

Project Metrics:
The Evolutionary Project
Management Method:
Practical Rules, Principles &
Templates to Practice Evolutionary
Project Management

*A practical and proven way to manage any project with
focus on high, immediate, measurable, estimated,
continuous, stakeholder-value delivery*

Monday 15th october 2007, Full Day

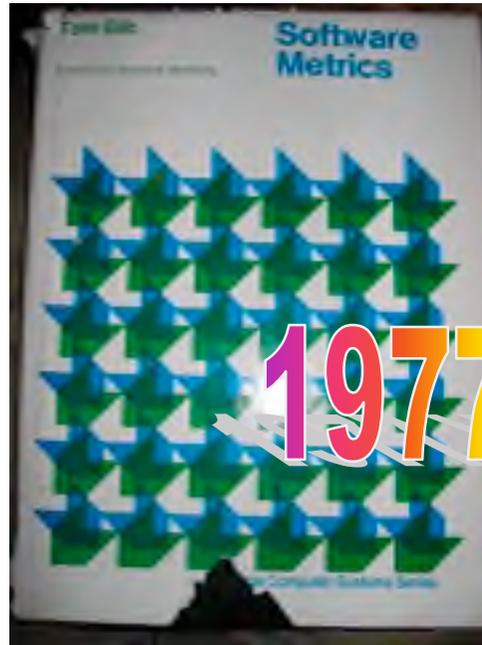
Presenter: Tom Gilb

Tom@Gilb.com

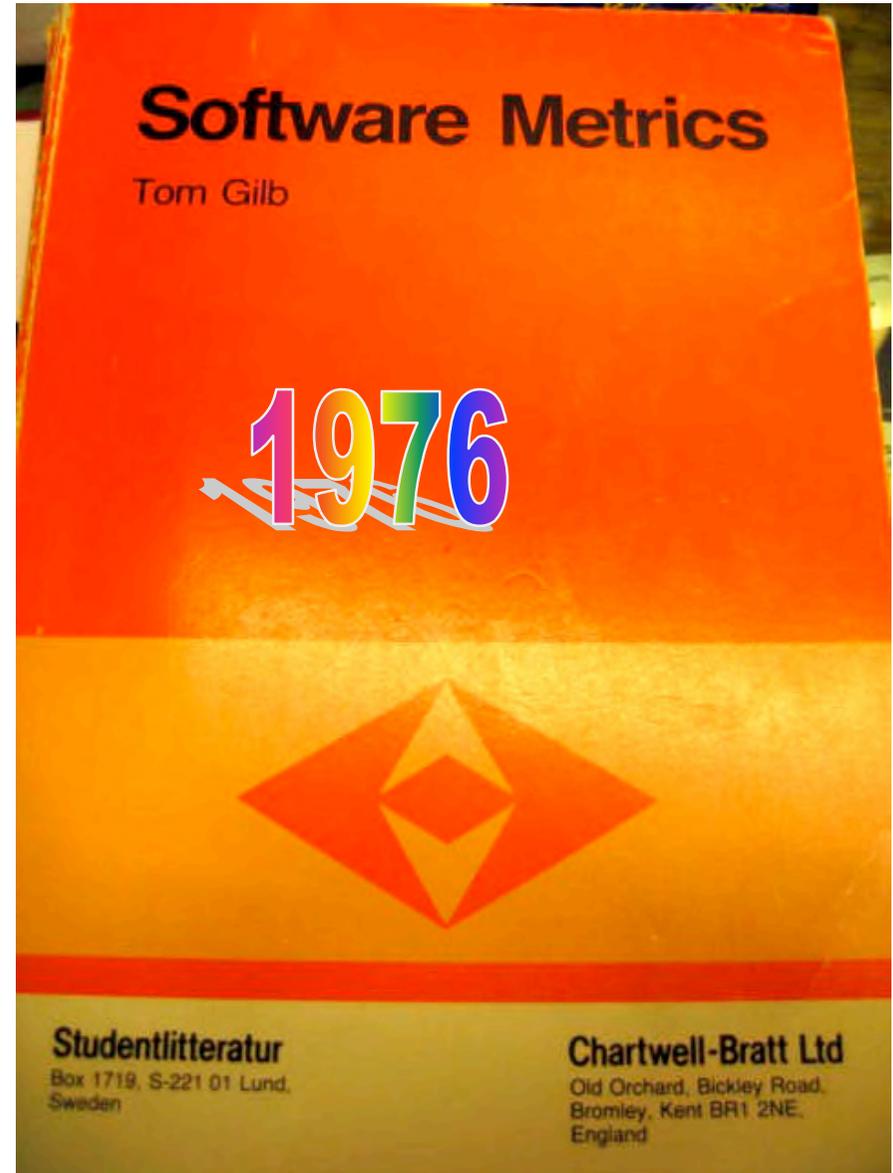
UK Software Metrics Association, London

Books 1976, 1977 and 2005

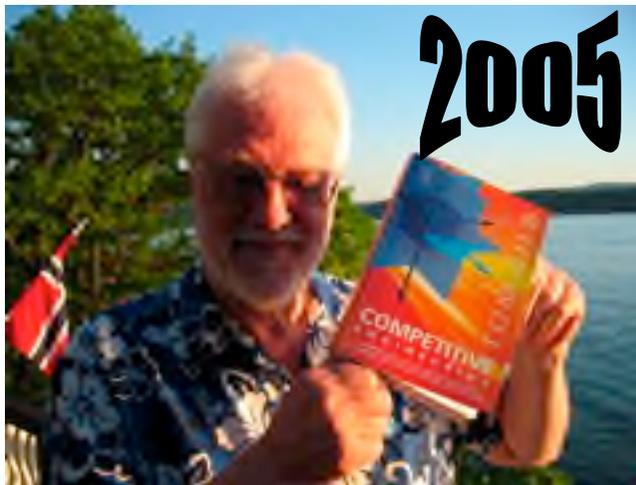
2



1977 USA



1976



2005

- Most people have only learned some form of 'Waterfall' (Grand Design) project management.
 - It is obsolete and dangerous to the health of your project.
- Evo is the most successful alternative project management method, if you look at practical experience, and is now a 'mandatory guideline' at US DoD.
 - Isn't it about time you learned more about it?
- This tutorial will supply the participant with the pragmatics of doing evolutionary project management - The Evo toolkit.
 - How do you specify objectives that you can evolve towards in small steps?
 - How do you specify designs that can be decomposed into smaller delivery steps?
 - How do you specify and control evolutionary stakeholder-value-delivery steps themselves?
 - The toolkit gives practical help.
- Evo has major impact on the whole way in which systems engineering is carried out.
 - All systems engineering processes (requirements, design, build, test, and quality control) are suddenly encapsulated into an early and frequent evolutionary result delivery step.
- If you know what you are doing you will soon produce results for stakeholders.
 - If not, you won't; and must consequently fix your engineering processes and designs.
- Who Should Attend:
 - Project managers, managers of project managers, software process specialists, IT Directors, software product company managers.

Workshop Content

4

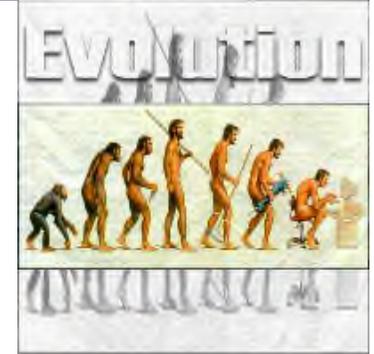
Introductory Slides

1. The Evo process description: a reusable template.
2. Basic Evo principles
3. Principles for decomposing into small Evo steps.
4. Defined Evo processes
5. Templates for Quantified Requirements and Quantified Design
6. Templates for Quantified Evo step specification
7. Impact Estimation Table Evo project management
8. Evo Policy template
9. Organizational considerations when doing Evo
10. Evo contracting template

Summary

Extra Slides

'Evo' defined



**A project management process
delivering evolutionary results
'high-value-first' progress
towards the desired goals, and
seeking to obtain, and use, realistic,
early feedback.**

”Complete focus on early rapid delivery of stakeholder value”

- *frequent* delivery of system changes (steps)
- steps delivered to stakeholders for *real* use
- feedback obtained from *stakeholders* to determine *next* step(s)
- the *existing* system is used as the initial system base
- *small* steps (ideally between 2%-5% of total project financial cost and time)
- steps with *highest value* and benefit-to-cost ratios given highest *priority* for delivery
- feedback used 'immediately' to modify long term plans and requirements and, also
- to decide on the *next* step total systems approach ('change *anything* that helps') -
- *results*-orientation ('delivering the results' is prime concern)

What are the major benefits of Evo?

7

- Management control of value
- Management control of costs
- Enforcing business thinking
 - Instead of technical thinking
- Flexibility for management to re-prioritize projects and spend
- Improves system maintenance culture
 - Because you 'maintain' at each step
 - Very low risk to do it and see if it works



Harlan Mills on Project Control 8



"Software Engineering began to emerge in FSD" (IBM Federal Systems Division, from 1996 a part of Lockheed Martin Marietta) "some ten years ago [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 [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.

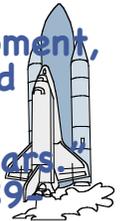
LAMPS software was a four-year project of over 200 person-years of effort, developing over three million, and integrating over seven million words of program and data for eight different processors distributed between a helicopter and a ship, in 45 incremental deliveries.

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. Harlan Mills [IBM Systems Journal No. 4, 1980, p. 415], Reprinted IBM SJ Vol. 38 1999, 289, 295



See note for Flight software. <http://history.nasa.gov/sts1/pages/computer.html> Case Study,

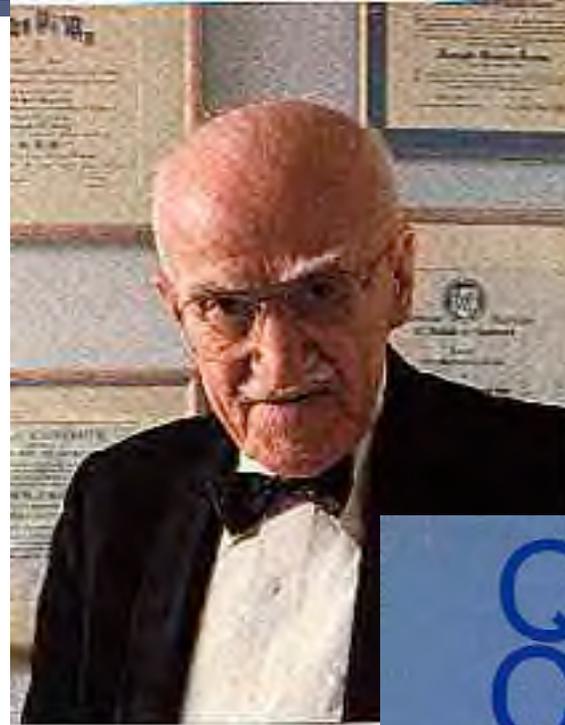
"The Space Shuttle Primary Computer System," *Communications of the ACM* 27, No. 9 (September 1984): 871-880.

See note for Weinberg history FSD via Mercury project

- “Design is an iterative process in which each design level is a refinement of the previous level. At each stage, design and cost alternatives are examined. Those that best satisfy the project objectives are prepared for review and selection by the project sponsor.
- If no alternative fits the cost target, several courses of action are available.
 - The most common one is to go back to the designer and ask for a less costly, and perhaps a less attractive design.
- If the target has been missed by a large amount - and cost is critical - redesign may not produce an answer.
 - In this case the sponsor has to consider giving up some of the planned capability of the system.
- Otherwise he has to recognize that the capability cannot be acquired without increasing the cost target.
- The design [to-cost] process is followed until the program design for a specific software increment has been completed. From that point, development of each increment can proceed concurrently with the program design of the others.
- When the development and test of an increment are complete, an estimate to complete the remaining increments is computed.
 - The algorithms used in this computation should reflect the various actual productivity rates experienced in developing and testing previous increments.
 - An alternative plan is prepared and reviewed, as previously described, whenever a cost projection is inconsistent with its cost plan....
 - The design-to-cost practice describes the management control procedures that balance cost, schedule, and functional capability.”
- ← Robert Quinnan, [IBM SJ No. 4 1980, page 474, web available]

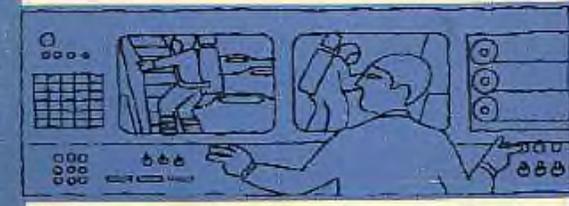
The Evo Cousins' Commonality

- Learning
- Measurement
- Future Improvement orientation
- Process Improvement
- The Deming/Shewhart (Juran) Statistical ideas
- Eternal learning
- Distinguish between 'chance causes' and 'common causes'
 - fix the common causes.



Quality Control Handbook

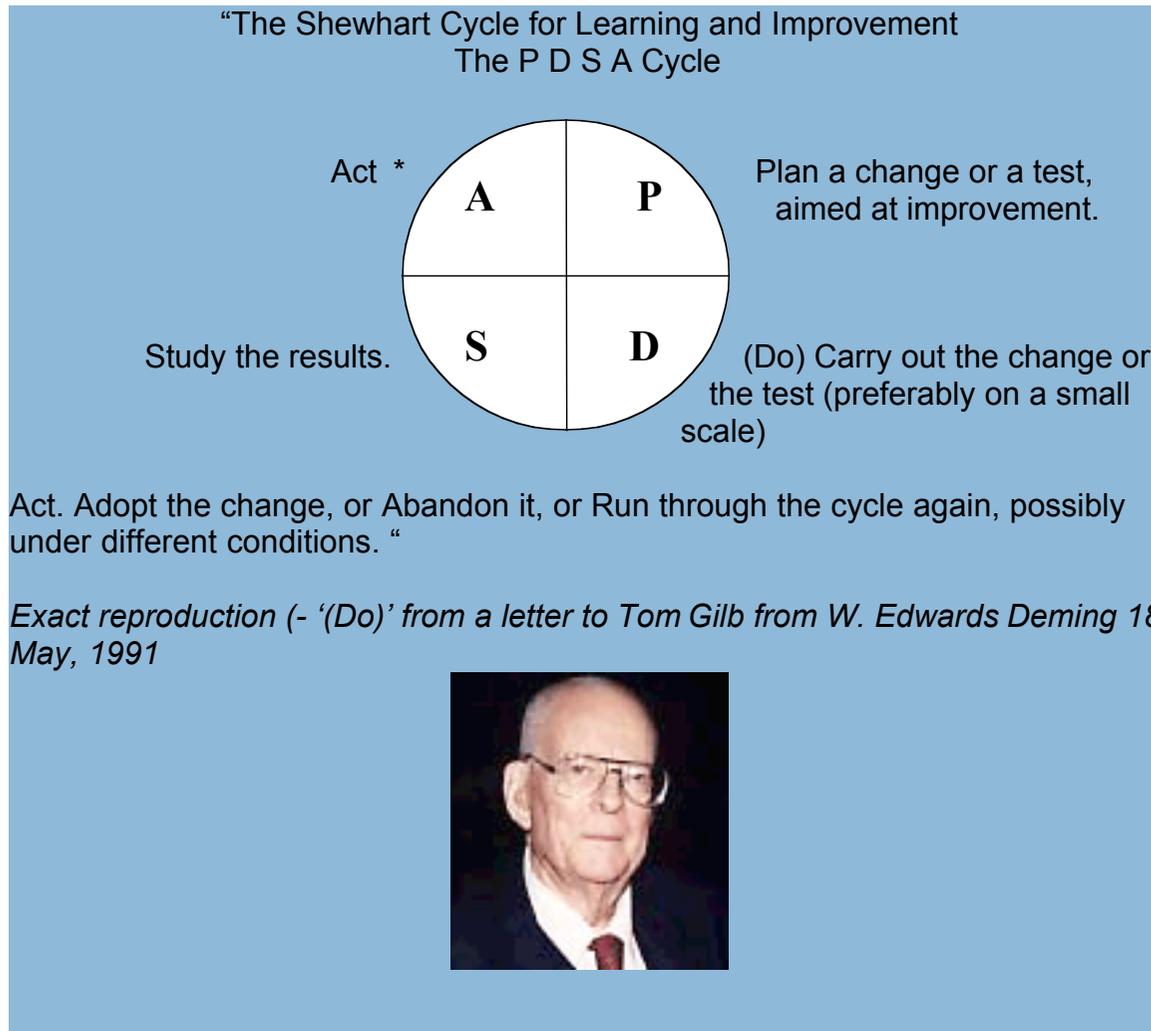
THIRD EDITION



J.M. JURAN, Editor-in-Chief; FRANK M. GRZYNA, JR., and R. S. BINGHAM, JR., Associate Editors

The P D S A Cycle from Deming

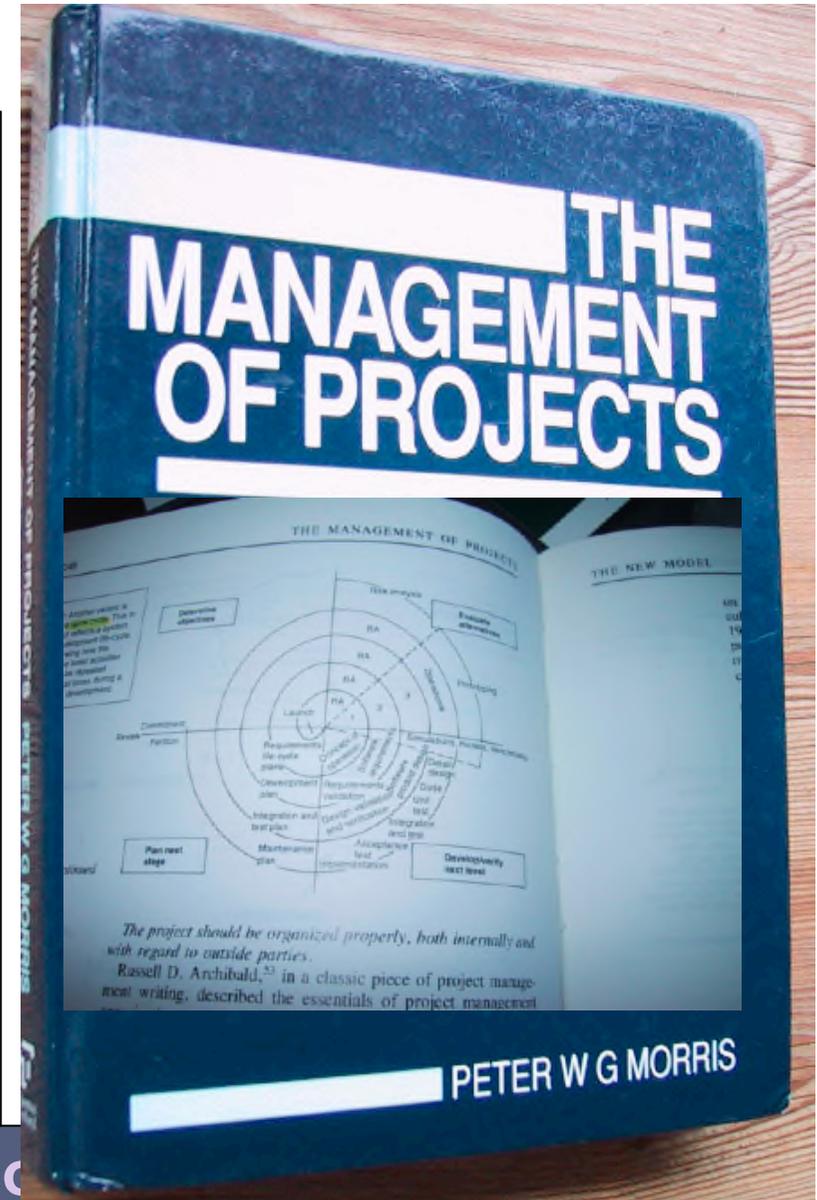
11

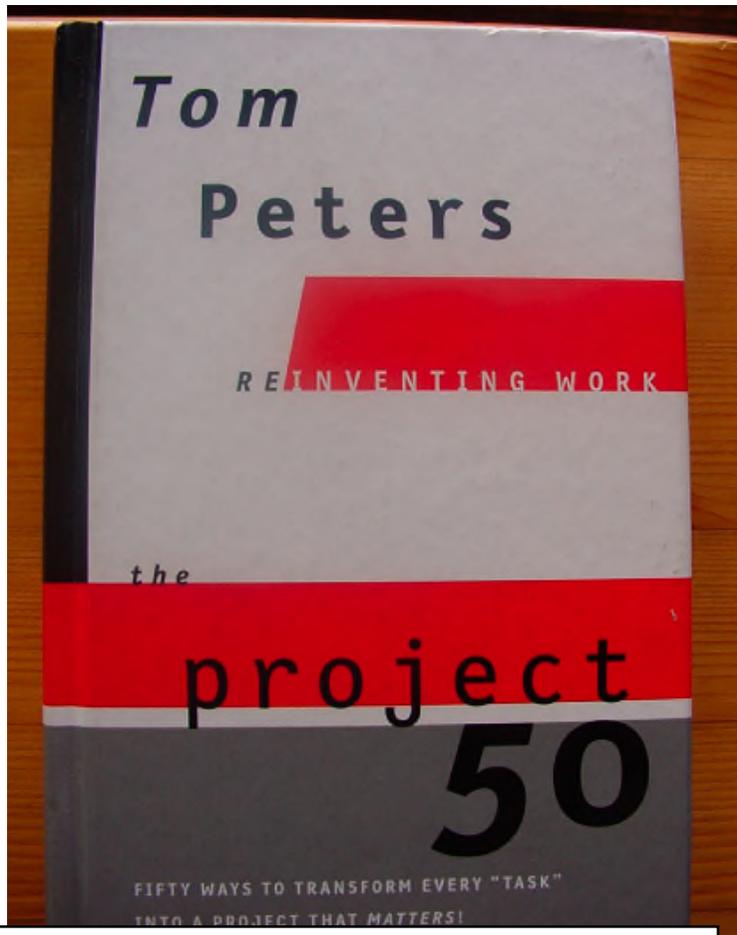


Professor Peter W G Morris

UMIST (Manchester), UCL (London)

- “The Management of Projects” (Telford, London, 1994)
- Manhattan Project to Channel Tunnel and Concorde
- Conclusion: There is no good project management method!
- Main culprit: Requirements problems
- New Model: Feedback, frequent, rapid: Plan Do Study Act, Spiral
- He did not cite, and admitted he was unaware of,
 - [Mills \(IBM Federal Systems Division\) military & space work published in 1980 \(IBM SJ No. 4\)](#)
 - Peter Morris Pwmorris@netcomuk.co.uk
 - www.INDECO.co.uk
 - Amazon.co.uk (NOT .com!)





Tom Peters

Reinventing Work, the project 50. Alfred A. Knopf, New York, 2000, ISBN 0-375-40773-1. See Peters' website www.tompeters.com, \$15.95

See also his book 'the Quick Prototype50'.

See especially his emphasis on 'quick prototyping' in relation to Evolutionary project management.

A.S.A.P.I.N.S.

As Soon As Possible If Not Sooner

“1. Now. Right now. Take some little - tiny! - element of your project. Corral a surrogate customer. Talk to him/her about it. That is ... test it. Now.

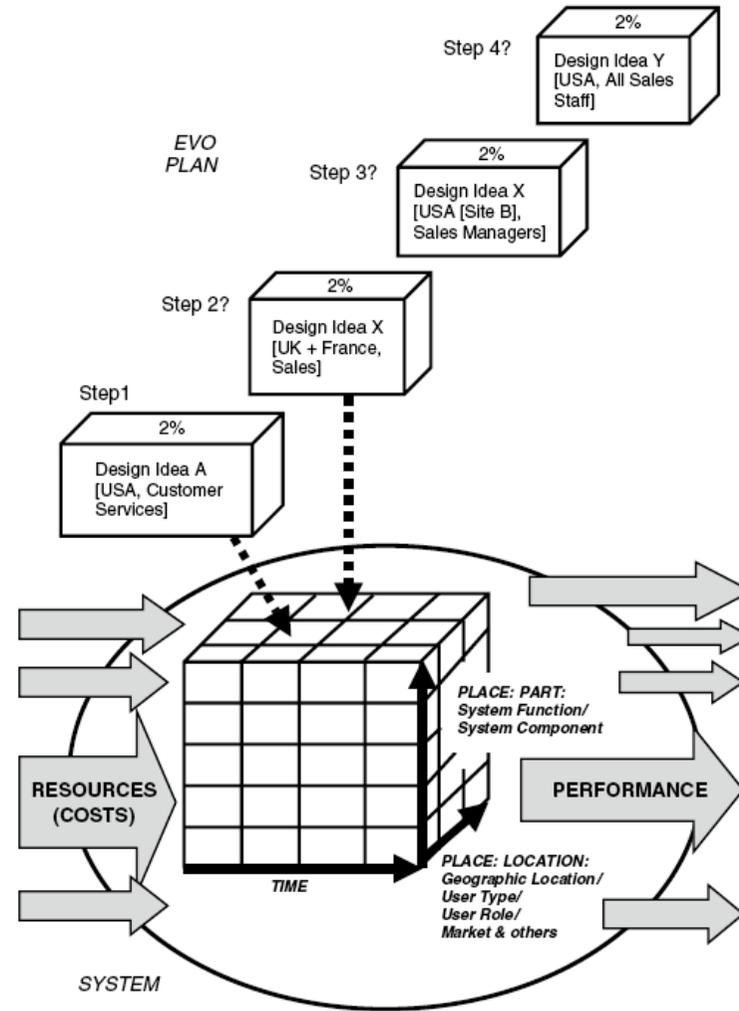
2. Your immediate goal: “Chunk up” the next three weeks. I.e.: Define a set of practical micro-bits ... that can be subjected to real-world tests..

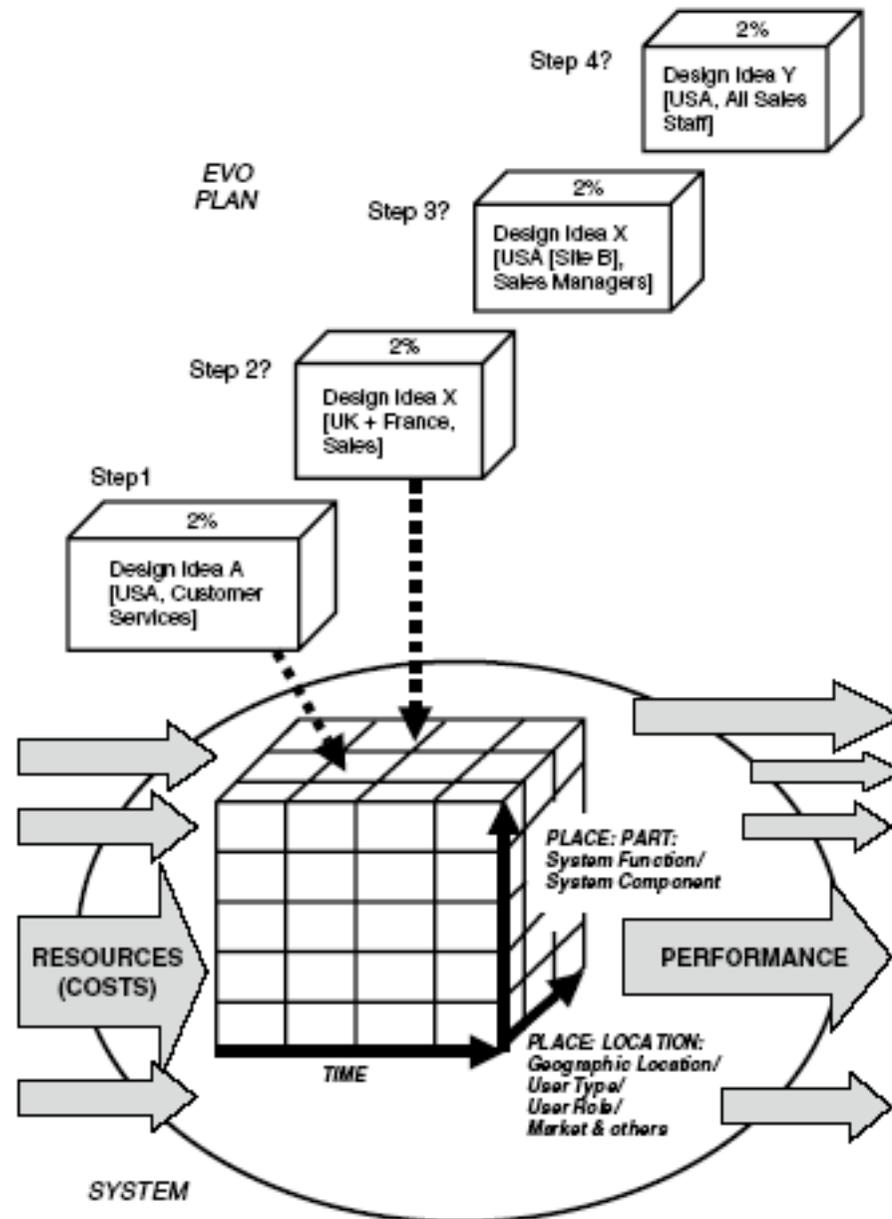
Observation: There is *no* situation - even at Boeing - where you cannot concoct a sorte-real-world-micro-test of some piece of your project .. **Within a few hours to two or three days.**

Quick Prototyping Excellence = Project Implementation Excellence. (No kidding... it's almost that basic!)”

Pages 138-9

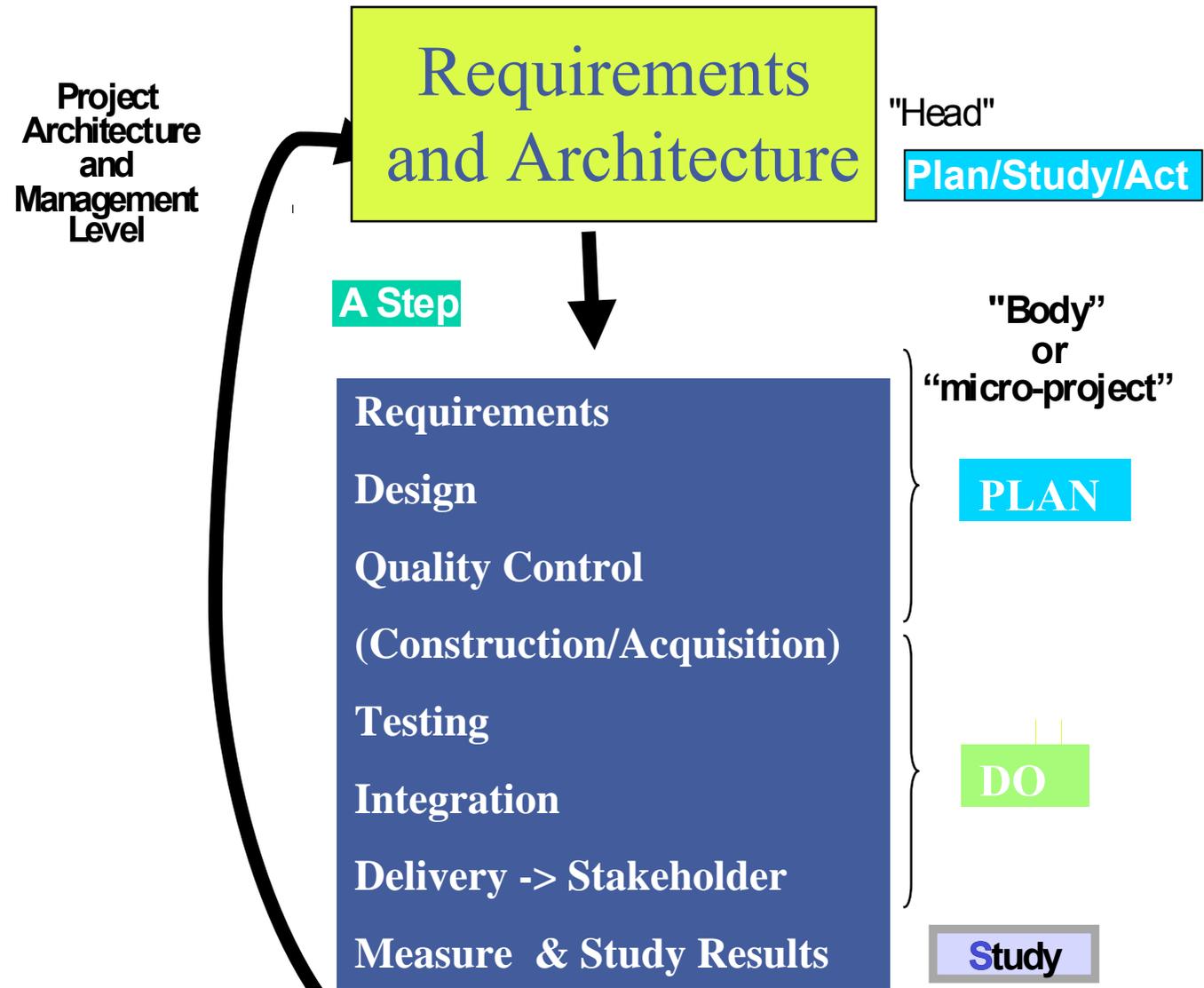
1. The Evo process description: a reusable template.





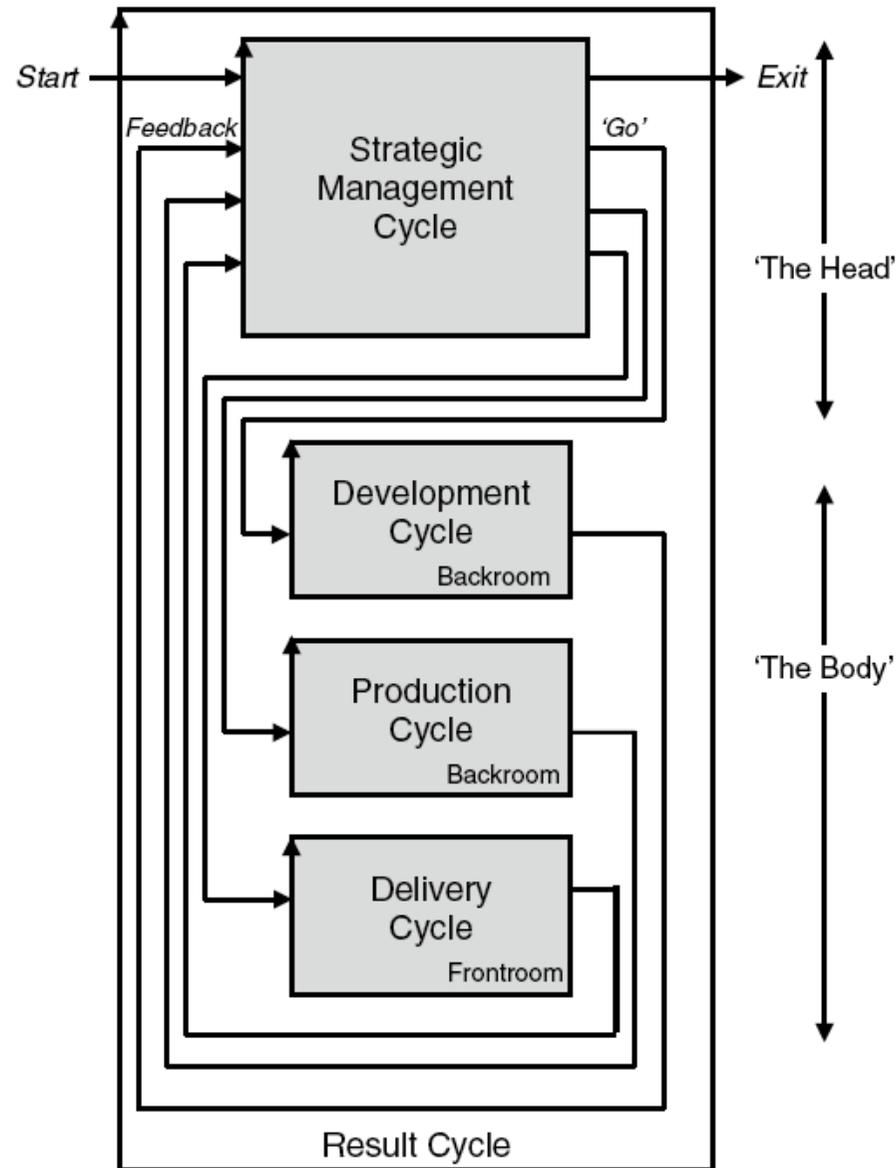
- **An Evo plan and the system: the diagram shows the steps being sequenced for delivery.**
- **Each step delivers a set of performance attributes (a subset of the long-term planned results),**
 - and consumes a set of resources (a subset of the long-term budgets),
 - in a specific place {location, system component}
 - and at a specific time (for delivering the benefits).
- **The purpose of this diagram is to show that each Evo Step will become a sub-component of the evolving system's long-term vision and plan.**
- **Source CE, Figure 10.4**

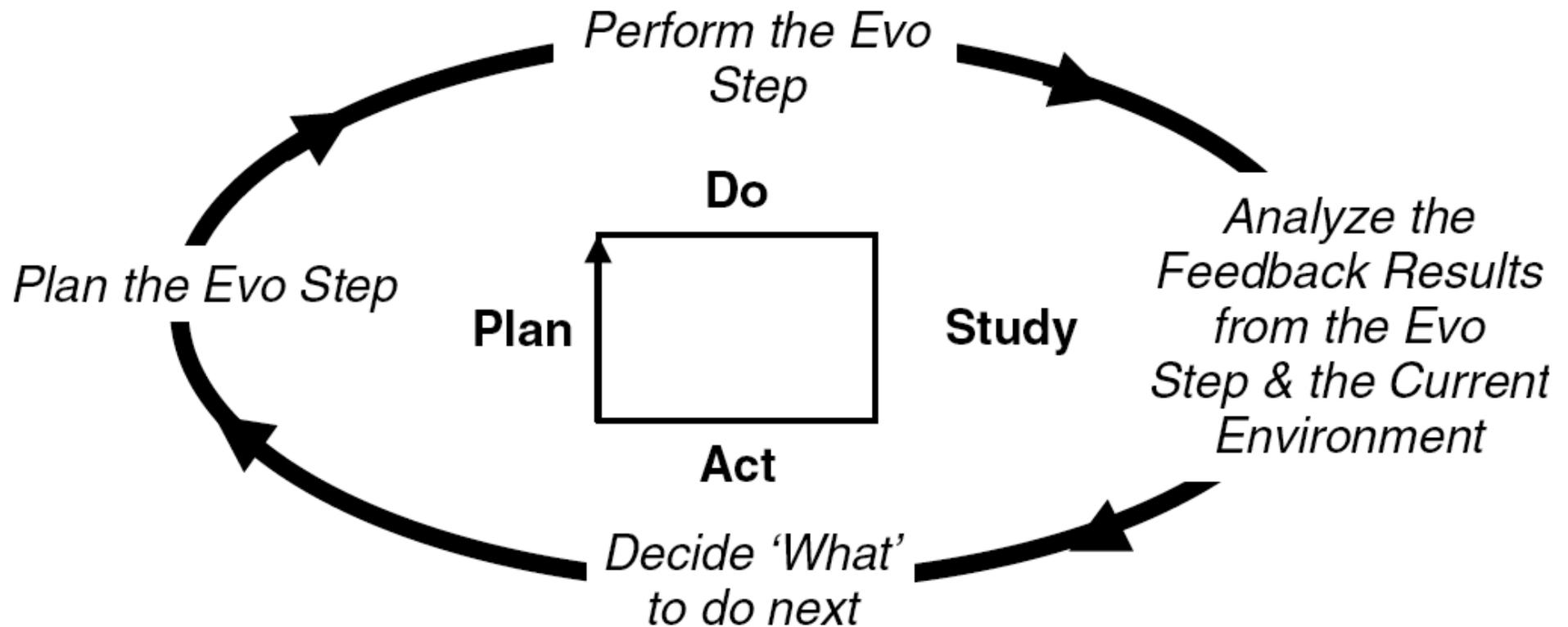
The Head:Body Model of Evo



The Result Cycle for an Evo Step

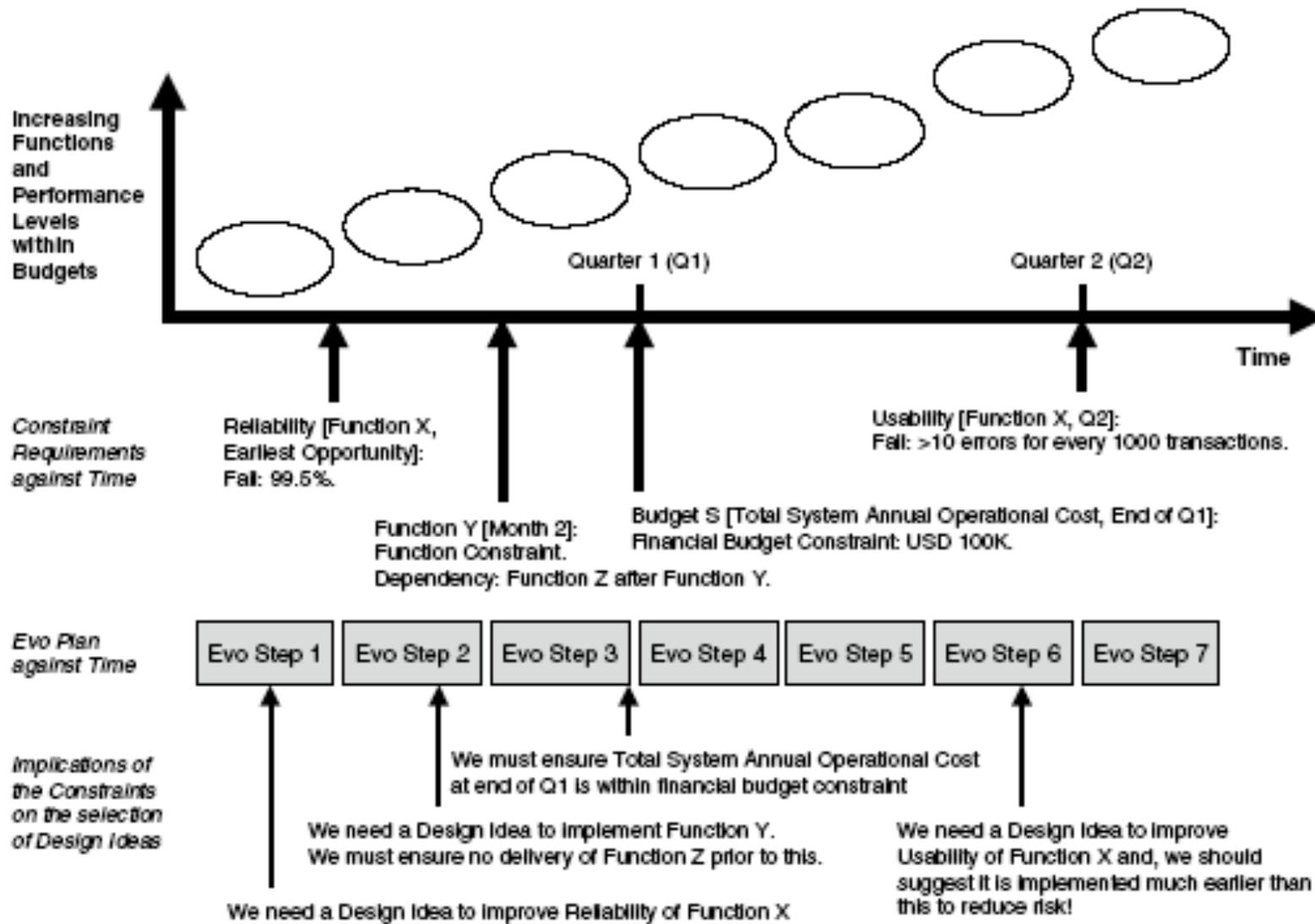
18





Source: Competitive Engineering Figure 10.3 pg 307

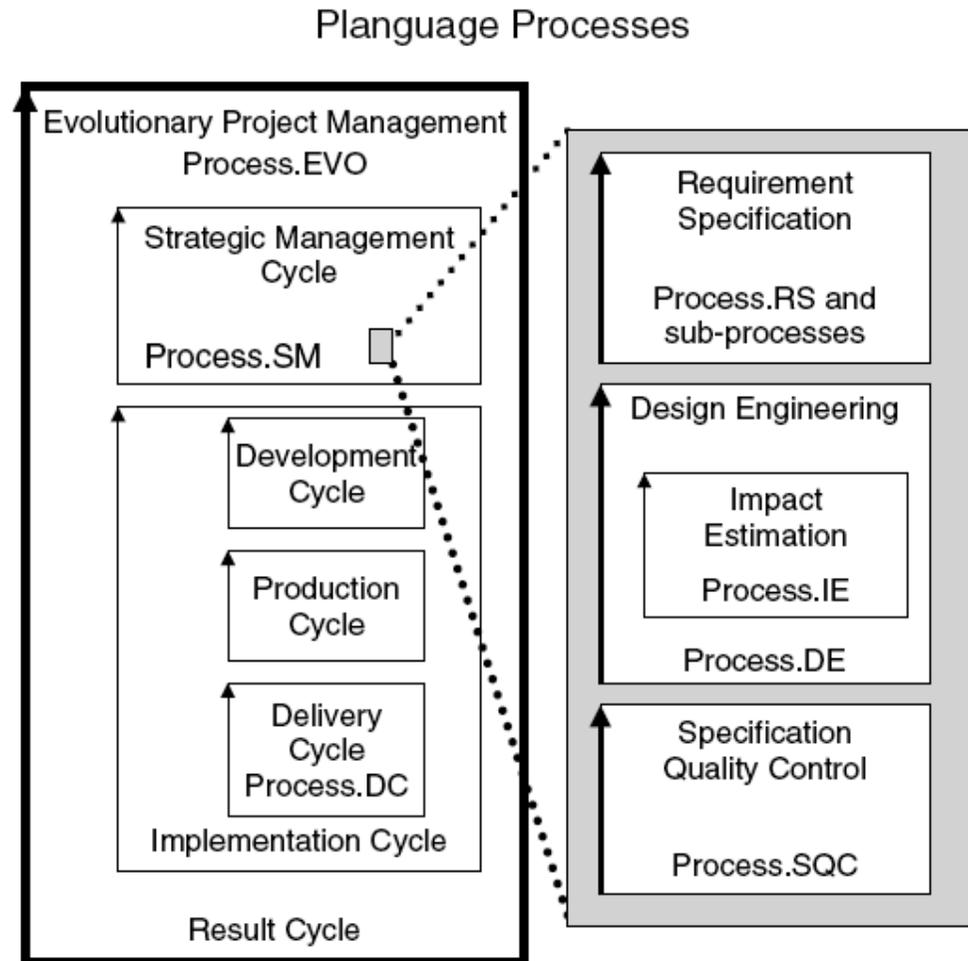
The cumulative effect of steps: Increments of function and performance



- **Rules: defined as standards for specification**
 - The Planning language, 'Planguage', defined in "Competitive Engineering" (CE) book has Rules applying to Evo for:
 - **Requirements Specification**
 - Function Requirements
 - Performance Requirements
 - Resource Requirements (cost, time, effort)
 - Generic Requirements Rules (Clarity, Consistency)
 - **Design Specification**
 - Describing a design fully
 - **Impact Estimation Table specification**
 - The multiple impacts of designs on all requirements
 - **Evolutionary Step Specification**
 - This set of rules will be illustrated in detail here.

Components of Planguage

22



Source: CE Fig 1.5

Tag: Rules.EVO. Version: October 7, 2004.

Owner: TG. Status: Draft.

Gist: Rules for Evo Plan Specification.

Base: The rules for generic specification, Rules.GS apply as well as all other Planguage rules needed to express requirements and design.

R1: Tags: All steps of an Evo plan will have a unique tag to enable cross- referencing from other specifications (such as test planning or costing).

R2: Detail: All detailed design idea specifications shall be kept separate from the Evo plan. For brevity, use Planguage step descriptions only. Any Evo plan elements yet to be defined in detail must be specified by a unique tag in fuzzy brackets (<Tag Name 1>). This will indicate that the detail is not specified yet. Rationale: We need to avoid the clutter of design idea definitions in the Evo plan itself. Tags are sufficient.

R3: Cost: Any planned step, that has an estimated incremental impact, for any resource attribute, which exceeds 5% of the total budget planned level, will be re-specified into smaller steps, to reduce risk. An average of 2%-of-budget steps is desirable (as risk of economic loss is then at 2% maximum), but individual projects may specify their own budget constraints. All planned steps still exceeding these single step budget constraints must be agreed by authorized signature.

R4: Time: Any step, which would take more than 5% of the total project calendar time (from project start up to the main long-term deadline), must be divided into smaller steps. An average of 2%-of- time steps is desirable, but individual projects may specify their own time constraints. All steps exceeding the 5% time constraint must be agreed by authorized signature. Rationale: Control time to deadline.

R5: Priority: The 'next step', at any point in the project, should ideally be selected using an Impact Estimation table to evaluate step options. Steps that you estimate to deliver the greatest stakeholder benefits, performance improvements (Sum of Percentage Impacts) to stakeholders, or that have the best performance to cost ratio, shall generally be done earliest, wherever logically possible, and when [other considerations] (such as a customer contract or request) do not have higher priority. Any specific priority factors, which override going for the greatest stakeholder benefits first, shall be clearly documented. There must be some specified clear rationale, policy or rule behind prioritizing steps differently from this rule. This could be some estimate of value of a step, which is outside the scope of the specific Impact Estimation table, which might have priority.

EXAMPLE Step 44: Type: Step. Consists Of: ABC [UK]: <- Contract Requirement 6.4. Rationale: The contract demands we deliver this step at this point. Optionally, there can be a project-defined constraint of a step having to achieve a minimum estimated value (financial growth or saving), overall performance improvement or performance to cost ratio before being considered for implementation at all.

R6: Next: Only the current step, or the approved next step, has 'commitment to implementation' (and even then, it could be terminated mid-implementation, if seen not to be delivering to plan). The sequencing specification of subsequent steps is not necessary, and is certainly not fixed. In practice, there is likely to be a tentative step sequencing mapped out, which captures any dependencies.

R7: Impact: The next step must be numerically estimated in detail for its impacts on all the critical performance and resource requirements. Other later steps may be more roughly estimated, either individually or in relevant groups. They will be estimated in greater detail as their 'turn' approaches. *Rationale: To force us to estimate, measure and consider deviation in small immediate steps.*

R8: Learn: The actual results of the steps already implemented (that is, the cumulative impacts on all requirement levels to date) and the estimated results for the next step must be specified in an IE table (see Table 10.1 example). Specific comment about negative deviations already experienced, and what you have specifically done in your plan to learn from them, should be included in some form of footnote or comment. (Note: We assume the use of an IE table, but other formats are possible.)

R9: Completeness: All the specified design ideas for a system, implemented or not, must be represented somewhere on an Evo plan. (Remember, you can use tags and you can declare a large set of designs with a single tag.

For example, A: Defined As: {B, C, D, E, F}.) Rationale: This is because failure to include all the specified design ideas somewhere on the Evo plan causes confusion. It leaves us to wonder: . Was it forgotten inadvertently? . Why is it specified, if it is planned never to be implemented? (If you are just keeping the idea in reserve, be specific.) <- CE Chapter 10

A 'Template' for Evo Step Specification

**Simplified example of filling out the
template**

An Evo Step Specification

Evo Step: Tutorial [Model 1234, Basic].

Stakeholders: {Marketing, Department XX}.

Implementers: Department XX.

Intended Audience: Marketing.

Gist: To prepare a written tutorial that teaches how to identify required information on internet web pages.

Step Content: HCTD12: <Hard Copy Text Document>. "This declares a design idea, HCTD12, that needs further detailed specification. Some additional notes about it are also given. See below."

Notes [HCTD12]:

- Can write the basic minimal functions, MMM, in 1 week. <-GF.
- Provide step by step instructions, in English.
- Questionnaire for Stakeholders.
- Intended audience: Marketing.
- Focus on <sales aspects>, not how to identify information in detail (not yet, in this step).
- Go to <specific web sites>.
- Process for Testing with Stakeholder (for example, observation, times).
- Pinpoint some characteristics of what we see on the terminal compared with what we see on a <PC or other terminal>.
- What instructions should be on the terminal to begin?
- No illustrations to be provided, just text.

Questionnaire: Defined As: Questionnaire to walkthrough with stakeholders.

Step Validation: Defined As: Process for Testing with Stakeholders. "Example observation, times."

Constraint: Step must be deliverable within one calendar week.

Assumptions [Applies = Step Cost [Effort], Source = MMM]: 10 hours per page.

Dependencies: <Feature list of WWW>, <77777 WWW Browser> <-MMM.

Risks: At least 3 hours needed of TTT's time for input and trial feedback.

Step Value:

{[Stakeholder = TTT, Saleability]: <some possibility of value>,

[Stakeholder = Developers]: <value of feedback on a tutorial>.

Step Cost [Effort]: < 10 hours <-MMM.

Process Description

1. Gather from all the key stakeholders the top few (5 to 20) most

critical goals that the project needs to deliver. Give each goal a reference name (a tag).

2. For each goal, define a scale of measure and a 'final' goal level.

For example: Reliable: Scale: Mean Time Before Failure, Goal: >1 month.

3. Define approximately 4 budgets for your most limited resources

(for example, time, people, money and equipment).

4. Write up these plans for the goals and budgets
(try to ensure this is kept to only one page).

5. Negotiate with the key stakeholders to formally agree the goals and budgets.

6. Plan to deliver some benefit

(that is, progress towards the goals)
in weekly (or shorter) increments (Evo steps).

7. Implement the project in Evo steps. Report to project sponsors after each Evo step (weekly, or shorter) with your best available estimates or measures, for each performance goal and each resource budget.

On a single page, summarize the progress to date towards achieving the goals and the costs incurred.

•Policy

- The project manager and the project will be judged exclusively on the relationship of progress towards achieving the goals versus the amounts of the budgets used.

-The project team will do anything legal and ethical to deliver the goal levels within the budgets.

- The team will be paid and rewarded for benefits delivered in relation to cost.

- The team will find their own work process and their own design.

- As experience dictates, the team will be free to suggest to the project sponsors (stakeholders) adjustments to 'more realistic levels' of the goals and budgets.

2. Basic Evo principles

The Principles of Tao Teh Ching (500 BC)

That which remains quiet, is easy to handle.

That which is not yet developed is easy to manage.

That which is weak is easy to control.

That which is still small is easy to direct.

Deal with little troubles before they become big.

Attend to little problems before they get out of hand.

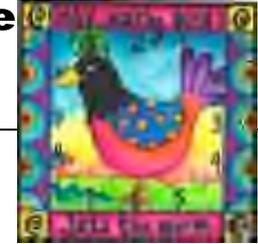
For the largest tree was once a sprout, the tallest tower started with

the first brick, and the longest journey started with the first step.

From Lao Tzu, in Bahn (1980).



Summary of Principles: Evolutionary Project Manage



1. The Principle of 'Capablanca's next move'

There is only one move that really counts, the next one.

2. The Principle of 'Do the juicy bits first'

Do whatever gives the biggest gains. Don't let the other stuff distract you!

3. The Principle of 'Better the devil you know'

Successful visionaries start from where they are, what they have and what their customers have.

4. The Principle of 'You eat an elephant one bite at a time'

System stakeholders need to digest new systems in small increments.

5. The Principle of 'Cause and Effect'

If you change in small stages, the causes of effects are clearer and easier to correct.

6. The Principle of

'The early bird catches the worm'

Your customers will be happier with an early long-term stream of their priority improvements, than years of promises, culminating in late disaster.

7. The Principle of 'Strike early, while the iron is still hot'

Install small steps quickly with people who are most interested and motivated.

8. The Principle of 'A bird in the hand is worth two in the bush'

Your next step should give the best result you can get now.

9. The Principle of 'No plan survives first contact with the enemy'

A little practical experience beats a lot of committee meetings.

10. The Principle of 'Adaptive Architecture'

Since you cannot be sure where or when you are going, your first priority is to equip yourself to go almost anywhere, anytime.

The Evo Principle of:

30

1. 'Capablanca's next move'

There is only one move that really counts, the next one.

2. 'Do the juicy bits first'

Do whatever gives the biggest gains. Don't let the other stuff distract you!

3. 'Better the devil you know'

Successful visionaries start from where they are, what they have and what their customers have.

4. 'You eat an elephant one bite at a time'

System stakeholders need to digest new systems in small increments.

5. 'Cause and Effect'

If you change in small stages, the causes of effects are clearer and easier to correct.

The Evo Principle of:

31

6. 'The early bird catches the worm'

Your customers will be happier with an early long-term stream of their priority improvements, than years of promises, culminating in late disaster.

7. 'Strike early, while the iron is still hot'

Install small steps quickly with people who are most interested and motivated.

8. 'A bird in the hand is worth two in the bush'

Your next step should give the best result you can get now.

9. 'No plan survives first contact with the enemy'²

A little practical experience beats a lot of committee meetings.

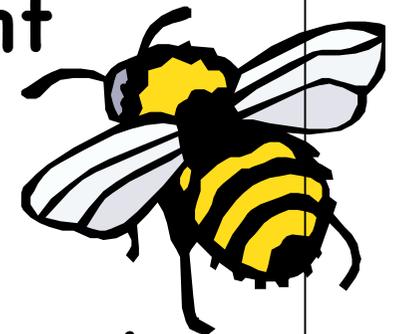
10. 'Adaptive Architecture'

Since you cannot be sure where or when you are going, your first priority is to equip yourself to go almost anywhere, anytime.

Evo as a 'Process Improvement' tactic?

32

- It can be hard to get co-operation to improve engineering processes 'in general'
- People are so busy meeting their project deadlines!
- Evo can be used for process improvement management within a project
- People have time for that
 - because it immediately benefits them
- Successful project improvements can then be made available for the rest of your organization



7 Da Vinci Principles: (Evo!)

<-Gelb, p.9

Curiosità

Insatiably curious, unrelenting quest for continuous learning

Dimostrazione

Commitment to test knowledge through experience, willingness to learn from mistakes. Learning for ones self, through practical experience

Sensazione

Continual refinement of senses. As means to enliven experience

Sfumato

Willingness to embrace ambiguity, paradox, uncertainty

Arte/Scienza

Balance science/art, logic & imagination, whole-brain thinking

Corporalità

Cultivation of grace, ambidexterity, fitness, poise

Connessione

Recognition & appreciation for interconnectedness of all things and phenomena, Systems thinking



Tom at Da Vinci birthplace 2007

Da Vinci on Practical Feedback Principle

Leonardo, proudly described himself as:

Uomo senza lettere
(man without letters)

Discepolo delle esperienze
(disciple of experience)

“To me it seems that those sciences are in vain and full of error which are not born of experience, mother of all certainty, first hand experience which in its origins, or means, or end has passed through one of the five senses.”

Source: Gelb page 78



Family at Da Vinci Museum, Vinci Italy
(Mimmo Paladino is sculpturer)



Leonardo's persistence principle

"Although generally recognized as the greatest genius of all time, Leonardo made many colossal mistakes and staggering blunders." <-Gelb

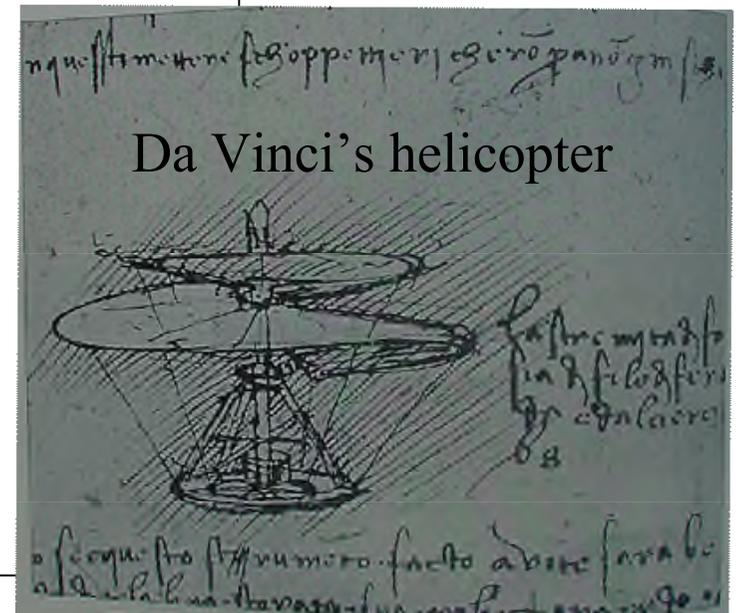
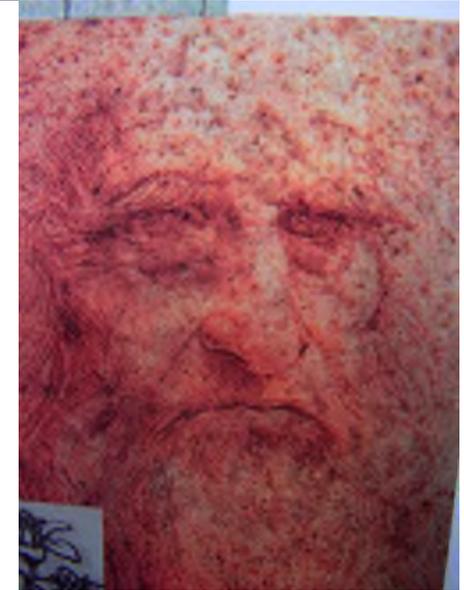
"Despite mistakes, disasters, failures, and disappointments, Leonardo never stopped learning, exploring, and experimenting. He demonstrated Herculean persistence in his quest for knowledge." <- Gelb

Leonardo wrote: <-Gelb p.79

"I do not depart from my furrow.

"Obstacles do not bend me"

"Every obstacle is destroyed through rigor"



Da Vinci's helicopter



Sol Gilb, view from
Da Vinci's Birth Home,
<- 2007

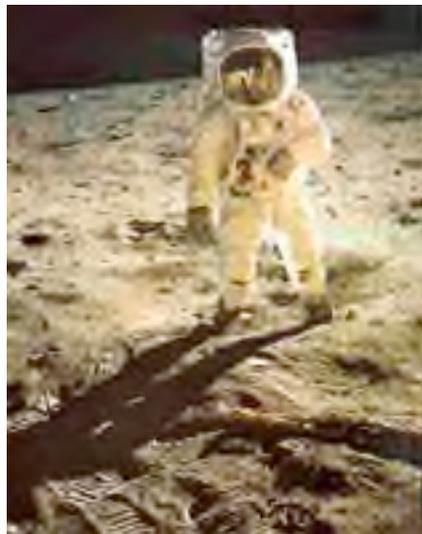
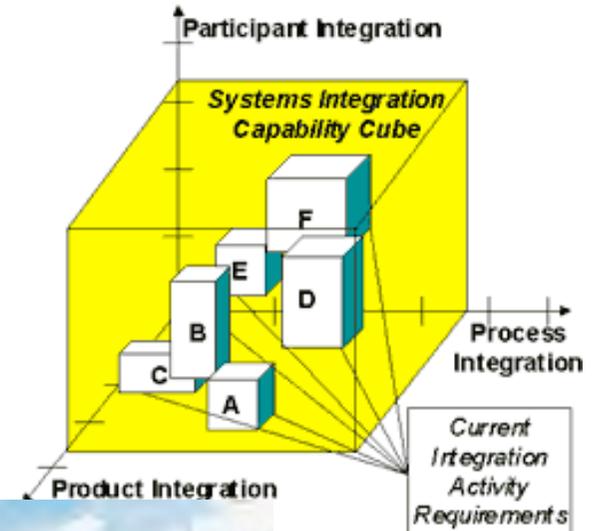
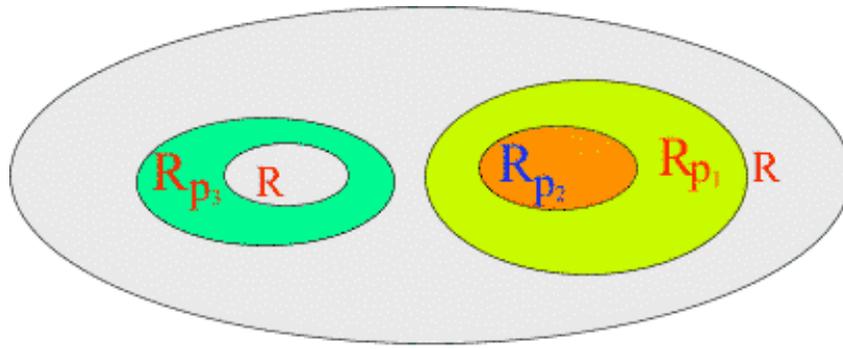
When You Do Not Need Evo

36

You do not need Evo if

1. There is no instability of requirements
2. There is no pressure on resources, to meet requirements
3. There is no volatility (frequent change) on the cost-or-ability of technology
4. There is no 'corruption', under pressure, to carry out planned 'architecture'
5. There is no need for early deliveries
6. Lateness of everything , by factor 3.14, is tolerable
7. Nobody is 'green',
(everybody knows all they need to know about the complex new advanced state-of-the-art system they are building: nothing to learn)

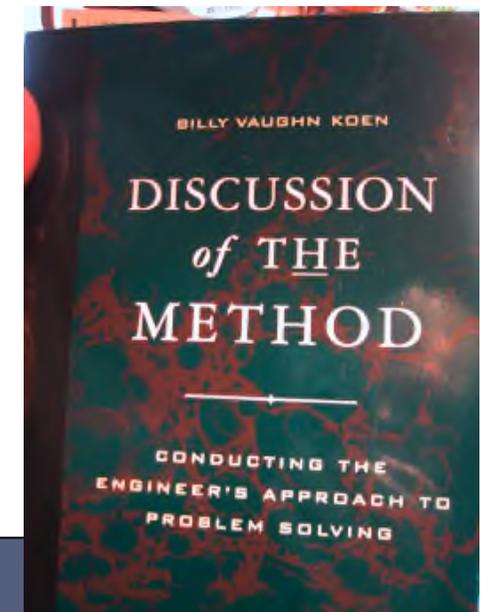
3. Principles for decomposing into small Evo steps.



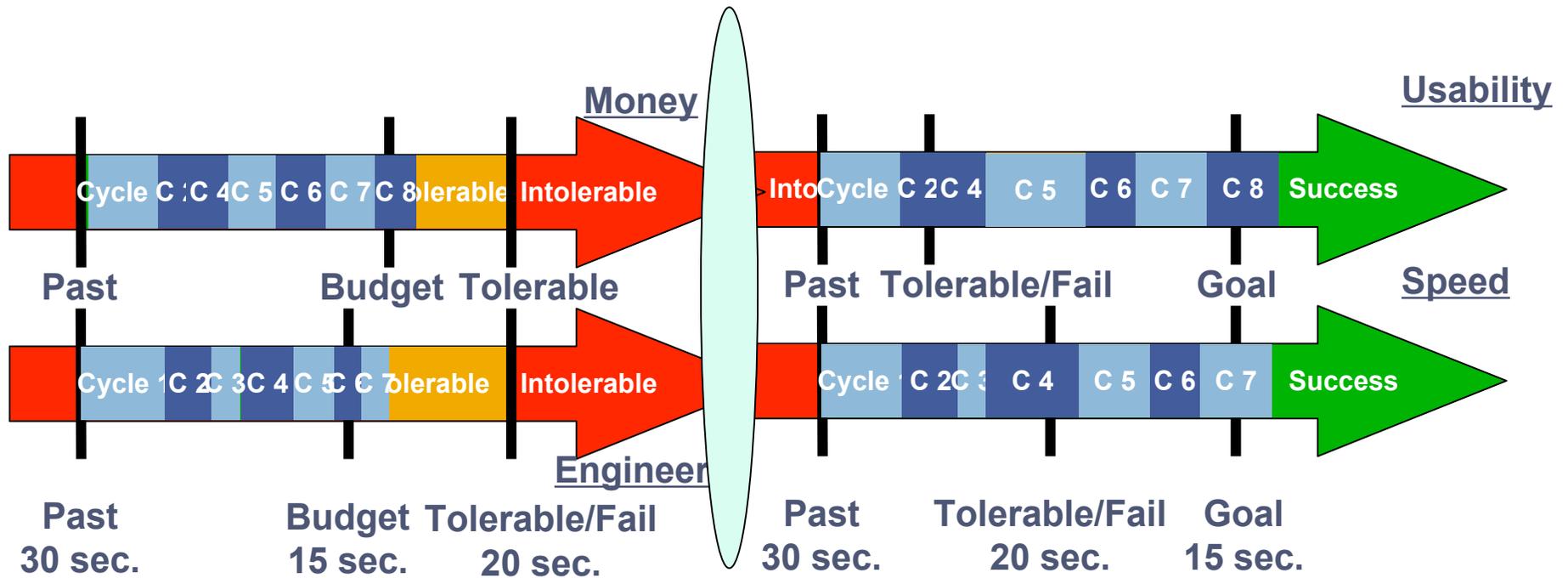
Make small changes in the sota:

'Sota' = Engineering State Of The Art Heuristics ←Koen,
Discussion, p. 48

**Always give yourself a chance to retreat; and
Use feedback to stabilize the design process**



Each Evolutionary Cycle uses a constrained budget of Development Resources



Descartes On Small

“We should bring the whole force of our minds to bear upon the most minute and simple details and to dwell upon them for a long time so that we become accustomed to perceive the truth clearly and distinctly.”

Rene Descartes, *Rules for the Direction of the Mind*, 1628



SUMMARY SLIDE: Decomposition is a teachable discipline
How to decompose systems into small evolutionary steps: (a list of practical tips)

41

1. Believe there is a way to do it, you just have not found it yet!
2. Identify obstacles, but don't use them as excuses: use your imagination to get rid of them!
3. Focus on some usefulness for the stakeholders: users, salesperson, installer, testers or customer. However small the positive contribution, something is better than nothing.
4. Do not focus on the design ideas themselves, they are distracting, especially for small initial cycles. Sometimes you have to ignore them entirely in the short term!
5. Think one stakeholder. Think 'tomorrow' or 'next week.' Think of one interesting improvement.
6. Focus on the results. (You should have them defined in your targets. Focus on moving towards the goal and budget levels.)
7. Don't be afraid to use temporary-scaffolding designs. Their cost must be seen in the light of the value of making some progress, and getting practical experience.
8. Don't be worried that your design is inelegant; it is results that count, not style.
9. Don't be afraid that the stakeholders won't like it. If you are focusing on the results they want, then by definition, they should like it. If you are not, then do!
10. Don't get so worried about "what might happen afterwards" that you can make no practical progress.
11. You cannot foresee everything. Don't even think about it!
12. If you focus on helping your stakeholder in practice, now, where they really need it, you will be forgiven a lot of 'sins'!

13. You can understand things much better, by getting some practical experience (and removing some of your fears).
14. Do early cycles, on willing local mature parts of your user/stakeholder community.
15. When some cycles, like a purchase-order cycle, take a long time, initiate them early (in the 'Backroom'), and do other useful cycles while you wait.
16. If something seems to need to wait for 'the big new system', ask if you cannot usefully do it with the 'awful old system', so as to pilot it realistically, and perhaps alleviate some 'pain' in the old system.
17. If something seems too costly to buy, for limited initial use, see if you can negotiate some kind of 'pay as you really use' contract. Most suppliers would like to do this to get your patronage, and to avoid competitors making the same deal.
18. If you can't think of some useful small cycles, then talk directly with the real 'customer', stakeholders, or end user. They probably have dozens of suggestions.
19. Talk with end users and other stakeholders in any case, they have insights you need.
20. Don't be afraid to use the old system and the old 'culture' as a launching platform for the radical new system. There is a lot of merit in this, and many people overlook it.

.Working within many varied technical cultures since 1960 I have never found an exception to this – there is always a way!

Advice on finding smaller implementation cycles when it seems difficult or impossible to achieve.

42

PART 1 of 4

Believe there is a way to do it,
you just have not found it yet!

Identify obstacles,
but don't use them as excuses: use your imagination to get rid of them!
(Garfield: Peak Performers)

Focus on some usefulness for the user or stakeholder,
however small.

Do not focus on the *design* ideas themselves,
they are distracting, especially for small initial cycles.
Focus on getting results and feedback.

Think; one stakeholder, tomorrow, one interesting improvement.
When that succeeds, multiply it.

Advice on finding smaller implementation cycles when it seems difficult or impossible to achieve.

43

PART 2 of 4

Focus on the results

(which you should have defined in your goals, moving toward Goal levels).

Don't be afraid to use temporary-scaffolding designs.

Their cost must be seen in the light of the value of making some progress, and getting practical experience.

Don't be worried that your design is inelegant;

it is results that count, not style.

Don't be afraid that the stakeholder won't like it.

If you are focusing on results they want, then by definition, they should like it. If you are not, then do!

Don't get so worried about "what might happen afterwards"

that you can make no practical progress.

You cannot foresee everything.

Don't even think about it!

Advice on finding smaller implementation cycles when it seems difficult or impossible to achieve.

44

PART 3 of 4

Focus on helping your stakeholder in practice, now,
where they really need it, you will be forgiven a lot of "sins"!

Get some practical experience

(and removing some of your fears). You will understand things
much better,

Do early cycles,

on willing local mature parts of your user community.

**When some cycles, like a purchase-order cycle, take a
long time,**

initiate them early, and do other useful cycles while you wait.
(Parallel activity is OK!)

**If something seems to need to wait for "the big new
system",**

ask if you cannot usefully do it with the 'awful old system', so as
to pilot it realistically, and perhaps alleviate some 'pain' in the
old system.

Almost ALWAYS start Evo from the existing system.

Advice on finding smaller implementation cycles when it seems difficult or impossible to achieve.

45

PART 4 of 4

If something seems too costly to buy, for limited initial use,

see if you can negotiate some kind of "pay as you really use" contract. Most suppliers would like to do this to get your patronage, and to avoid competitors making the same deal.

If you can't think of some useful small cycles,

then talk/observe directly with the real "stakeholder" or end user. They probably have dozens of suggestions.

Talk with end users in any case,

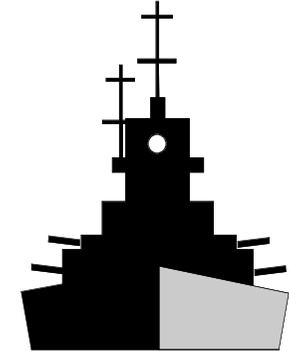
they have insights you need.

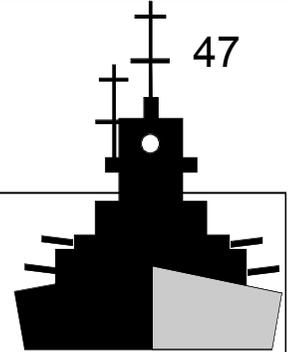
Don't be afraid to use the old system and the old "culture"

as a launching platform for the radical new system. There is a lot of merit in this, and many people overlook it.

Here is a case of finding small steps when the client did not believe there were any

- Two principles are used to solve the decomposition problem
- The entire sequence took about 30 minutes over a meal





Once, when holding a public course on the EVO method in London, a participant came to me in the first break and said he did not think he could use this early Evolutionary method.

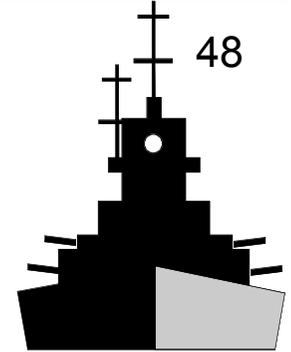
Why?

"Because my system is to be mounted on a new ship not destined to be launched for three years."

The Barrier:

"It cannot be done until the new {thing, building, organization, system}.... is ready in some years time".





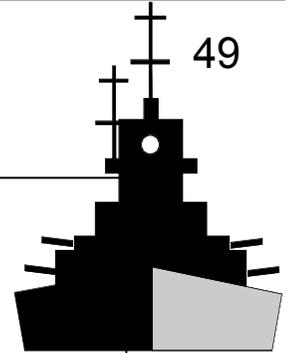
Faith:

I did not know anything about his system, at that point. But I expressed confidence that there is always a solution, and bet that we could find one during the lunch hour.

The Case:

He started our lunch by explaining that his weapons research team made a radar-like device that had two antennas instead of the usual one, which had their signals analyzed by a computer before presenting their data. It was for ship-and-air traffic, surrounding the ship it was on.





The Shift of attention:

I made a stab at the "results" he was delivering, and who his "customer" was, two vital pieces of insight for making Evolutionary delivery plans.

"May I assume that the main result you provide is "increased accuracy of perception", and that your "customer" is Her Majesty's Navy?"

"Correct." He replied.

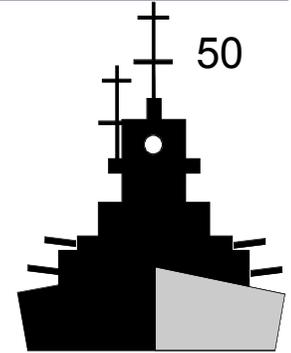
"Does your 'box' work more or less, now, in your labs?", I ventured. (Because if it did, that opened for immediate use of some kind)

"Yes", he replied.

"Then what is to prevent you from putting it aboard one of Her Majesty's current ships, and ironing out any problems in practice, enhancing it, and possibly giving that ship increased capability in a real war?" I tried, innocently.

"Nothing!", he replied. And at that point I had won my bet, 20 minutes into the lunch.





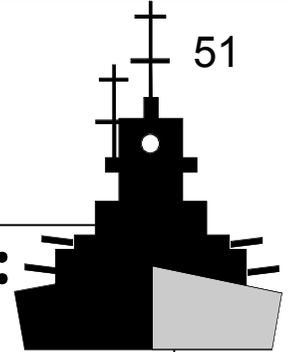
"You know, Tom", he said after five minutes of silent contemplation, "the thing that really amazes me, is that not one person at our research labs has ever dared think that thought!".

The necessary insights:

the customer was not the new ship,
and the project was not to put the
electronics box on the new ship.

The project was to give increased perception
to the real customer, The Royal Navy.





Notice the “method” emerging from this example:

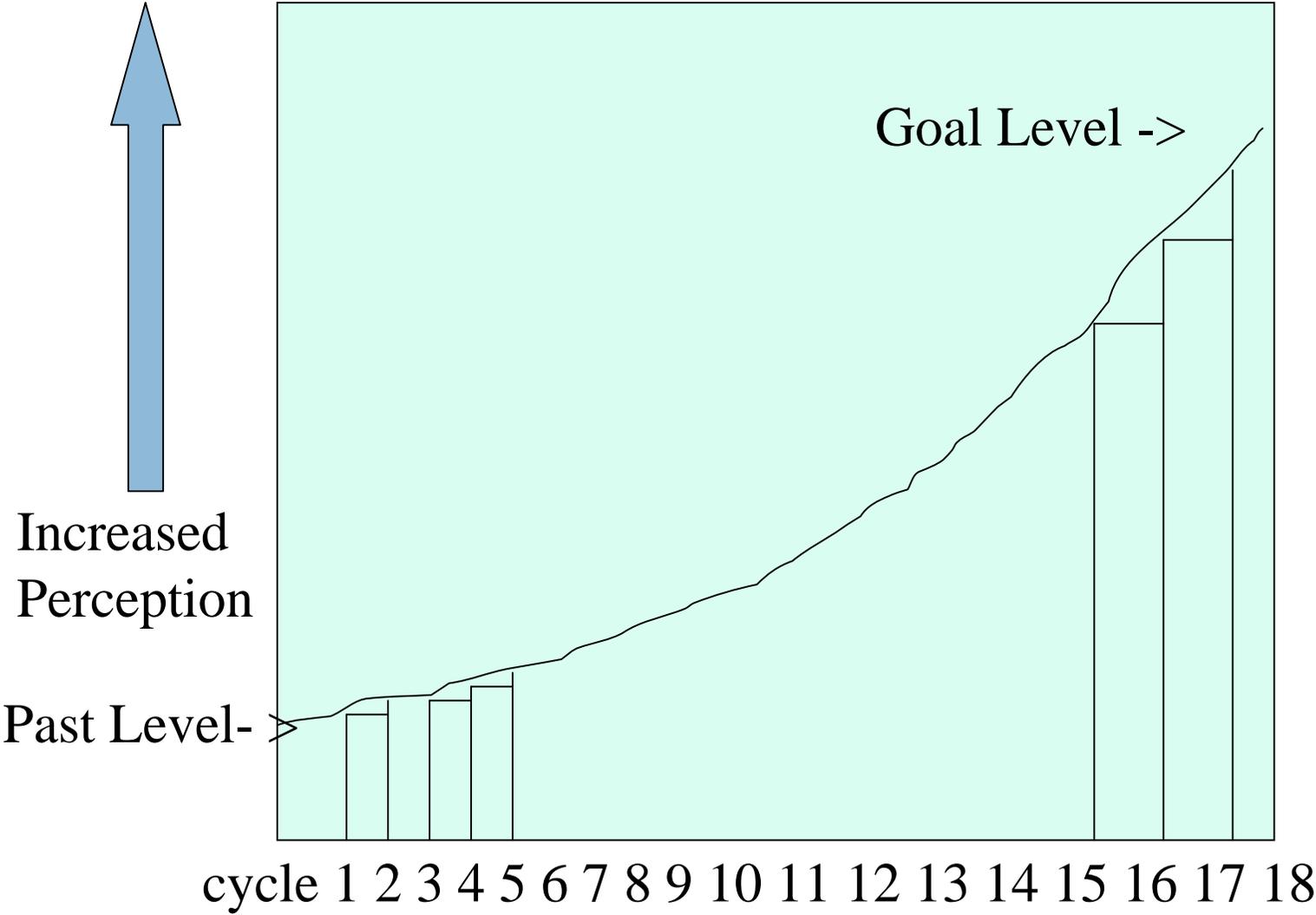
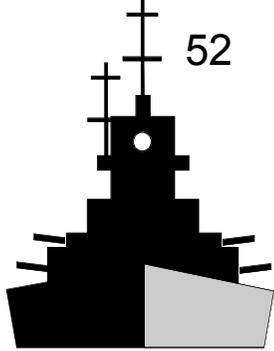
1. Identify the real stakeholder, and plan to deliver results to them.
2. Identify the real improvement results and focus on delivering those results to the real stakeholder.

in other words:

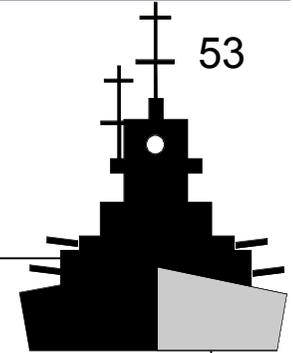
1. Do not get distracted by intermediaries (the new ship)
think “The Royal Navy” or even “The Western Alliance”.
2. Do not get distracted by the perceived project product (the new radar device for the new ship):
think “increased accuracy of perception”.



The Naval Weapons System: Evo increase of Perception. slide 6 of 7



The Naval Weapons System: Lessons Learned (7 of 7)

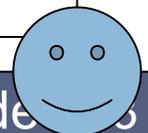


Evolutionary Projects are not normal thinking even amongst well educated engineers.

Evo is a systems method not limited to a software method

Focus on 'evolving' the results of the project (increased accuracy of perception, not 'deliver a black box')

Focus on your real stakeholder (The Royal Navy, not a ship)



Philips Evo Pilot May 2001

# Jobs	Week	[- 5%,+10%]	[-10%,+20%]	[-15%,+30%]	out of range			
6	wk 8	1	5					
11	wk 9	3	1	7				
19	wk 10	6	3	7	3			
25	wk 11	6	4	6	9			
25	wk 12	17		3	5			
42	wk 13	31			3	2	6	
55	wk 14	37			11		1	6
55	wk 15	39			9		1	6
55	wk 16	48				4	1	2
55	wk 17	50				4		1

Frank van Latum
The Manager



The GxxLine PXX Optimizer EVO team proudly presents the success of the Timing Prediction Improvement EVO steps. Shown are the results of the test set used to monitor the improvement process.

The size of the test set has grown, as can be seen in the first column. (In the second column the week number is shown.)

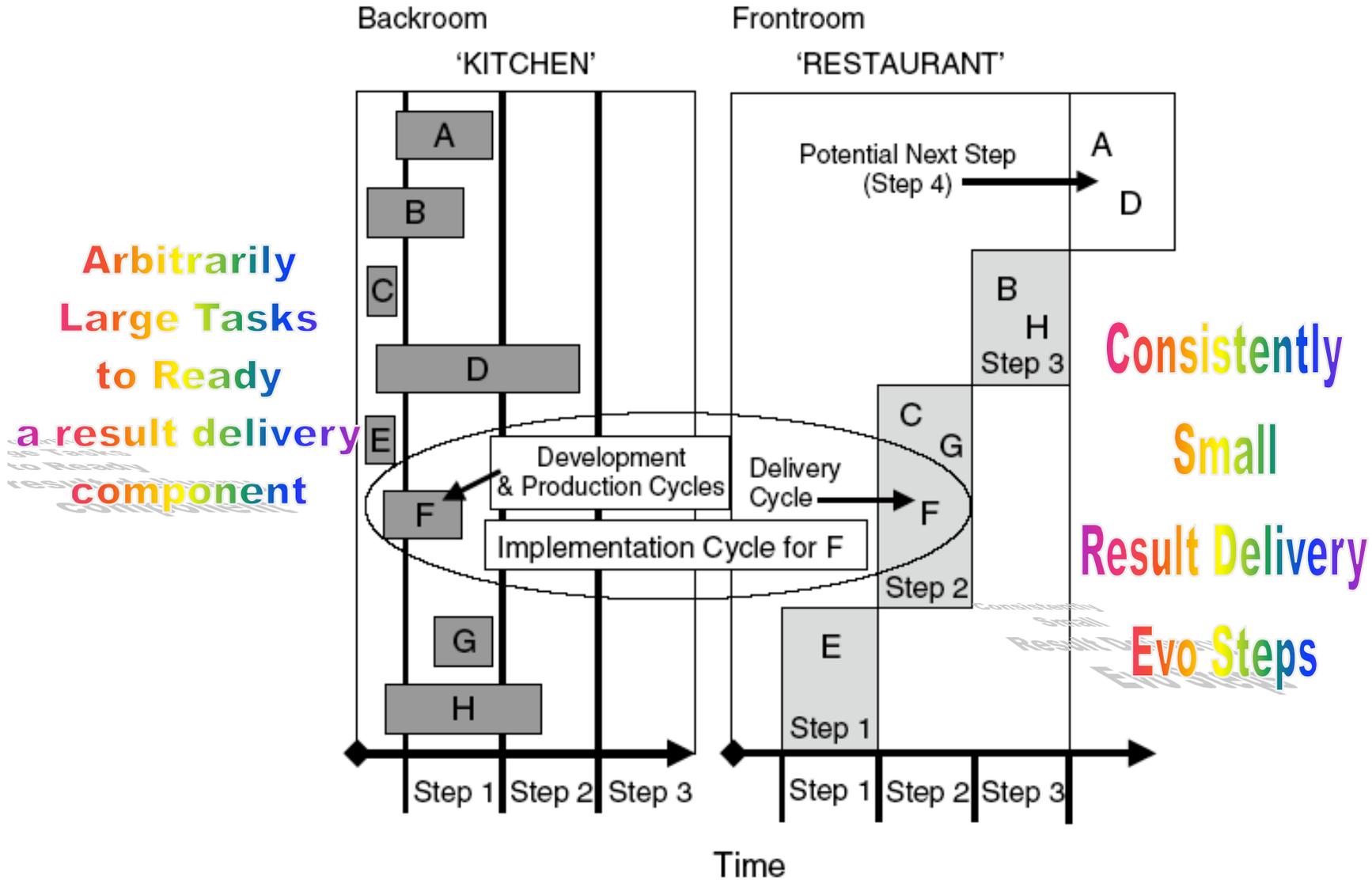
We measured the quality of the timing prediction in percentages, in which -5% means that the prediction by the optimizer is 5% too optimistic.

Excellent quality (-5% to +10%) is given the color green, very good quality quality is yellow, good quality is orange, & the rest is red. The results are for the ToXXXz X(i) and EXXX X(i), and are accomplished by thorough analysis of the machines, and appropriate adaptation of the software.

The GXXline Optimiser Team presented the word document below to the Business Creation Process review team.

The results were received with great applause. The graphics are based on the timing accuracy scale of measure that was defined with Jan verbakel. Classification: Unclassified

Backroom and Frontroom



The Backroom and Frontroom model

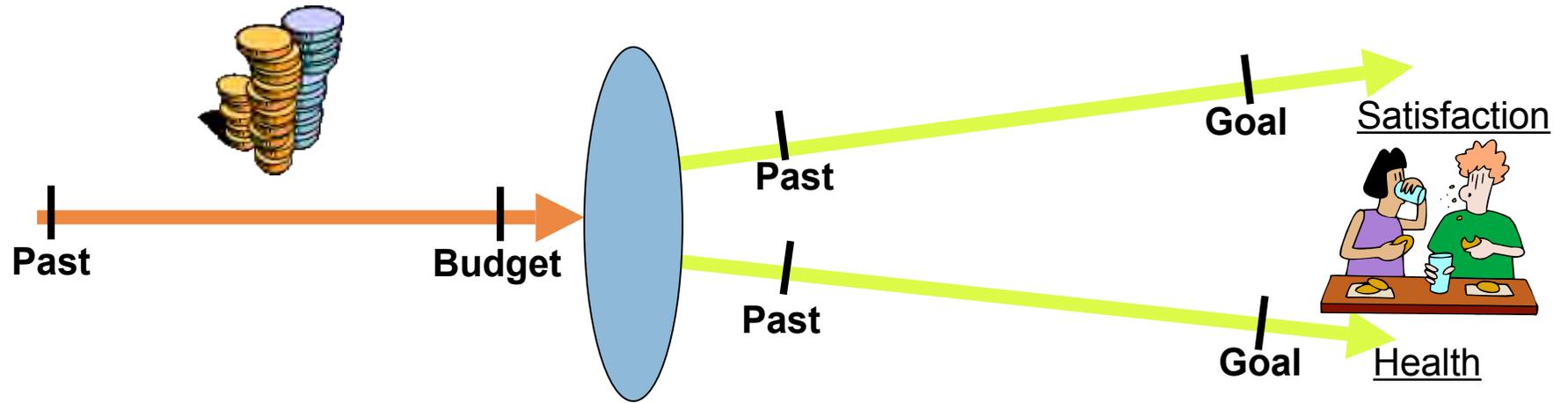
helps decompose into small steps

56

● Because:

- Long-duration Tasks, exceeding an Evo Step length, are carried out in the backroom, using whatever time is necessary
- The backroom tasks need to be started early enough to be able to probably deliver them when stakeholder want them.
- Small Evo steps only apply to the 'Frontroom'.
 - They are the cycle of result delivery.
 - They are a cycle of installation and integration - from the stakeholder point of view.
- If a task is delayed, or takes a long time, then we will try to find something else to deliver on a regular cycle.
- The Backroom contains (hopefully) a set of potential result deliveries.
- The frontroom decides when and where to deliver these ready potential results.

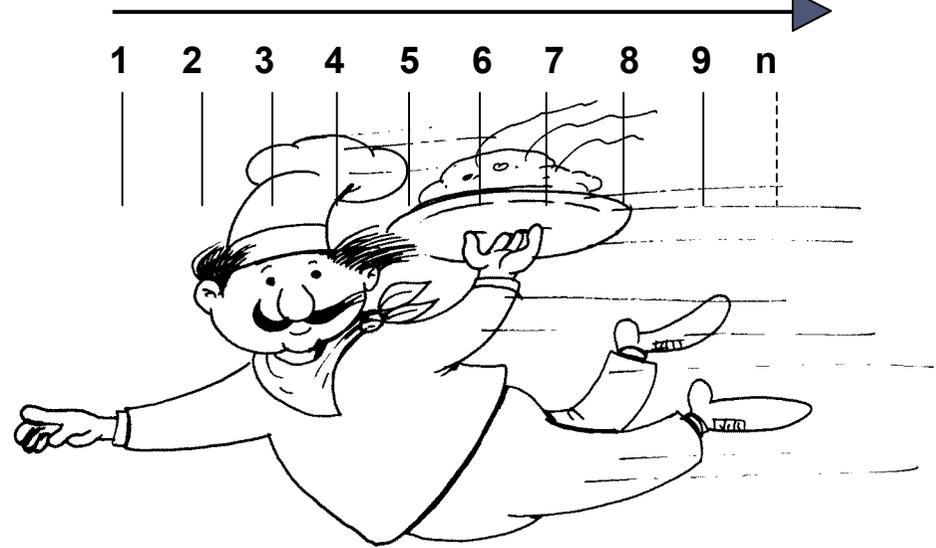
Costs / Effects

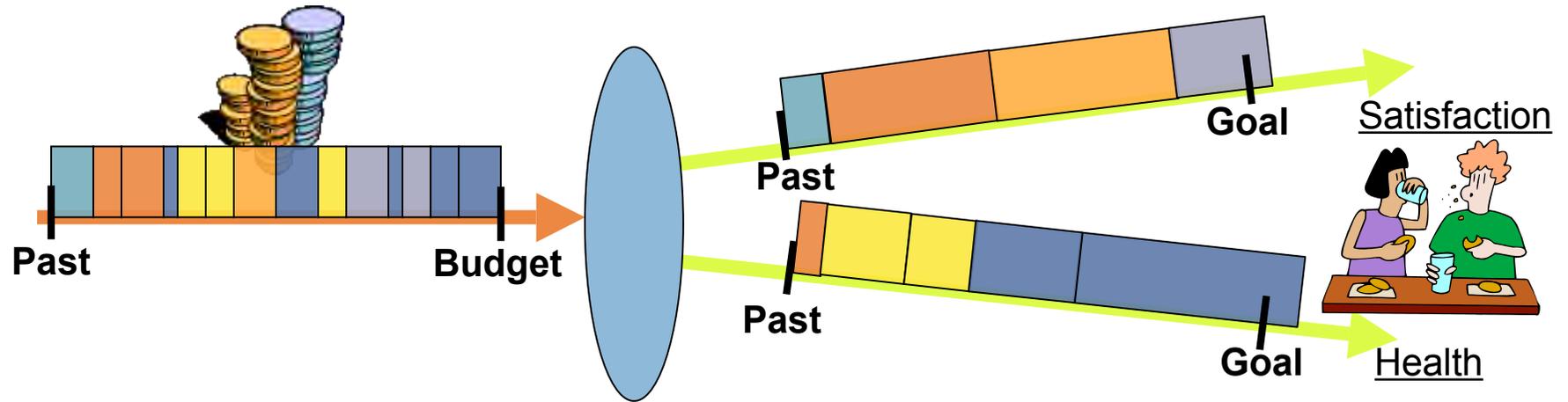


Back-room Design Development

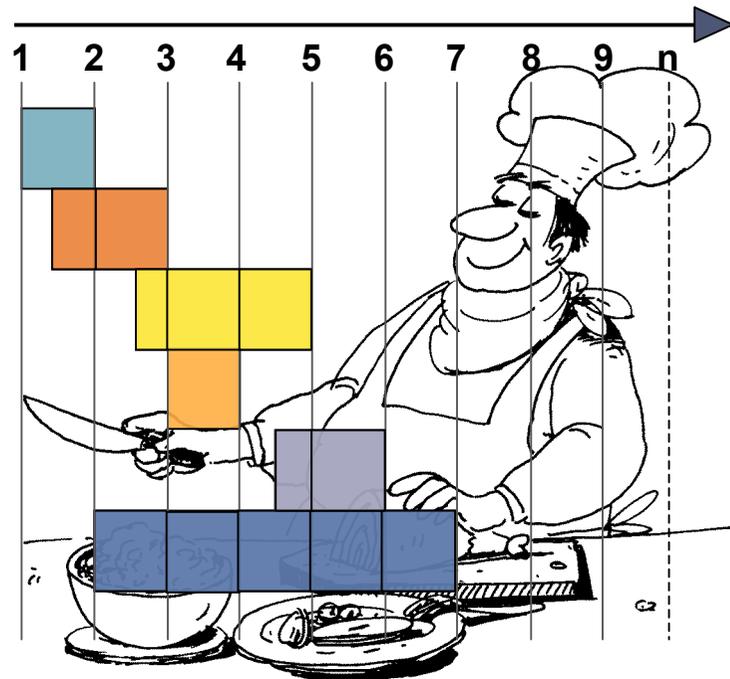


Front-room Evolutionary Delivery

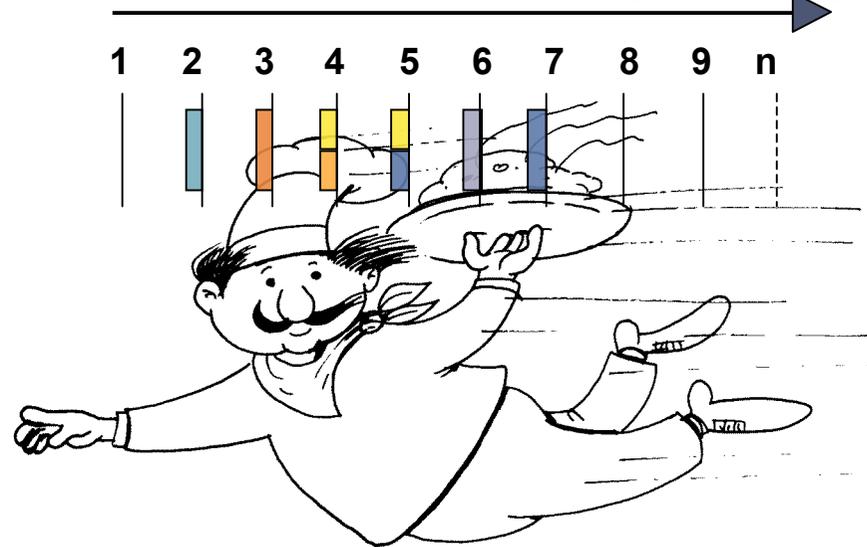




Back-room Design Development



Front-room Evolutionary Delivery



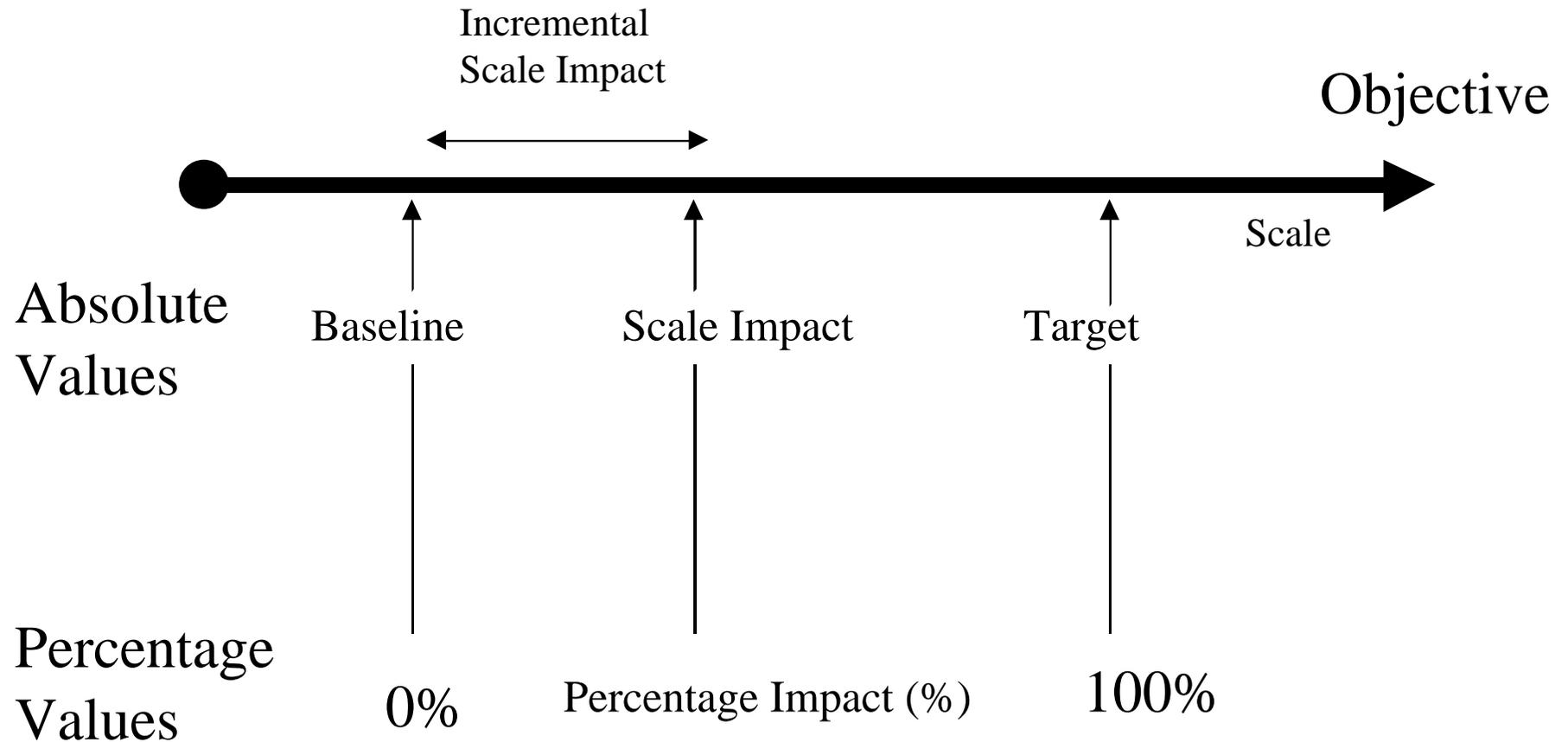
Using the Impact Estimation Table to decompose into Evo Steps

Method: Look for high impact, and for high impact for resource use, to spot promising rough (many steps) and detailed (single step) Evo steps

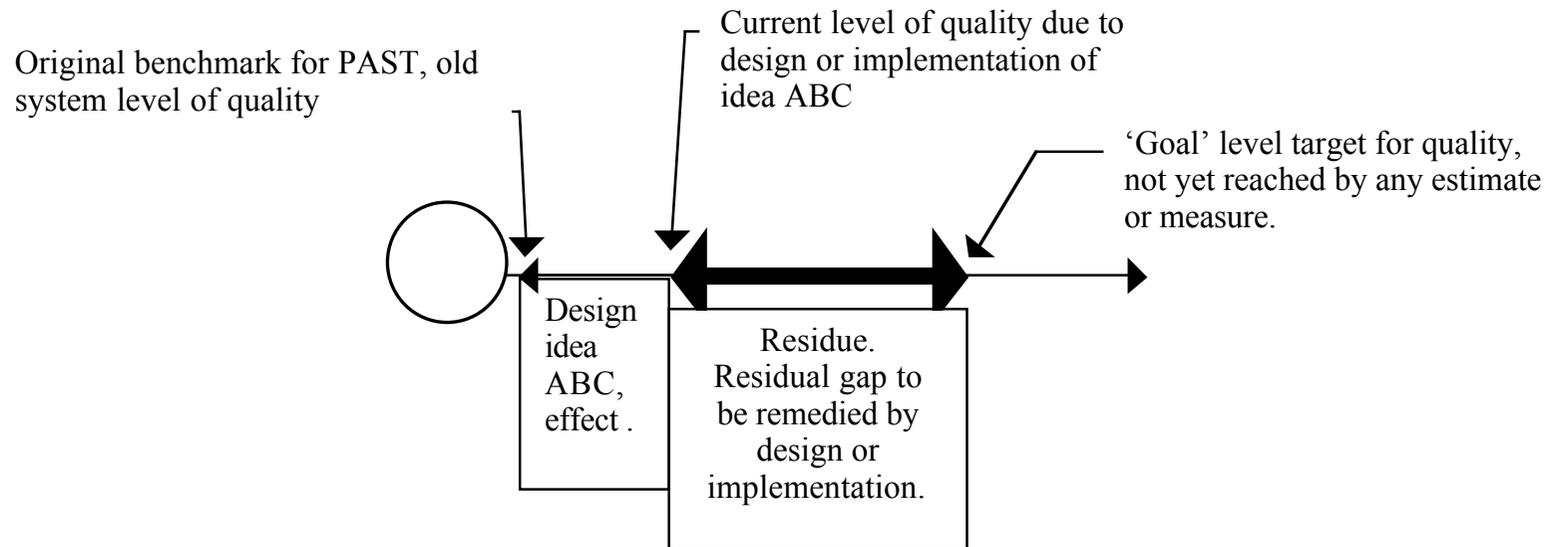
For more info on Impact Estimation,
see Gilb, Competitive Engineering (IE Chapter), or
Free downloads at www.gilb.com (books (Evo, PM) and papers)

Impact Estimation Basic Concepts

60



Source: Lindsey Brodie, Editor of Competitive Engineering May 2000



We must estimate, or measure, the numeric cumulative impact of the design

on a defined Scale,
using a defined Meter,
with respect to target and constraint levels.

'Cumulative' - the effect it has after other designs are in place.

Consider synergy effect of other designs ($2+2 > 4$)

Consider thrashing effect of other designs ($2+2 < 4$)

Nordic Road Building Software IE

“Look for high impact numbers” to identify promising Evo steps

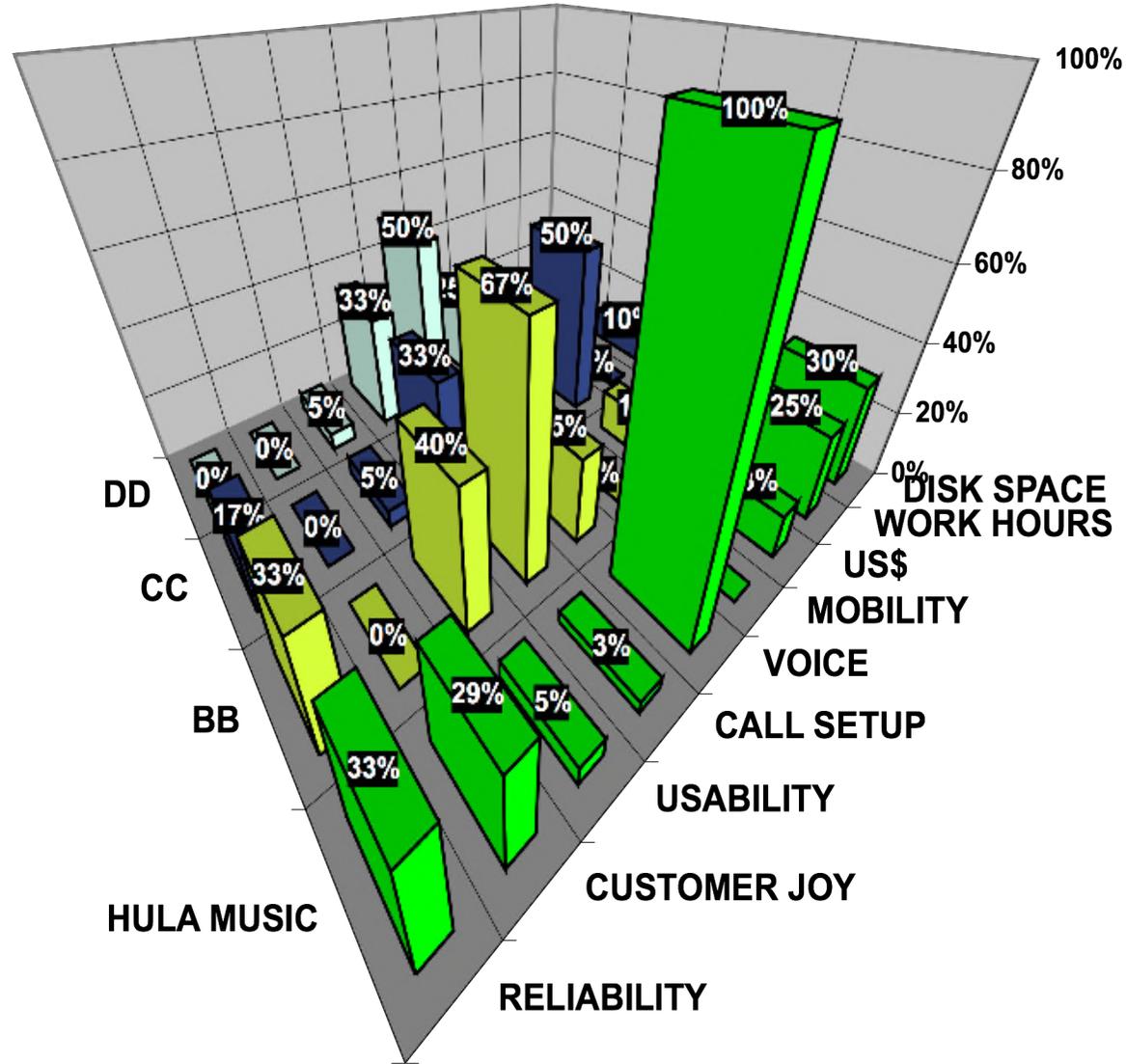
	Road Design Functions										
	Road Design Functions						Road Data Model		Drawing Production		CAD Compor
	Road Standard (Requirements)	Road Network	Alignment Design	Road modelling	Intersection modelling (3D!)	Analyse the Design	Storage of road model	Storage of Alignments	Drawing Functions	Drawing Factory	
Product Qualities											
Efficiency.Design,	5%	30%	20%	40%	15%	20%	10%	15%	30%	20%	0%
Efficiency.Construction	0%	5%	0%	40%	20%	10%	10%	0%	0%	0%	0%
Efficiency. Facility management	0%	20%	0%	10%	5%	0%	10%	10%	0%	0%	0%
Efficient.Localisation	-20%	0%	0%	0%	15%	-5%	10%	0%	30%	20%	0%
Quality.Localisation	-20%	0%	0%	0%	0%	0%	10%	0%	20%	15%	0%
Usability.Learnability	0%	10%	30%	30%	15%	-5%	5%	10%	10%	10%	0%
Usability.Intuitive	-5%	10%	20%	30%	15%	-5%	10%	10%	10%	10%	0%
Usability.Fun	10%	10%	20%	20%	10%	5%	5%	0%	15%	15%	0%
Usability.Workflow	20%	40%	10%	20%	15%	0%	5%	10%	10%	10%	0%
Availability.Reliability	0%	-10%	-10%	-10%	-10%	0%	10%	0%	5%	5%	0%
Availability.Maintainability	0%	-10%	-10%	-10%	-10%	0%	10%	0%	5%	5%	0%
Availability.Scaleability	0%	-10%	-10%	-10%	20%	0%	20%	0%	10%	10%	0%
Portability	0%	0%	0%	0%	20%	0%	15%	10%	10%	10%	0%
Identity. Novapoint	30%	30%	30%	0%	10%	15%	30%	10%	5%	5%	0%
	20%	125%	100%	160%	140%	35%	160%	75%	160%	135%	0%
Engineers.Innhouse											
	15,000	300	1000	80	1000	1000	100	2500	100	0	
Engineers.External											
Thai		300								1000	
Vietnam							300				
Partners			300	200		1000			80		
Sweden											800
Denmark											
Finland											
Others											
Total Development Resources	600	1300	280	1000	2000	400	2500	180	1000	800	
Benefit / Dev. Resources	0.03%	0.10%	0.36%	0.16%	0.07%	0.09%	0.06%	0.42%	0.16%	0.17%	0
			2	3				1	3	4	

US Army Example: PERSINSCOM

63

STRATEGIES → OBJECTIVES	Technolog y Investment	Business Practice s	People	Empow -erment	Principles of IMA Management	Business Process Re- engineering	SUM
Customer Service ? → 0 Violation of agreement	50%	10%	5%	5%	5%	60%	185%
Availability 90% → 99.5% Up time	50%	5%	5-10%	0	0	200%	265%
Usability 200 → 60 Requests by Users	50%	5-10%	5-10%	50%	0	10%	130%
Responsiveness 70% → ECP's on time	50%	10%	90%	25%	5%	50%	180%
Productivity 3:1 Return on Investment	45%	60%	10%	35%	100%	53%	303%
Morale 72 → 60 per mo. Sick Leave	50%	5%	75%	45%	15%	61%	251%
Data Integrity 88% → 97% Data Error %	42%	10%	25%	5%	70%	25%	177%
Technology Adaptability 75% Adapt Technology	5%	30%	5%	60%	0	60%	160%
Requirement Adaptability ? → 2.6% Adapt to Change	80%	20%	60%	75%	20%	5%	260%
Resource Adaptability 2.1M → ? Resource Change	10%	80%	5%	50%	50%	75%	270%
Cost Reduction FADS → 30% Total Funding	50%	40%	10%	40%	50%	50%	240%
SUM IMPACT FOR EACH SOLUTION	482%	280%	305%	390%	315%	649%	
Money % of total budget	15%	4%	3%	4%	6%	4%	
Time % total work months/year	15%	15%	20%	10%	20%	18%	
SUM RESOURCES	30	19	23	14	26	22	
BENEFIT/RESOURCES RATIO	16:1	14:7	13:3	27:9	12:1	29:5	

Impact Estimation



•Figure 1: Real (NON-CONFIDENTIAL version) example of an initial draft of setting the objectives that engineering processes must meet. 65

Business objective	Measure	Goal (200X)	Stretch goal (0X)	Volume	Value	Profit	Cash
Time to market	Normal project time from GT to GT5	<9 mo.	<12 mo.	X		X	X
Mid-range	Min BoM for The Corp phone	<\$90	<\$100	X		X	X
Platformisation Technology	# of Technology 66 Lic. shipping > 3M/yr	4	6	X		X	X
Interface	Interface units	>11M	>13M	X		X	X
Operator preference	Top-3 operators issue RFQ spec The Corp			X		X	X
Productivity				X		X	X
Get Torden	Lyn goes for Technology 66 in Sep-04	Yes		X		X	X
Fragmentation	Share of components modified	<10%	<5%		X	X	X
Commoditisation	Switching cost for a UI to another System	>1yr	>2yrs		X	X	X
Duplication	The Corp share of 'in scope' code in best-selling device	>90%	>95%		X	X	X
Competitiveness	Major feature comparison with MX	Same	Better	X		X	X
User experience	Key use cases superior vs. competition	5	10	X	X	X	X
Downstream cost saving	Project ROI for Licensees	>33%	>66%	X	X	X	X
Platformisation IFace	Number of shipping Lic.	33	55	X		X	X
Japan	Share of of XXXX sales	>50%	>60%	X		X	X

Numbers are intentionally changed from real ones

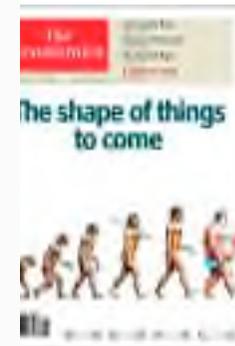
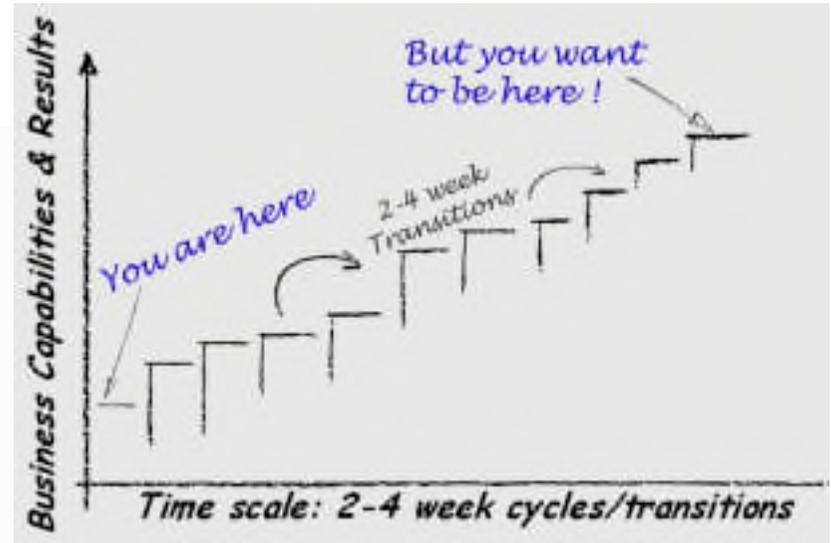
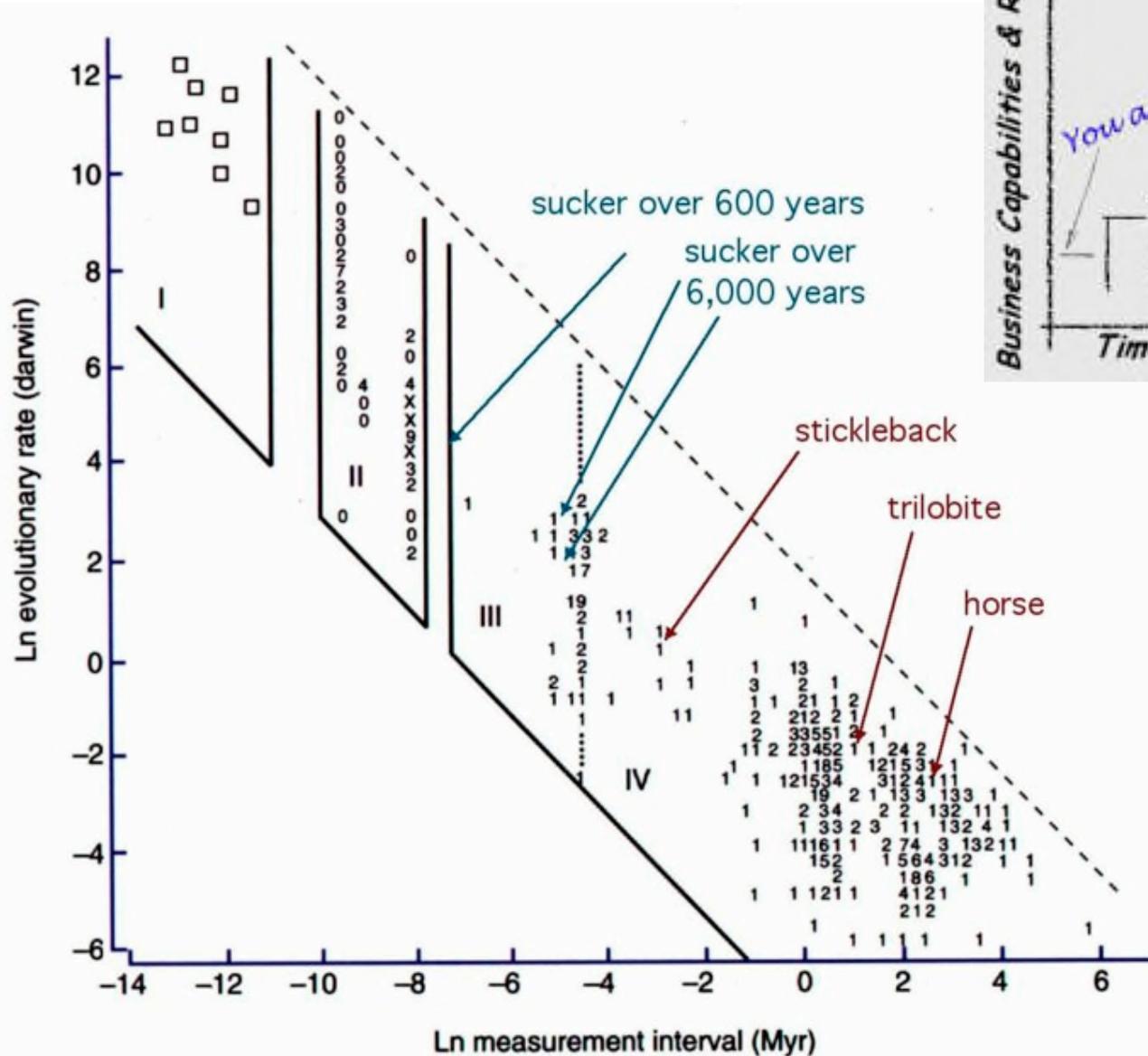
Business Objectives Quantified

A set of 12 proposed engineering processes

	Deliverables						
		Telephony	Modularity	Tools	User Experience	GUI & Graphics	Security	Enterprise
Business Objective								
Time to Market		10%	10%	15%	0%	0%	0%	5%
Product Range		0%	30%	5%	10%	5%	5%	0%
Platform Technology		10%	0%	0%	5%	0%	10%	5%
Units		15%	5%	5%	0%	0%	10%	10%
Operator Preference		10%	5%	5%	10%	10%	20%	10%
Commoditization		10%	-20%	15%	0%	0%	5%	5%
Duplication		10%	0%	0%	0%	0%	5%	5%
Competitiveness		15%	10%	10%	10%	20%	10%	10%
User Experience		0%	20%	0%	30%	10%	0%	0%
Downstream Cost Saving		5%	10%	0%	10%	0%	0%	5%
Other Country		5%	10%	0%	10%	5%	0%	0%
Total Contribution		90%	80%	55%	85%	50%	65%	55%
Cost (£M)		0.49	1.92	0.81	1.21	2.68	0.79	0.60
Contribution to Cost Ratio		184	42	68	70	19	82	92

- A set of 12 proposed engineering Deliverables, for about \$100,000,000 of investment projected over time, are evaluated theoretically for their impact on 13 Business Objectives (as defined in previous slide).
- This real example is altered substantially to protect confidentiality. It appropriately ignited the imagination of top management to really plan their engineering business in a quantified manner.
- Notice the overall impact to cost ratio (ROI Index) is estimated for each process. The actual definitions of the strategy deliverables are elsewhere, and are confidential. But that detail would be needed to estimate and to check these estimates

4. Defined Evo processes

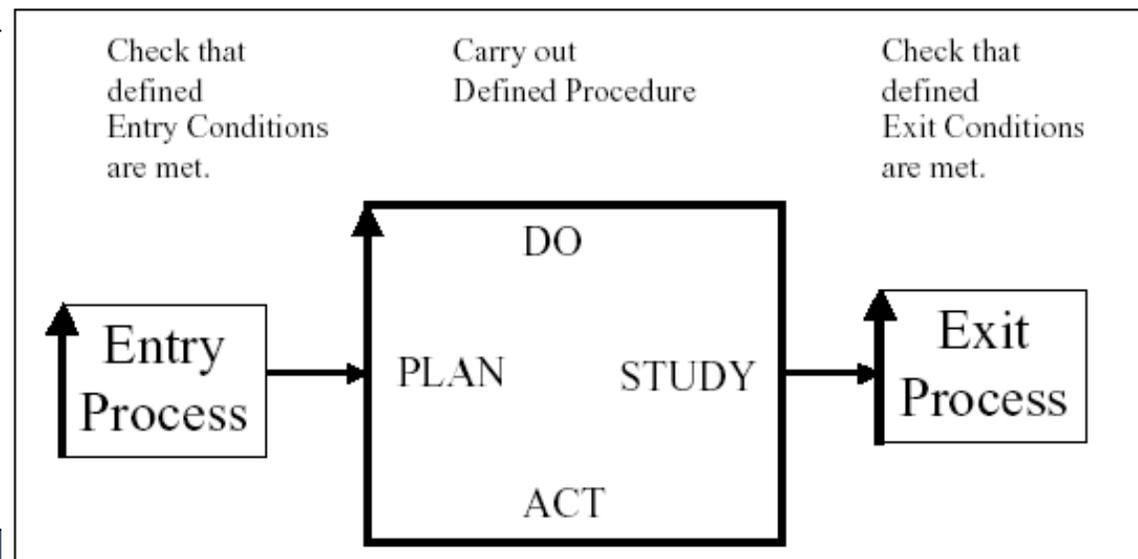


The process format used for Planguage process descriptions consists of three basic elements 68

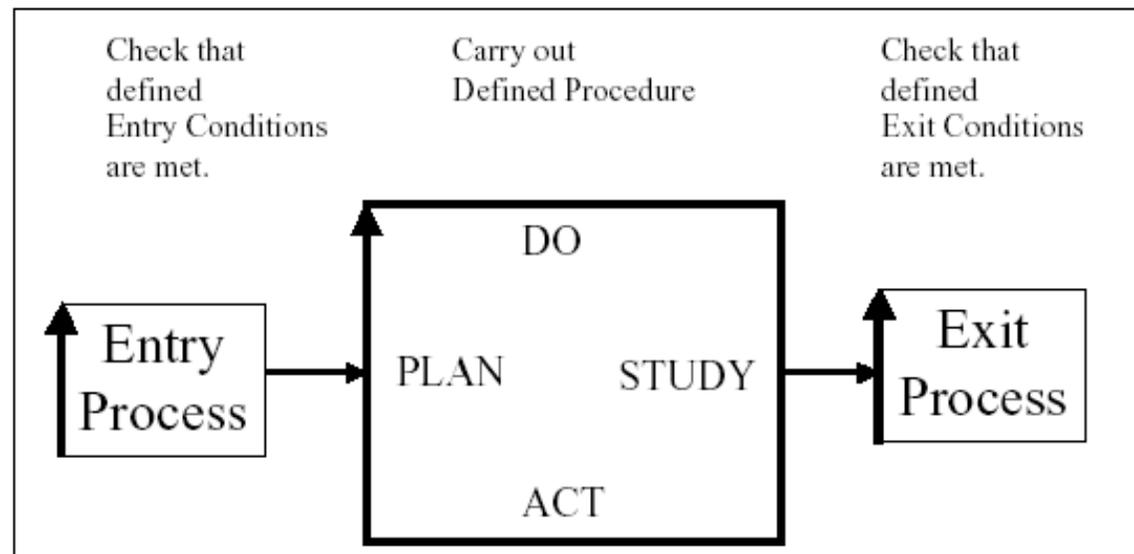
Entry Conditions : to determine whether it is wise to start the procedure.

Procedure : specifying for a task what work needs to be done and how best to do it.

Exit Conditions : to help determine if the work is "truly finished".



The quantified Exit and Entry controls



Entry and Exit Condition example:

Maximum estimated 1.0 Major defects per logical page remaining.

This was the MOST important lesson IBM learned about software processes (source Ron Radice, co-inventor Inspections, Inventor of CMM)

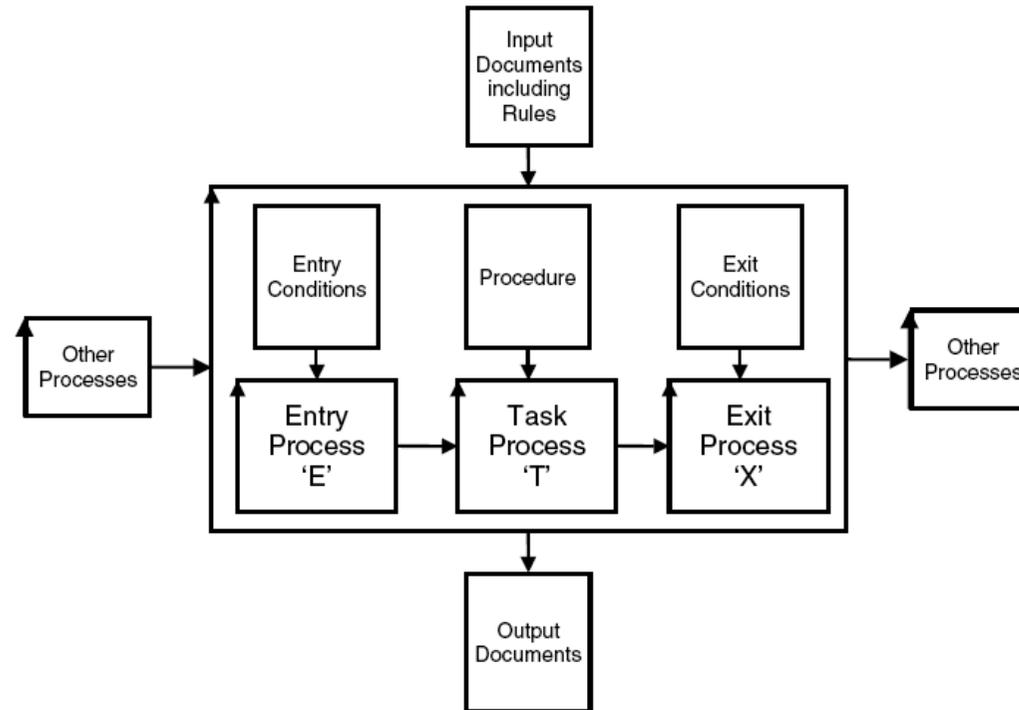


Figure 1.4

- Diagram of a simple process showing its sub-processes and its relationship to other processes and documents.
- The input documents for each process include the rules, the entry conditions, the procedure and the exit conditions.
- The diagram also shows how the !ETX" concept for a process is derived.
 - A rectangle is the symbol for a !written document."
 - A rectangle with arrow is a !process" symbol.
- An example of such a process could be !Requirement Specification." <- CE, figure 1.4

From Competitive Engineering, Chapter 10

Process Description: Evolutionary Project Management

These Evo processes are generalized.

Modification to suit individual circumstances might well be required.

Head and Body Process

72

306 Competitive Engineering

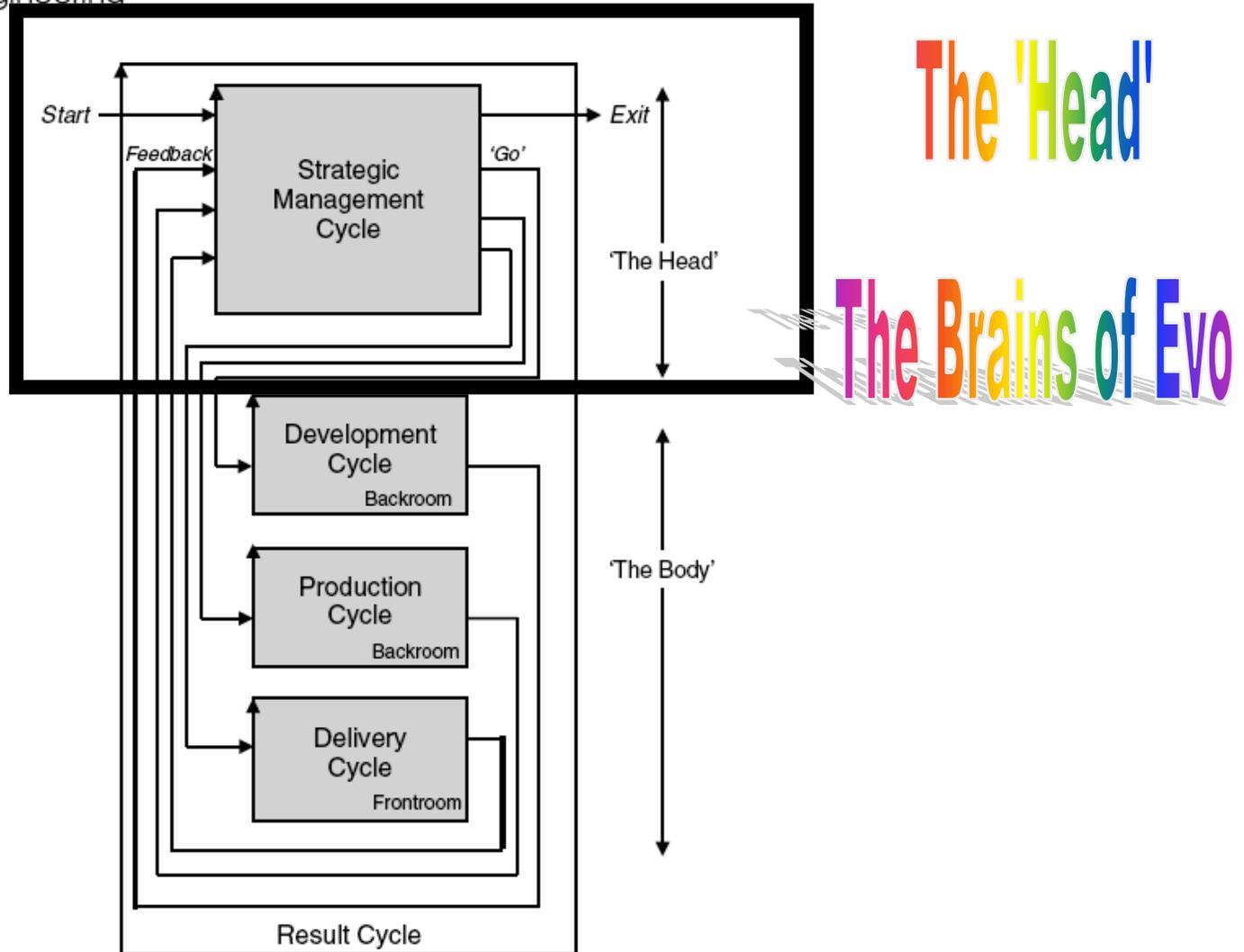


Figure 10.2
The result cycle for an Evo Step.

Process: Strategic Management Cycle (!The Head") Entry

73

Process: Strategic Management Cycle (!The Head")

Tag: Process.SM. Version: October 7, 2004.

Owner: TG. Status: Draft. Note: Process.DC (Delivery Cycle) is a separate process defined below.

Entry Conditions: Entry.SM

E1: All necessary input information for Evo is available to the project management and design team.

E2: All input documents have successfully exited from their own quality control process. The specification quality control (SQC) entry condition applies to the project requirements and the design idea specifications. Note: This usually implies between 0.2 and 1 remaining major defect(s)/page (A page is 300 words of non-commentary text.)

E3: The design idea specifications have been evaluated using IE and, the IE table has exited from SQC (Spec Quality Control, see CE Chapter on SQC).

E4: The level of uncertainty acceptable to the project has been formally determined (deviation (%) from plan). Default level 10%.

E5: The project management and design team are adequately trained or, assisted by a qualified person to analyze and specify evolutionary plans.

E6: There is relevant approval, including funding, for the project to proceed.

Procedure: Procedure.SM

P1: Plan:

1. Modify if necessary top-level project requirements and design ideas.
2. Update the long-term Evo plan.
3. Initiate any backroom development cycles and/or production cycles required for future steps.
4. Decide on the next step for delivery (to the frontroom).
5. For next step: Set step targets, select step design ideas, decide step [qualifiers].
6. Produce maximum one page overview plan for the step delivery (see template in Figure 10.8 and, also the example in Figure 10.5).

The step delivery cycle (DC) can start once the next step (for delivery) has been decided and when the relevant development and production cycles are complete.

P2: Do: Initiate the Delivery Cycle (that is, the step delivery to the stakeholder. Others may carry out the detailed work).

P3: Study:

1. On completion of the Delivery Cycle, identify the numeric differences between the system's actual attribute levels and the target requirements. Where are the large 'gaps'?
2. Note numeric differences between estimated step results and actual results.
3. Monitor the progress of any current 'backroom' development cycles and/or production cycles. Ensure they have sufficient resources to be completed on-time.
4. Note any stakeholder needs, technological, political or economic changes, which should be reflected in the Evo step sequencing, or even the requirement or design specification.

P4: Act: Adopt the change, or abandon it (revert to previous state before step implementation). Or, decide to run through the cycle again, but possibly under changed conditions (paraphrased from W.E. Deming 1986). Go to P1 (that is, continue cycling), unless Exit Conditions are met.

Exit Conditions:

Exit.SM

X1: If resources used up, stop project. Keep results achieved so far!

X2: If all existing Goal levels are reached, stop using resources.

Head and Body Process

76

306 Competitive Engineering

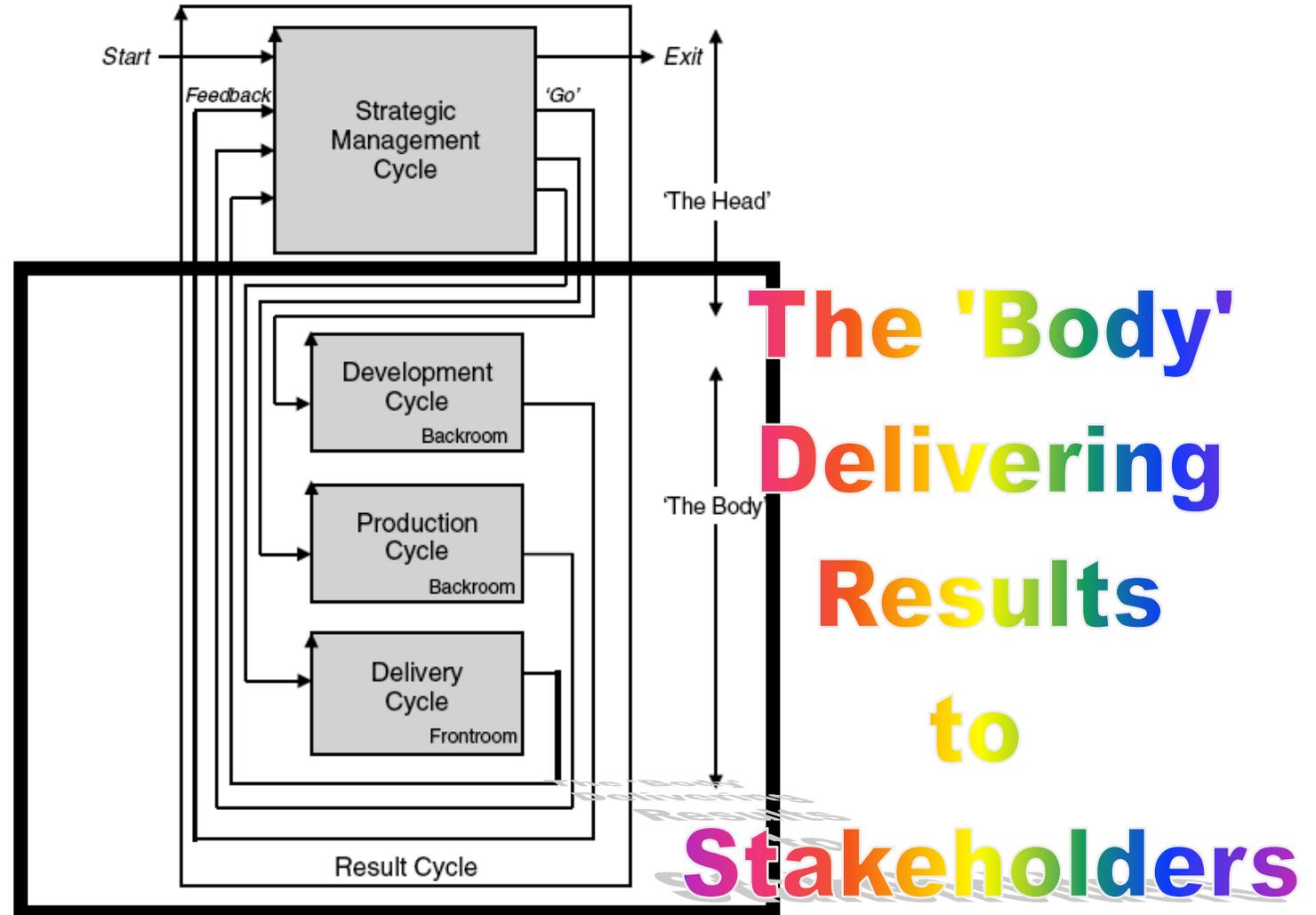


Figure 10.2
The result cycle for an Evo Step.

Delivery Cycle

77

Process: Delivery Cycle (Part of 'The Body')

Tag: Process.DC.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Gist: This process is for delivery of a single step, not the larger project totality.

Entry Conditions: Delivery Cycle

78

Entry Conditions: Entry.DC [Step n]

E1: All logically prerequisite steps to this one, which were specified, have been completed.

E2: The numeric feedback results from any previously completed steps must be available to the design team and must have been studied.

(You may want to re-do the previous step before proceeding.)

Procedure: Procedure.DC [Step n]

79

Procedure: Procedure.DC [Step n]

P1: Plan:

1. Specify the delivery of the step in detail. See Figure 10.8 in Section 10.9 for a template.
2. Agree the plan with the relevant, affected stakeholders (For example, management and customers). The list of topics to consider includes: changes to working practices, training, installing, regression testing, field trials, hand-over and criteria for success: all system-wide considerations.

P2: Do:

Deliver the step. Install it with real stakeholders, so they get some of the planned measurable benefits.

P3: Study:

1. Determine the results of delivering the step. Obtain any relevant measurements: test, measure and sample, to establish the new performance levels and the new operational cost levels. Compare results to the short-term and long-term targets.
2. Analyze the data and produce a feedback report for management. For example, use an Impact Estimation table as a tool to do this study task.

P4: Act:

1. Decide if this step succeeded, must be redone in whole or part, or totally rejected.
2. Take any required minor corrective actions (for example, bug fixing) to 'stabilize' the system.



Exit Conditions: Exit.DC [Step n]

X1: Step completed, or dropped.

- Exit a step only when all step performance levels and function requirements are reached (or wavered formally).
- Give up if reaching planned requirements is impractical, or if you run out of resources.

1. Gather from all the key stakeholders the top few (5 to 20) most critical goals that the project needs to deliver.

Give each goal a reference name (a tag).

2. For each goal, define a scale of measure and a 'final' goal level.

For example:

Reliable: Scale: Mean Time Before Failure, Goal: >1 month.

**3. Define approximately 4 budgets for your most limited resources
(for example, time, people, money and equipment).**

4. Write up these plans for the goals and budgets

(try to ensure this is kept to only one page).

5. Negotiate with the key stakeholders to formally agree the goals and budgets.

6. Plan to deliver some benefit (that is, progress towards the goals) in weekly (or shorter) increments (Evo steps).

7. Implement the project in Evo steps.

Report to project sponsors after each Evo step (weekly, or shorter) with your best available estimates or measures, for each performance goal and each resource budget.

8. On a single page, summarize the progress to date towards achieving the goals and the costs incurred. <- CE p.308

Policy for the Agile Evo Process

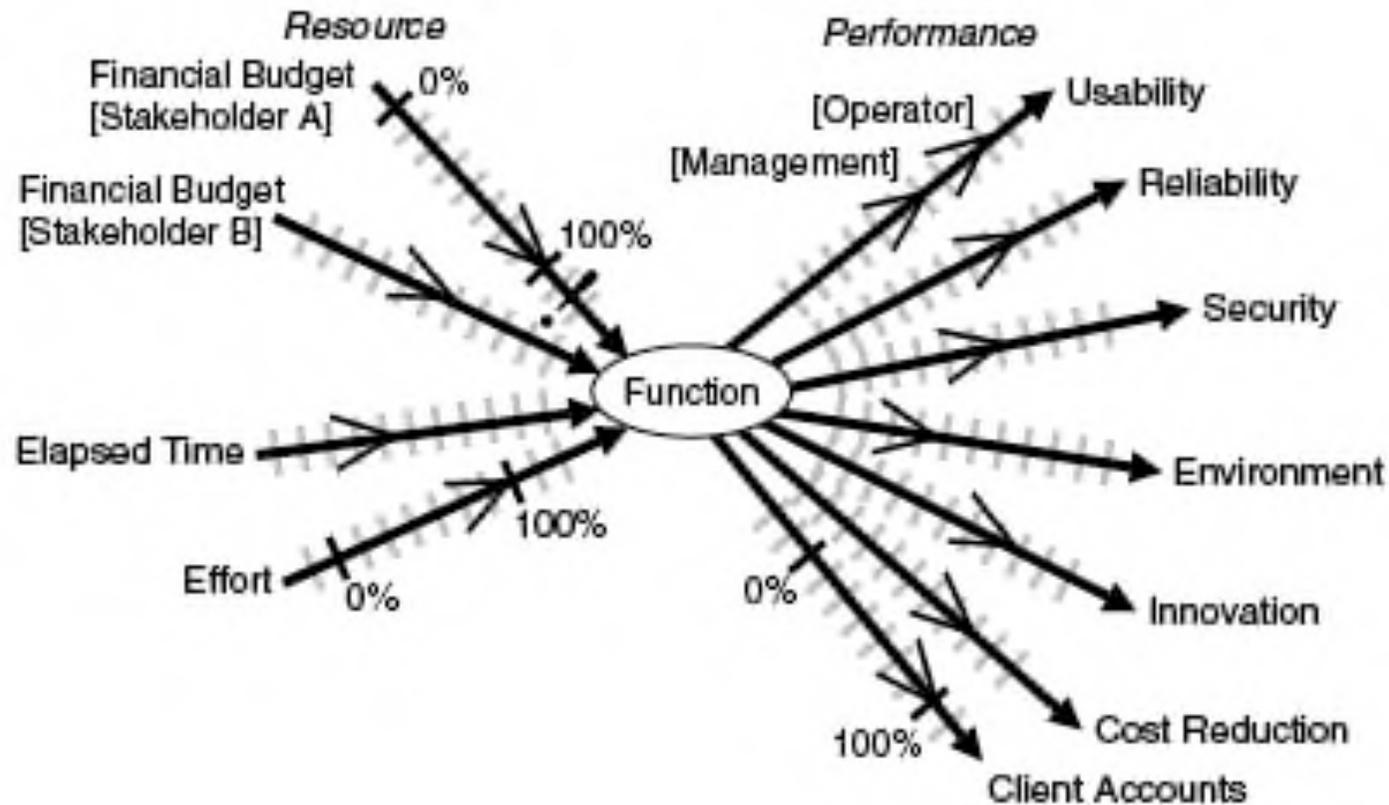
82

- 1. The project manager and the project will be judged exclusively on the relationship of *progress towards achieving the goals* versus the amounts of the *budgets used*.**
- 2. The project team will do *anything* legal and ethical to deliver the goal levels within the budgets.**
- 3. The team will be paid and rewarded for *benefits delivered* in relation to *cost*.**
- 4. The team will find their own *work process* and their own *design*.**
- 5. As experience dictates, the team will be free to suggest to the project sponsors (stakeholders) *adjustments* to ‘more realistic levels’ of the goals and budgets.**

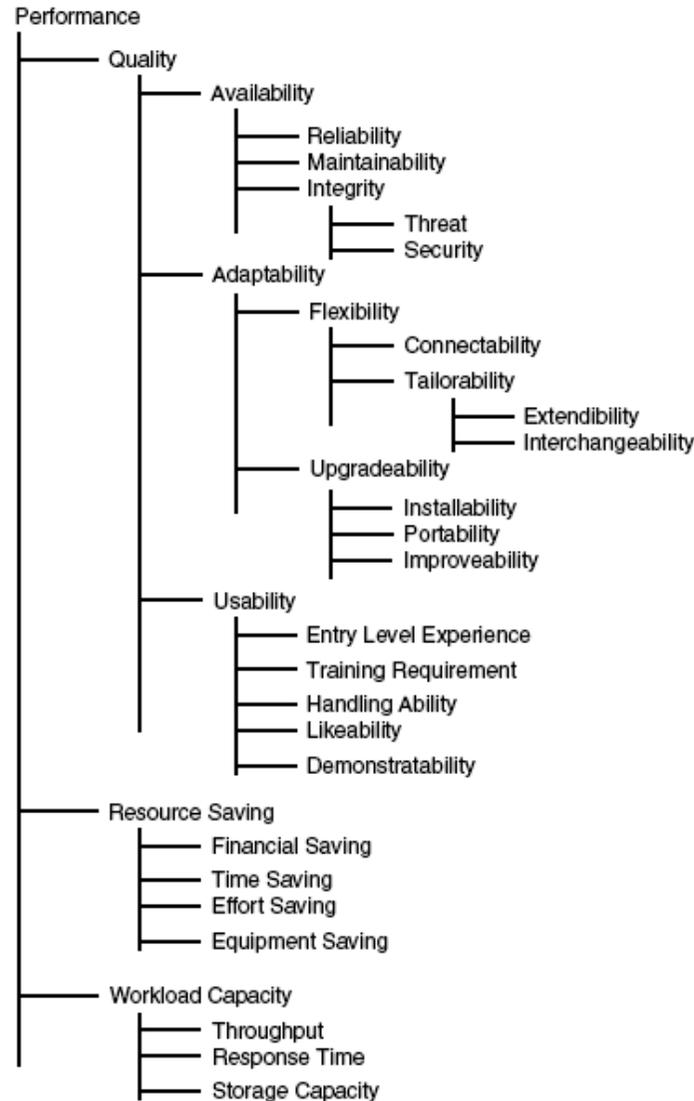
<-CE p. 308

5. Templates for Quantified Requirements and Quantified Design

Evolution towards Multiple Goals



A Hierarchy of performance attributes 84



A Quantified Quality Requirement

Usability:

Ambition: Operator ease of learning & doing tasks under <all conditions> should be maximum possible ease & speed of performance with minimum training & minimum possibility of <unchecked error(s)>.

Usability.Intuitiveness:

Ambition: High probability that an operator will within a specified time from deciding the need to perform a specific task (without reference to handbooks or help facility) find a way to accomplish their desired task.

Scale: Percentage Probability that a defined [Individual Person: Default: Trained Operator] will find a way to perform a defined [Task Type] without reference to any written instructions, other than the help or guidance instructions offered by the immediate system screen (that is, no additional paper or on-line system reference information), within a defined [Time Period: Default: Within one second from deciding that it is necessary to perform the task].

Comment [Intuitiveness:Scale]: "I'm not sure if one second is acceptable or realistic, it's just a guess" <-

MAB.

Meter: To be defined. Not crucial this 1st draft <- TG.

Past [System R]: 80%? <- LN.

Record [Mac User Interface]: 95%? <- TG.

Fail [Trained Operator, Rare Tasks [{<1/week, <1/year}]]: From 50% to 90%? <- MAB.

Goal [Tasks Done [<1/week (but more than 1/Month)]]: 99%? <- LN.

[Tasks Done [<1/year]]: 20%? <- JB.

[Turbulence, Tasks Done [<1/year]]: 10%? <- TG.

===== User Defined Terms =====

Trained Operator: Defined As: Command and Control Onboard Operator, who has been through approved training course of at least 200 hours duration.

Rare Tasks: Defined As: Types of tasks performed by an Onboard Operator less than once a week on average.

Tasks Done: Defined As: Distinct tasks carried out by Onboard Operator.

===== Usability.Intelligibility:

Ambition: High ability for an operator to <correctly> interpret the meaning of given information.

Scale: Percentage Probability of <objectively correct> interpretation(s) of a defined [Set of <Inputs>] by a defined [Individual Person: Default: Trained Operator] within a defined [Time Period].

Meter [Acceptance]: Use about 10 Trained Operators, and use about 100 <representative sets of information per operator within 15 minutes?> - MAB.

Comment [Meter]: "Not sure if the 15 minutes are realistic" <- MAB.

Comment [Meter]: "This is a client & contract determined detail" <- MAB.

M1: Past: [XXX, 20 Trained Operators, 300 <data sets>, 30 minutes]: 99.0%

<- Acceptance Test Report from XXX, MAB.

Record [XXX]: 99.0%. "None other than XXX known by me" <- MAB.

Fail [First Delivery Step]: 99.0%? <- MAB.

Fail [Acceptance]: 99.5%? <- MAB.

Goal [XXX, 20 Trained Operators, 300 <data sets>, 30 minutes]: 99.9% <- LN.

Usability.Productivity *(taken from Confirmit 8.5 development)*

Scale for quantification: Time in minutes to set up a typical specified Market Research-report

Past Level [Release 8.0]: 65 mins.,

Tolerable Limit [Release 8.5]: 35 mins.,

Goal [Release 8.5]: 25 mins.

Note: end result was actually 20 minutes ☺

Meter [Weekly Step]: Candidates with Reportal experience, and with knowledge of MR-specific reporting features, performed a set of predefined steps, to produce a standard MR Report.

Our new focus is on the day-to-day operations of our Market Research users,

not a list of features that they might or might not like. 50% never used!

We KNOW that increased efficiency, which leads to more profit, will please them.

The '45 minutes actually saved x thousands of customer reports'
= big \$\$\$ saved

After one week we had defined more or less all the requirements for the next version (8.5) of Confirmit.



Trond Johansen

confirmit✓



Scalar Requirement Template

Elementary scalar requirement template <with hints>

87

Tag: <Tag name of the elementary scalar requirement>.
Type: <{Performance Requirement: {Quality Requirement, etc.}>
===== **Basic Information** =====
Version: <Date or other version number>.
Status: <{Draft, SQC Exited, Approved, Rejected}>.
Quality Level: <Maximum remaining major defects/page, sample size, date>.
Owner: <Role/e-mail/name of the person responsible for this specification>.
Stakeholders: <Name any stakeholders with an interest in this specification>.
Gist: <Brief description, capturing the essential meaning of the requirement>.
Description: <Optional, full description of the requirement>.
Ambition: <Summarize the ambition level of only the targets below. Give the overall real ambition level in 5-20 words>.
===== **Scale of Measure** =====
Scale: <Scale of measure for the requirement (States the units of measure for all the targets, constraints and benchmarks) and the scale qualifiers>.
===== **Measurement** =====
Meter: <The method to be used to obtain measurements on the defined Scale>.
===== **Benchmarks** ===== "Past Numeric Values" =====
Past [<when, where, if>]: <Past or current level. State if it is an estimate> <- <Source>.
Record [<when, where, if>]: <State-of-the-art level> <- <Source>.
Trend [<when, where, if>]: <Prediction of rate of change or future state-of-the-art level> <- <Source>.
===== **Targets** ===== "Future Numeric Values" =====
Goal/Budget [<when, where, if>]: <Planned target level> <- <Source>.
Stretch [<when, where, if>]: <Motivating ambition level> <- <Source>.
Wish [<when, where, if>]: <Dream level (unbudgeted)> <- <Source>.

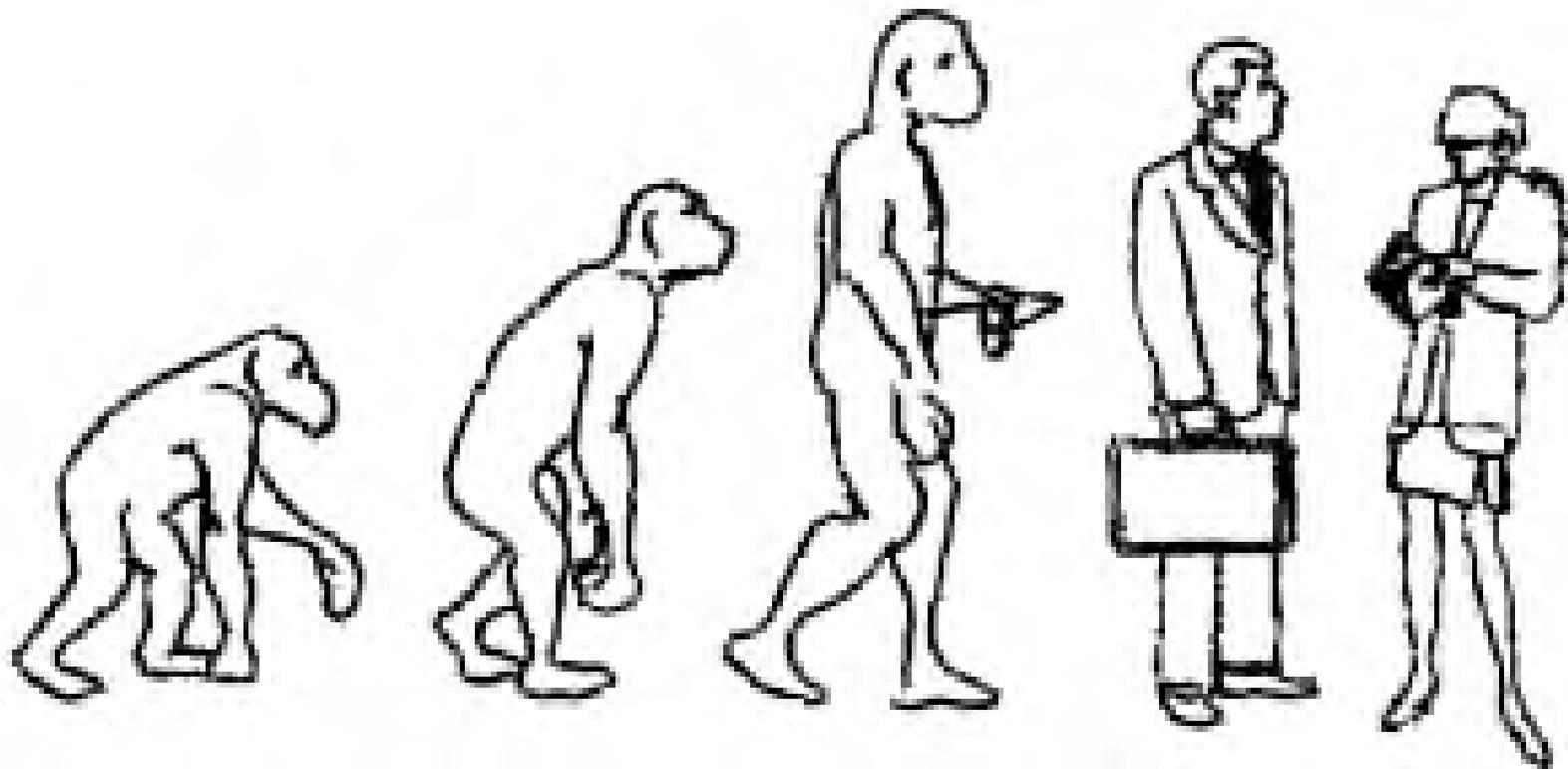
===== **Constraints** == "Specific Restrictions" =====
Fail [<when, where, if>]: <Failure level> <- <Source>.
Survival [<when, where, if>]: <Survival level> <- <Source>.
===== **Relationships** =====
Is Part Of: <Refer to the tags of any supra-requirements (complex requirements) that this requirement is part of. A hierarchy of tags (For example, A.B.C) is preferable>.
Is Impacted By: <Refer to the tags of any design ideas that impact this requirement> <- <Source>.
Impacts: <Name any requirements or designs or plans that are impacted significantly by this>.
===== **Priority and Risk Management** =====
Rationale: <Justify why this requirement exists>.
Value: <Name [stakeholder, time, place, event]: Quantify, or express in words, the value claimed as a result of delivering the requirement>.
Assumptions: <State any assumptions made in connection with this requirement> <- <Source>.
Dependencies: <State anything that achieving the planned requirement level is dependent on> <- <Source>.
Risks: <List or refer to tags of anything that could cause delay or negative impact> <- <Source>.
Priority: <List the tags of any system elements that must be implemented before or after this requirement>.
Issues: <State any known issues>.

Tag: OPP Integration.
Type: Design Idea [Architectural].
===== Basic Information
Version:
Status:
Quality Level:
Owner:
Expert:
Authority:
Source: System Specification Volume 1 Version 1.1, SIG, February 4 – Precise reference <to be supplied by Andy>.
Gist: The X-999 would integrate both 'Push Server' and 'Push Client' roles of the Object Push Profile (OPP).
Description: Defined X-999 software acts in accordance with the <specification> defined for both the Push Server and Push Client roles of the Object Push Profile (OPP).
Only when official certification is actually and correctly granted; has the {developer or supplier or any real integrator, whoever it really is doing the integration} completed their task correctly.
This includes correct proven interface to any other related modules specified in the specification.
Stakeholders: Phonebook, Scheduler, Testers, <Product Architect>, Product Planner, Software Engineers,

User Interface Designer, Project Team Leader, Company engineers, Developers from other Company
 product departments which we interface with, the supplier of the TTT, CC. "Other than Owner and Expert. The people we are writing this particular requirement for."
 ===== Design Relationships =====
 Reuse of Other Design:
 Reuse of This Design:
 Design Constraints:
 Sub-Designs:
 ===== Impacts Relationships =====
 Impacts [Functions]:
 Impacts [Intended]: Interoperability.
 Impacts [Side Effects]:
 Impacts [Costs]:
 Impacts [Other Designs]:
 Interoperability: Defined As: Certified that this device can exchange information with any other device produced by this project.
 ===== Impact Estimation/Feedback =====
 Tag: Interoperability.
 Scale:
 Percentage Impact [Interoperability, Estimate]: <100% of Interoperability objective with other devices that support OPP on time is estimated to be the result>.
 ===== Priority and Risk Management =====
 Rationale:
 Value:
 Assumptions: There are some performance requirements within our certification process regarding probability of connection and transmission etc. that we do not remember <-TG.
 Dependencies:
 Risks:
 We do not 'understand' fully (because we don't have information to hand here) our certification requirements, so we risk that our design will fail certification <-TG.
 Priority:
 Issues:
 ===== Implementation Control =====
 Not yet filled in.
 ===== Location of Specification =====
 Location of Master Specification: <Give the intranet web location of this master specification>.

6. Templates for Quantified Evo step specification

89



Real Step Specification Example

90

An Evo Step Specification

Evo Step: Tutorial [Model 1234, Basic].

Stakeholders: {Marketing, Department XX}.

Implementers: Department XX.

Intended Audience: Marketing.

Gist: To prepare a written tutorial that teaches how to identify required information on internet web pages.

Step Content: HCTD12: <Hard Copy Text Document>. "This declares a design idea, HCTD12, that needs further detailed specification. Some additional notes about it are also given. See below."

Notes [HCTD12]:

- . Can write the basic minimal functions, MMM, in 1 week. <-GF.
- . Provide step by step instructions, in English.
- . Questionnaire for Stakeholders.
- . Intended audience: Marketing.
- . Focus on <sales aspects>, not how to identify information in detail (not yet, in this step).
- . Go to <specific web sites>.
- . Process for Testing with Stakeholder (for example, observation, times).
- . Pinpoint some characteristics of what we see on the terminal compared with what we see on a <PC or other terminal>.
- . What instructions should be on the terminal to begin?
- . No illustrations to be provided, just text.

Questionnaire: Defined As: Questionnaire to walkthrough with stakeholders.

Step Validation: DefinedAs:Process for TestingwithStakeholders. "Example observation, times."

Constraint: Step must be deliverable within one calendar week.

Assumptions [Applies?Step Cost [Effort], Source?MMM]: 10 hours per page.

Dependencies: <Feature list of WWW>, <77777 WWW Browser> <-MMM.

Risks: At least 3 hours needed of TTT's time for input and trial feedback.

Step Value:

{[Stakeholder?TTT, Saleability]: <some possibility of value>, [Stakeholder?Developers]: <value of feedback on a tutorial>}

Step Cost [Effort]: < 10 hours <-MMM.

Figure 10.5

An example of using the specification template for an Evo step.

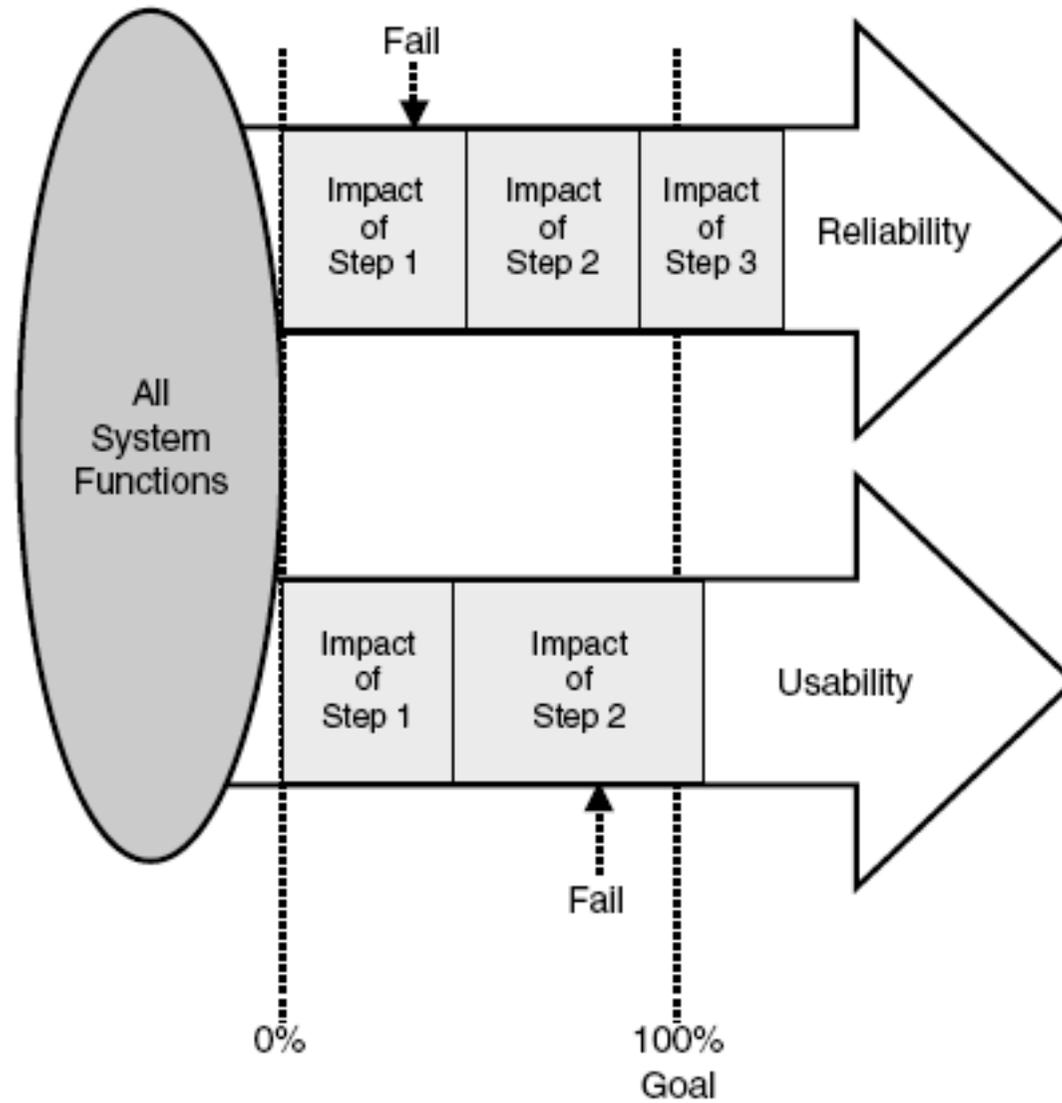
Source CE, Evolutionary Project Management Page 313

A Template For EVO Step Specification

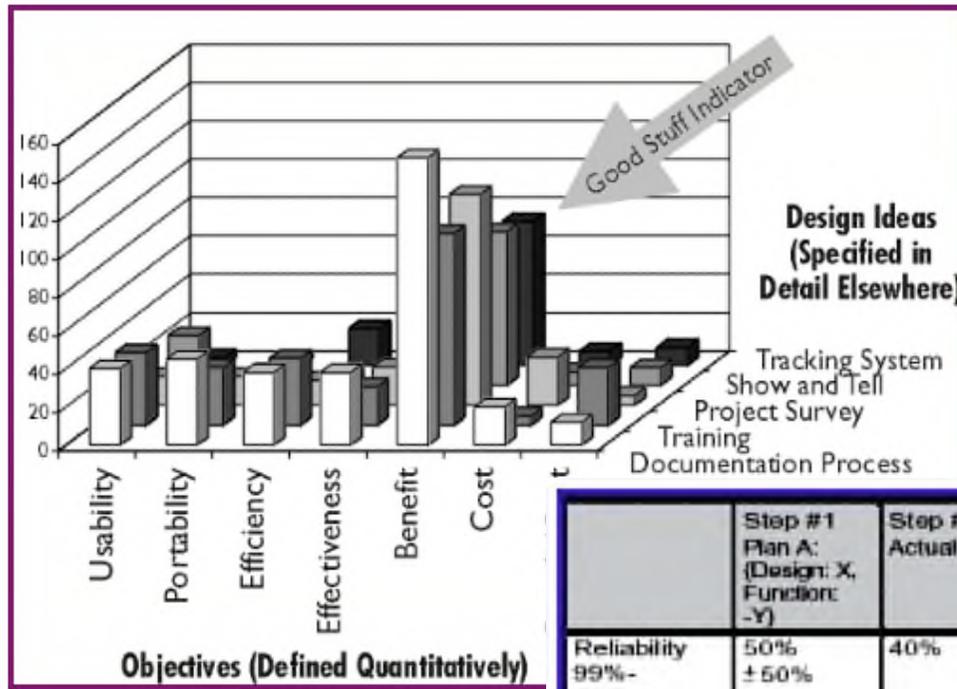
Tag: <Tag name for the step>.
Type: Evo Step.
===== Basic Information =====
Version: <Date or version of last update to step specification>.
Status: <{Specification Stage [{Draft, SQC Exited, Approved}], In
Evo Plan, Scheduled Next,
Under Implementation, Delivered awaiting Feedback, Feedback
Obtained}, date> <- <Source
(who says 'Status' is true?)>.
Quality Level: <Maximum remaining major defects/page, sample
size, SQC date>.
Owner: <Who is taking responsibility for the step in terms of
specification>.
Stakeholders: <Who are you going to deliver requirements to? >.
Implementers: <Who is in charge of implementing this step>.
Gist: <Brief description of the main idea of this step>.
Description: <Give a detailed, unambiguous description of the step,
or a tag reference to a
place where it is described. Remember to include definitions of any
local terms>.
Implementation Details: "Includes relevant details, such as <which
product>, <which area of
application system>."
Evo Plan: <Tag of the Evo Plan that this step is associated with>.
Step Content: <Step Elements: {Design Ideas, Functions, Tasks, re-
used step definitions}>.

===== Measurement =====
Test: <Refer to tags of any test plan and/or test cases, which apply to this
step>.
Step Validation/Feedback:
Specification Quality Control (SQC): <outcome, date>,
Pre-Delivery Test: <outcome, date>,
Post Delivery Results: <{problems, stakeholder feedback}, date>,
Certification Specification: <refer to the certification plans>.
===== Priority and Risk Management =====
Constraints:
<Any legal, political, economic, security or other constraints imposed on
implementation> <- <Source (who says this is true?)>.
Assumptions: <Any assumptions that have been made>.
Dependencies:
<Anything which must be in place, finished, working properly, for us to be
able to start this Evo step or to complete it> <- <Source (who says this is
true?)>.
Risks: <Any risks that need to be taken into account>.
Priority:
<Name, using tags, any system elements, which must clearly be done after
or must clearly be done before. Give any relevant reasons>.
Issues: <Unresolved concerns or problems in the step specification or the
system>.
===== Benefits and Costs =====
Rationale: <Justify the existence of this step>.
Step Value:
<Real measurements or estimates of numeric value to stakeholders>.
"Value in terms of
meeting the requirements. At least, the value on scale 0 (none) to 9
(highest)." <- <Source (who says this is true?)>.
Step Cost:
<Budgets or real costs>. "For example, financial costs and engineering
hours. These must be
constrained by the Evo 2% policy. At least, the value on scale 0 (very
cheap) to 9 (high and unpredictable)." <- <Source (who says this is
true?)>.

Dynamic Step Priority



7. Impact Estimation Table Evo project management 93



	Step #1 Plan A: (Design: X, Function: -Y)	Step #1 Actual	Step #1 Difference - Is Bad + Is Good	Total Step #1	Step #2 Plan B: (Design: Z, Design: F)	Step #2 Actual	Step #2 Difference	Total Steps #1 and #2	Step #3 Next Step Plan
Reliability 99%- 99.9%	50% ± 50%	40%	-10%	40%	30% ± 20%	20%	-10%	60%	0%
Performance 11 sec.- 1 sec.	80% ± 40%	40%	-40%	40%	30% ± 50%	30%	0	70%	30%
Usability 30 min.- 30 sec.	10% ± 20%	12%	+2%	12%	20% ± 15%	5%	-15%	17%	83%
Capital Cost 1 mill.	20% ± 1%	10%	+10%	10%	5% ± 2%	10%	-5%	20%	5%
Engineering Hours 10,000	2% ± 1%	4%	-2%	4%	10% ± 2.5%	3%	+7%	7%	5%
Calendar Time	1 week	2 weeks	-1 week	2 weeks	1 week	0.5 week	+0.5 week	2.5 weeks	1 week

Tracking the Project Evolution using an Impact Estimation Table
(conceptual diagram)

Step	Step 1 Plan % (of Target)	Actual %	Deviation %	Step 2 to Step 20 Plan %	Plan % cumulated to here	Step 21 [CA, NV, WA] Plan %	Plan % cumulated to here	Step 22 [all others] Plan %	Plan % cumulated to here
Target Requirement									
<i>Performance 1</i>	5	3	-2	40	43	40	83	-20	63
<i>Performance 2</i>	10	12	+2	50	62	30	92	60	152
<i>Performance 3</i>	20	13	-7	20	33	20	53	30	83
<i>Cost A</i>	1	3	+2	25	28	10	38	20	58
<i>Cost B</i>	4	6	+2	38	44	0	44	5	49



Trond Johansen

IET for MR Project - Confirmit (<-FIRM Product Brand) 8.5

Solution: Recoding

Make it possible to recode variable on the fly from Reportal.

Estimated effort: 4 days

Estimated Productivity Improvement: 20 minutes (50% way to Goal)

actual result 38 minutes (95% progress towards Goal)

	A	B	C	D	E	F	G	BX	BY	BZ	CA
1											
2		Current Status	Improvements	Goals			Step9				
3	Recoding										
4								Estimated impact		Actual impact	
5		Units	Units	%	Past	Tolerable	Goal	Units	%	Units	%
6					Usability.Replacability (feature count)						
7		1,00	1,0	50,0	2	1	0				
8					Usability.Speed.NewFeaturesImpact (%)						
9		5,00	5,0	100,0	0	15	5				
10		10,00	10,0	200,0	0	15	5				
11		0,00	0,0	0,0	0	30	10				
12					Usability.Intuitiveness (%)						
13		0,00	0,0	0,0	0	60	80				
14					Usability.Productivity (minutes)						
15		20,00	45,0	112,5	65	35	25	20,00	50,00	38,00	95,00
20					Development resources						
21			101,0	91,8	0		110	4,00	3,64	4,00	3,64

Product quality - "green" week

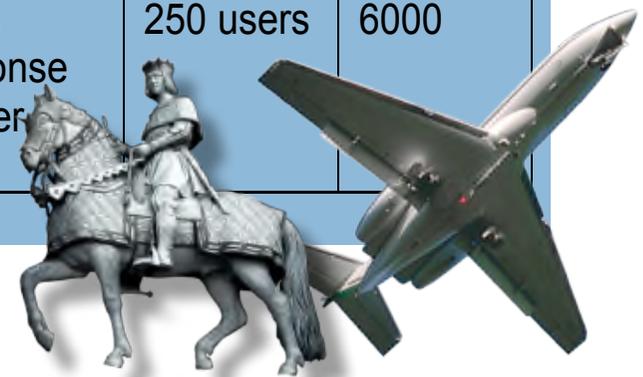
In these "green" weeks, some of the deliverables will be less visible for the end users, but more visible for our QA department.

We manage product quality through an Impact Estimation table.

Current Status	Improvement	Goals			Step 6 (week 14)		Step 7 (week 15)		
		Units	Past	Tolerable	Goal	Estimated Impact	Actual Impact	Estimated Impact	Actual Impact
	100,0	100,0	0	80	100		100	100	
Speed									
	100,0	100,0	0	80	100	100	100		
Maintainability.Doc.Code									
	100,0	100,0	0	80	100	100	100		
InterviewerConsole									
NUnitTests									
	0,0	0,0	0	90	100				
PeerTests									
	100,0	100,0	0	90	100		100	100	
FxCop									
	0,0	10,0	10	0	0				
TestDirectorTests									
	100,0	100,0	0	90	100		100	100	
Robustness.Correctness									
	2,0	2,0	0	1	2	2	2		
Robustness.BoundaryConditions									
	0,0	0,0	0	80	100				
Speed									
	0,0	0,0	0	80	100				
ResourceUsage.CPU									
	100,0	0,0	100	80	70	70			
Maintainability.Doc.Code									
	100,0	100,0	0	80	100	100	100		
SynchronizationStatus									
NUnitTests									

Only highlights of the impacts are listed here

Description of requirement/work task	Past	Status
Usability.Productivity: Time for the system to generate a survey	7200 sec	15 sec
Usability.Productivity: Time to set up a typical specified Market Research-report (MR)	65 min	20 min
Usability.Productivity: Time to grant a set of End-users access to a Report set and distribute report login info.	80 min	5 min
Usability.Intuitiveness: The time in minutes it takes a medium experienced programmer to define a complete and correct data transfer definition with Confirmit Web Services without any user documentation or any other aid	15 min	5 min
Performance.Runtime.Concurrency: Maximum number of simultaneous respondents executing a survey with a click rate of 20 sec and an response time<500 ms, given a defined [Survey-Complexity] and a defined [Server Configuration, Typical]	250 users	6000



Release 8.5

Product quality	Description	Customer value
Intuitiveness	Probability that an inexperienced user can intuitively figure out how to set up a defined Simple Survey correctly.	Probability increased by 175%
Productivity	Time in minutes for a defined advanced user, with full knowledge of 9.0 functionality, to set up a defined advanced survey correctly.	Time reduced by 38%

Product quality	Description	Customer value
Productivity	Time (in minutes) to test a defined survey and identify 4 inserted script errors, starting from when the questionnaire is finished to the time testing is complete and is ready for production. (Defined Survey: Complex survey, 60 questions, comprehensive JScripting.)	Time reduced by 83% and error tracking increased by 25%

Product quality	Description	Customer value
Performance	Max number of panelists that the system can support without exceeding a defined time for the defined task, with all components of the panel system performing acceptable.	Number of panelists increased by 1500%
Scalability	Ability to accomplish a bulk-update of X panelists within a timeframe of Z second	Number of panelists increased by 700%
Performance	Number of responses a database can contain if the generation of a defined table should be run in 5 seconds.	Number of responses increased by 1400%

8. Evo Policy template

100



A Basic Evolutionary Planning Policy

1. Financial Control:

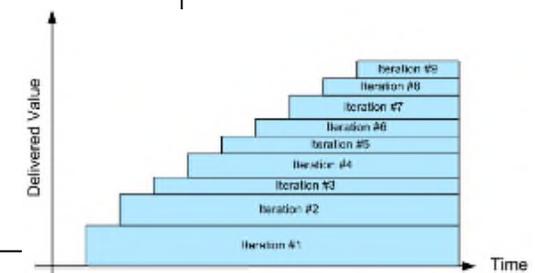
No project cycle can exceed 2% of total initial financial budget before delivering some measurable, required results to stakeholders.

2. Deadline Control:

No project cycle can exceed 2% of total project time (For example, one week for a one year project) before delivering some measurable, required results to stakeholders.

3. Value Control:

The next step should always be the one that delivers best stakeholder value for its costs.

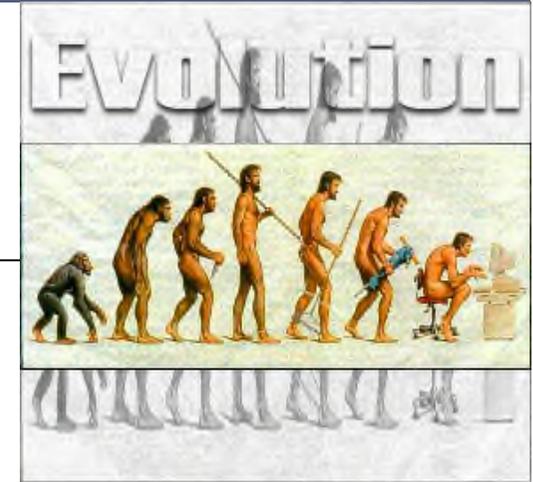


9. Organizational considerations when doing Evo

102

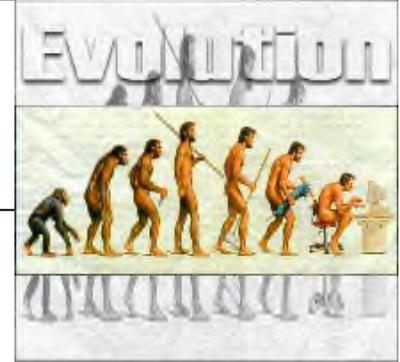
**How does Evo influence your organization?
And
How can your organization influence Evo?**

Evo Objective:
To learn as much as possible:
To contribute to your 'learning organization' continuously



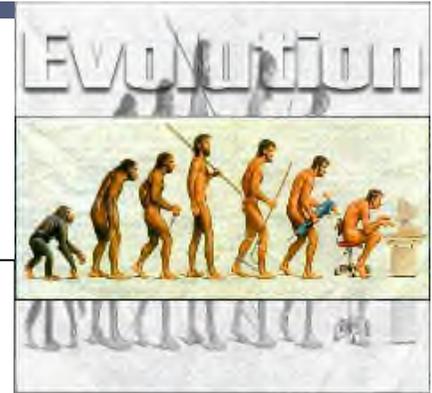
- Pick high 'unknown' areas first
 - New markets
 - New stakeholders
 - New technology
 - New combinations of technology
 - Areas which are critical for your Stakeholder Value & Product Quality Goals
- Try them out and make sure they work before 'scaling up'.
- This is where you will learn and as a consequence
 - Change requirements
 - Change design
 - Change Evo scheduling

Evo Objective: To reduce risks



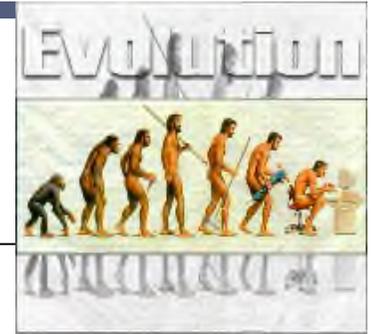
- **Decompose and schedule early those things which are high risk.**
- **How do we know they are high risk?**
 - Experts say they are
 - Estimates have large \pm uncertainty ($> \pm 30\%$)
 - Estimates have low Impact Estimation credibility (under 0.5)
 - Your organization has never done this particular thing before
- **Have an Evo planning policy that says:**
 - We schedule the high risk, high value factors before the low risk, high value factors.

Evo Objective: To deliver value to stakeholders



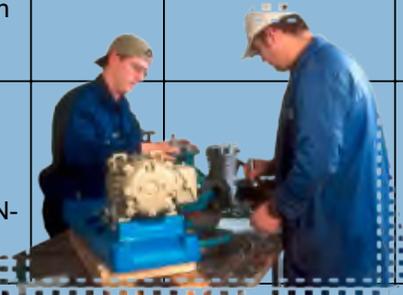
- Decompose and schedule high value to cost cycles first
 - Recognize the Internal Stakeholders are usually opportunity for early value at low risk
- Use Impact Estimation tables to compare value alternatives
- Use Evo cycle templates which force you to estimate value and cost.
 - See next Evo cycle template slides
 - General template with hints
 - Real example
- Have a clear Evo planning policy:
 - `Evo cycles will, after high risks are tested,
 - primarily select and deliver cycles
 - based on overall numeric value delivery to Stakeholder Values & Product Qualities requirements
 - in relation to overall cost in terms of resource budgets.

Evo Objective: To get 'political' advantage



- Ask who are your most critical stakeholders.
- Ask what are their most urgent priorities
- Make a plan to deliver those priorities first
- Confirm with the stakeholders that you have still understood their real current priorities
- Make sure you get correct feedback from them when the cycle is actually implemented.

FIRM EVO-week cycle

	Development Team	Users (PMT, Pros, Doc writer, other)	CTO (Sys Arch, Process Mgr)	QA (Configuration Manager & Test Manager)
Friday	<ul style="list-style-type: none"> ✓ PM: Send Version N detail plan to CTO + prior to Project Mgmt meeting ✓ PM: Attend Project Mgmt meeting: 12.00-15.00 ✓ Developers: Focus on general maintenance work, documentation. 		<ul style="list-style-type: none"> ✓ Approve/reject design & Step N ✓ Attend Project Mgmt meeting: 12-15 	<ul style="list-style-type: none"> ✓ Run final build and create setup for Version N-1. ✓ Install setup on test servers (external and internal) ✓ Perform initial crash test and then release Version N-1
Monday	<ul style="list-style-type: none"> ✓ Develop test code & code for Version N 	<ul style="list-style-type: none"> ✓ Use Version N-1 		<ul style="list-style-type: none"> ✓ Follow up CI ✓ Review test plans, tests
Tuesday	<ul style="list-style-type: none"> ✓ Develop Test Code & Code for Version N ✓ Meet with users to Discuss Action Taken Regarding Feedback From Version N-1 	<ul style="list-style-type: none"> ✓ Meet with developers to give Feedback and Discuss Action Taken from previous actions 	<ul style="list-style-type: none"> ✓ System Architect to review code and test code 	<ul style="list-style-type: none"> ✓ Follow up CI ✓ Review test plans, tests
Wednesday	<ul style="list-style-type: none"> ✓ Develop test code & code for Version N 			<ul style="list-style-type: none"> ✓ Review test plans, tests ✓ Follow up CI
Thursday	<ul style="list-style-type: none"> ✓ Complete Test Code & Code for Version N ✓ Complete GUI tests for Version N-2 			<ul style="list-style-type: none"> ✓ Review test plans, tests ✓ Follow up CI



10. Evo contracting template



A Basic Evolutionary Planning Policy

1. Financial Control:

**No project cycle can exceed 2% of total initial financial budget,
before delivering
some measurable, required
results to stakeholders.**

2. Deadline Control:

**No project cycle can exceed 2% of total project time
(For example, one week for a one year project)
before delivering
some measurable,
required
results
to stakeholders.**



3. Value Control:

The next step should always be the one that delivers best stakeholder value for its costs

One way to avoid software project failure is
to refuse to pay for failure.

This will **motivate** software suppliers to
make use of already well-known and well-practiced methods for successful IT and software projects [Larman03, Gilb05].

There are two key ideas that too many people do not practice,
are not trained to practice,
and are not managed well.

The first is the **quantification** of
the values expected by stakeholders of the system,
especially the 'qualitative' aspects.

This gives the **basis for payment.**

The second idea is to **divide** large projects
into an incremental *series of smaller projects*.
This means roughly weekly, or 2% of current projects, per step of value delivery.
Each increment must attempt to **increase** some aspect of stakeholder **value**,
in the *direction of the longer-term requirements*.

This small-step discipline makes sure that
suppliers really know what they are doing,
and are really focussed on value delivery,
rather than their traditional concern for 'technical construction'.

This culture change must be **top management led.**

The software technologists have **consistently failed** for decades,
and the problem has never been technological.



- A clear framework for controlling the project.

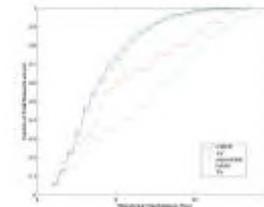
- **The general framework.**

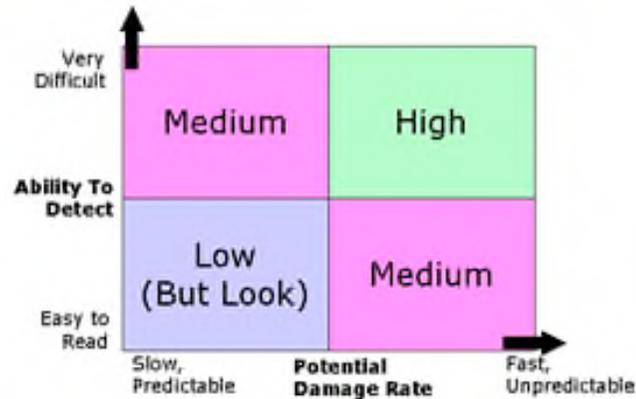
The contract needs to totally avoid fixed commitments, when these are not realistic.

The contract needs to define a framework for helping the partners (supplier (s) and customer)

to produce useful results

for the customer.





The framework should be designed to deal with the inevitable *risks*, and changing *priorities*.

- The notion of defining *results* at each delivery cycle.

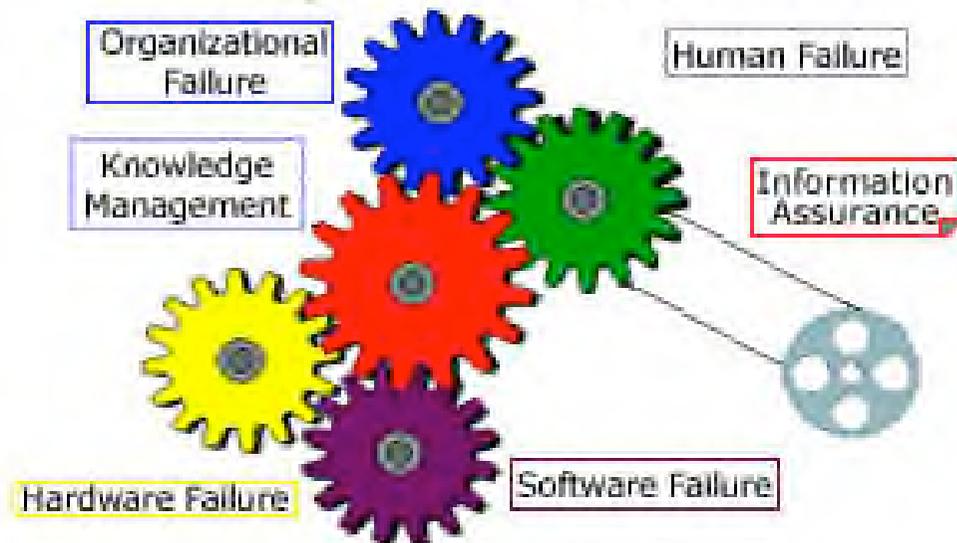
• **Example:**
 “ At the beginning of each cycle the customer will define the primary measurable and testable results they want to achieve, by the end of the cycle.

- The supplier will then suggest the degree of those results that they believe would be possible within given constraints.

- The customer will then agree to this level, or repeat this cycle of setting delivery cycle requirements.

”

System Failure



CYCLE COST ESTIMATION

The notion of defining *costs* at each delivery cycle.

On the basis of a **mutually agreed measurable results;**

the supplier will estimate

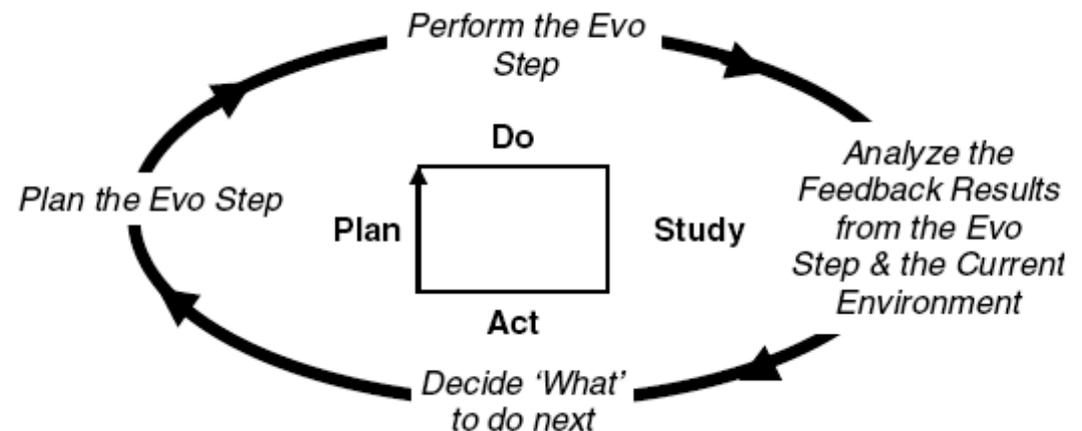
the total costs for their own efforts,

and for all other related costs

the customer is likely to incur.

- If the estimated costs are acceptable to the customer,

they form the basis for invoicing for the cycle.



Drop this in to a Conventional Contract

- Author Tom Gilb .
-

• **Contract Design idea:** designed to work within the scope of a present contract with minimum modification.

An Evo step is considered a step on the path to delivering a phase.

You can choose to declare this paragraph has priority over conflicting statements (30.1), or to clean up other conflicting statements in the initial contract basis.

• **§30. Evolutionary Result Delivery Management.**

• 30.1 **Precedence.** This paragraph has precedence over conflicting paragraphs.

• 30.2 **Steps of a Phase.** The Customer may optionally undertake to specify, accept and pay for evolutionary usable increments of delivery, of the defined Phase, of any size. These are hereafter called "**Steps**".

• 30.3 **Step Size.** Step size can vary as needed and desired by the Customer, but is assumed to usually be based on a regular weekly cycle duration.

• 30.4 **Intent.** The intent of this evolutionary project management method is that the Customer shall gain several benefits: earlier delivery of prioritized system components, limited risk, ability to improve specification after gaining experience, incremental learning of use of the new system, better visibility of project progress, and many other benefits. This method is the best known way to control software projects [Larman03].

• 30.5 **Specification Improvement.** All specification of requirements and design for a phase will be considered a framework for planning, not a frozen definition. The Customer shall be free to improve upon such specification in any way that suits their interests, at any time. This includes any extension, change or retraction of framework specification which the Customer needs.

• 30.6 **Payment for Acceptable Results.** Estimates given in proposals are based on initial requirements, and are for budgeting and planning purposes. Actual payment will be based on successful acceptable delivery to the Customer in Evolutionary Step deliveries, fully under Customer Control. The Customer is not obliged to pay for results which do not conform to the Customer-agreed Step Requirements Specification.

• 30.7 **Payment Mechanism.** Invoicing will be on a Step basis triggered by end of Step preliminary (same day) signed acceptance that the Step is apparently as defined in Step Requirements. If Customer experience during the 30 day payment due period demonstrates that there is a breach of specified Step requirements, and this is not satisfactorily resolved by the Company, then a Stop Payment signal for that Step can be sent and will be respected until the problem is resolved to meet specified Step Requirements.

• 30.8 **Invoicing Basis.** The documented time and materials will be the basis for invoicing a Step. An estimate of the Step costs will be made by

• the Company in advance and form a part of the Step Plan, approved by the Customer.

• 30.9 **Deviation.** Deviation plus or minus of up to 100% from Step cost and times estimates will normally be acceptable (because they are small in absolute terms), as long as the Step Requirements are met. (The Customer prioritises quality above cost). Larger deviations must be approved by the Customer in writing before proceeding with the Step or its invoicing.

• 30.9 **Scope.** This project management and payment method can include any aspect of work which the Company delivers including software, documentation and training, maintenance, testing and any requested form of assistance.



- **The notion of *learning* from results-to-date, at each cycle.** (same as CMMI 5, Defect Prevention Process)

- **The contract will stipulate the allocation of specific time, each cycle,**

- to analyze results to date,
 - problems,
 - risks,
- and to change
 - plans,
 - processes,
 - suppliers and
 - anything necessary
 - in order to maintain progress towards necessary results and costs.

- **At the extreme this can include**

- shutting down the project,
- or removing suppliers,
 - when they clearly are incapable of delivering the results expected.



•The contract should give specific general guidance

•regarding the method of prioritization

• of what to do at each cycle.

•For example:

•At each cycle the customer has the right

•to select the implementation

•that they estimate

•will give them the greatest numeric progress

•towards their long term objectives.

Hold these. I have to go back for my wife!



'Hold these. I have to go back for my Wife'

Measurable Statement of 'The most critical target improvements *intended*'

- **A set of no more than ten of**
 - the most critical business improvement targets
 - that will be delivered or enabled by the contracted system,
 - will be stated in an appendix to the contract,
 - as a statement of purpose.
- **The customer has the right to**
 - update this and
 - change it at any time
 - for any reason.
- **It serves to**
 - inform the suppliers
 - as to the long range objectives
 - of their client.
- **It helps**
 - focus the customer
 - on working towards
 - what they have stated there.
 -



• **Productivity:**

Ambition: at least 10% increase per year.

•**Savings:**

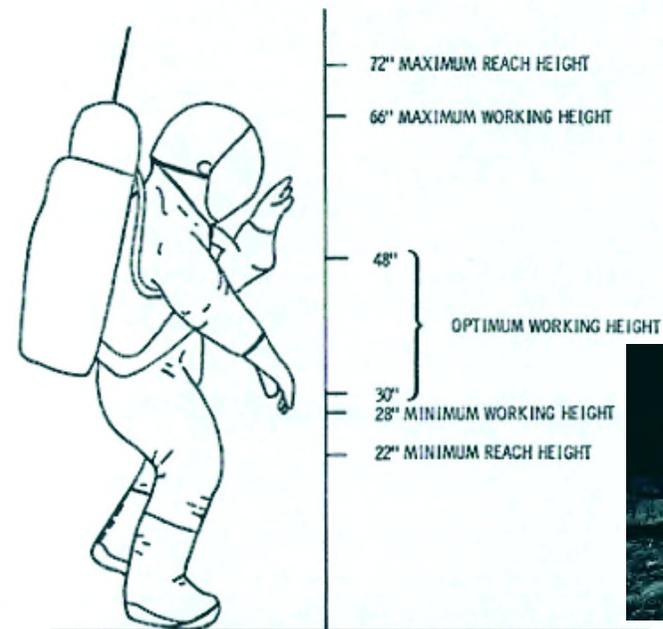
Ambition: Cost per customer reduced by 50% within 3 years.

•**Service:**

Ambition: reduction by 90% of customer call wait time within 2 years.



- **There need to be clear statements of all overall constraints**
 - in the contract appendix,
- **and allowance for specific cycle constraints**
 - as need dictates.
- **The point is that**
 - estimates of results and costs
 - must be made with knowledge of those constraints.
- **Constraints can be of two kinds:**
 - **scalar constraints** –
 - regarding performance and quality levels;
 - **and non-scalar constraints**
 - such as legal, cultural, contractual requirements.



ASTRONAUT REACH CONSTRAINTS

Evo Step a Result Delivery Step: Here is a template made for this client to document each Evo step:
Evolutionary Delivery Step Plan (the Form)

Buyer Requirements.

Functional Requirements: _____

Benefit/Quality/Performance Requirements:

Reference Tag: _____
 Ambition Level: _____
 Quantification Scale: _____
 Meter [END STEP ACCEPTANCE TEST] _____
 Past Level [<when?, where?>] _____. Source: _____.
 Fail Level [<when?, where?>] _____. Source: _____.
 Goal level [<when?, where?>] _____. Source: _____
 Cross Reference to more detail:

REPEAT THE ABOVE TEMPLATE FOR ALL DELIVERABLE RESULTS

Resource Constraints:

- Calendar Time:
- Work-Hours:
- Qualified People:
- Money (Specific Cost Constraints for this step):
- Other Constraints
- Design Constraints
- Legal Constraints
- Generic Cost Constraints
- Quality Constraints

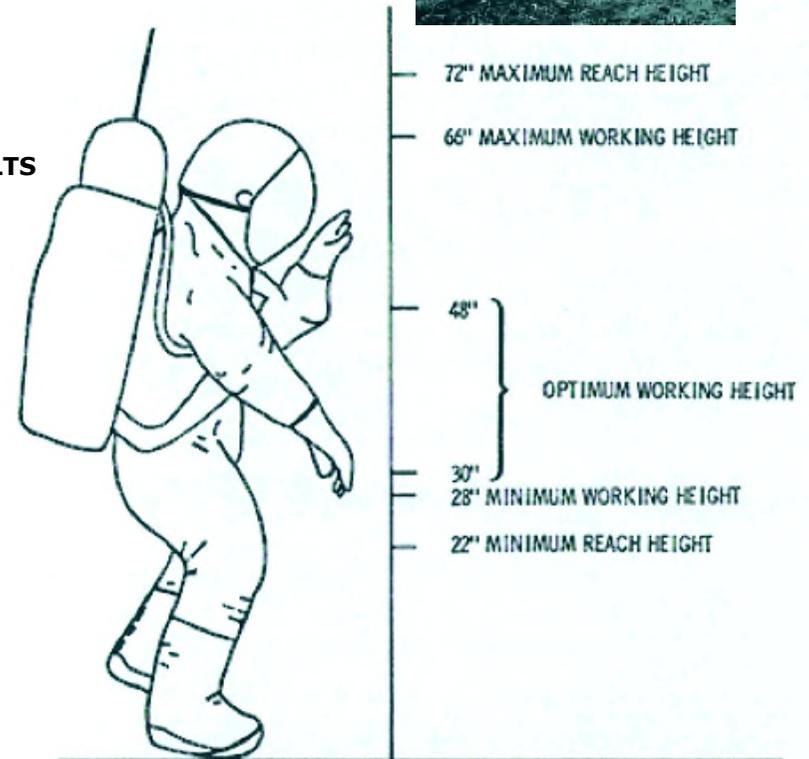
Assumptions:

A1:

Dependencies:

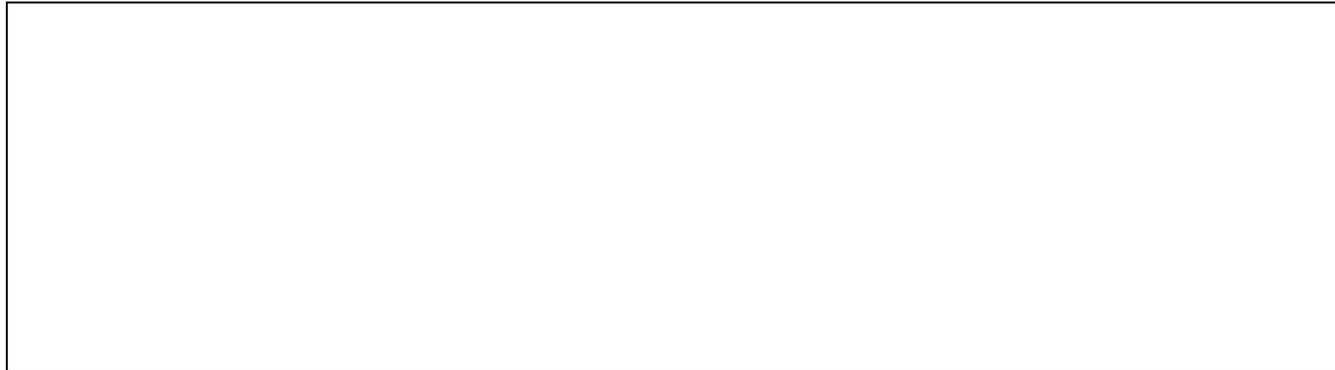
D1:

The Resource Constraints can be specified for the sum of all defined results, or for each one of them.



ASTRONAUT REACH CONSTRAINTS

Summary Slides



COVER FEATURE

Iterative and Incremental Development: A Brief History



Although many view iterative and incremental development as a modern practice, its application dates as far back as the mid-1950s. Prominent software-engineering thought leaders from each succeeding decade supported IID practices, and many large projects used them successfully.

Craig Larman
Valtech

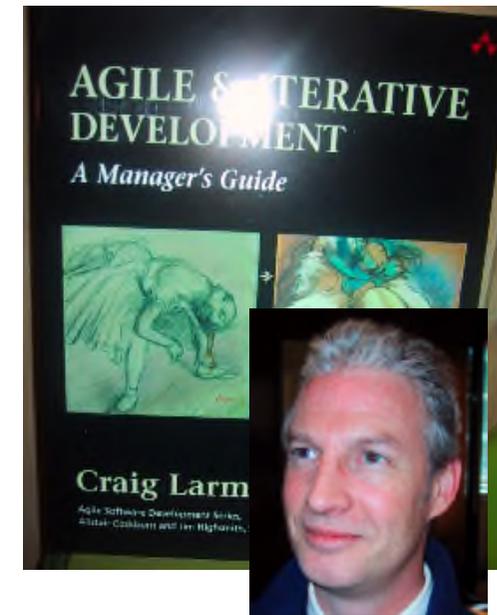
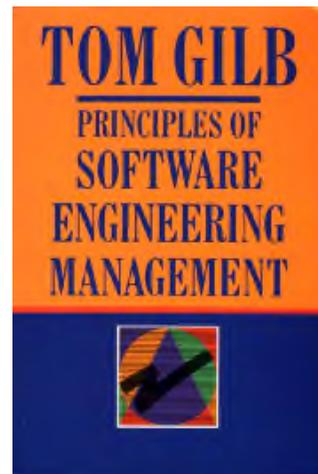
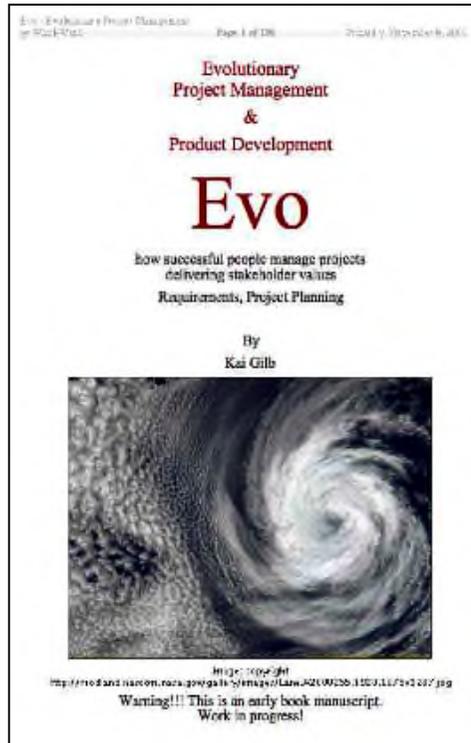
Victor R. Basili

As agile methods become more popular, some view iterative, evolutionary, and incremental software development—a cornerstone of these methods—as the “modern” replacement of the waterfall model, but its practiced and published roots go back

opment” merely for rework, in modern agile methods the term implies not just revisiting work, but also evolutionary advancement—a usage that dates from at least 1968.

PRE-1970

Some Literature



Evo Wisdom

"The shortest way to do many things is to do only one thing at once."

-- Samuel Smiles (1812-1904): Self-Help, 1859.

"Little drops of water,
little grains of sand -
Make the mighty ocean,
and the pleasant land."

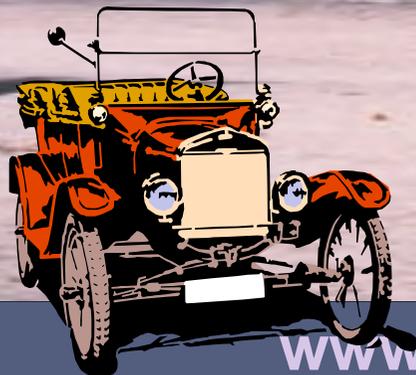
--Julia Carney

"Nothing is particularly hard if you divide it into small cycles."

--Henry Ford

"Great things are not done by impulse, but by a series of small things brought together."

--Vincent van Gogh



Tom@Gilb.com

