estimate

estimate • analyze • plan • control

Use Case Estimating 2009

Bob Hunt Vice President, Services
(703-201-0651)
bhunt@Galorath.com
June 2009

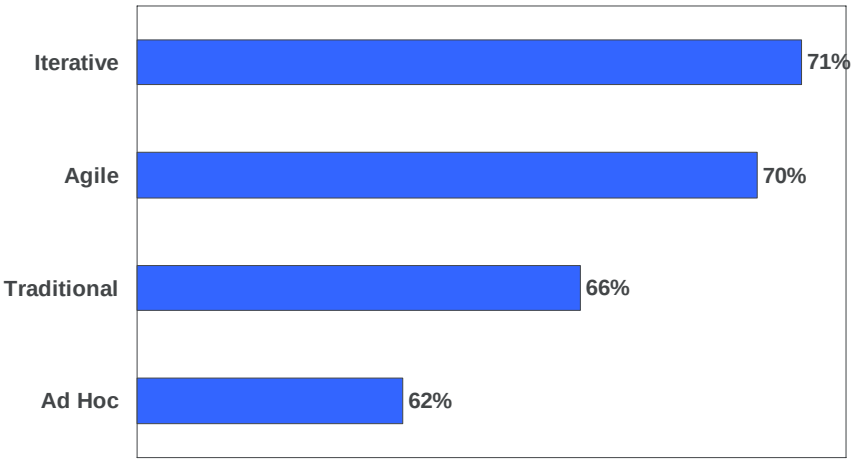
SEER by GALORATH

The Good News

- **U.S. Improves Success Rate Of Federal IT Projects**
  - The OMB now identifies 489 items as "high risk" due to factors such as complexity, scope, or level of importance, down 19% from the 601 projects that were on the list in February.
  - The U.S. government currently has about one-fifth fewer IT projects at risk of failing or having other serious flaws compared to earlier this year, according to an updated report released today by the Office of Management and Budget.

Marianne Kolbasuk McGee; InformationWeek; April 17, 2008

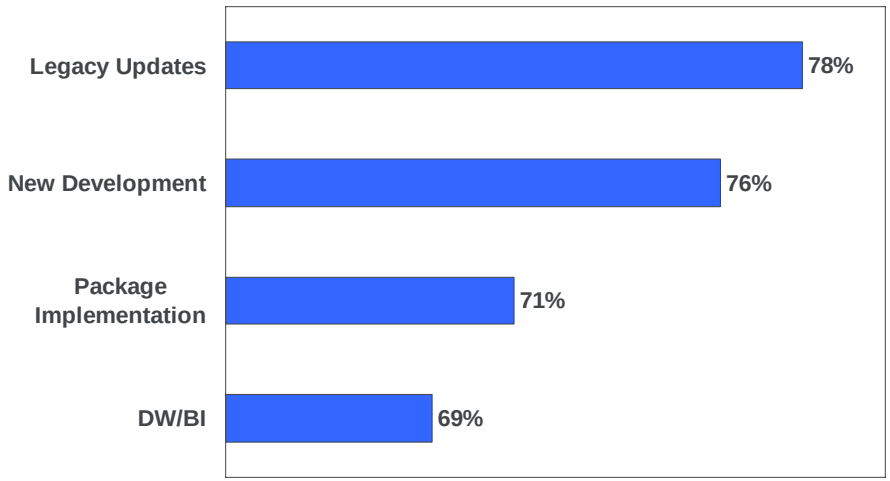
Software Development Project Success Survey 2008*



Project Success Rate by Paradigm

*Copyright 2009 Scott W. Ambler; www.ambysoft.com/surveys/

Success Rate by Project Type



*Copyright 2009 Scott W. Ambler; www.ambysoft.com/surveys/

SOFTWARE SIZING AND COSTING POINTS TO CONSIDER

SEER by GALORATH

- New/different software development methodologies **do not imply that**
  - Old methods must have been bad or
  - Everything learned in the past is obsolete
- C++ without object oriented design techniques is not building an object oriented system
- Use Cases are a "Synthetic" rather than an "Analytic" Technique – a building up rather than a breaking down
- Lines of Code and Function Points perform just as well on Object-Oriented systems as other methods
- **But, the closer we get to the actual artifacts the developers produce, the simpler it may be to obtain definitions of size / effort that can easily be understood and measured – this is the driver behind the current emphasis on Use Case Points**

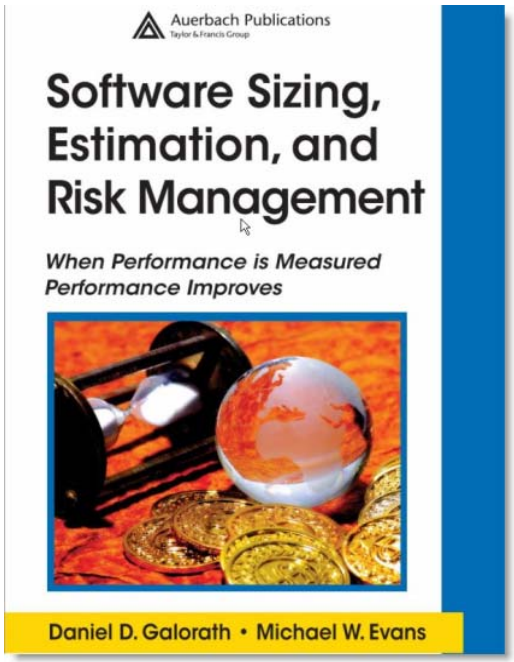
BRIEF HISTORY OF SOFTWARE DEVELOPMENT AND SIZING CONCEPTS & TECHNIQUES

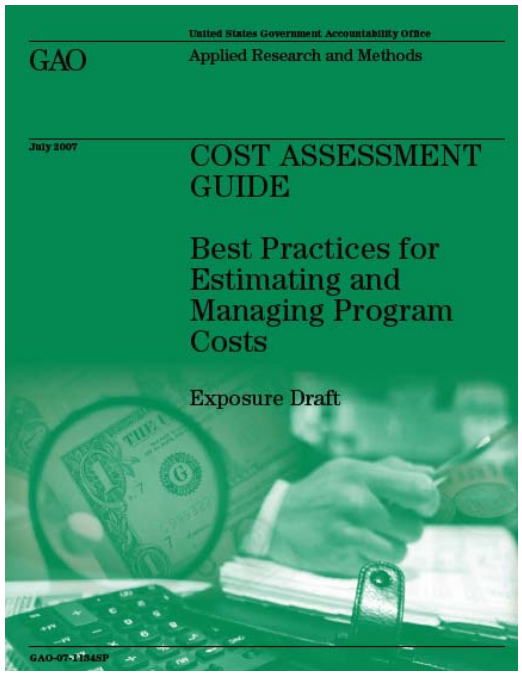
SEER by GALORATH

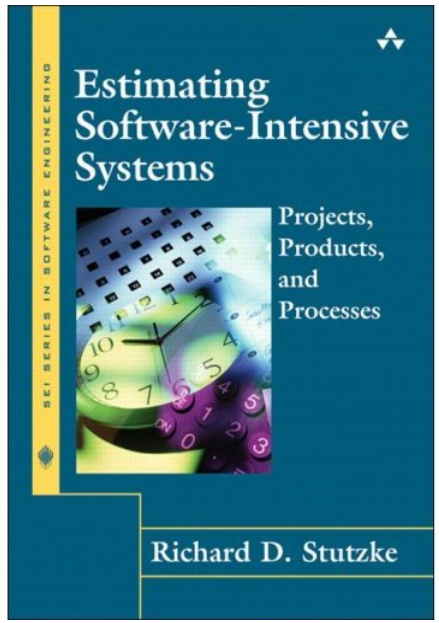
Time Frame	Development Concepts*	Sizing concepts
1950	Engineering is engineering	LOC - SLOC
1960	Software Crafting	
1970	Process, process, process (formality; waterfall)	Function Points
1980	Software tools/processes	
1990	Concurrent Processes	Sizing By Artifacts
2000	Agile methods, Objects, Use Cases	Use Cases
2010	?	?
2020	?	?

*adapted from "A View of the 20th and 21st Century software Engineering; Boehm

THREE SOURCES





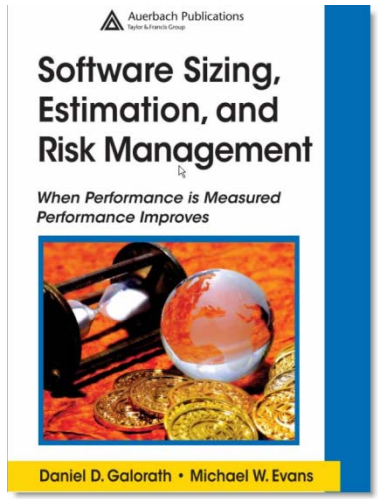


There are others

USE A PROCESS

10 Step Software Estimation Process:

Consistent Processes = 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

GAO COST GUIDE CHAPTER 12: SOFTWARE COST ESTIMATION - THE GAO VIEW

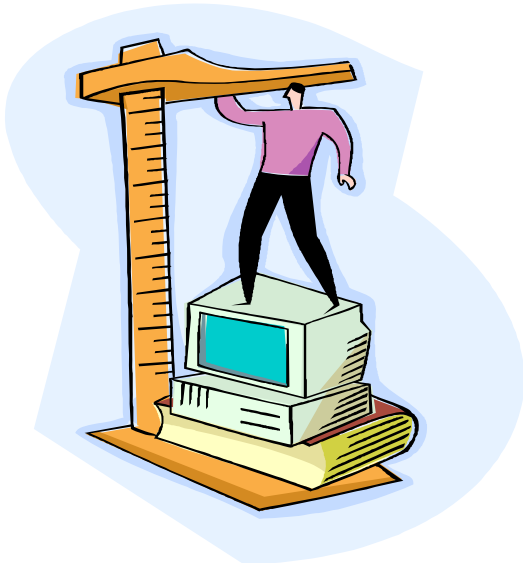
- **Estimating software development is a difficult and complex task**
  - **Close to 31 percent of software programs are canceled**
  - **More than half overrun original estimates by 50 percent, according to a Standish Group International, Inc. study (2000)**
- **There is an overwhelming sense of optimism about how quickly software can be developed**
  - **Stems from a lack of understanding how staffing, schedule, software complexity, and technology all interrelate**
  - **Optimism about new technology and reuse savings result in unachievable schedules**
- **Software costs are comprised of two basic elements**
  - **The amount, or size, of software to be developed**
  - **The development effort, or manpower, necessary to accomplish the requirements**

SOFTWARE SIZE IS STILL THE DRIVER

SIZING METHODS, TOOLS, PRACTICES FOR VIABLE ESTIMATES

SEER by GALORATH

- **Traditional & Artifact Size Measures**
  - Source lines of Code
  - Function Points
  - Sizing By Artifacts
    - Base Classes
    - Web Pages
    - Objects/Use Cases
    - Evolved Function Points
    - Pre-existing Lines
    - Others



- **SEER Extended Function Based Sizing**
  - 3 Years Research Beyond IFPUG
  - Easier To Estimate Before Detailed Architecture Available
- **Sizing Tools**
  - SEER Estimate by Comparison
  - SEER IBM RSx and Rose Adaptor
  - Analogies from Sizing Databases
  - Galorath Sizing Methodology
  - Other
- **Can we elevate Use Case Points to the level of other size metrics?**

EXECUTIVE ISSUES FOR SOFTWARE SIZING

- Sizing must be supportive of the level of estimates required
  - E.g. Use Cases or relative sizing for early rough order of magnitude estimates
  - E.g. Function Based Sizing or function points for detailed estimates
    - SEER Function based sizing is preferred
    - Your users will understand this without special training … because it **doesn't** require IFPUG certification
- Some organizations use their own unique sizing proxies
  - Standard component types

SIZE STUDY METHODOLOGY

SEER by GALORATH

Evaluate Multiple Sources of Software Size…

New Size

Pre-existing Size (rework)

Generated Code

COTS/GOTS

Glue Code

Integrated Code

Total Size Estimates	Least	Likely	Most
Expert Judgement	12000	15500	17000
Relevant Range by Analogy	19850	24750	32540
Sizing Database	8000	32000	46000
Functional Analysis	19680	27540	35400
SEER-AccuScope	15450	22650	29850

Expert Judgment

Functional Analysis

SEER-AccuScope Analysis

Counts for Pre-existing

Sizing Databases

Analogies

…Using Multiple Methods

ACCOUNT FOR SIZE GROWTH

- Software Size Growth May Be Driven By Many Factors
- Operational Environment Volatility
  - The mission/scope changes
- Essence (Requirements) Volatility
  - Initially the customer doesn't know what he/she wants
- Essence Understanding (Requirements Completeness and Correctness)
  - Don't understand the problem
  - The specifications are vague because problem not understood
- Essence versus Implementation Correspondence
  - The developer adds extra features (gold plating)

Bottom Line: You must understand size, and factor code growth in your estimates, or your effort & schedule projections will probably be low

USE CASES ARE NOT A PHENOMENON UNTO THEMSELVES

SEER by GALORATH

- Everything said in the previous charts about size is applicable to sizing with Use Case Points
- Because requirements are so abstract and different form computer programs, it is difficult for programmers to get them right
- Traditional requirements gathering often:

Takes too long	Documents the wrong thing
Makes bad assumptions	Are completed late

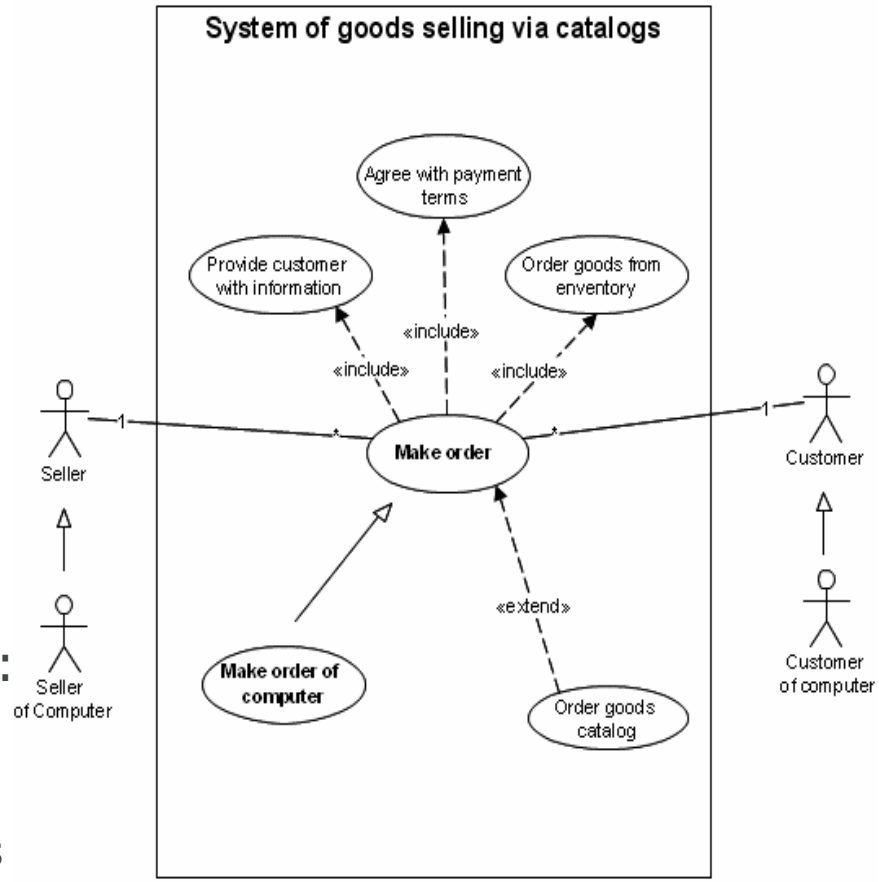
- The computer industry is struggling to find a way to represent functionality to users – Use Cases are a potential answer
- Use Cases are a part of comprehensive Unified Modeling Language (UML) composed of nine diagrams

Use Case	Sequence	Collaboration
Statechart	Activity	Class
Object	Component	Development

WHAT IS A USE CASE POINT?

A weighted count of actors and use cases.

- Actor weight is classified as:
  - 1 – Simple : highly defined and elemental, such as a simple API call
  - 2 – Average : protocol-driven interaction, allowing some freedom
  - 3 – Complex : potentially complex interaction
- Use Case weight is classified as:
  - 5 – Simple : 3 or fewer transactions
  - 10 – Average : 4-7 transactions
  - 15 – Complex : more than 7 transactions



USE CASE AND OBJECT SIZING

- Object Sizing
  - **If you already know how to count function points, you will find that counting objects is a matter of simple analogy**
  - **If you are new to function points, counting objects is in some ways easier because less judgment is involved**
- Use-Case Sizing
  - **Use cases, actors and relations, and the complexity of each is used to estimate a use case point (UCP) count; UCPs are a widely accepted metric for sizing use cases**
  - **Using statistically valid means, UCPs are automatically translated to an alternative metric such as source lines of code, function points or a metric you define**

ESTIMATING SOFTWARE VIA USE CASES

SEER by GALORATH

History

- Mid-1990s-Rumbaugh,Booch, and Jacobson of Rational Software Corporation developed the Unified Modeling Language (UML) as notation and methodology for developing object-oriented software
- UML was incorporated into the Rational Unified Process (RUP) by Rational Software
- Within UML is the concept of defining the requirements for software products with Use Cases
- Rational Software Corporation, created a software project estimating technique based on Use Case Points and including statistical and weighted modifiers
- Karner's technique is now incorporated into RUP.

  - Use Cases, as defined by UML, describe the things actors want the system to do and have proven to be an easy method for capturing the scope of a project early in its lifecycle

  - Use Cases may allow a consistent artifact to base an early project estimate.

SCHNEIDER AND WINTERS MODEL - APPLYING USE CASES

SEER by GALORATH

Weighting Actors for Complexity

Actor Type	Description	Quantity	Weight Factor	Subtotal
Simple	Defined API	3	1	3
Average	Interactive or protocol-driven interface	2	2	4
Complex	Graphical user interface	1	3	3
Total Actor Points				10

Weighting Use Cases for Complexity

Use Case Type	Description	Quantity	Weight Factor	Subtotal
Simple	Up to 3 transactions	3	5	15
Average	4 to 7 transactions	2	10	20
Complex	More than 7 transactions	1	15	15
Total Use Cases				50

Add the total for Actors to the total for Use Cases to determine the Unadjusted Use Case Points (UUCP) = 60

SCHNEIDER AND WINTERS MODEL - APPLYING USE CASES

Weighting technical factors is an exercise to calculate a Use Case Point modifier, called the technical complexity factor (TCF)

Weighting Technical Factors

Technical Factor	Factor Description	Weight Factor	Project Rating	Subtotal
T1	Must have a distributed solution	2	5	10
T2	Must respond to specific performance objectives	1	3	3
T3	Must meet end-user efficiency desires	1	5	5
T4	Complex internal processing	1	5	5
T5	Code must be reusable	1	3	3
T6	Must be easy to install	.5	3	1.5
T7	Must be easy to use	.5	3	1.5
T8	Must be portable	2	0	0
T9	Must be easy to change	1	5	5
T10	Must allow concurrent users	1	0	0
T11	Includes special security features	1	5	5
T12	Must provide direct access for third-parties	1	0	0
T13	Requires special user training facilities	1	3	3
Total TFactor				42

SCHNEIDER AND WINTERS MODEL - APPLYING USE CASES

(Weighting Factor) * ∑(Tlevel) = TFactor
The TFactor does not directly modify the UUCP. To calculate Technical Complexity Factor (TCF), multiply TFactor by 0.01 and then add 0.6.
(0.01 * Tfactor) + 0.6 = TCF
(0.01 * 42) + 0.6 = 1.02 TCF
Calculate the size of the software (Use Case) project by multiplying UUCP times TCF.
UUCP * TCF = SzUC
60 * 1.02 = 61.2

Note on Reusable Components: Reusable software components should not be included in this estimate. Identify the UUCP associated with the reusable components and Adjust the size of SzUC accordingly.

SCHNEIDER AND WINTERS MODEL - APPLYING USE CASES

The experience level of each team member can have a great effect on the accuracy of an estimate. This is called the Experience Factor (EF).

Weighting Experience Factors

ExperienceFactor	Factor Description	Weight Factor	Project Rating	Subtotal
E1	Familiar with FPT software process	1	4	4
E2	Application experience	0.5	2	1
E3	Paradigm experience (OO)	1	4	4
E4	Lead analyst capability	0.5	4	2
E5	Motivation	0	4	0
E6	Stable Requirements	2	2	4
E7	Part-time workers	-1	0	0
E8	Difficulty of programming language	-3	1	-3
Total EFactor				12

To calculate EF, go through the preceding table and rate each factor from 0 to 5. ∑(Elevel) * (Weighting Factor) * = Efactor
Calculate the Experience Factor (EF) by multiplying Efactor times –0.03 and adding 1.4. (-0.03 * 12) + 1.4 = 1.04
To calculate Use Case Points (UCP), multiply SzUC by EF SzUC * EF = UCP
61.2 * 1.04 = 63.648 or UUCP * TCF * EF = UCP = 60 * 1.02 * 1.04 = 63.648

SCHNEIDER AND WINTERS MODEL - APPLYING USE CASES

To calculate Use Case Points (UCP), multiply SzUC by EF SzUC * EF = UCP 61.2 * 1.04 = 63.648 or UUCP * TCF * EF = UCP = 60 * 1.02 * 1.04 = 63.648

Now that we have estimated the Use Case Point, where do we go from here?

- **Use the Use Case Point Count to directly estimate man-hours**
- **Use the Use Case Point count as a direct factor in the estimating model**

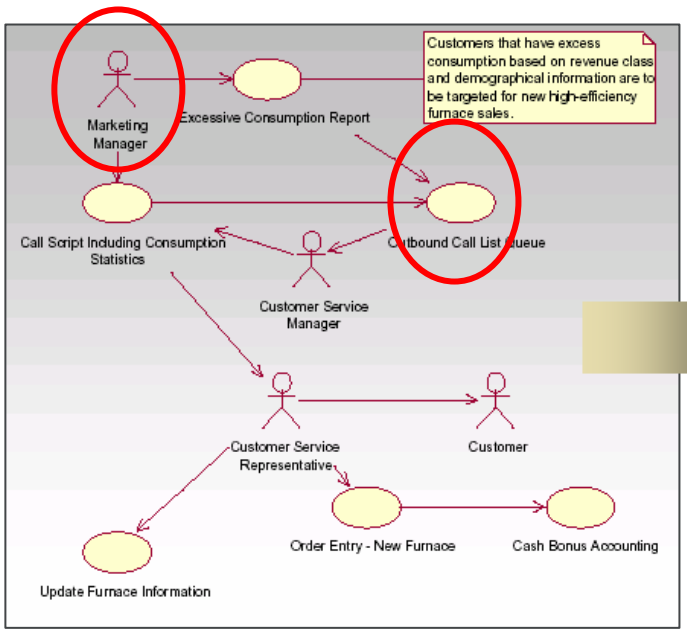
The Galorath Approach for IBM Rational/UML (RSx and Rose)

SEER by GALORATH

User Enters Use Case…

Chart Is Read In A Machine-Readable Format…

SEER Adaptor UML Estimates Desired Metric



1. Actor(s)

1.1 IT Support Clerk

2. Flow of Events

2.1Basic Flow

2.1.1 IT Support Clerk selects Diagnose.
2.1.2 *System examines disk for errors*
2.1.3 *System displays results*
2.1.4 IT Support Clerk confirms results
2.1.5 End of Use Case.

3. Alternative Flows

3.1 Continuing from 2.1.2 - System identifies errors on the disk

3.1.1 *System identifies errors on the disk and displays fix option*
3.1.2 IT Support Clerk chooses to correct errors
3.1.3 *System corrects errors and displays results.*
3.1.4 End of use case

3.2Continuing from 2.1.2 - System identifies errors on the disk

3.2.1 *System identifies errors on the disk and displays fix option*
3.2.2 IT Support Clerk chooses not to fix errors on disk
3.2.3 *System skips fix and displays results.*
3.2.3 End of Use Case

4. Special Requirements

5. Pre-Conditions

5.1 System navigated from Norton SystemWorks to Norton Utilities to Norton Disk Doctor
5.2 Norton Disk Doctors is correctly installed on PC

6. Post Condition

6.1 Norton Disk Doctor closed and System returns to idle condition

Scope

XXXXX lines of code
YYY unadjusted function points
ZZZ object measures

TO SEER-IBM Rational RSx and Rose Adaptor (CriticalMass)

SEER-CriticalMass can scope a project directly from use cases specified in

IBM Rational Rose or RSX

- It can be used for:
  - Very early size estimates
  - A sanity check against alternative size estimates
  - On a continuous basis, to measure changes in scope
- Tied into the SEER product line, you can achieve very early project estimates and plans

CLOSING THOUGHTS

SEER by GALORATH

- The Use Case points method is a very valuable addition to the tools available for the project manager
- Use Case Points may become as reliable as other effort estimation tools such as function point and lines of code with time
- All of the estimation methods are susceptible to error, and require accurate historical data to be useful within the context of the organization
- The Use Case points method is especially valuable in those system development projects where use cases are produced anyway
- The standardization and international efforts that have helped the Function Point method become widely accepted should be applied to Use Case Points
- At this time, Use Case points are subject to some variability and they need to be calibrated for your organization

- **MORE TO COME**

The Good, the Bad and the Ugly

- **Parametric estimation provides consistency**
- **Less bias – "facts on the table"**
- **Requires organizational training & discipline**

"Software estimation is neither hard nor new. What is hard, is accepting that the easy-to digest answer we seek when estimating is simply not there…

Using these [estimation] tools turns the practice of estimation ... that engrains a disciplined approach and pays heed to underlying behavioral and attitude challenges, into an exercise that is simply about playing with numbers"

Douglas Muir, *The New Relevance Of Estimation*, Software Productivity Center