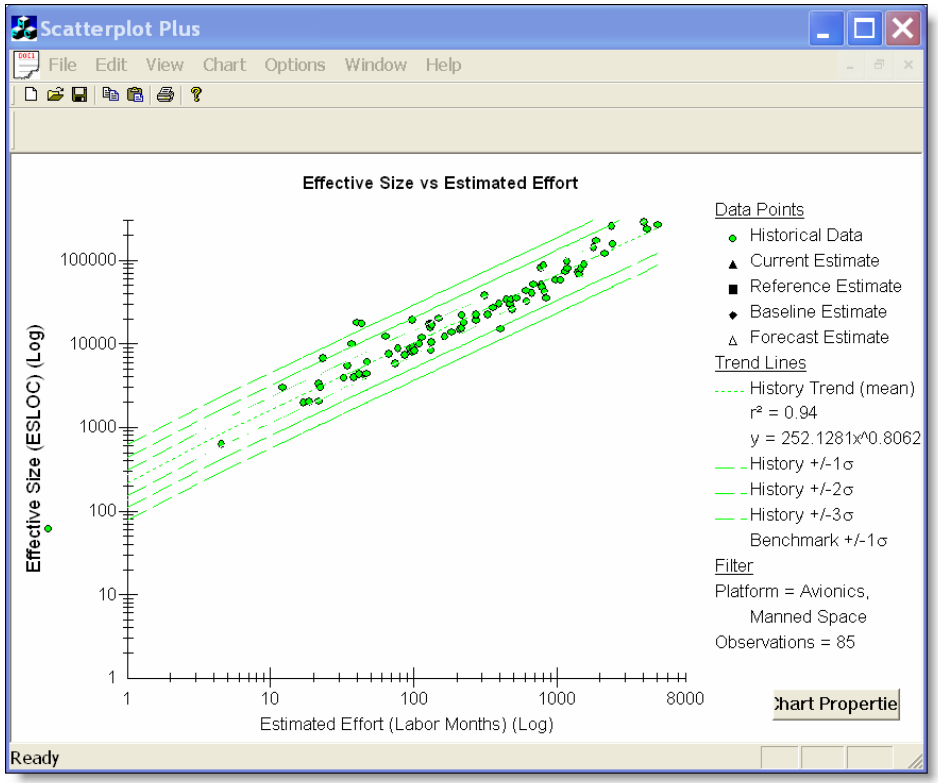
- **Using your chosen methodology and tool, do a first run**
- **Never report preliminary results!**
- **Focus on the inputs**
  - Verify completeness
  - Verify accuracy
- **Focus on the outputs**
  - Sanity check for reasonableness, completeness
- What's driving the estimate?
- Use "fresh eyes" to review
  - Ask a colleague for help
  - Set aside overnight



Compare Parametrics With Metrics and Sanity Checks

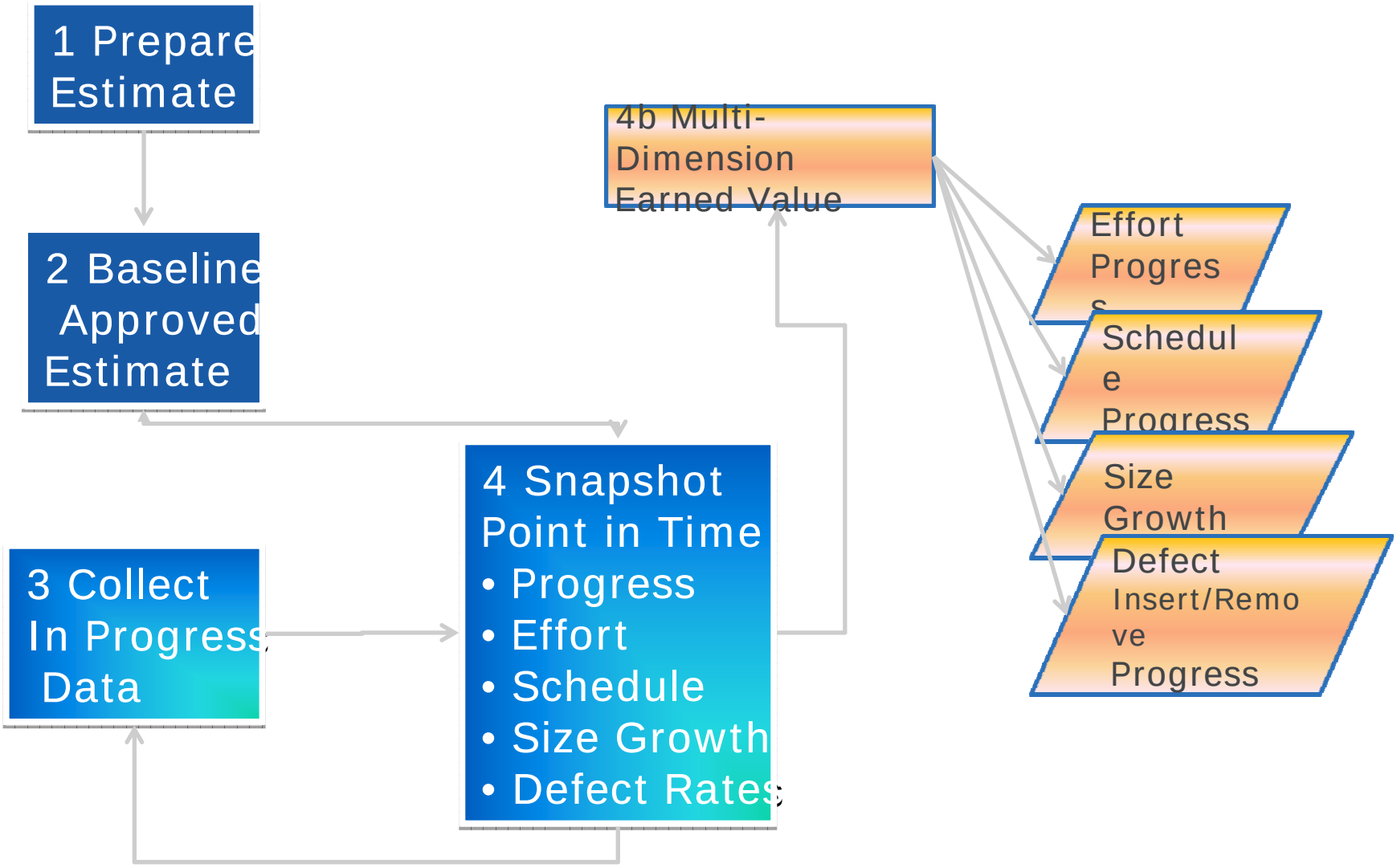
- Works with common repository
- Shows actual data, ranges, and correlations
- Plots estimates and contrasts with data points
- Plots actual data and / or trends





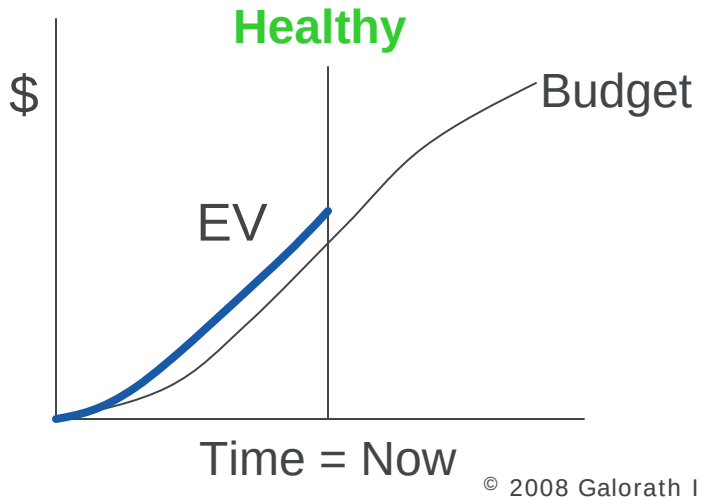
"In God we trust,
all others bring data."
- W. Edwards Deming

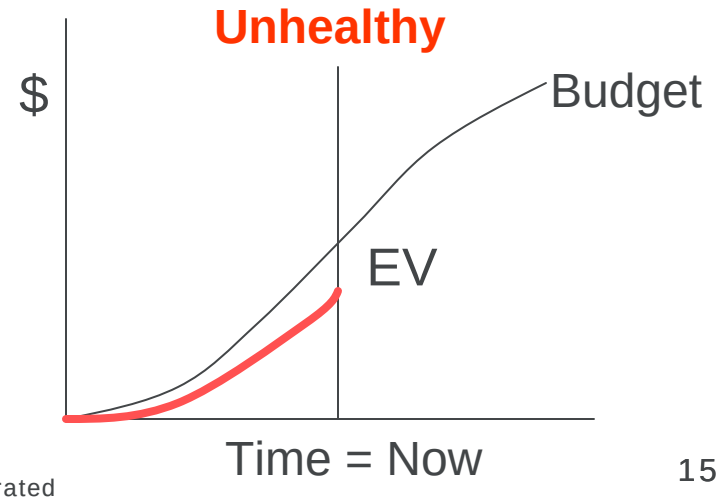
Process For Combining Estimation, Planning & Control, Measurement & Analysis



Use Earned Value TO Quantify Progress Versus Effort

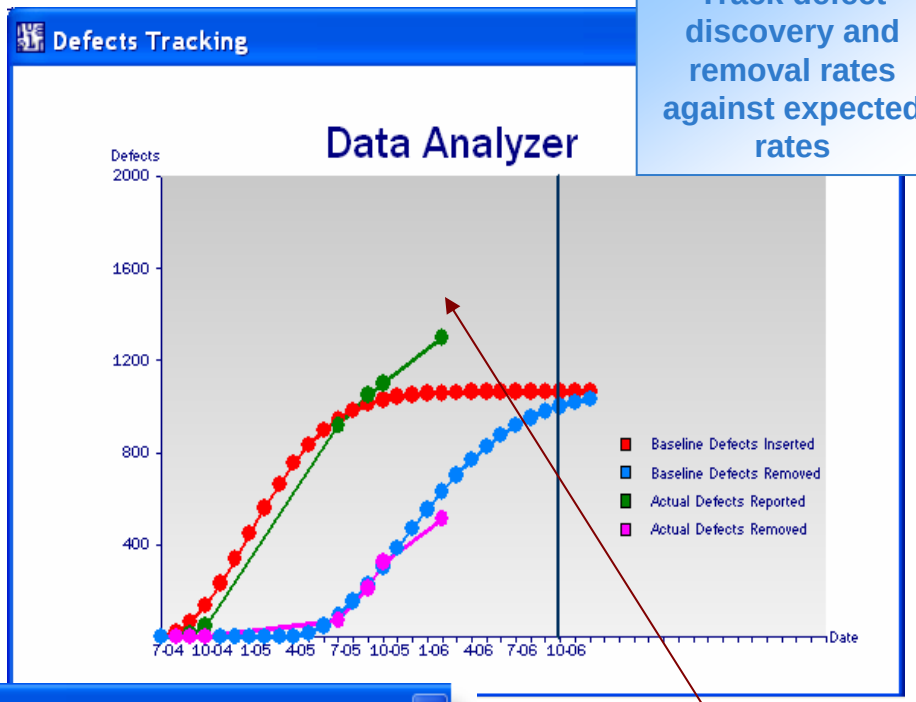
- The main concern of EVM is what has been accomplished in a given time and budget, versus what was planned for the same time and budget
  - A project is generally deemed healthy if what has been accomplished is what was planned, or more
  - A project is deemed unhealthy if accomplishment lags expectations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





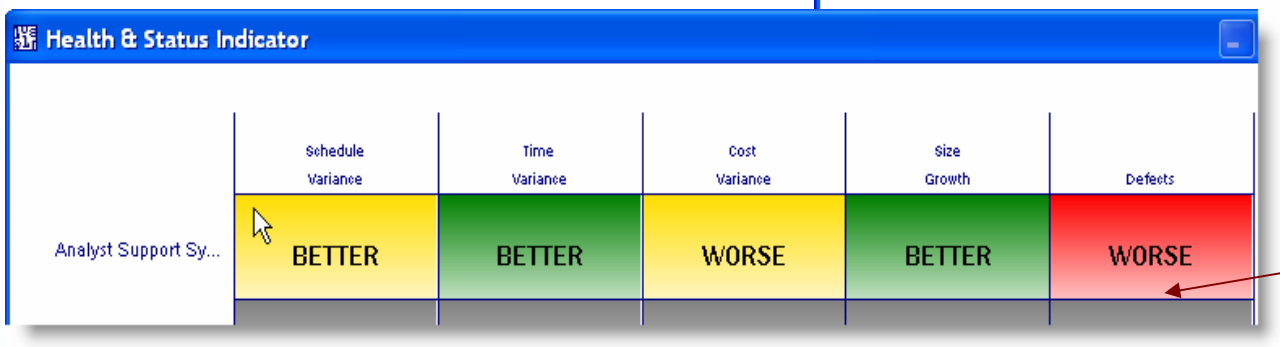
Defects and Growth Impact Software Process

Track defect discovery and removal rates against expected rates



Heath and Status Indicator shows status and trends from the previous snapshot

Thresholds are user definable

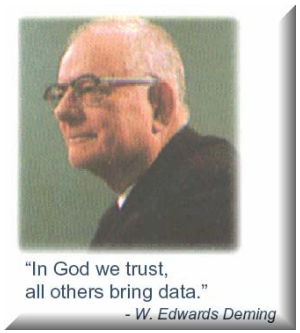


Increased defect reporting rate shows a worsening trend

Measurement During Development & Maintenance

Some Measurement Heroes

- Frederick Taylor: The Principals of Scientific Management 1901 "Let data and facts do the talking"
- W. Edwards Demming: "In God We Trust… All Others Bring Data"
- Frederick Brooks: "There is an incremental person when added to a software project that makes it take longer"
- Ed Yourdon: "Avoiding Death Marches in Software Projects"
- Steven Covey: "Sharpen the Saw" Focus on improvement
- Eli Goldratt: Improvements should increase profit Effectiveness



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

Core Metric: Value Provided By Software

- Concept: Spend where you obtain the most value
  - Value = savings to company or additional revenue due to the software
- Software Fails to add value much too often
  - Users enamored with concept
  - Concept deployed
  - Little to no value contributed to company…
  - Many reasons… often no changes in business rules
- MRP is a classic example of software hyped but which did not provide value

Many Organizations May Not Be Mature Enough To Consider Value From the Software Team

Theory of Constraints Questions Regarding Value (Source Goldratt)

1. What is the main power of the technology?

2. What limitation does it diminish?

3. What rules helped us to accommodate the limitation?

4. What rules should we use now?

Measurement Job Not Over When Development Is Complete Maintenance GQM (Adapted from Mitre)

Goal	Question	Metric(s)
Maximize Customer Satisfaction	How many problems affect the customer?	1. Current Change Backlog 2. Software Reliability
Minimize cost	How much does a software maintenance delivery cost?	
	How are costs allocated	Cost per activity
	What kinds of changes are being made?	Number of changes by type
	How much effort is expended per change	Staff hours expended by change /type
Minimize Schedule	How difficult is the delivery?	Complexity Assessment Software Maintainability Computer resource Utilization
	Are we meeting delivery schedules?	Percentage of On-Time Deliveries

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

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

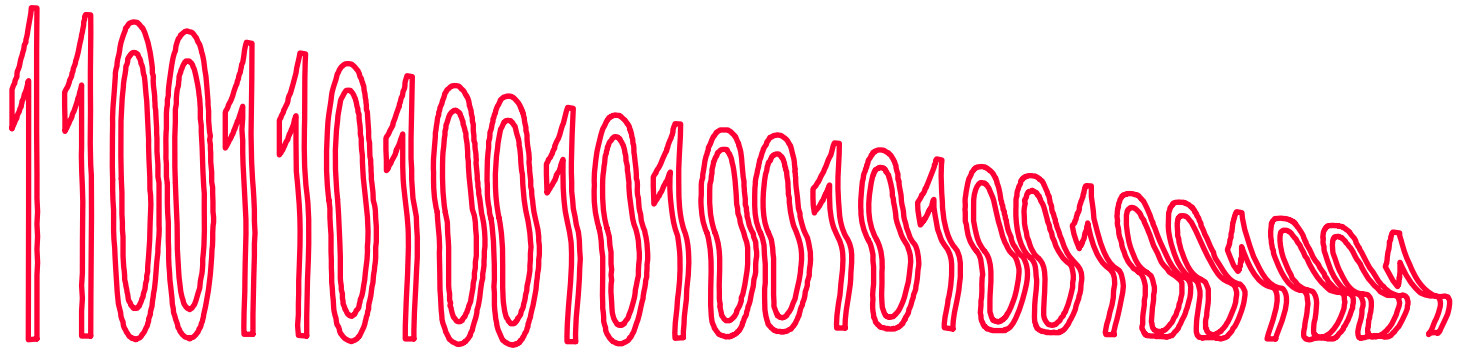
- Becomes increasingly difficult to maintain older technology
- Would you recommend a student study COBOL, Ada or PASCAL

- People become less available
- Tools an practices become obsolete

Maintenance & Total Ownership Costs

Maintenance Defined

- Dictionary: "The work of keeping something in proper order"
- Software maintenance is different from hardware maintenance because:
  - Software doesn't physically wear out, but...
  - Software often gets less useful with age and...
  - It may be delivered with undiscovered flaws
- Software maintenance is: "The process of modifying existing operational software while leaving its primary functions intact."



Development Quality Impacts Maintenance

http://www.bcs.org/server.php?show=ConWebDoc.3063

- IEEE Std 1919-1993: Software maintenance defines maintenance as:

Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment

  - States that maintenance starts after delivery

- **Largest costs of software production occur after the 'development phase' is complete**
  - Maintenance up to 75 per cent of the total ownership cost.
- **Maintenance costs generally not result of poor requirements or design**
- **Even if "right the first time" change is inevitable:**
  - Political decisions (e.g. introduction of a new tax).
  - Hardware related changes.
  - Operating system upgrades over time.
  - Competition - new features to be added.
  - System almost instantly complying to outdated requirements
- **Construction may not affect function, but greatly affects future maintainability**
- **Maintainability goals during development can significantly reduce total ownership costs**

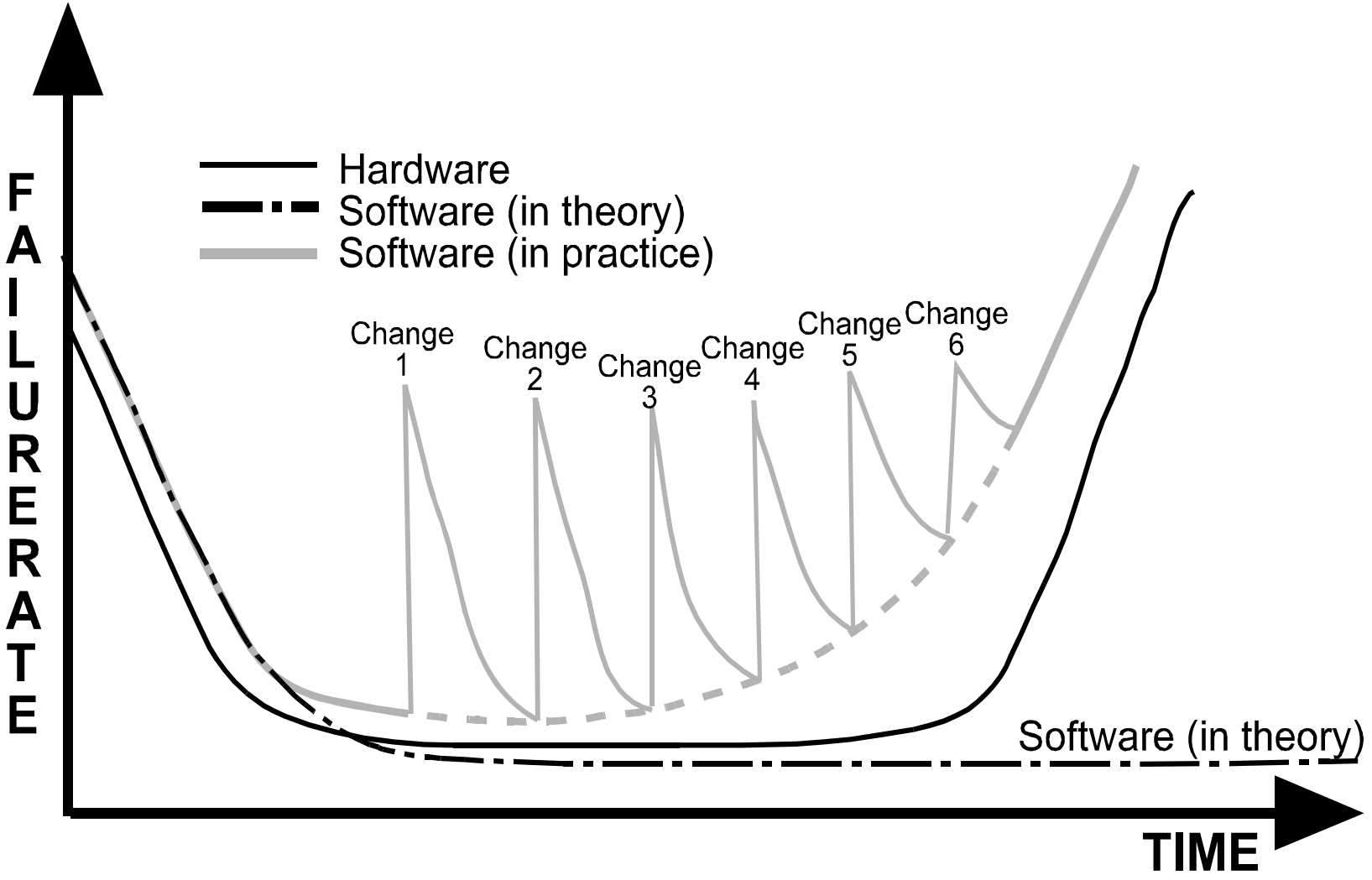
Why Total Lifecycle Measurement matters

- 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

Maintenance Dissected

- Maintenance typically 50% + of the total software workload:
  - Highly dependent on maintenance rigor & operational “life expectancy”
  - Reducing maintenance costs can reduce life cycle costs significantly
- Generally includes sustaining engineering & new function development:
  - Corrective changes (fixing bugs)
  - Adapting to new requirements (OS upgrade, new processor)
  - Perfecting or improving existing functions (improve speed, performance)
  - Enhancing application with (minor) new functions (new feature)
- For every new software product we develop, we get one more to maintain -- for ?? years

Software Maintenance Is Often A Series of Block Changes



Software Maintenance Goals, Questions, Metrics Adapted from Mitre 1997

Goal	Question	Metric(s)
Maximize Customer Satisfaction	How many problems affect the customer?	1. Current Change Backlog 2. Software Reliability
Minimize cost	How much does a software maintenance delivery cost?	
	How are costs allocated	Cost per activity
	What kinds of changes are being made?	Number of changes by type
	How much effort is expended per change	Staff hours expended by change /type
Minimize Schedule	How difficult is the delivery?	Complexity Assessment Software Maintainability Computer resource Utilization
	Are we meeting delivery schedules?	Percentage of On-Time Deliveries

Software Maintenance Critical Success Factors (Source IEEE)

Why Maintenance Is Hard

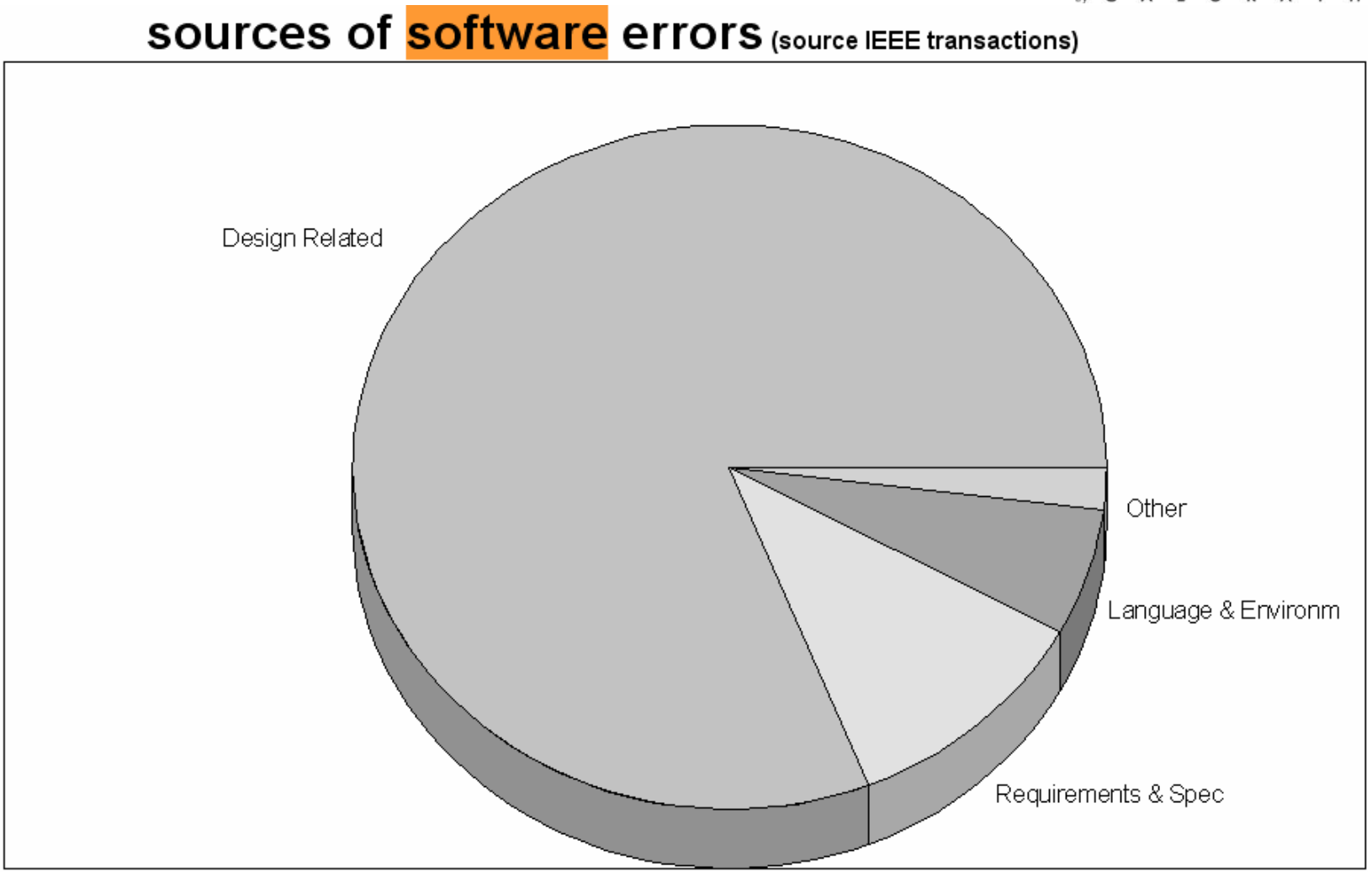
- May not have had maintenance as a goal
- System may not have been fully tested
- Documentation may be inadequate
- Maintenance staff may be inexperienced
- The tendency to produce quick & dirty fixes
- Process or language experience may have left a mess
- The "but I only changed 1 line syndrome"

Why Software Maintenance Costing Is 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

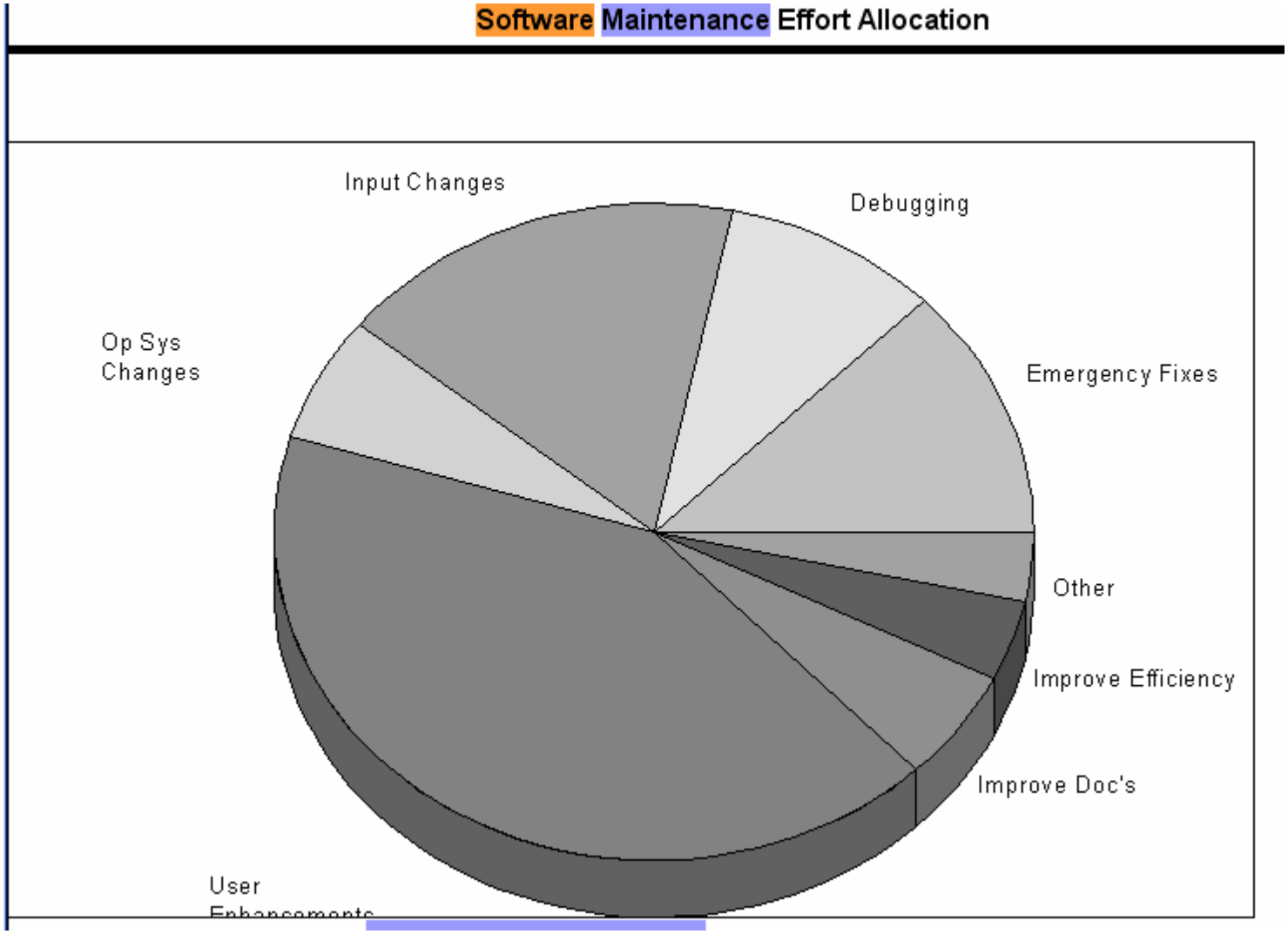
Maintaining A Car	Maintaining Software
High Maintenance: Go By The Book (Regular Oil Changes, Etc.)	• Fix emergencies • Provide new functionality as needed • Adapt as necessary • Software may not degenerate over time
Nominal Maintenance: Go Partially By The Book (Less Frequent Oil Changes, Etc.	• Fix 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	• Fix only emergencies and small adaptations • Software will degenerate over time

Sources of Software Errors

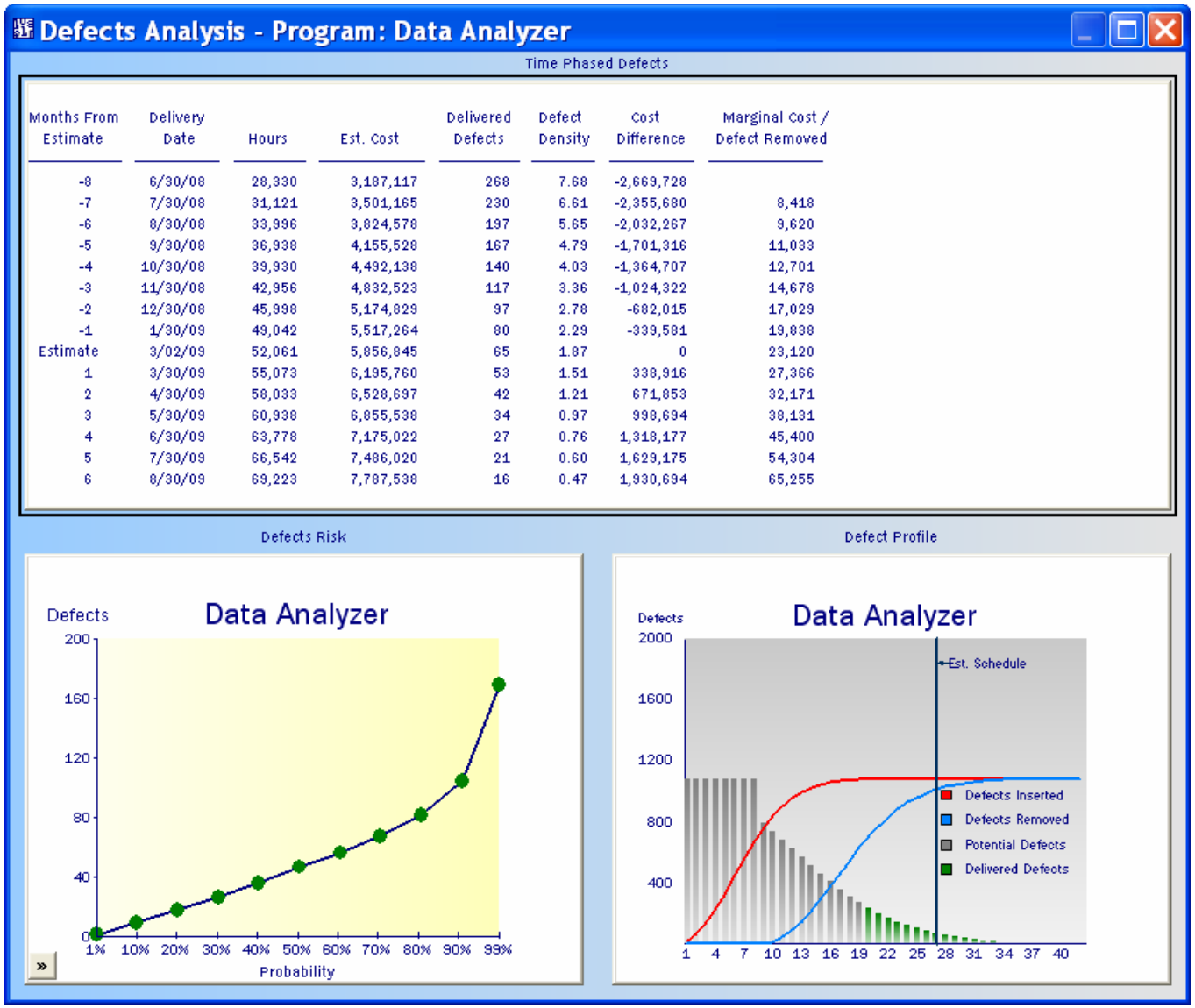


Software Maintenance Effort Allocation

Allocation of Softwaree Effort Source IEEE

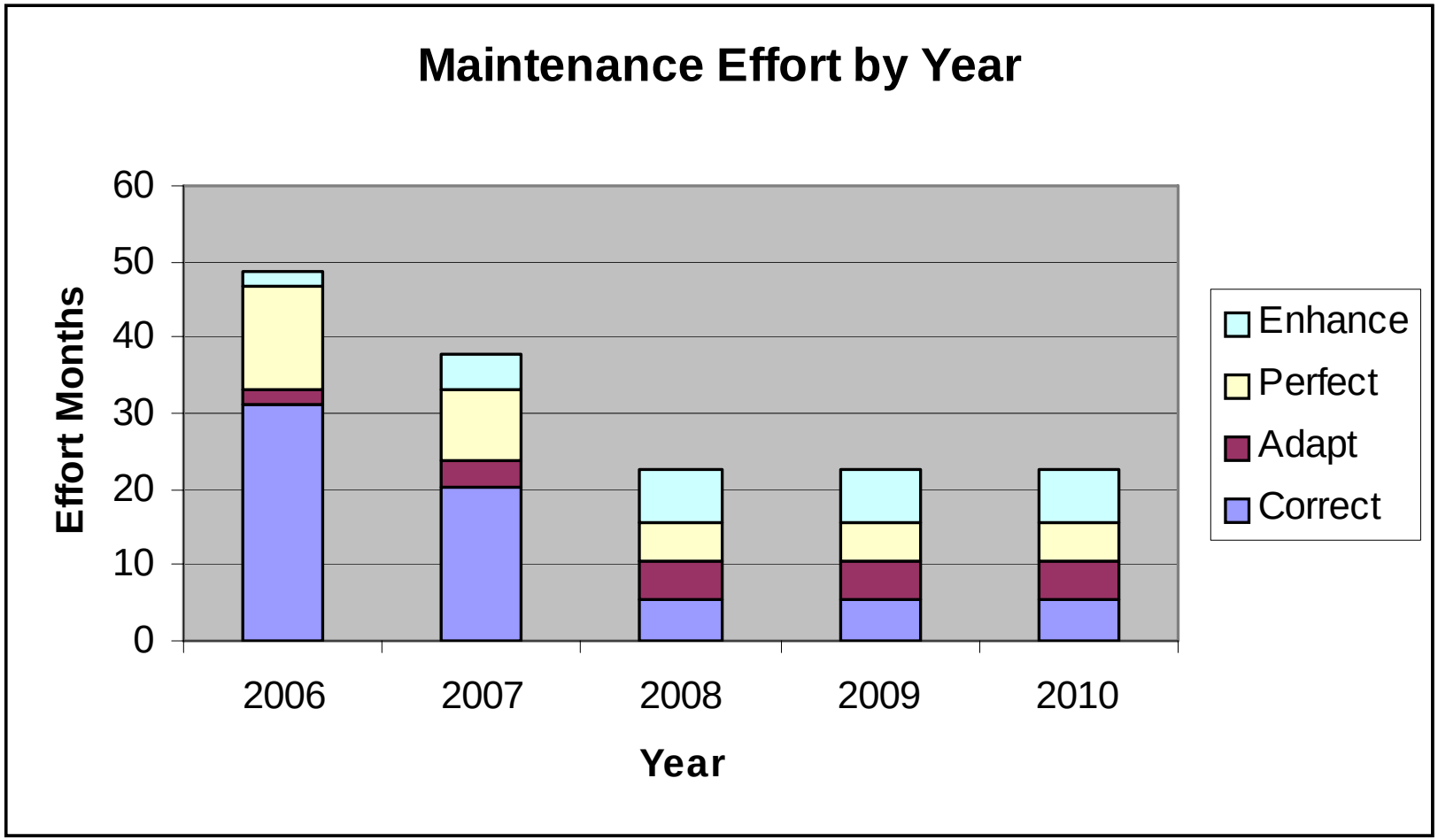


Development Defects Analysis Is a Clue to Maintenance Issues



Months From Estimate	Delivery Date	Hours	Est. Cost	Delivered Defects	Defect Density	Cost Difference	Marginal Cost / Defect Removed
-8	6/30/08	28,330	3,187,117	268	7.68	-2,669,728	
-7	7/30/08	31,121	3,501,165	230	6.61	-2,355,680	8,418
-6	8/30/08	33,996	3,824,578	197	5.65	-2,032,267	9,620
-5	9/30/08	36,938	4,155,528	167	4.79	-1,701,316	11,033
-4	10/30/08	39,930	4,492,138	140	4.03	-1,364,707	12,701
-3	11/30/08	42,956	4,832,523	117	3.36	-1,024,322	14,678
-2	12/30/08	45,998	5,174,829	97	2.78	-682,015	17,029
-1	1/30/09	49,042	5,517,264	80	2.29	-339,581	19,838
Estimate	3/02/09	52,061	5,856,845	65	1.87	0	23,120
1	3/30/09	55,073	6,195,760	53	1.51	338,916	27,366
2	4/30/09	58,033	6,528,697	42	1.21	671,853	32,171
3	5/30/09	60,938	6,855,538	34	0.97	998,694	38,131
4	6/30/09	63,778	7,175,022	27	0.76	1,318,177	45,400
5	7/30/09	66,542	7,486,020	21	0.60	1,629,175	54,304
6	8/30/09	69,223	7,787,538	16	0.47	1,930,694	65,255

Typical Maintenance Staffing

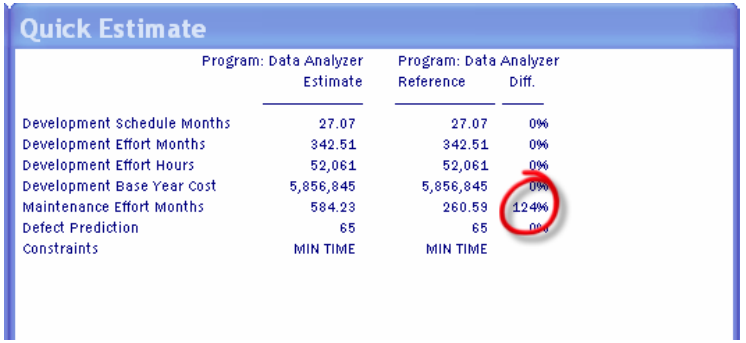


Maintenance Growth Over Life

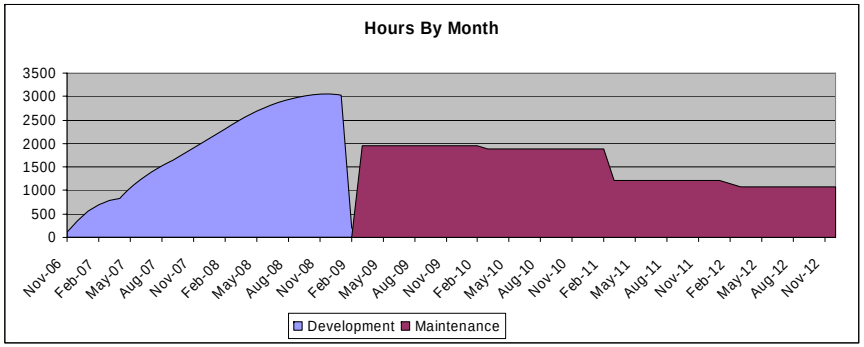
- Anticipated size growth from the point immediately after the software is turned over to maintenance to the end of the maintenance cycle
- May include additions of new functionality

Rating	Description
100%	Major updates adding many new functions
35%	Moderate updates adding some new functions
20%	Minor updates & enhancements to existing functions
5%	No updates expected, some minor enhancements
0%	Sustaining engineering only

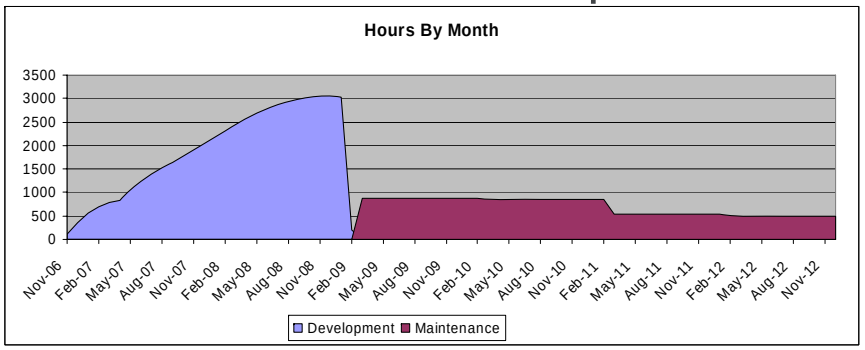
0 vs 100% growth over 5 years



100% growth over 5 years
Initial 27 mo development



0% growth over 5 years
Initial 27 mo development

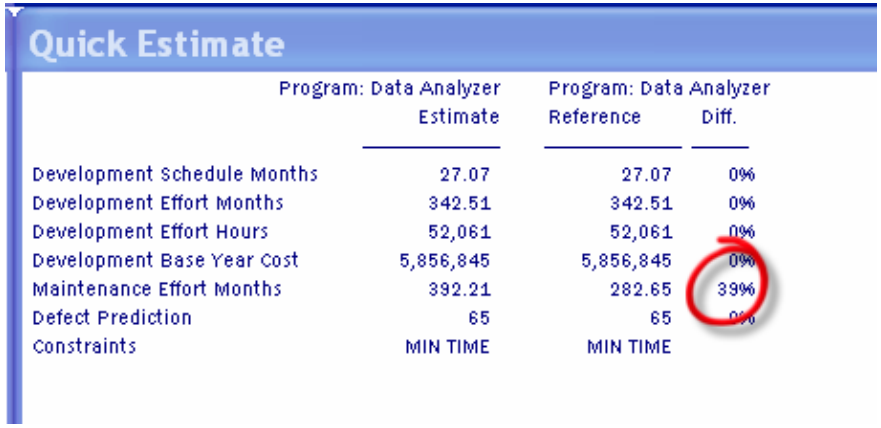


Annual Change Rate

- Average percent of the software impacted by software maintenance and sustaining engineering per year
- May include changes, revalidation, reverse engineering, redocumentation, minor changes for new hardware, or recertification

Rating	Description
35%	Very High
15%	High
11%	Nominal
5%	Low
0%	Very Low

50% vs 0 annual change over 5 years



Quick Estimate

	Program: Data Analyzer Estimate	Program: Data Analyzer Reference	Diff.
Development Schedule Months	27.07	27.07	0%
Development Effort Months	342.51	342.51	0%
Development Effort Hours	52,061	52,061	0%
Development Base Year Cost	5,856,845	5,856,845	0%
Maintenance Effort Months	392.21	282.65	39%
Defect Prediction	65	65	0%
Constraints	MIN TIME	MIN TIME	

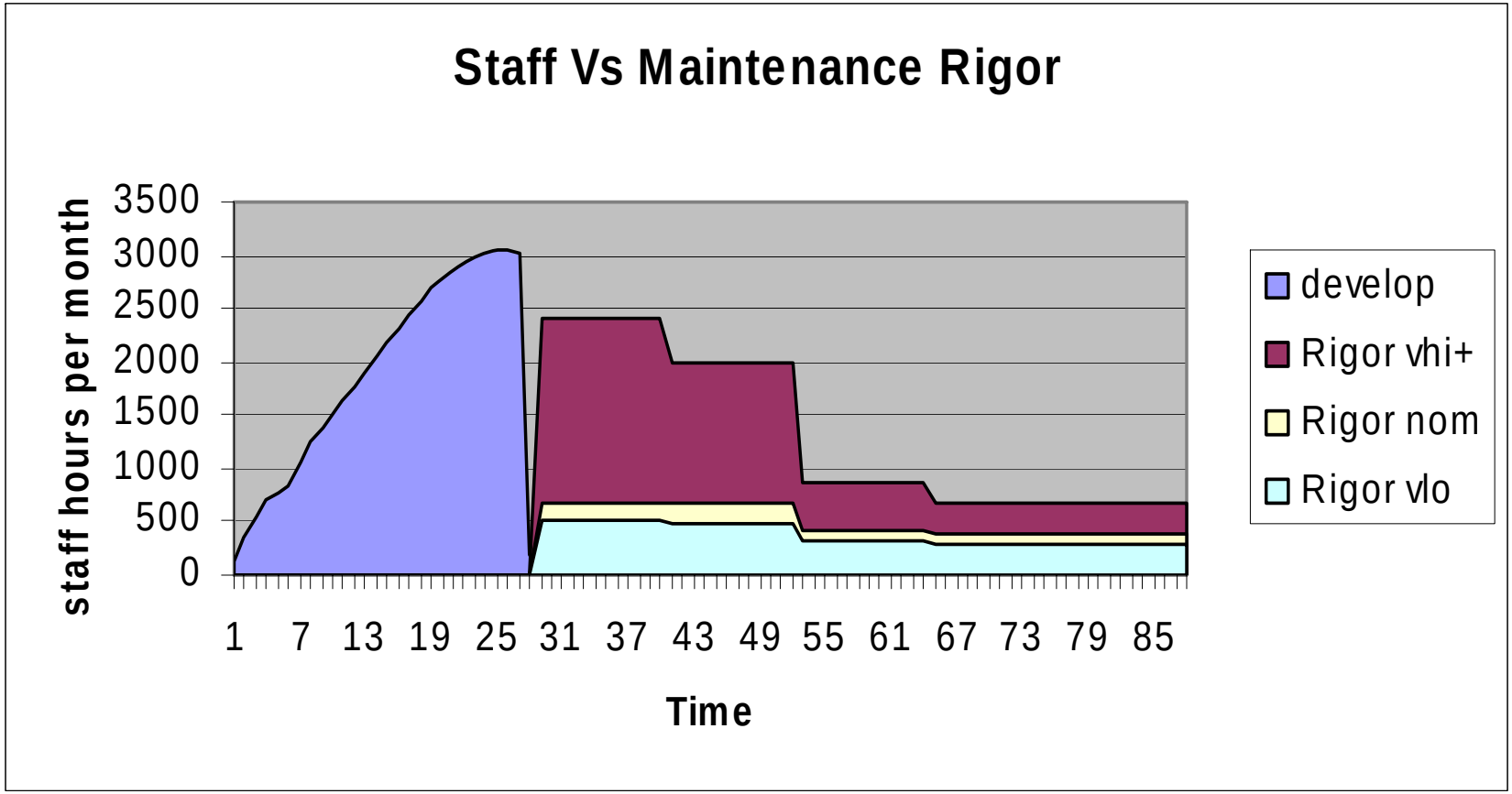
Maintenance Level (Rigor)

- Rates the thoroughness with which maintenance activities will be performed

Rating	Description
Very High +	Full complete maintenance estimate (From Raleigh Curve)
Very High	**Thorough maintenance** for all types of software maintenance activities, including regular documentation updates. Well planned in both the long and short term with frequent reviews of priorities. Dedicated maintenance staff
High	**Complete maintenance** including maintenance planning and priority review. Software documentation is updated on a semi-regular basis. Software will not degenerate over time
Nominal	**Average maintenance activity.** Short term planning and prioritization of maintenance activity. Documentation is updated less than once a year (change pages and addenda). Software will become less useful over time
Low	**Basic maintenance, reactive to emergencies and problems** as they arise. No planning of maintenance activity. Documentation is updated only with change pages and addenda. Software will degenerate over time
Very Low	**Bare bones maintenance. Non-dedicated team** doing emergency fixes. Little to no documentation update. Software will degenerate rapidly. May also represent sustaining engineering effort of a delivered incremental build during development of subsequent builds

Key Driver: Maintenance Level (Rigor)

Most Projects Spend Low During Maintenance



Percent to be Maintained

- Enter the percent of the total code that will be maintained
- If maintenance will be shared with another organization, enter only the portion to be included in this estimate
- If software cannot be changed, do not include it in the percent to be maintained (e.g. non updateable embedded processors)

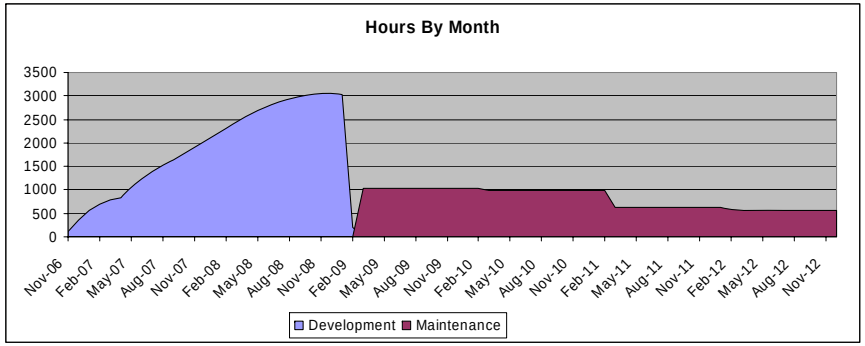
Rating	Description
100%	Maintenance for entire WBS element will be included in the estimate
15%	Maintenance effort is outside the estimate, but some maintenance integration effort is required
0%	No maintenance effort is included in the estimate

Steady State Maintenance Only

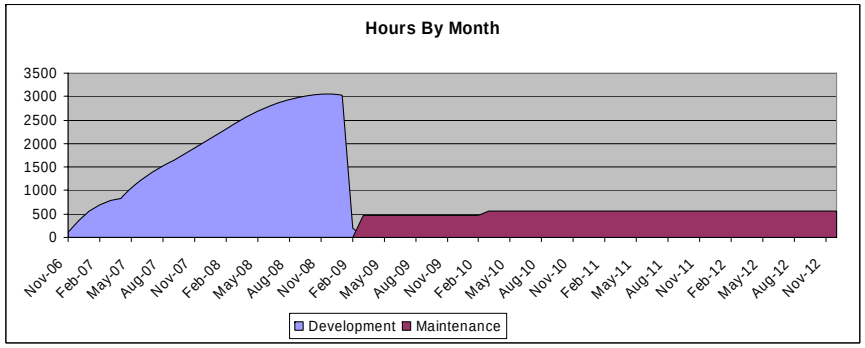
- Indicates whether maintenance profile should be effort-based, or fixed staff.

Rating	Description
YES	Estimate maintenance with a fixed annual staff level. (For Contracts where level of effort will not allow rampdown or planned initial block change will be added to effort)
NO	Estimate maintenance with additional effort in the first years.

no

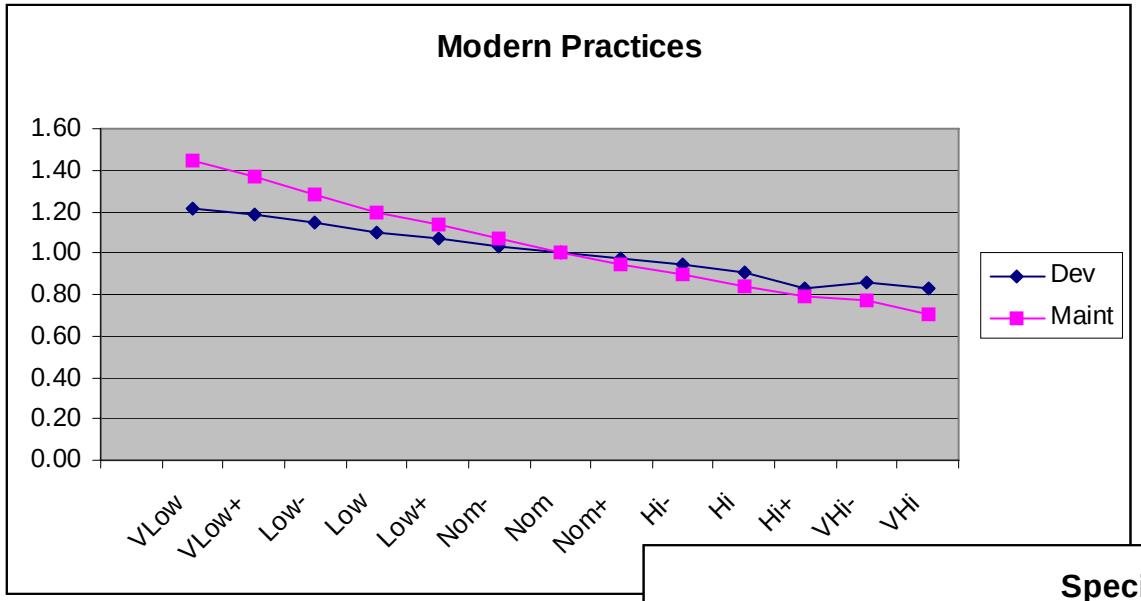


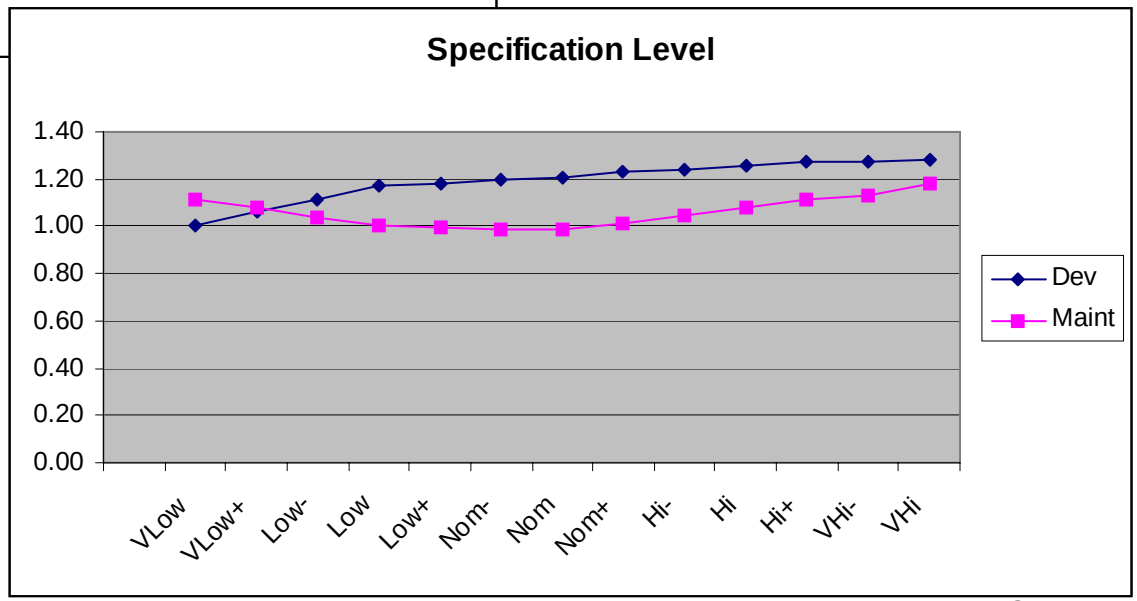
yes



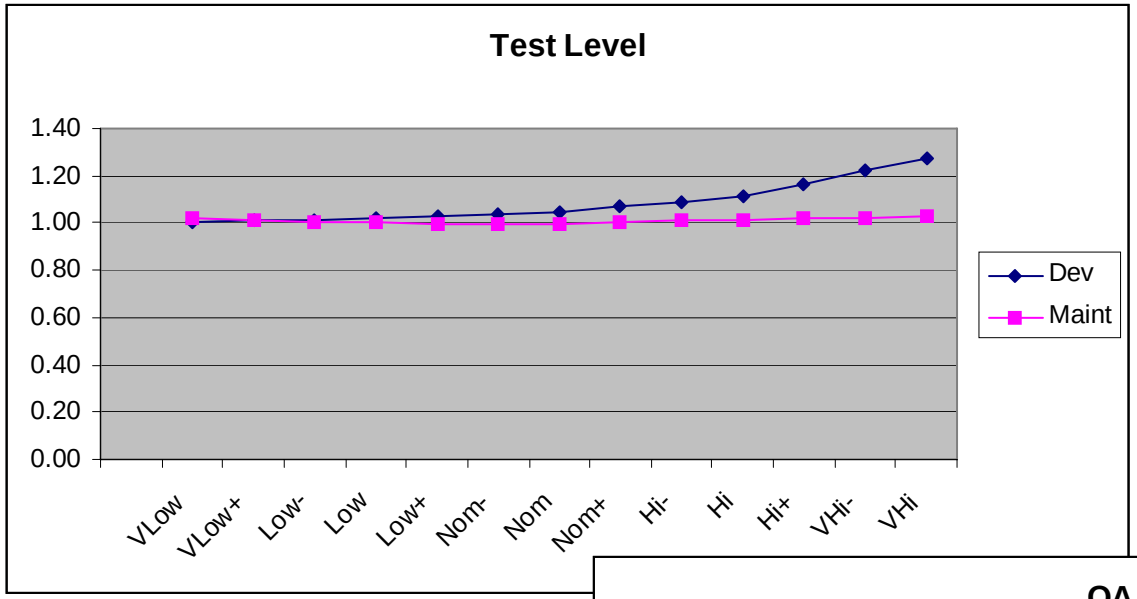
Some Trades…. Costs During Development and Maintenance Impacts

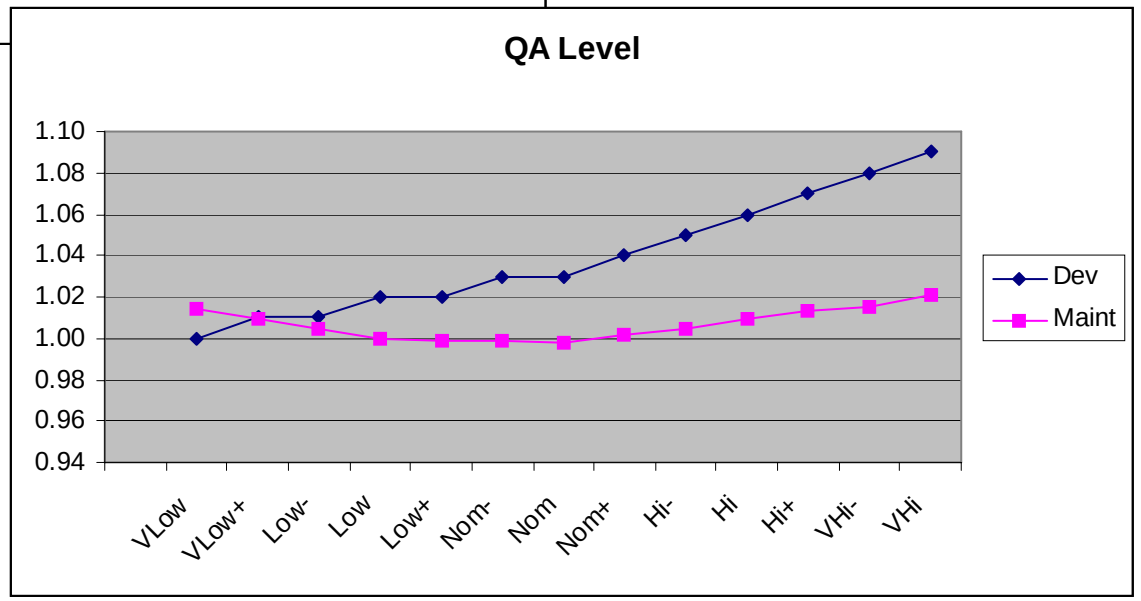
Parameter Sensitivity Development Vs Maintenance - 1



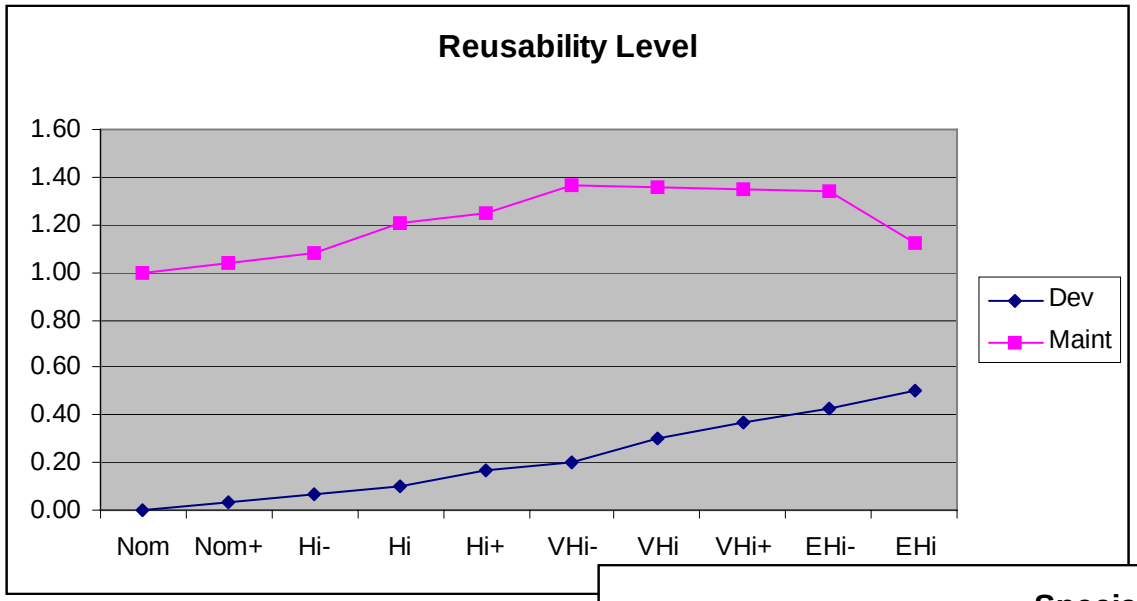


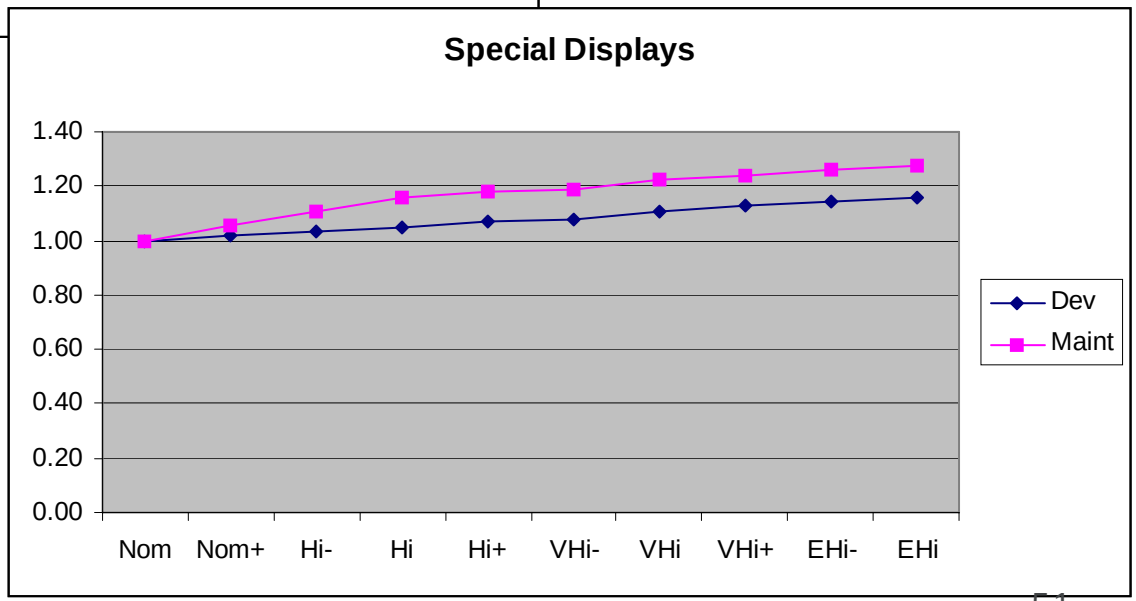
Parameter Sensitivity Development Vs Maintenance - 2





Parameter Sensitivity Development Vs Maintenance - 3





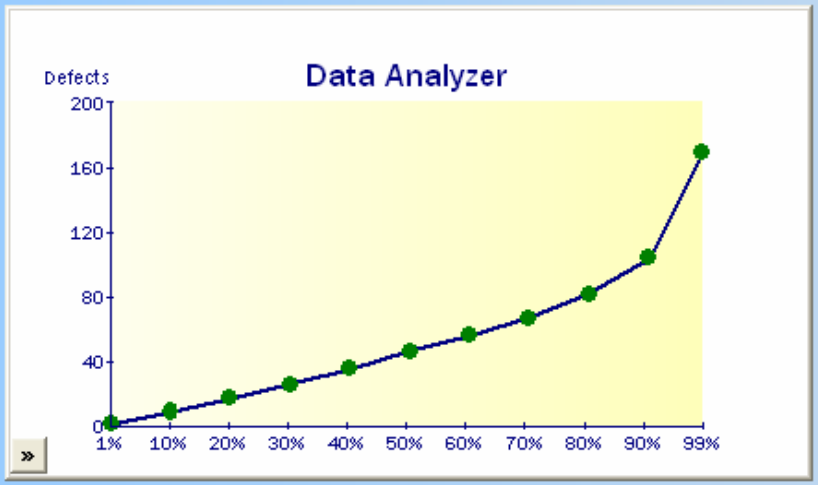
Defects Can Be Reduced By Further Development Testing but Not Eliminated

Defects Analysis - Program: Data Analyzer

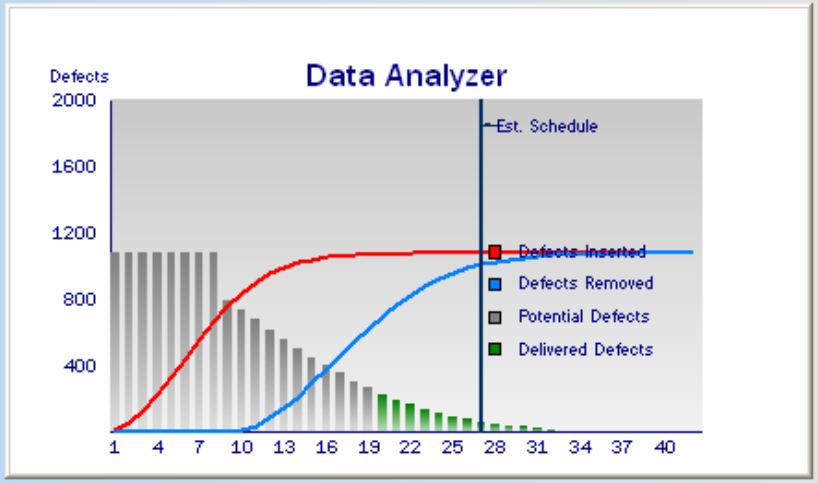
Time Phased Defects

Months From Estimate	Delivery Date	Hours	Est. Cost	Delivered Defects	Defect Density	Cost Difference	Marginal Cost / Defect Removed
-8	7/01/08	28,330	3,187,117	268	7.68	-2,669,728	
-7	7/31/08	31,121	3,501,165	230	6.61	-2,355,680	8,418
-6	8/31/08	33,996	3,824,578	197	5.65	-2,032,267	9,620
-5	10/01/08	36,938	4,155,528	167	4.79	-1,701,316	11,033
-4	10/31/08	39,930	4,492,138	140	4.03	-1,364,707	12,701
-3	12/01/08	42,956	4,832,523	117	3.36	-1,024,322	14,678
-2	12/31/08	45,998	5,174,829	97	2.78	-682,015	17,029
-1	1/31/09	49,042	5,517,264	80	2.29	-339,581	19,838
Estimate	3/03/09	52,061	5,856,845	65	1.87	0	23,120
1	3/31/09	55,073	6,195,760	53	1.51	338,916	27,366
2	5/01/09	58,033	6,528,697	42	1.21	671,853	32,171
3	5/31/09	60,938	6,855,538	34	0.97	998,694	38,131
4	7/01/09	63,778	7,175,022	27	0.76	1,318,177	45,400

Defects Risk



Defect Profile



Conclusions

- Software Maintenance can be 75% of total ownership costs
- Development decisions, processes and tools can impact maintenance costs
- Generally even a perfect delivered system quickly needs upgrade
- While software maintenance is often treated as a level of effort activity there are consequences:
  - Quality, functionality and reliability
- Software total ownership costs and risks can be estimated using SEER for Software