

Estimation – a Waste of Time? *(and some alternatives that actually work !)*

For BCS SPA, London
June 1 2011
Evening
By Tom Gilb

Based On A Paper

Software Quality
Professional

Series Software Engineering Techniques Improving
the Software Development Life Cycle for Critical
Applications
Michael A. Gilb
Origins and Analysis of Software Engineering
Michael A. Gilb, Ed. David L. King, Steven D. King,
James R. King, and Robert L. King
Are We Doing Good? Or Are We Doing Bad?
Linda Winters
A Study of Test Engineering by EITC's Computer
Lab the "Test to Break" Initiative
Michael A. Gilb, Steven D. King, Robert L. King,
and David L. King



“Estimation: A Paradigm Shift Toward Dynamic Design-to Cost and Radical Management”

-  Volume 13 Issue 2 of SQP journal - the March 2011 version.
 -  Software Quality Professional, USA
 -  The American Society for Quality (ASQ)
-  http://www.gilb.com/tiki-download_file.php?fileId=460

The Obligatory Dilbert

December 7, 2009

About Latest News

I NEED A BUDGET ESTIMATE FOR MY PROJECT, BUT I DON'T HAVE A SCOPE OR A DESIGN FOR IT YET.

OKAY, MY ESTIMATE IS \$3,583,729.






YOU DON'T KNOW ANYTHING ABOUT MY PROJECT.

THAT MAKES TWO OF US.

Dilbert.com DilbertCartoonist@gmail.com

12-7-09 © 2009 Scott Adams, Inc./Dist. by UFS, Inc.

The Risk Principles

-  1. DRIVERS: If you have not specified all critical performance and quality levels numerically – you cannot estimate project resources for those vague requirements.
-  2. EXPERIENCE: If you do not have experience data, about the resources needed for your technical solutions, then you cannot estimate the project resources.
-  3. ARCHITECTURE: If you implement your project solutions *all at once*, without learning their costs and interactions incrementally – you cannot expect to be able to understand the results of many interactions.
-  4. STAFF: If a complex and large professional project staff is an unknown set of people, or changes mid-project – you cannot expect to estimate the costs for so many human variables.
-  5. SENSITIVITY: If even the *slightest change* is made, after an ‘accurate’ estimation, to *any* of the requirements, designs or constraints – then the estimate might need to be changed *radically*. And – you probably will not have the information necessary to do it, nor the insight that you *need* to do it.

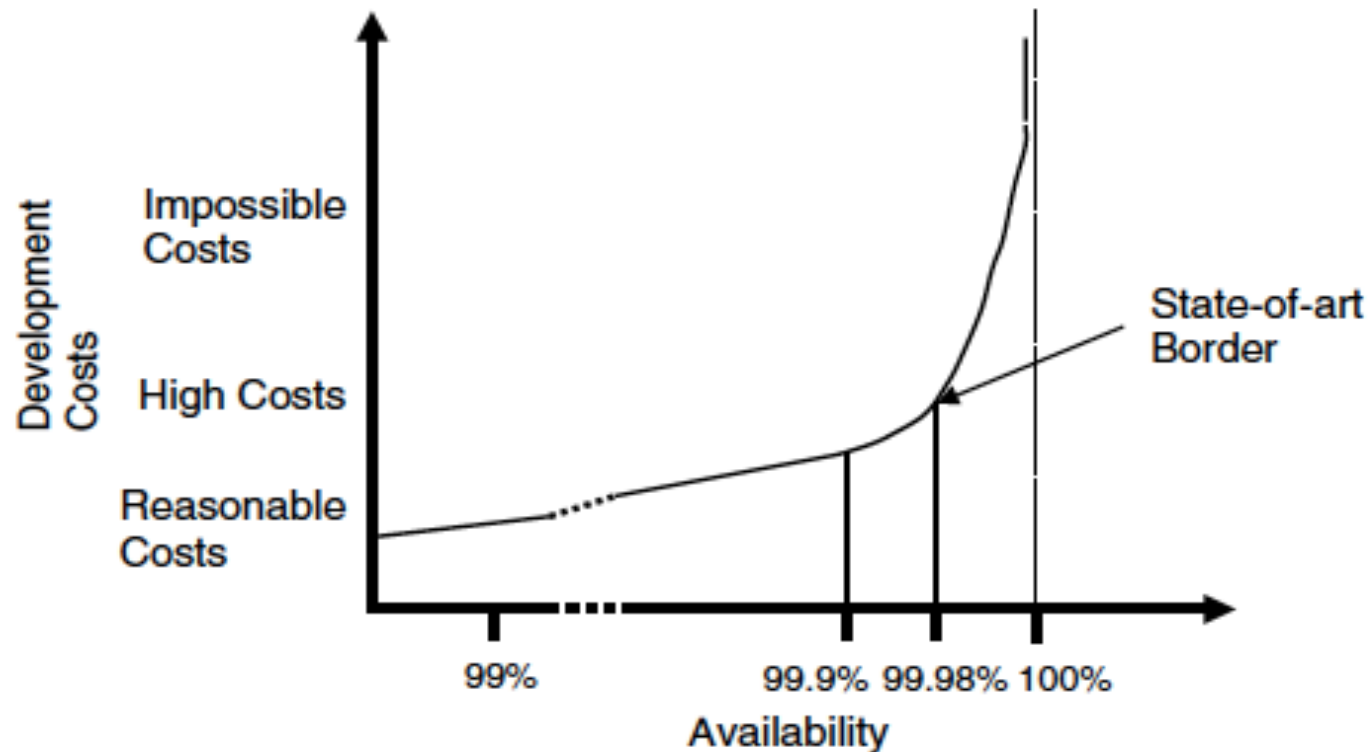
The Risk Principles (in Detail)

- 🌐 The point being
 - 🌐 that I want you to lose faith in convention notions of project estimation
 - 🌐 The risk of being very wrong is very high!
 - 🌐 The probability of being reasonably right is as big as you winning the Euro Lottery prize this week
 - 🌐 In fact if you sometime experience being 'right', it is Not due to estimation
 - 🌐 Just probably due to slamming on the brakes, when the resources are used up.

1. DRIVERS

- 🌐 If you have not specified
 - 🌐 all critical performance and quality levels numerically –
 - 🌐 you *cannot* estimate project resources for those vague requirements.

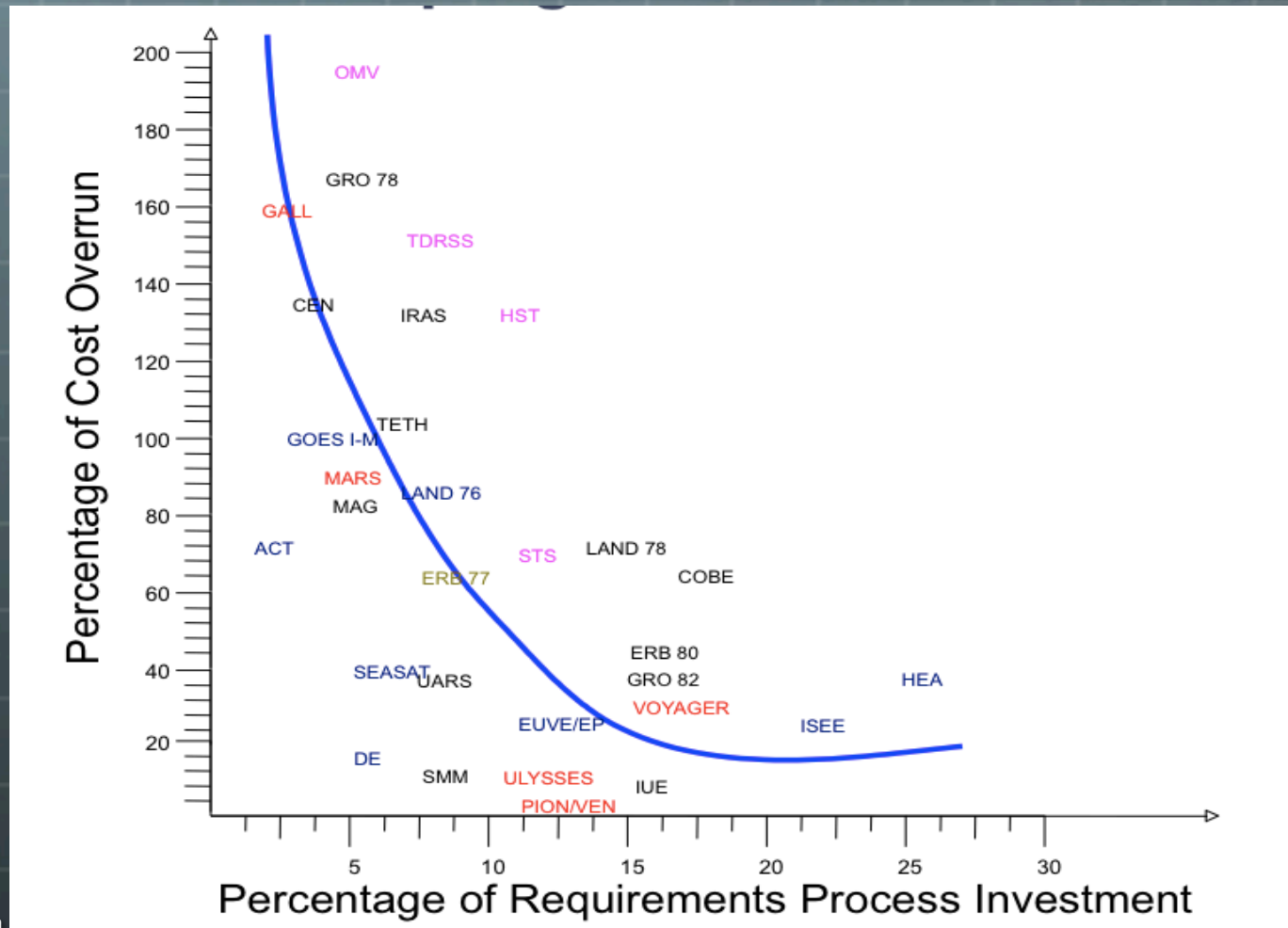
How much will 'High Availability' Cost?



2. EXPERIENCE

- 🌐 If you do not have experience data,
 - 🌐 about the resources needed for your technical solutions,
 - 🌐 then you *cannot* estimate the project resources.

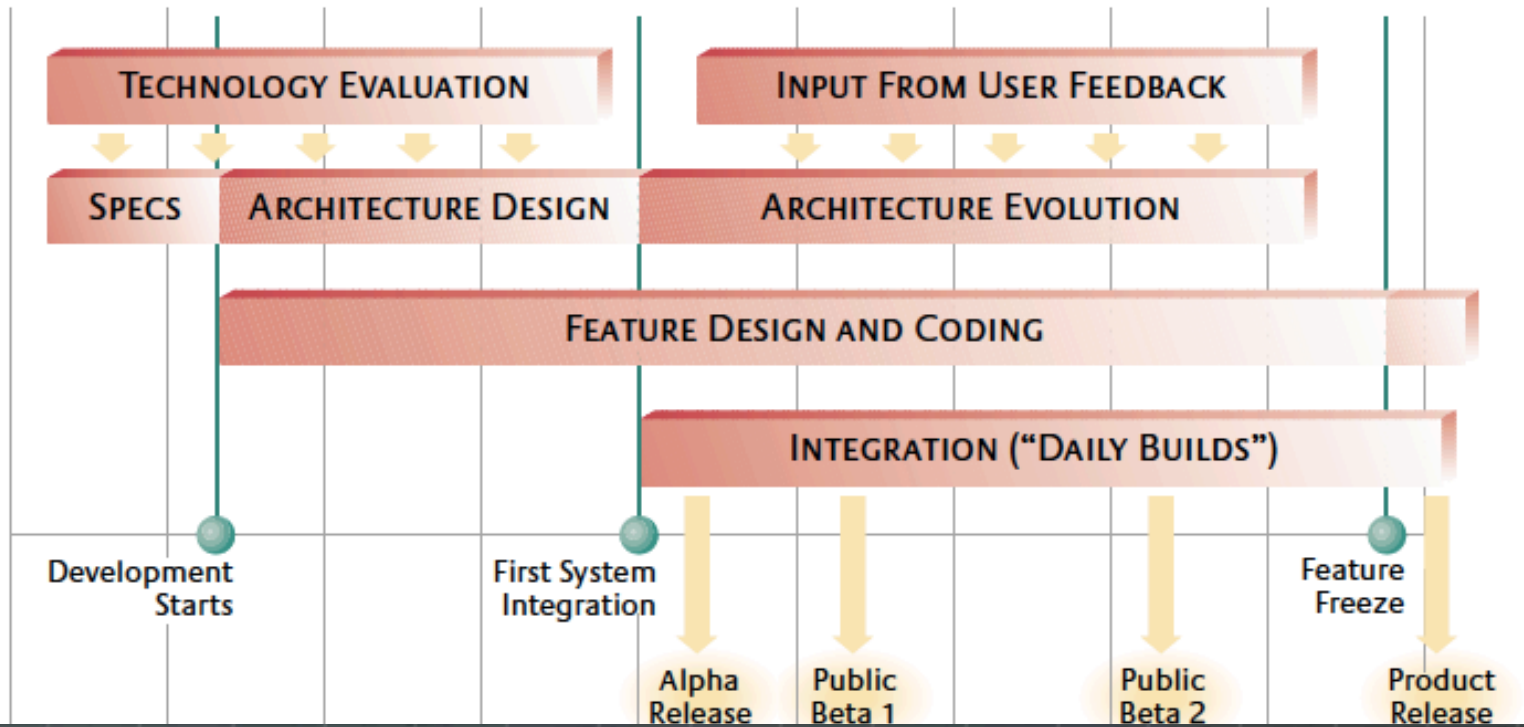
What is the cost difference if we use 5% for requirements, rather than 25%, if we are NASA?



3. ARCHITECTURE

- If you implement your project solutions *all at once*,
 - without learning their costs and interactions incrementally –
 - you cannot expect to be able to understand the results of many interactions.

Big Bang Fails



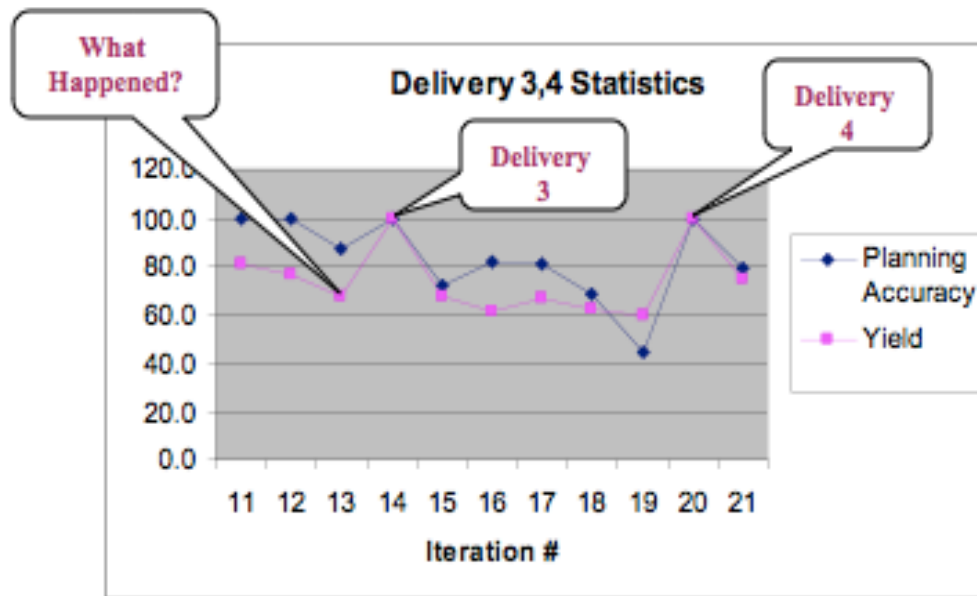
Small Delivery Steps Give Better Control

	Design Idea: Step 9 - Recoding			
Requirements	Estimated Scale Impact	Estimated % Impact	Actual Scale Impact	Actual % Impact
Objectives				
Usability.Productivity 65 <-> 25 minutes Past: 65 minutes. Tolerable: 35 minutes. Goal: 25 minutes.	65 – 20 = 45 minutes	50%	65 - 38 = 27 minutes	95%
Resources				
Development Cost 0 <-> 110 days	4 days	3.64%	4 days	3.64%

4. STAFF

- If a complex and large professional project staff is
 - an unknown set of people,
 - or changes mid-project –
- you cannot expect to estimate the costs for so many human variables.

Real Case: Iterative measures, detected bad staff change (Honeywell, Berntsen)



Measures

- Planning Accuracy - % of planned work that was completed.
- Build Yield - % of completed work that passed verification testing.

5. SENSITIVITY

- 🌐 If even the *slightest change* is made,
 - 🌐 after an ‘accurate’ estimation,
 - 🌐 to *any* of the requirements, designs or constraints –
 - 🌐 then the estimate might need to be changed *radically*.
- 🌐 And – you probably will not have the information necessary to do it,
 - 🌐 nor the insight that you *need* to do it.

Primary Objectives for a £100 mill. Project

- 🌐 Central to the Corporation's business strategy is to be the world's premier integrated <domain> service provider
- 🌐 Will provide a much more efficient user experience
- 🌐 Dramatically scale back the time frequently needed after the last data is acquired to time align, depth correct, splice, merge, recompute and/or do whatever else is needed to generate the desired products
- 🌐 Make the system much easier to understand and use than has been the case for the previous system
- 🌐 A primary goal is to provide a much more productive systems development environment than was previously the case
- 🌐 Will provide a richer set of functionality for supporting next-generation logging tools and applications
- 🌐 Robustness is an essential system requirement (see rewrite in example below)
- 🌐 Major improvements in data quality over current practices.

The Control Principles

- 6. LEARN SMALL: Carry out projects in *small increments* of delivering requirements – so you can measure *results* and *costs*, against (short term) estimates.
- 7. LEARN ROOT: If incremental costs for a given requirement level (and its designs) deviate negatively from estimates – analyze the root cause, and change anything about the next increments that you believe might get you back on track.
- 8. PRIORITIZE CRITICAL: You will have to *prioritize* your most critical requirements and constraints: there is no guarantee you can achieve them all. Deliver ‘high-value for resources-used’ *first*.
- 9. RISK FAST: You should probably implement the design ideas with the highest value, with regard to cost and risk, early.
- 10. APPLY NOW: Learn early, learn often, learn well; and apply the learning to your current project.

The Control Principles (In Detail)

- 🌐 The point here is that :
 - 🌐 Given *any* estimate of reasonable resources
 - 🌐 You should be able to deliver so much prioritised value
 - 🌐 that you will stay in business, forever

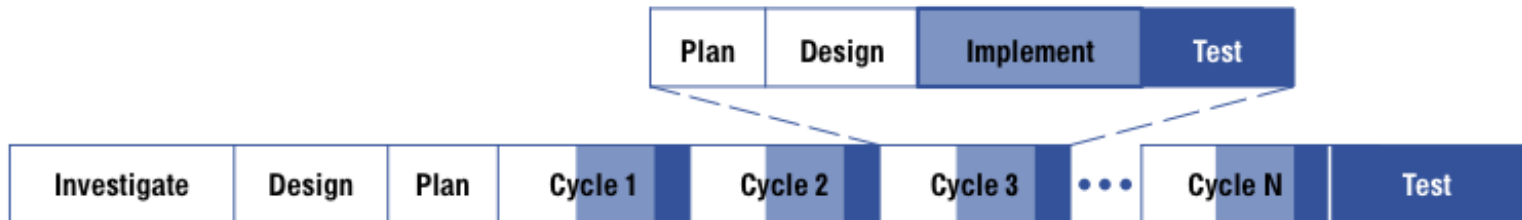
6. LEARN SMALL

- 🌐 Carry out projects in *small increments* of delivering requirements –
 - 🌐 so you can measure *results* and *costs*,
 - 🌐 against (short term) estimates.

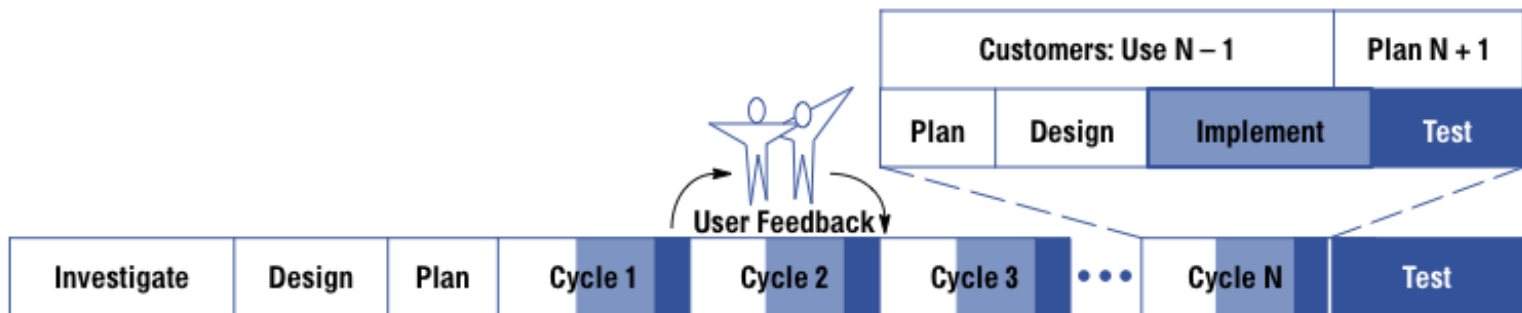
Breaking Result Deliveries into Small Chunks (Evo, HP)



Waterfall Development Life Cycle



Incremental Development Life Cycle



Evolutionary Development Life Cycle

7. LEARN ROOT

- If incremental costs for a given requirement level (and its designs) deviate negatively from estimates –
 - analyze the root cause, and
 - change anything about the next increments that you believe might get you back on track.

5 'Why's find roots

Customers wait too long on the phone at the end of the month.

WHY?

The last week of the month is the busiest for sales.

WHY?

The company offers more incentives to customers late in the month.

WHY?

Sales are usually behind the goal late in the month.

WHY?

Customers have learned that if they wait, they will get incentives.

WHY?

Root Cause

Sales targets are done on a monthly basis, letting a big deficit form.



Action: Make weekly sales goals instead of monthly targets to prevent getting so far behind.

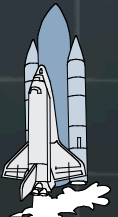
8. PRIORITIZE CRITICAL

- 🌐 You will have to *prioritize* your most critical requirements and constraints:
 - 🌐 there is no guarantee you can achieve them all.
- 🌐 Deliver ‘high-value for resources-used’ *first*.

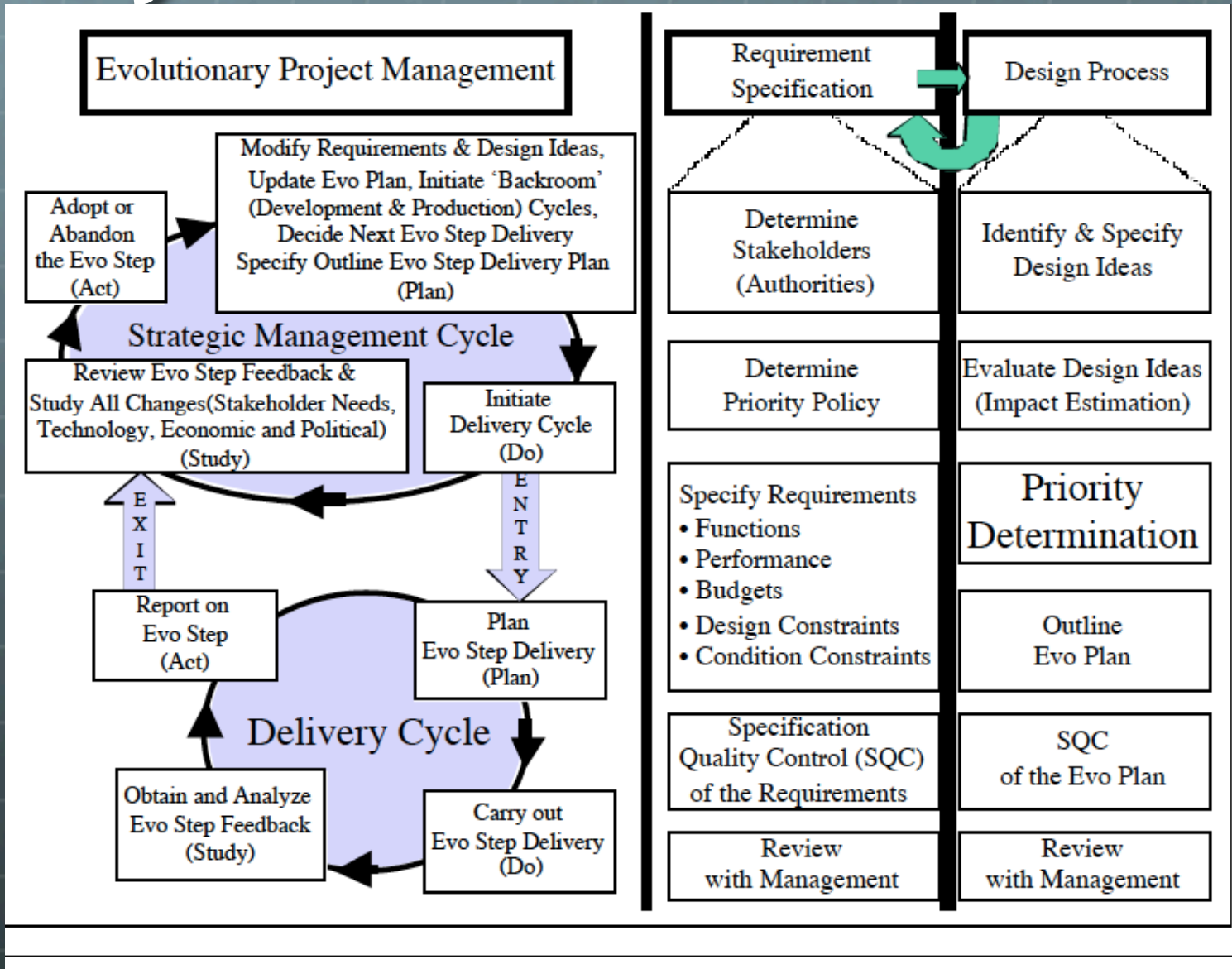
In the Cleanroom Method, developed by IBM's Harlan Mills (1980) they reported:



- “Software Engineering began to emerge in FSD” (IBM Federal Systems Division, from 1996 a part of Lockheed Martin Marietta) “some ten years ago [Ed. about 1970] in a continuing evolution that is still underway:
- Ten years ago general management expected the worst from software projects – cost overruns, late deliveries, unreliable and incomplete software
- Today [Ed. 1980!], management has learned to expect on-time, within budget, deliveries of high-quality software. A Navy helicopter ship system, called LAMPS, provides a recent example. LAMPS software was a four-year project of over 200 person-years of effort, developing over three million, and integrating over seven million words of program and data for eight different processors distributed between a helicopter and a ship in 45 incremental deliveries [Ed. Note 2%!]. Every one of those deliveries was on time and under budget
- A more extended example can be found in the NASA space program,
 - - Where in the past ten years, FSD has managed some 7,000 person-years of software development, developing and integrating over a hundred million bytes of program and data for ground and space processors in over a dozen projects.
 - - There were few late or overrun deliveries in that decade, and none at all in the past four years.”



Dynamic Prioritisation



9. RISK FAST

- 🌐 You should probably implement the design ideas with the highest value, with regard to cost and risk, early.

Which Designs are 'Risky' ?

Design Ideas

On-line Support: Gist: Provide an optional alternative user interface, with the users' task information for defined task(s) embedded into it.

On-line Help: Gist: Integrate the users' task information for defined task(s) into the user interface as a 'Help' facility.

Picture Handbook: Gist: Produce a radically changed handbook that uses pictures and concrete examples to *instruct*, without the need for *any* other text.

Access Index: Gist: Make detailed *keyword indexes*, using *experience* from *at least ten* real users learning to carry out the defined task(s). What do *they* want to look things up under?

'Impact Estimation' Making 'Risk' Visible

	<i>On-line Support</i>	<i>On-line Help</i>	<i>Picture Handbook</i>	<i>On-line Help + Access Index</i>
Learning 60 minutes <-> 10 minutes				
Scale Impact	5 min.	10 min.	30 min.	8 min.
Scale Uncertainty	±3 min.	±5 min.	±10 min.	±5 min.
Percentage Impact	110%	100%	60%	104%
Percentage Uncertainty	±6% (3 of 50 minutes)	±10%	±20%?	±10%
Evidence	Project Ajax: 7 minutes	Other Systems	Guess	Other Systems + Guess
Source	Ajax Report, p.6	World Report, p.17	John B	World Report, p.17 + John B
Credibility	0.7	0.8	0.2	0.6
Development Cost	120 K	25 K	10 K	26 K
Performance to Cost Ratio	$110/120 = 0.92$	$100/25 = 4.0$	$60/10 = 6.0$	$104/26 = 4.0$
Credibility-adjusted Performance to Cost Ratio (to 1 decimal place)	$0.92 * 0.7 = 0.6$	$4.0 * 0.8 = 3.2$	$6.0 * 0.2 = 1.2$	$4.0 * 0.6 = 2.4$

10. APPLY NOW








Learn early, learn often, learn well; and apply the learning to your current project.

We can *simplify* The Control Principles

- 1. Do something of value in a short time
 - 2. Measure values and costs
 - 3. Adjust what you do next, if necessary
-
- Repeat until you no longer can find value for money

Advantages with Control Principles

-  1. You cannot waste much time or money before you realize that you have false ideas
-  2. You *can* deliver value early, and keep people happy
-  3. You are forced to think about the *whole* system, including *people* (not just code)
-  4. So you are destined to see the true costs of delivering value – not just the code costs
-  5. You will learn a general method that you can apply for the rest of your career.

Disadvantages Control Principles

- 1. You cannot hide your ignorance from yourself any longer
- 2. You might have to do something not taught at school, or not taught in textbooks
- 3. There will always be people who criticize anything different or new
- 4. You cannot continue to hide your lack of ability to produce results, inside a multi-year delayed project.

Finis

 Tom @ Gilb . Com

 www.Gilb.com