Evolutionary Project Management: Multiple Performance, Quality and Cost Metrics for Early and Continuous Stakeholder Value Delivery - An agile approach.

1-hour lecture at the ICEIS conference, Porto, Portugal, on the 14th APRIL 2004

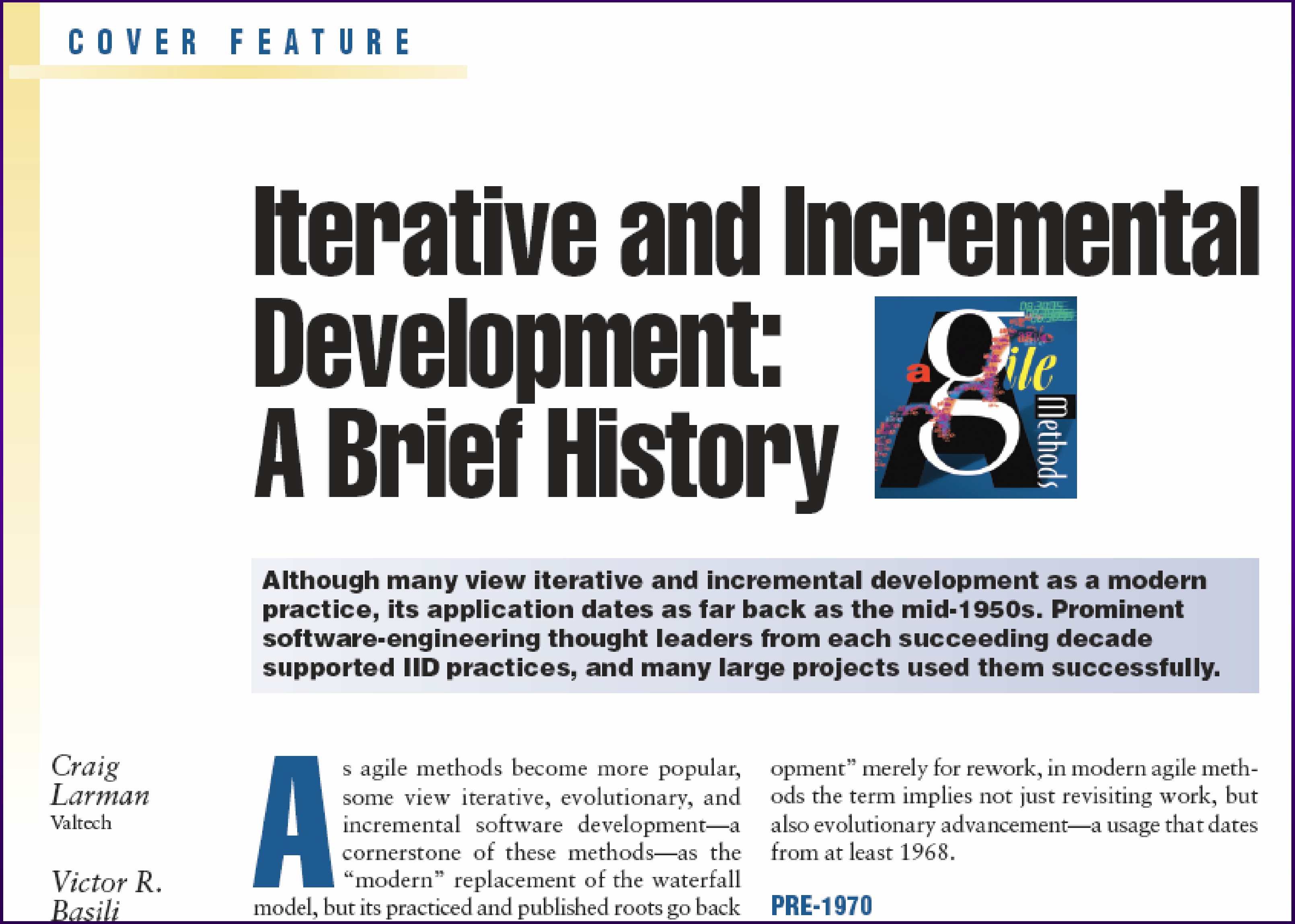
By Tom Gilb

Evo Values: the top 5

- **learn** rapidly by realistic measurement
- deliver real value to stakeholders early, frequently, at every step.
- be humble about complex systems: simplify and attack problems one small step at a time
- delegate power to the coalface, by focusing on end results, and not on methods, or on well intended bureaucracy.
- admire, applaud and reward a team based on the flow of measurable results: stakeholder value in relation to costs.

Evo

Evo History June 03 IEEE ‘Computer’



COVER FEATURE

Iterative and Incremental Development: A Brief History

agile
methods

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

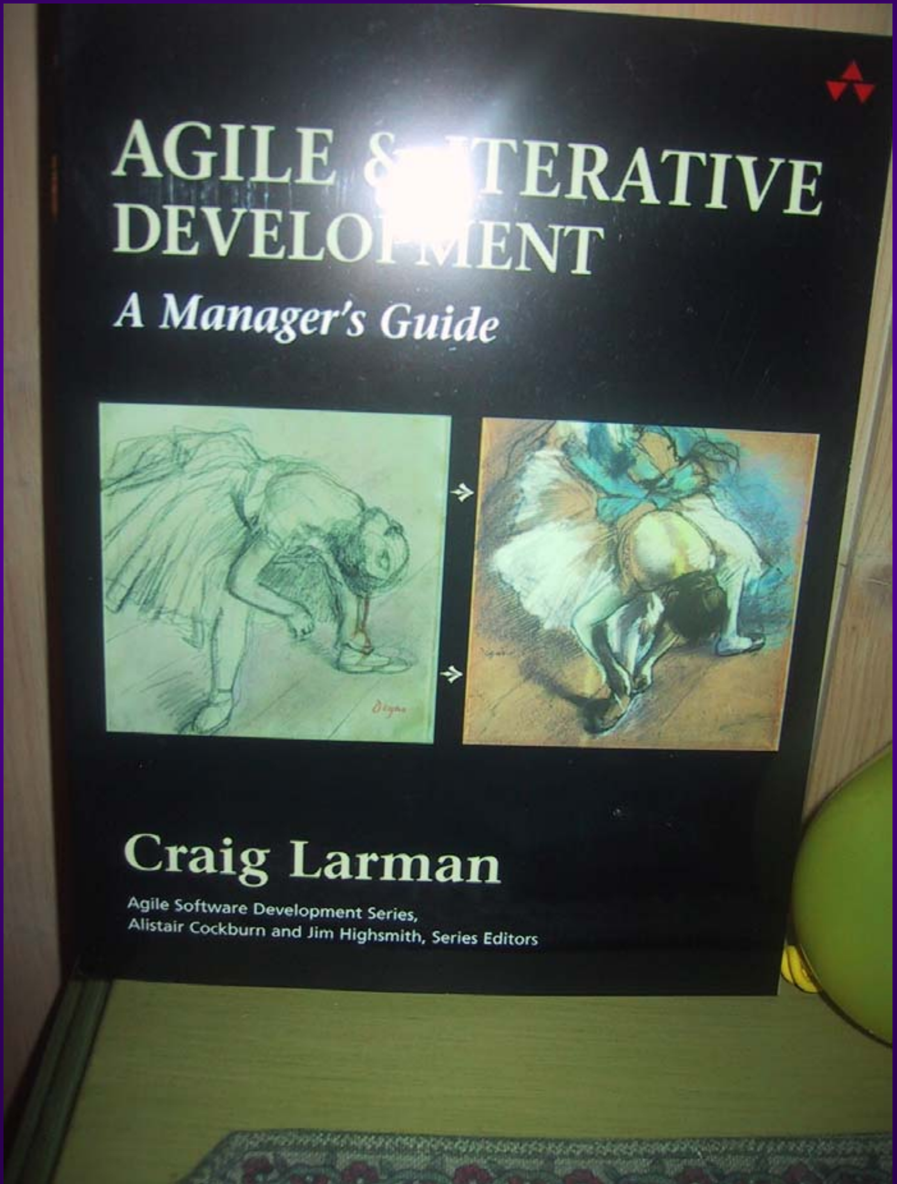
Victor R. Basili

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.

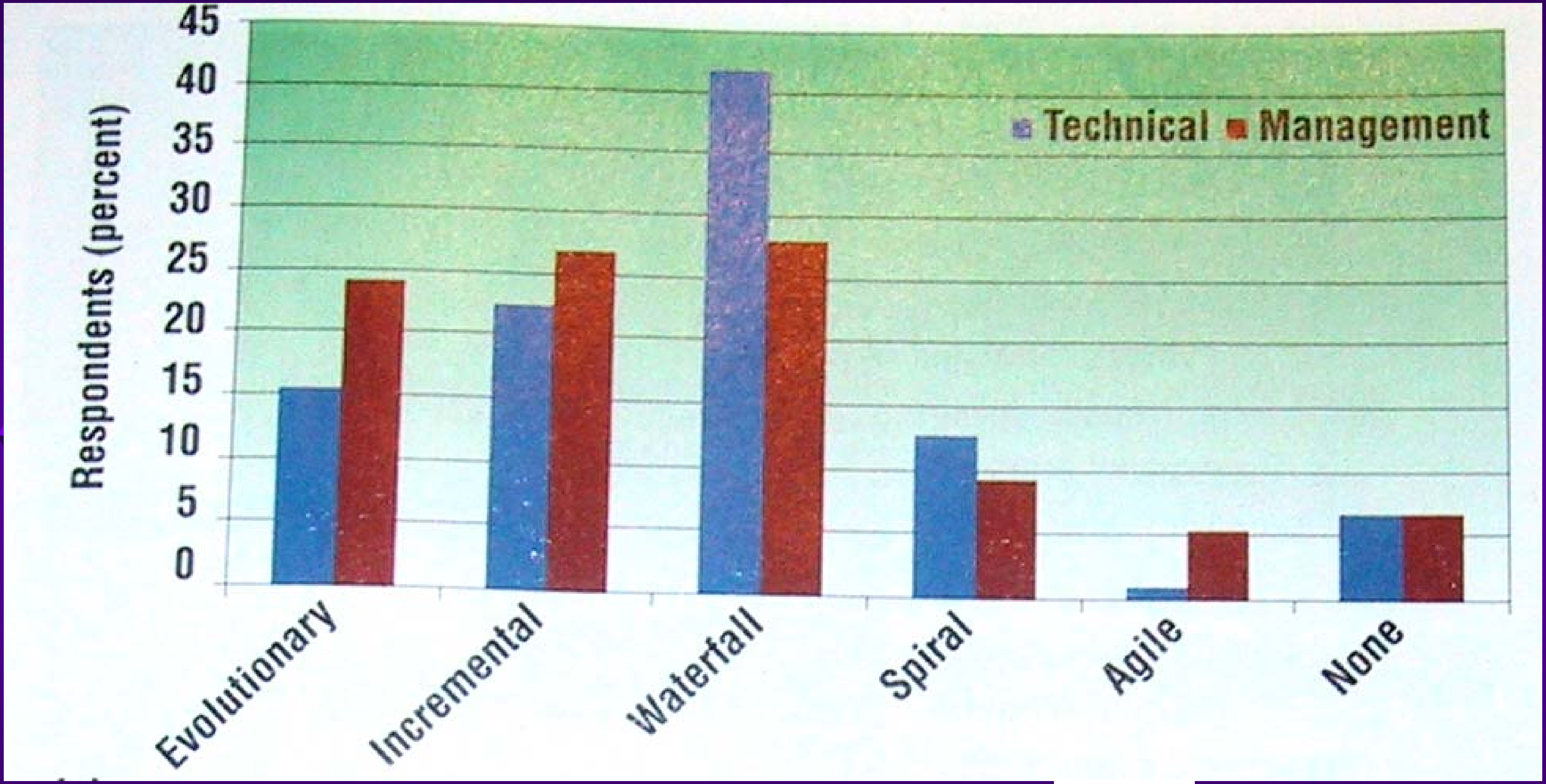
PRE-1970

Larman's Book Comparing Evo methods, including Gilb's Evo (Chapter 10)


AGILE & ITERATIVE DEVELOPMENT
A Manager's Guide
Craig Larman
Agile Software Development Series,
Alistair Cockburn and Jim Highsmith, Series Editors

Craig Larman

% Use of Evo and other life cycle models 2002



QuickTime™ and a TIFF (LZW) decompressor are needed to see this picture.

Colin J Neill and Philip A. Laplante

Requirements Engineering:

The State of the Practice

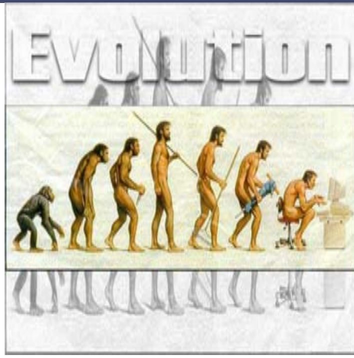
IEEE Software Nov/Dec 2003, Pp 40-45

QuickTime™ and a TIFF (LZW) decompressor are needed to see this picture.

When You Do Not Need Evo

- 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)

’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”

Evo characteristics

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?

- 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

What are the major technology process changes?

You need clear, quantified *requirements* to ‘evolve’ towards - ‘stakeholders view’ requirements

Test process: changes - rapid, early

User involvement continuous

Teamwork towards one user result

Open Ended Architecture to Evo in

Backroom and Frontroom management

How do you best manage it?

Motivate development team by results

Empower stakeholders to think value

Train development in Evo

Equip with Evo ‘tools’ (templates etc)

Support and advise (new) teams

Feed budget to teams with best value

What are the pitfalls?

Failing to focus on real value

Failing to use value/cost priority

Failure to train and support after training

Giving up too early and falling back on old habits

Lack of management commitment

Lack of management support

Defeatism: giving up rather than cracking problems.

What are the pre-requisites?

(eg componentised architecture)

Clear management policy

Evo tools (standards)

Trained Project Management

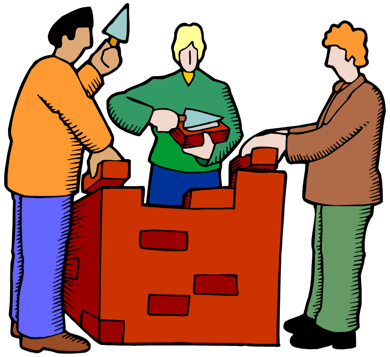
Reward structure

Long term quantified objectives

Evo plan for Evo method

Enthusiastic volunteer projects

Open architecture is useful but not a start condition!

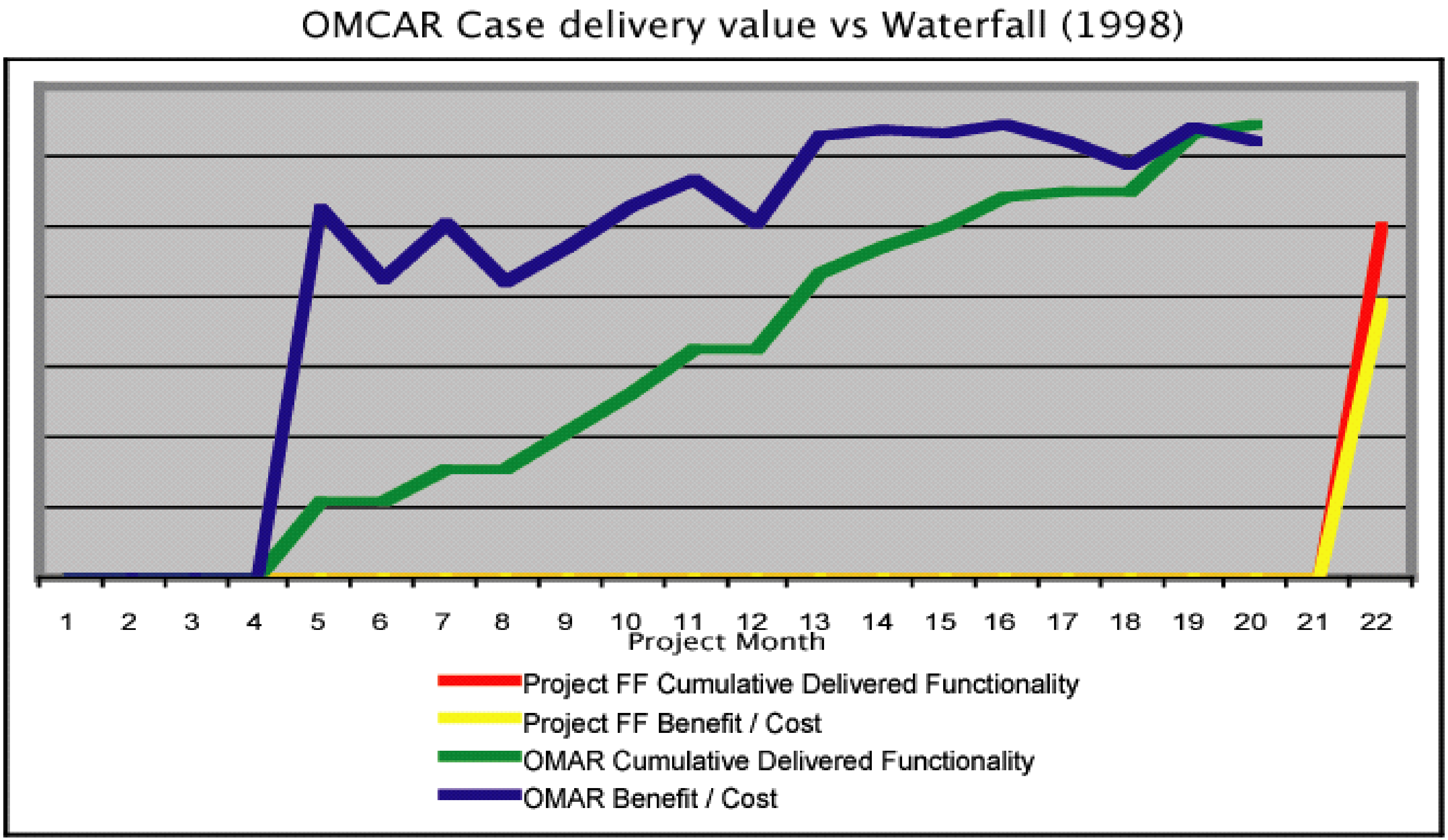


Are there types of apps/users that EVO might not be appropriate for?

In principle no, but

- Some projects will have greater benefits
- Even ‘old’ failing projects can be ‘saved’ by Evo restructuring
- Bigger projects will have more benefit
- There may be some projects with ‘constraints’ (like dates for laws or consortium agreements) so you can’t really deliver much before a distant time.

Omar

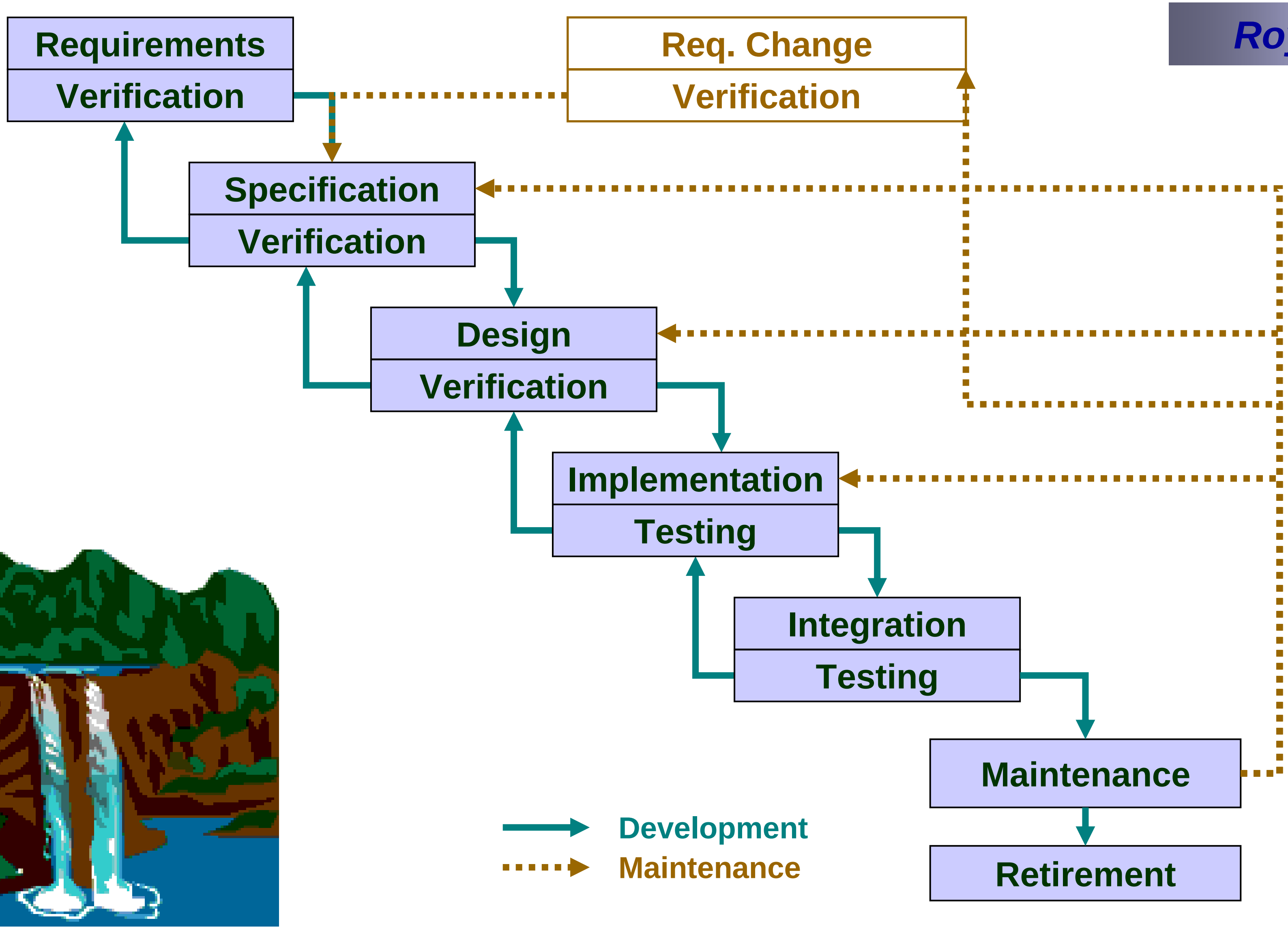


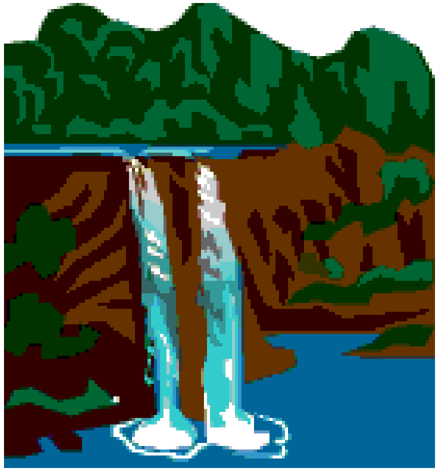
Using Evolutionary Project Management, ToGet More Quality, From Fewer Resources, In Less Time; By Stuart Woodward, DoubleHelix Software & Services Ltd.

IEEE Computer Oct 1999, Stuart Woodward , “More Quality From Fewer Resources in Less Time”

The Waterfall Model

Royce, 1970





What’s wrong with the Waterfall model?

Risk mitigation postponed until late stages.

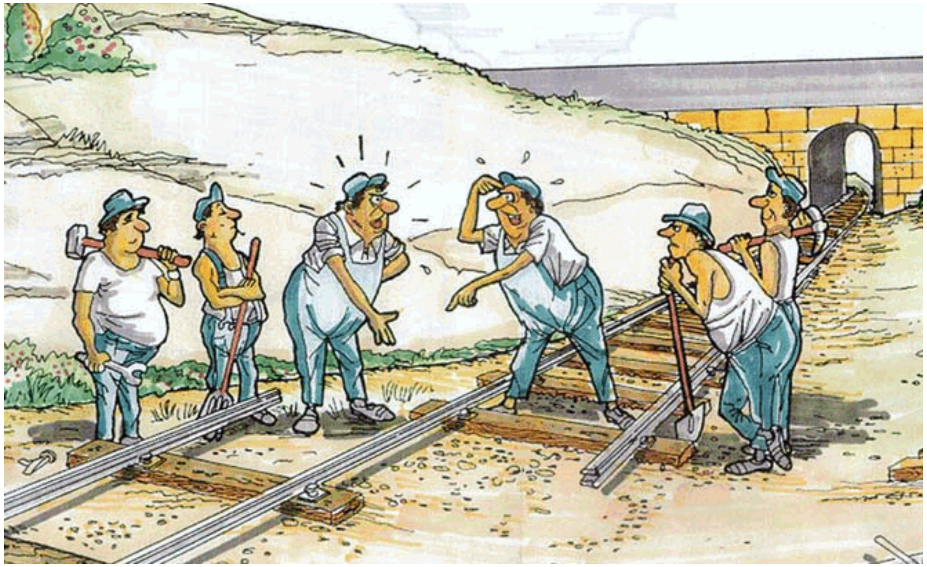
Document-based verification until late stages.

Attempt to stipulate unstable requirements too early.

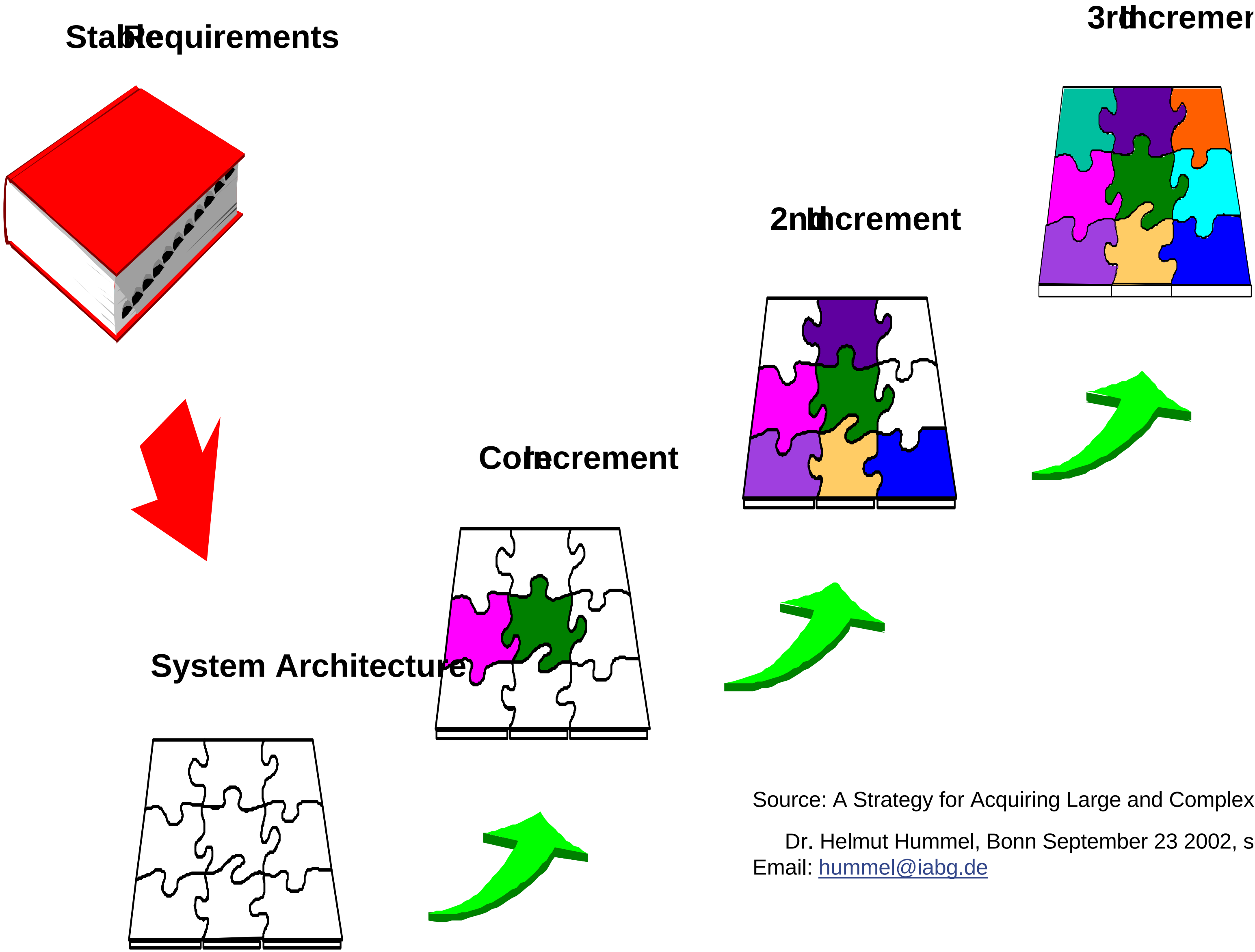
Operational problems discovered too late.

Lengthy modification cycles and much rework.

The
inevitable
result...



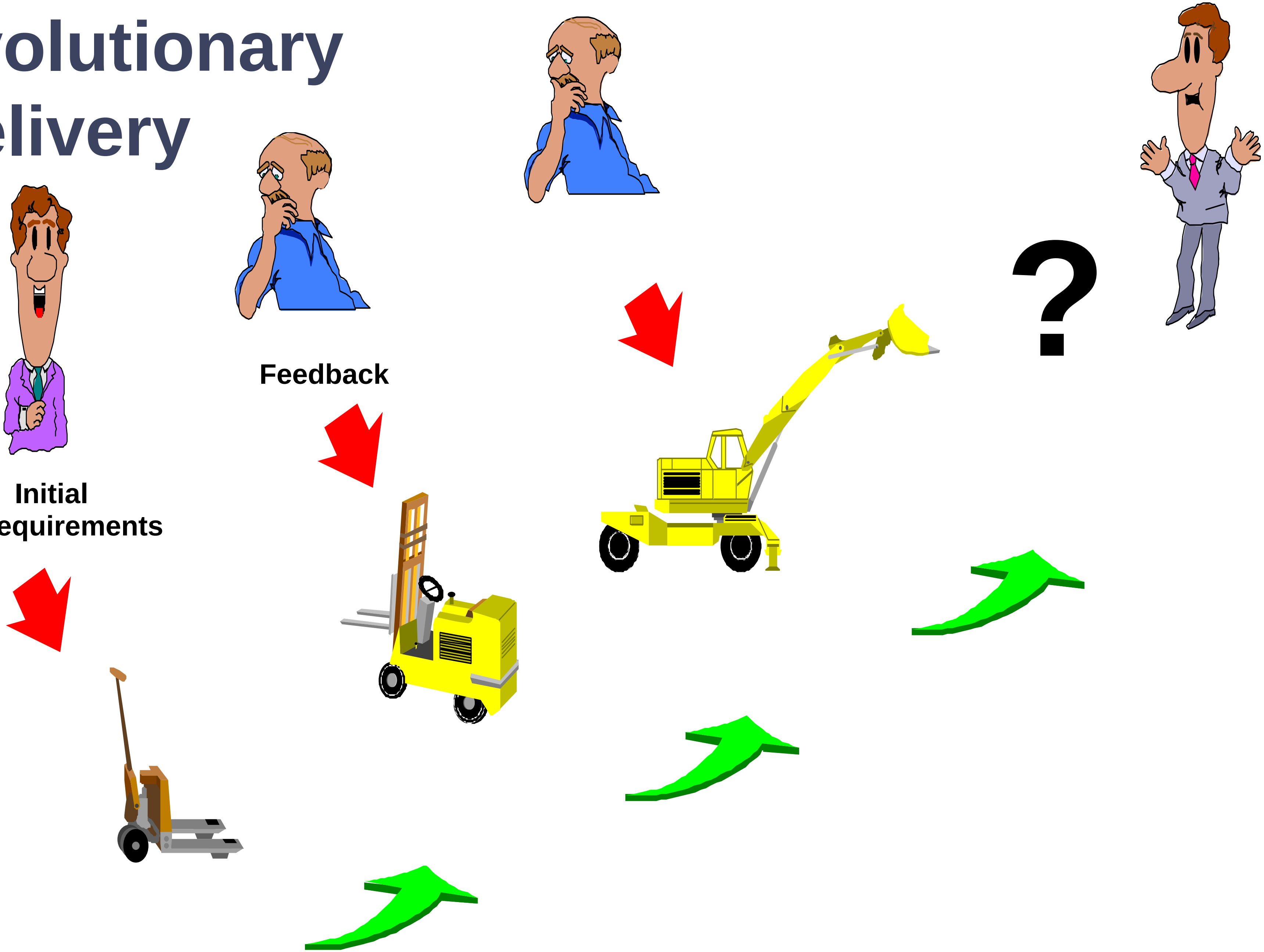
Incremental Development



Source: A Strategy for Acquiring Large and Complex Systems

Dr. Helmut Hummel, Bonn September 23 2002, see note for paper
Email: hummel@iabg.de

Evolutionary Delivery

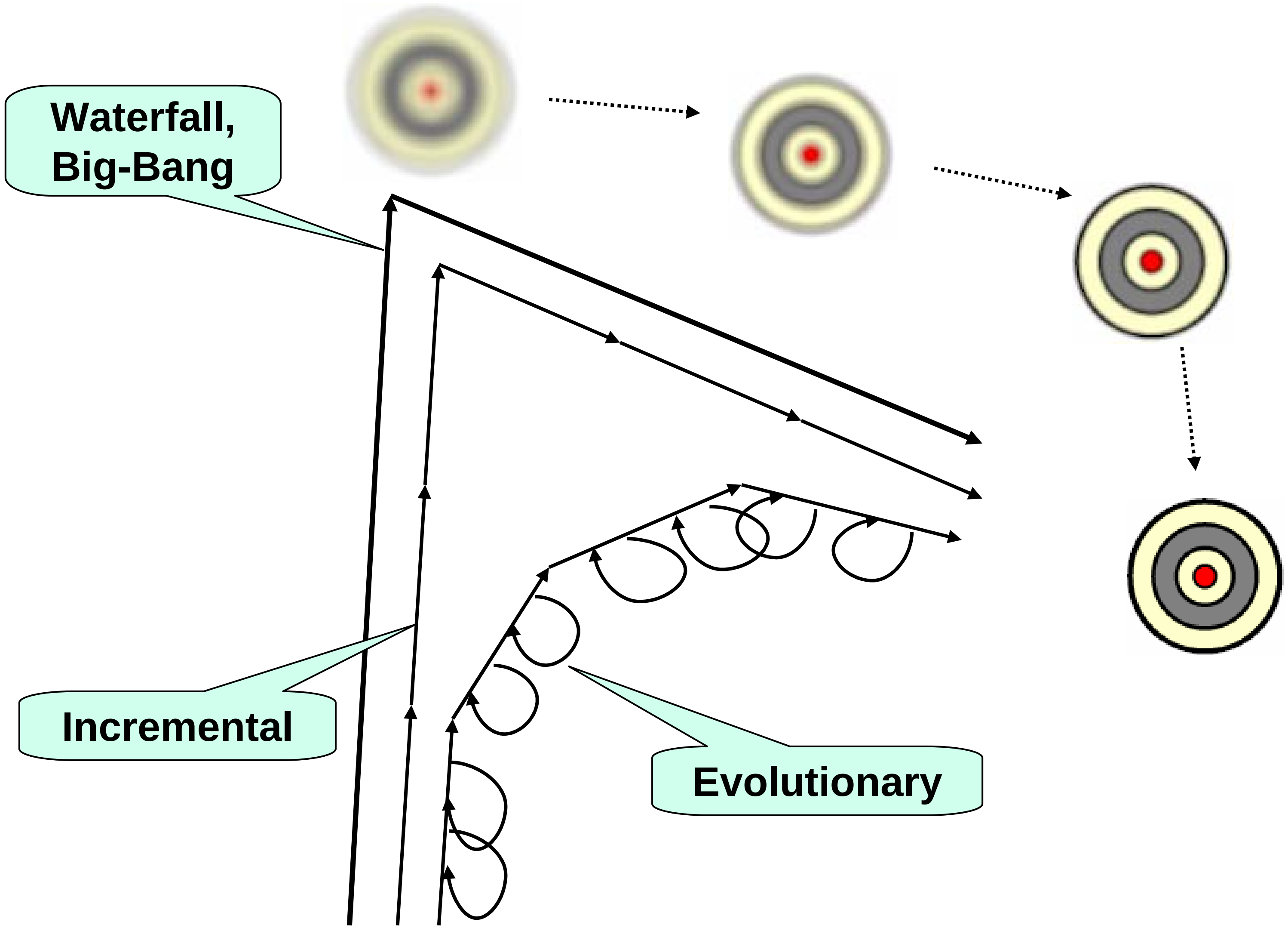


Feedback

Initial Requirements

?

Evo adjusts to changing requirements



Courtesy Niels Malotaux July 16 2002 based on a diagram in Gilb88

Evo Software Project Management:
The Agile Metrics Option

Agile Evo Summary:

A recent London Times survey report indicated that only 13% of 1500 surveyed IT projects were ‘successful’ [Times].

Other reports ([Standish], Chaos) indicate that about half of the surveyed projects were considered total failures,

- the same percentage as US Department of Defense estimated its software projects failed.

We must be doing something very wrong.

What can the IT Manager and IT Project Manager do about this situation in practice?

Some people recommend complex development process standards such as CMM, CMMI, SPICE and their like.

- I am not convinced that these are good medicine for even very large systems engineering projects,
- and certainly they are overly complex for most IT projects in Europe.

Some people recommend agile and extreme programming methods –

- these are closer to my heart –
- but maybe, for non-trivial projects -
- they are ‘too simple’?

I will offer you my advice in the form of a short simple defined process.

My main addition to the agile concepts is that I believe they need to focus on the top few critical stakeholder objectives.

These top objectives need to be quantified and measurable in practice.

This simple quantification device is missing from most methods,

- but I believe that quantified management a necessary minimum to control all but the smallest upgrade efforts.

The Simplest and Best Agile Project Method 1

Background:

A number of 'agile' methods have appeared, trying to simplify project management and systems implementation.

They have all missed the central point,

- namely evolutionary project management (Evo),
- using quantified feedback about central goals and budgets
- which would allow them complete freedom to simplify, and to succeed.
- Here is my suggestion for ultimate agility.

The Simplest and Best Agile Project Method 2

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.**

8. **When all Goals are reached: 'Claim success and move on'**

   a. **Free remaining resources for more profitable ventures**

Agile project Management Policy

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.

“The End”.

That is the end of this slides. You need read no more. But I can write an ‘appendix’, in case anyone would like more detail! Here it is.

APPENDIX!

I will comment on the process definition, statement by statement.

‘The Simplest and Best Agile Project Method’

The Gilb Agile Process (“GAP” of course) is ‘simplest method because of its sharp focus at a ‘high level’, on the ‘end results’.

This allows us to avoid distracting management attention

with the supporting processes, designs and requirements, needed to deliver the results.

The supporting processes, designs, and requirements do need to exist of course,

but our GAP process is neutral,

and in fact encourages competition and selection

of the fittest supporting processes at any step.

This is essentially different from making user-driven lists of functions to program into the system – typical of conventional Agile methods.

It focuses on the main outcome, for example high security, ease of use, or flexibility.

GAP is the ‘Best’ Agile process because it focuses on *numerically defined* and tracked critical business or technical goals of a project.

This numeric focus is in sharp contrast to the non-numeric ‘yellow sticky’ mentality of Conventional Routine Agile Processes.

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).

- **Projects need to learn to focus on *all stakeholders* that arguably can affect the success or failure.**

- **The *needs* of these stakeholders must be *determined* – by any useful methods – and *converted* into project *requirements*.**

-

- **By contrast the Conventional Agile Model**
  - **focuses on a User/Customer ('in the next room').**
  - **Good enough if they *were* the only stakeholder.**
  - **But disastrous for most real projects,**
    - **where the critical stakeholders are more varied in type and number.**

- **Conventional Agile processes, due to this dangerously narrow requirements focus, risk outright failure,**
  - **even if the 'Customer' gets all *their* needs fulfilled.**

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.*

- **In the Gilb Agile Process, the project is initially defined in terms of clearly stated, quantified, critical objectives.**

- **During the project, these long-term (Project completion term)**
  - **objectives can be changed, and tuned,**
  - **based on practical experience and feedback,**
  - **from each Evo step.**
  - **They are not cast in concrete, even though they are extremely clear.**

- **Conventional Agile methods do not have any such quantification concept.**
-
- **Conventional vague ideas, un-measurable, un-testable, un-quantified, and un-deadlined requirements, do not count as true long term goals, in our view.**

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

- **Conventional methods**
  - **do not seem to directly, and in detail, manage the array of limited resources we have.**
  - **But admittedly there are some such devices in place in the Conventional Agile methods,**
    - **such as the incremental weekly (or so) development cycle.**

- **the GAP method sets an explicit numeric budget for any useful set of limited resources – but it does not stop there!**

- **Our Evo cycles will both**
  - **estimate,**
  - **record actual resource use,**
  - **and analyze the deviation, on *every* Evo cycle,**
  - **in order to understand and control the economics of the project –**
  - **concurrently with the performance characteristics.**

- **This is the essential distinction between incremental and evolutionary development methods.**

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

- **all these key quantified performance targets, and resource budgets, are presented simultaneously on a single overview page.**
- **additional detail about them can, of course, be captured off of this one ‘focus’ page.**
- **this set of top level objectives is not frozen.**
- **It can be updated as the result of**
  - **both internal Evolutionary (Evo) step learning,**
  - **or of external pressures and insights.**

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

- **once the objectives, the version derived from our *developer's* understanding of stakeholder needs, are clearly articulated –**
  - **we need to go back to the *real stakeholders***
  - **and check that they agree with our 'clear' (but potentially incorrect or outdated) interpretation.**

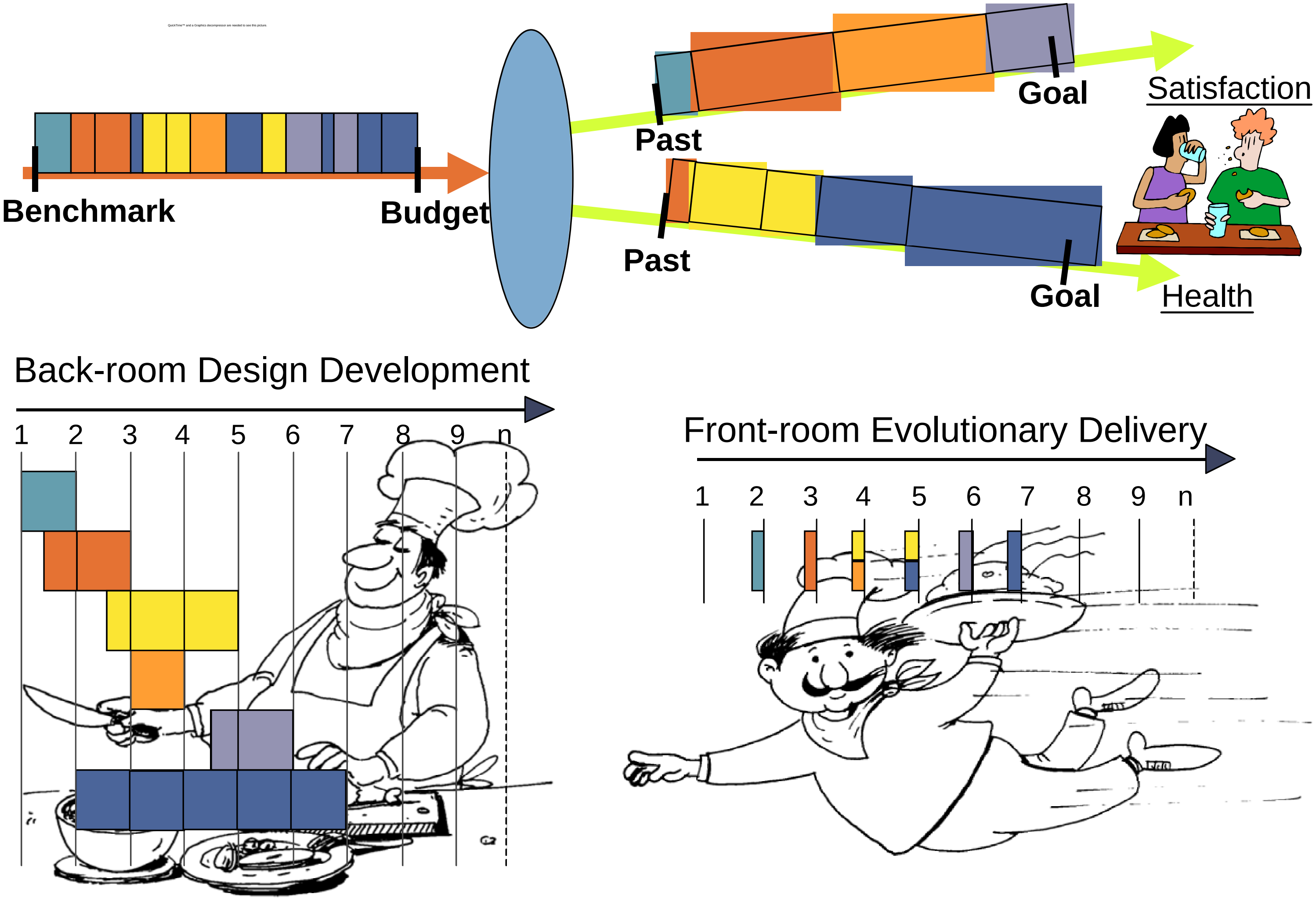
- **it is certainly a wise precaution to *check* back later,**
  - ***during* the project evolution,**
  - **with the *specific stakeholders***
  - **that will be *impacted* with a *particular* Evo step,**
    - as to how they feel about a particular choice of step content (design) - (that impacts the performance and cost aspect estimates):
    - are estimates realistic in the real implementation environment?,
    - and to check for any new insights regarding the long term objectives.

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

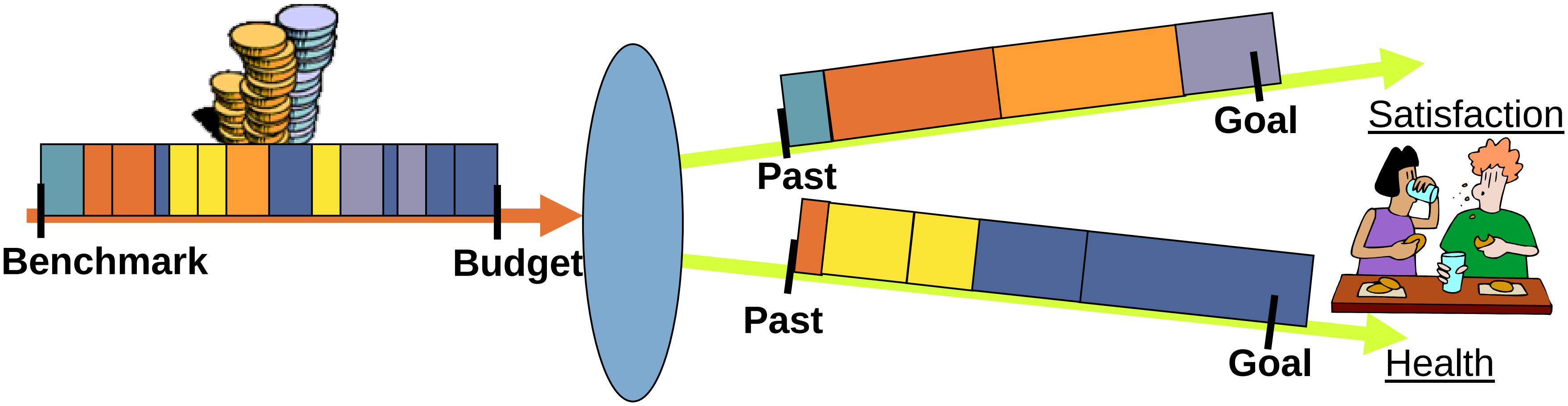
- **the weekly delivery cycle is adopted by Conventional Agile methods – good.**
- **but the notion of *measurement*, on multiple performance and resource objectives, is absent.**
- **the Conventional notion of *agreeing with a user*, about function to be built, during that weekly cycle is healthy, but**
  - **the GAP method is focused on**
    - **systematic, weekly cycle, measured delivery**
    - **towards long-range higher-level objectives, w**
    - **ithin numeric, multiple, resource-constraints.**
- **this means that the GAP method is more clearly focused on**
  - **the wider stakeholder set values,**
  - **and on the total resource cost management.**
- **the GAP method is NOT focused on system ‘construction’ (‘we are programmers, therefore we write code’).**
- **the GAP method is focused on delivering useful results from an organically whole system.**
  - **This means that we are not focused on ‘writing code’.**
    we reuse, buy, or exploit existing code just as happily as to write our own.
    - **We build databases, train and motivate users, improve hardware, telecommunications, websites, improve working environment, improve motivation.**
    - **So we become more like systems engineers (‘any technology to deliver the results!’), than programmers (‘what can we code for you today?’).**

Figure: Evolutionary result delivery takes system components readied for integration in the 'backroom' using arbitrary acquisition duration (as in kitchens), and presents them to stakeholders in frequent short Evolutionary result delivery cycles (as in the dining room). (Ill. By Kai Gilb)

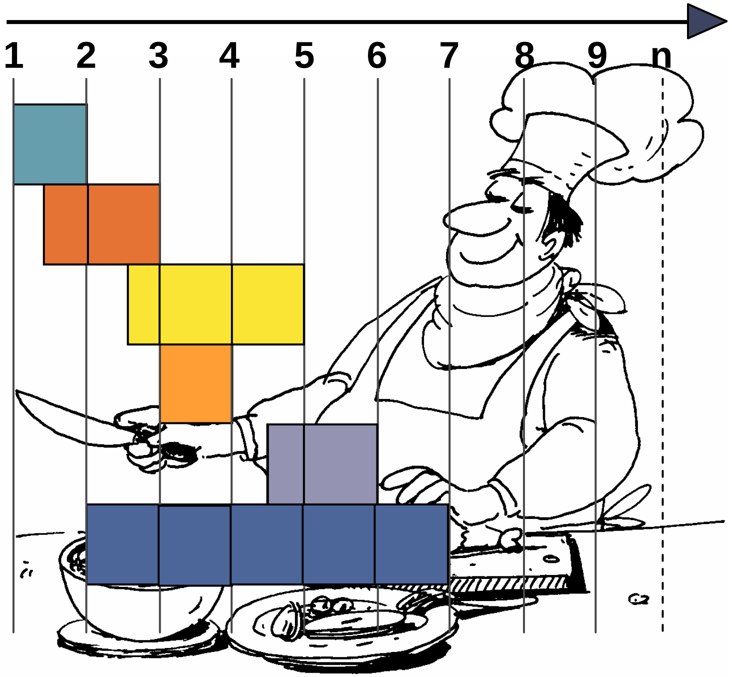
Costs / Effects


QuickTime™ and a Graphics decompressor are needed to see this picture.
Benchmark
Budget
Past
Goal
Satisfaction
Past
Goal
Health
Back-room Design Development
1 2 3 4 5 6 7 8 9 n
Front-room Evolutionary Delivery
1 2 3 4 5 6 7 8 9 n

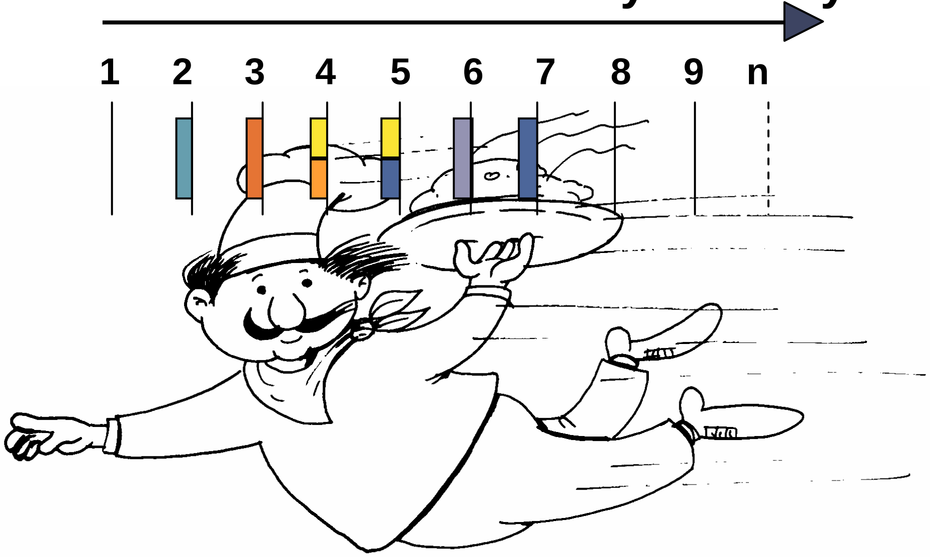
Costs / Effects



Back-room Design Development



Front-room Evolutionary Delivery



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.

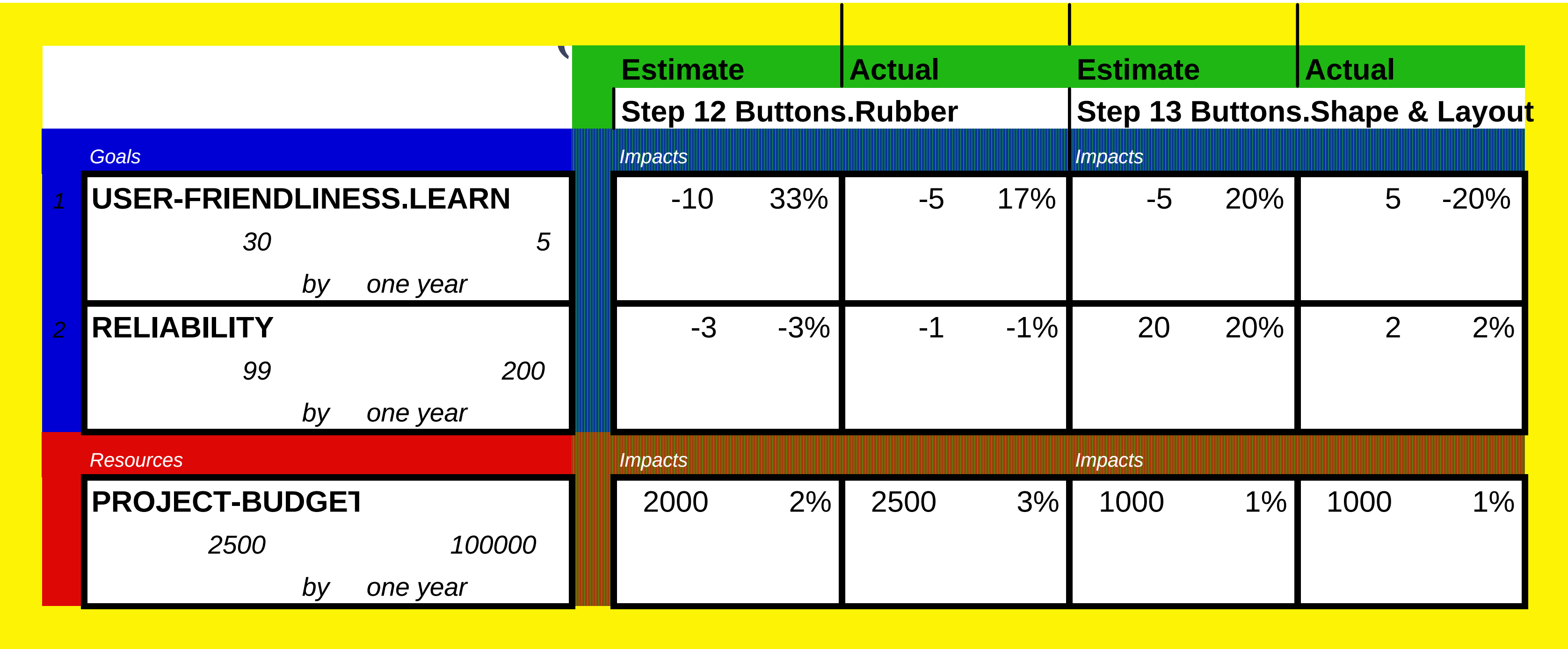
- **All agile methods agree that the development needs to be done in short, frequent, delivery cycles.**

- **the GAP method, specifically insists that the closed loop control of each cycle is:**

  - done by numeric pre-cycle estimates,
  - end-cycle measurements,
  - analysis of deviation from estimates,
  - and appropriate change to immediate planned cycles,
  - to estimates,
  - and to stakeholder expectation management

    – **(‘this is going to late, if we don’t do X’).**

Figure: the use of an Impact Estimation table [CE, POSEM, WWW] to plan and track critical performance and cost characteristics of a system



		Estimate		Actual		Estimate		Actual	
		Step 12 Buttons.Rubber				Step 13 Buttons.Shape & Layout			
	Goals	Impacts				Impacts			
1	**USER-FRIENDLINESS.LEARN** 30 5 by one year	-10	33%	-5	17%	-5	20%	5	-20%
2	**RELIABILITY** 99 200 by one year	-3	-3%	-1	-1%	20	20%	2	2%
	Resources	Impacts				Impacts			
	PROJECT-BUDGET 2500 100000 by one year	2000	2%	2500	3%	1000	1%	1000	1%

- *The pair of numbers in the three left hand columns (30, 5 etc.) are defined benchmarks (30, 99, 2500) and Goal levels (5, 200, 100,000).*
- *The '%' figures are the real scale impacts (like 20) converted to a % of the way from benchmark to the Goal levels (like 20% of the distance from benchmark to Goal).*

8. When all Goals are reached: *'Claim success and move* on' [Gerstner] *Free remaining resources for more profitable ventures.*

- **one advantage with numeric Goal levels,**
  - **compared to a stream of yellow stickies from users,**
  - **is that it is quite clear when your objective is reached.**
  - **No additional effort should be expended to improve upon it,**
    - **unless a new improved target level is set.**

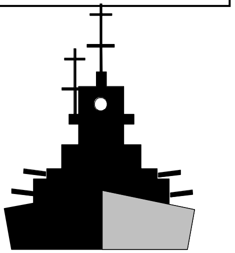
- **the numeric goal level is the success level,**
  - **success is well defined formally in advance.**

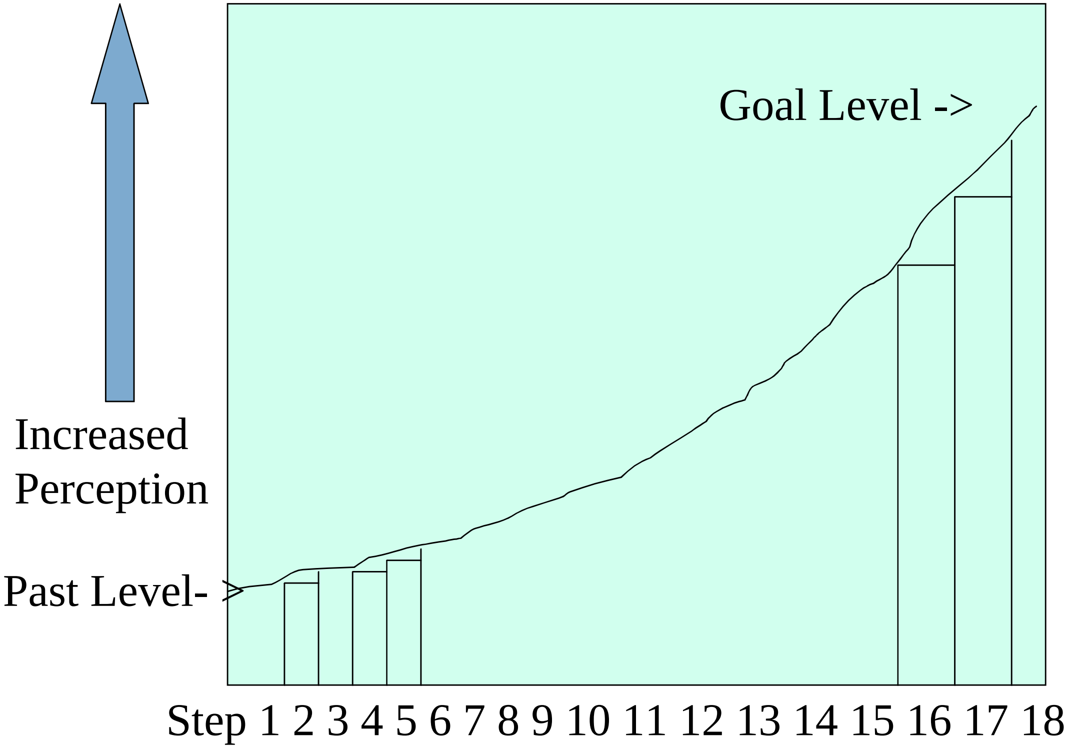
- **a 'Fail' level (a 'constraint', not a 'target') can also be set,**
  - **in each required objective's specification,**
  - **to announce a lower limit (constraint).**
  - **Fail levels define an 'acceptable' (if not yet 'successful') range of each performance and cost characteristic.**
- **Fail and Goal levels can be used to manage project decisions [CE].**

- **projects need to be evaluated on performance delivered in relation to resources used.**
- **This is a measure of project management 'efficiency'.**
- **When targets are reached,**
  - **we need to avoid misusing resources to deliver more than is required.**
  - **Perfect performance and quality costs infinite resources.**

Results are cumulated numerically step by step until the Goal level is reached.

The Naval Weapons System: Evo increase of Perception.





In a UK Radar system the system was delivered by gradually building database info about plans and ships, tuning recognition logic and tuning the radar hardware.

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.***

Projects need to be judged primarily

on their ability to meet critical performance characteristics,

in a timely and profitable way.

This cannot be expected if the project team is paid ‘by effort expended’.

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

- Teams need to be paid be results delivered in relationship to costs. By their project efficiency.
- **Even if this means that super efficient teams get terribly rich! And failure teams go ‘bankrupt’. Long live the capitalist free market mechanism!**

- **When only 13% of 1500 IT projects are ‘successful’ [Times],**
  - **we clearly need to find better mechanisms for rewarding success,**
  - **and for not rewarding failure.**
  - **I suggest that sharp numeric definition of success levels**
    - **(Goal [China, End 2005] 65%),**
    - **and consequent rewards for reaching them,**
    - **is minimum appropriate behavior for any software project.**

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

- **Conventional Agile processes believe we need to reduce unnecessarily cumbersome corporate mandated processes.**
  - **I agree.**

- **They also believe in empowering the project team to find the processes, designs and methods that really work for them locally.**
  - **I heartily agree!**

- **But I believe that**
  - **sharp numeric definition of objectives,**
  - **coupled with frequent estimation and measurement of progress,**
  - **is a clearly superior mechanism**
    - **for enabling this empowerment.**

- **The price for this,**
  - **a few estimates**
  - **and measures weekly,**
  - **seems a small price to pay**
  - **for superior control over project efficiency.**

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.

- ***No project team should be***
  - ***'stuck' with trying to satisfy unrealistic***
  - ***or conflicting stakeholder dreams***
  - ***within constrained resources.***

- ***The project team can only be charged with reasonable capability***
  - ***to deliver inside 'state of the art' performance levels***
  - ***and deliver inside 'state of the art' costs.***

References:

[POSEM] Tom Gilb: Principles of Software Engineering Management, Addison-Wesley, 1988.

[CE] Tom Gilb, Competitive Engineering: A Handbook for Systems & Software Engineering Management using Planguage, Addison-Wesley, 2004.

[Gerstner] (Louis V. Gerstner, Jr.
Retired Chairman and CEO, IBM, "Who Says Elephants Can't Dance? Inside IBM's Historic Turnaround"**)**

[WWW] www.Gilb.com

[Times] "A survey of 1,027, mainly private sector, IT projects published in the 2001 *British Computer Society Review* showed that only 130 (12.7 per cent) succeeded**." is an article online at Web address: http://www.BCS.org.UK/review/2001/html/p061.htm under the title "**IT projects sink or swim"
To see the online report in all its HTML glory, go to http://www.BCS.org.uk/review/2001/html/p061.htm

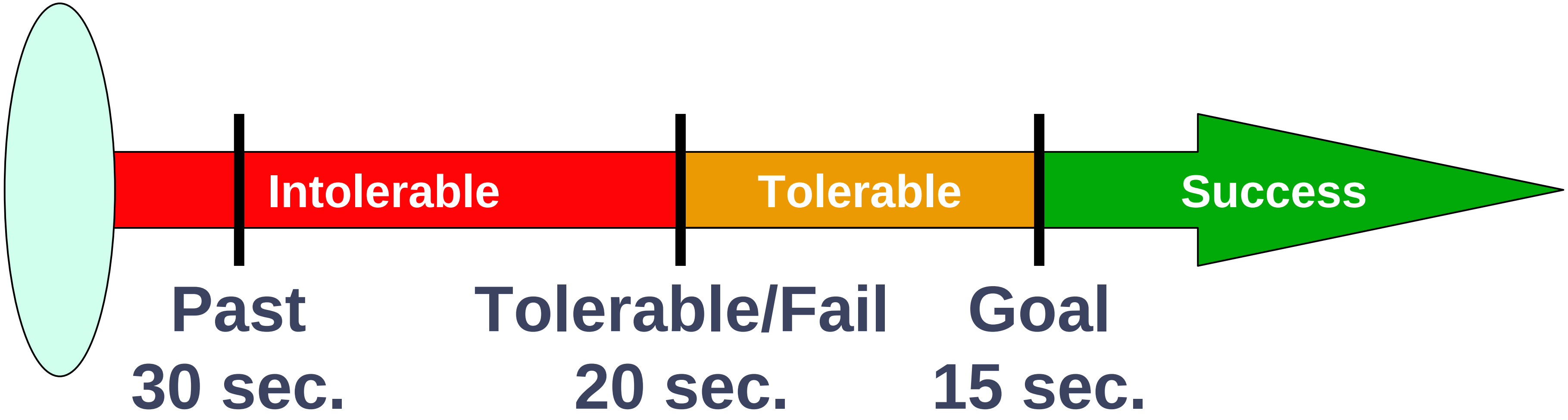
[Standish] Turning CHAOS into SUCCESS By Jim Johnson, SOFTWAREmag.com*, December 1999: http://www.softwaremag.com/archive/1999dec/Success.html.

**[Larman] Agile and Iterative Development: A Manager's Guide
Craig Larman
Addison-Wesley, 2003**

http://www.amazon.com/exec/obidos/ASIN/0131111558/qid%3D1055614131/sr%3D11-1/ref%3Dsr%5F11%5F1/104-5178070-9819101 www.craiglarman.com

[Basili & Larman] IEEE Computer paper June 2003, A History of Iterative and Incremental Development, ***Craig Larman, Chief Scientist, Valtech, USA,*** craig@craiglarman.com***. Dr. Victor R. Basili, Professor, Dept. of Computer Science, University of Maryland, basili@cs.umd.edu***

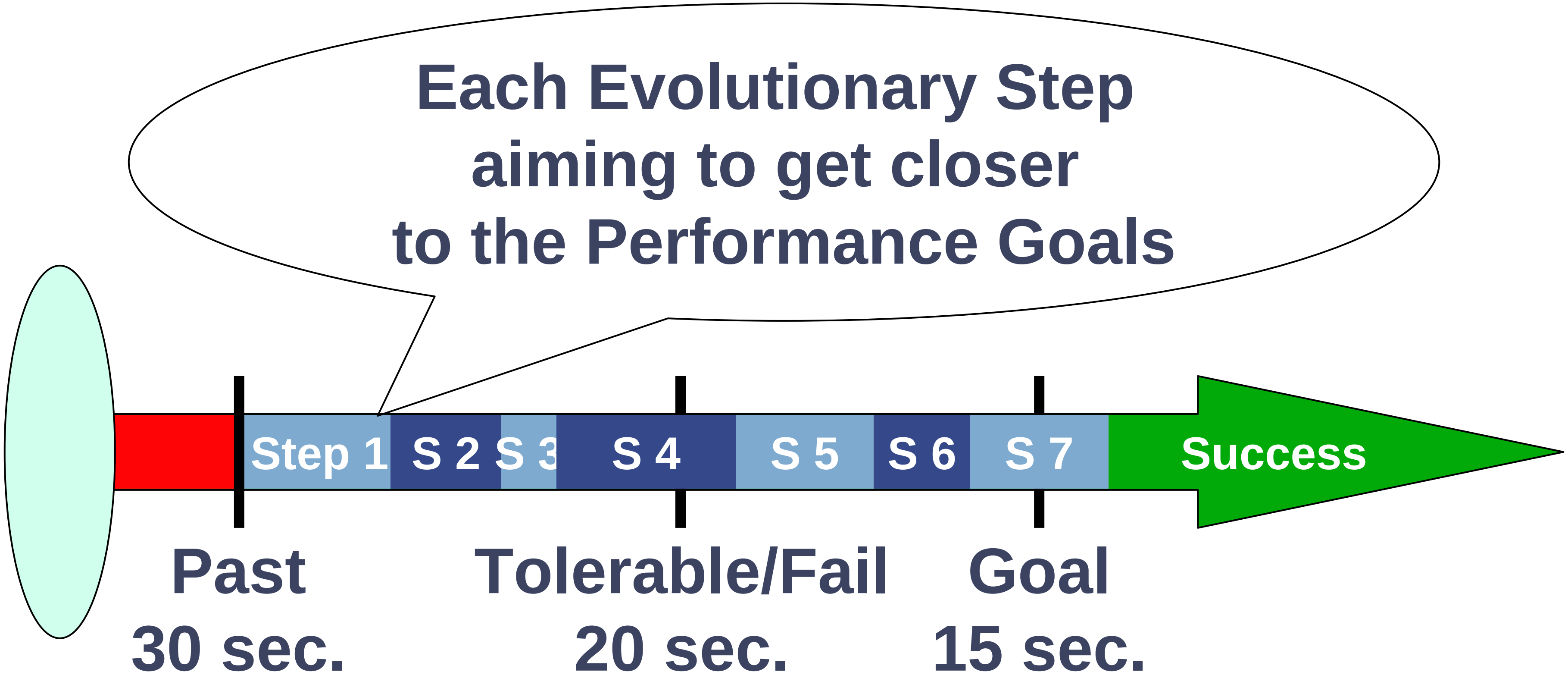
Evolutionary Delivery is driven by meeting Performance Requirements



Speed

Scale: seconds to do task

Evolutionary Delivery is driven by meeting Performance Requirements

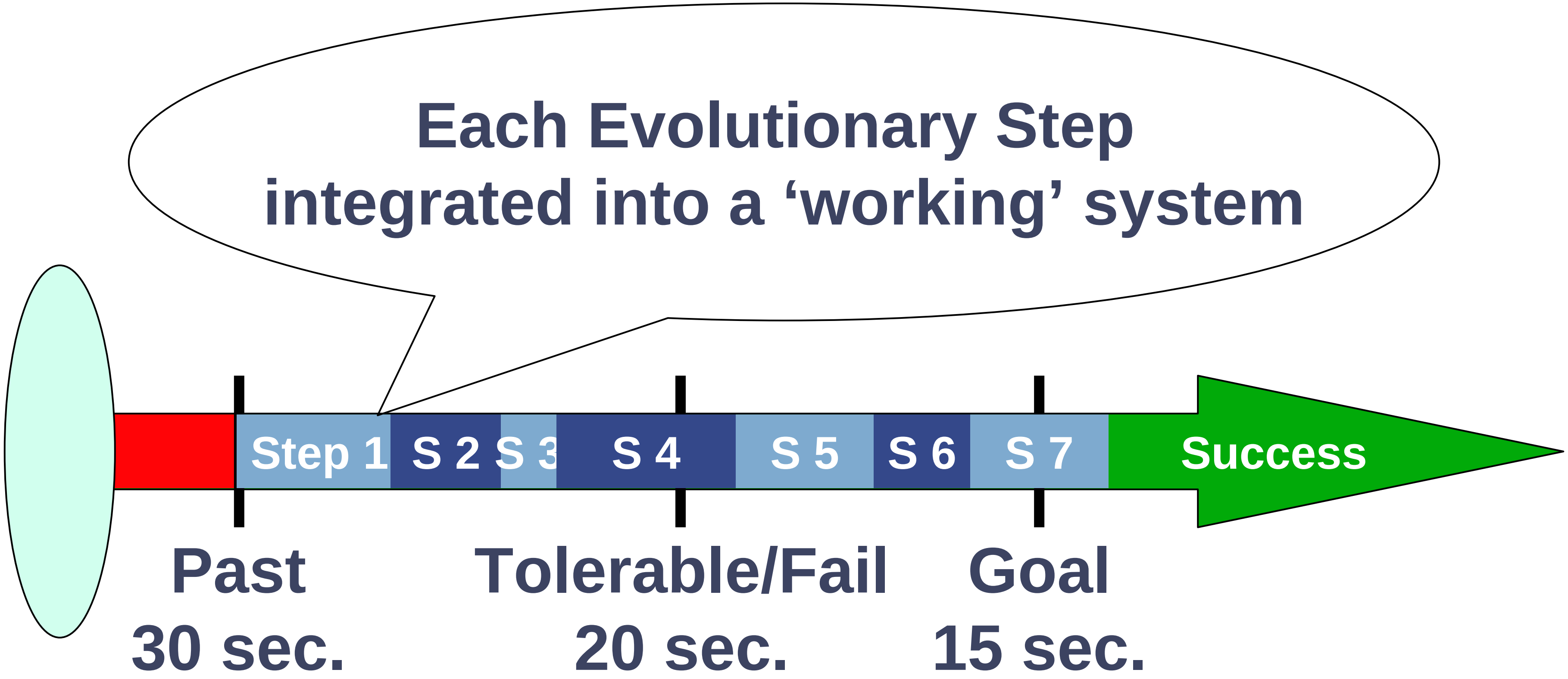


Each Evolutionary Step aiming to get closer to the Performance Goals

Speed

Scale: seconds to do task

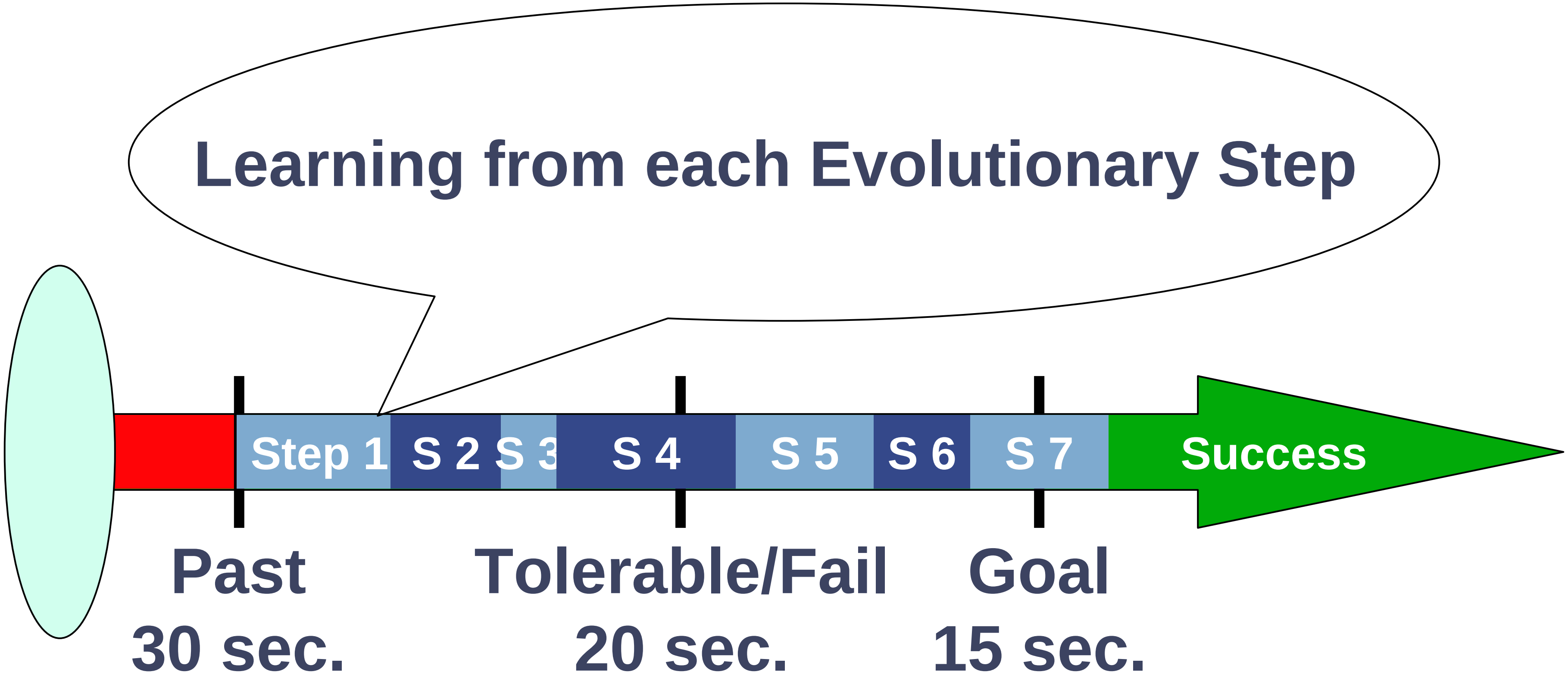
Evolutionary Delivery is driven by meeting Performance Requirements



Speed

Scale: seconds to do task

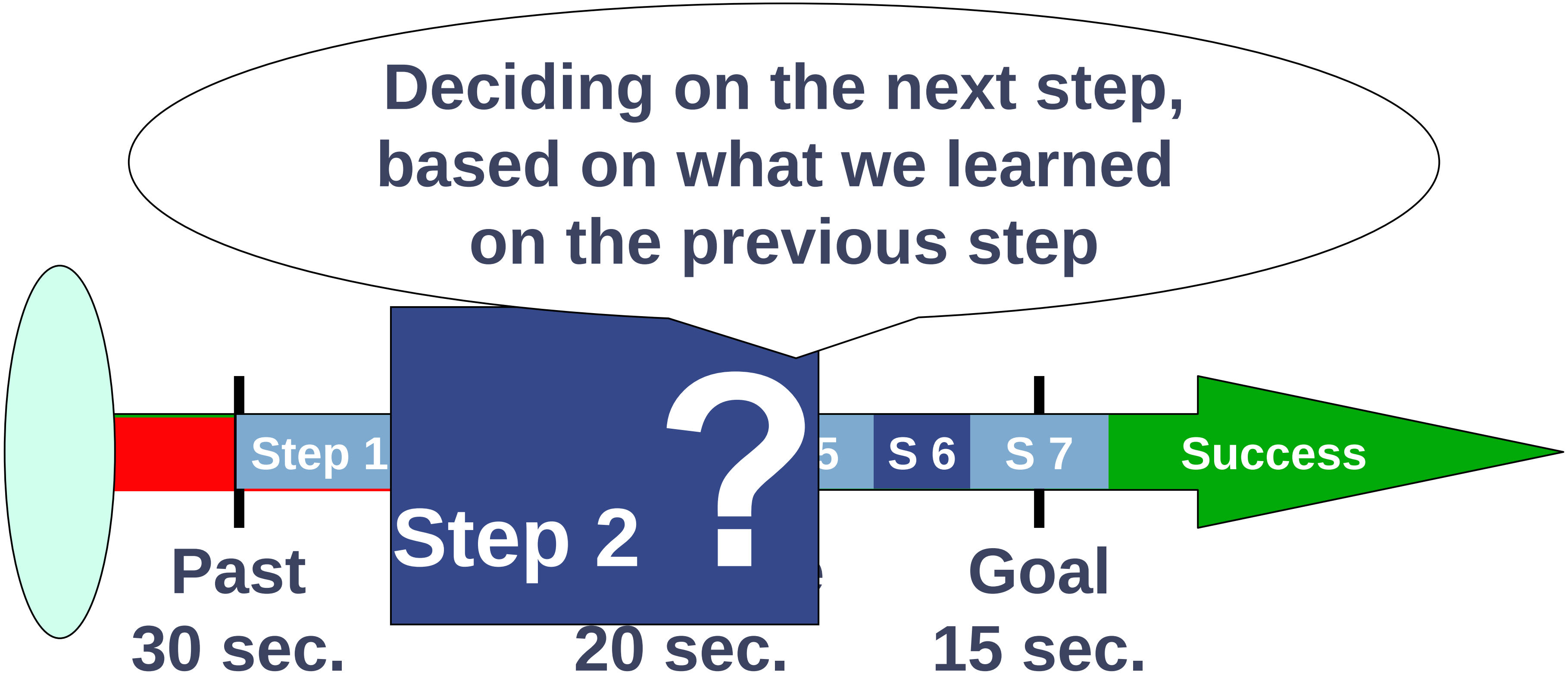
Evolutionary Delivery is driven by meeting Performance Requirements


Learning from each Evolutionary Step

Speed

Scale: seconds to do task

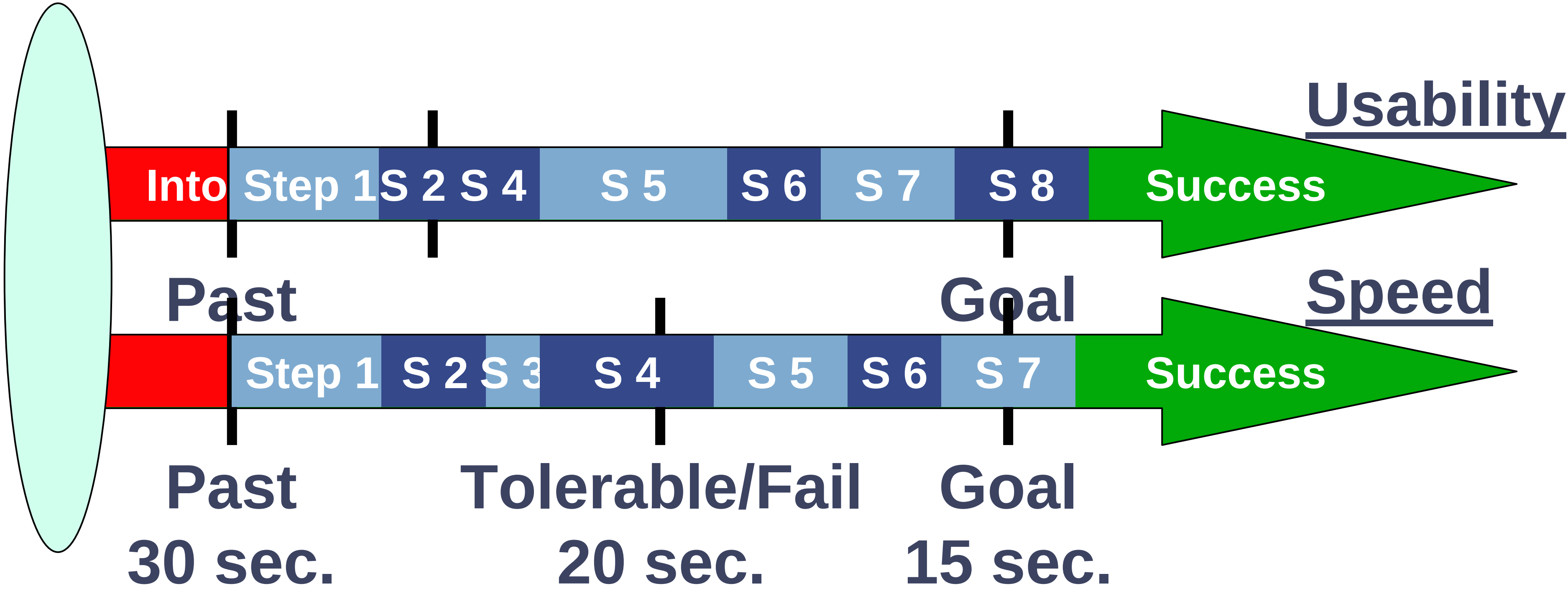
Evolutionary Delivery is driven by meeting Performance Requirements



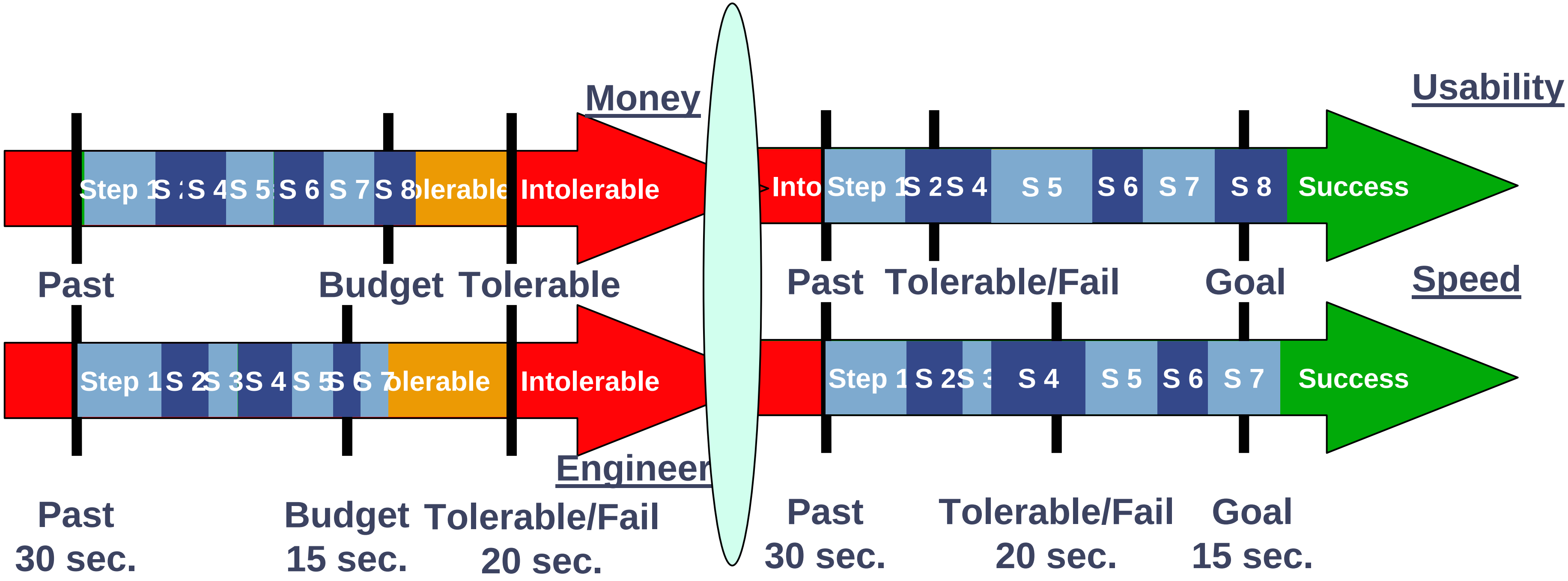
Speed

Scale: seconds to do task

Evolutionary Delivery is driven by meeting multiple Performance Requirements Simultaneously

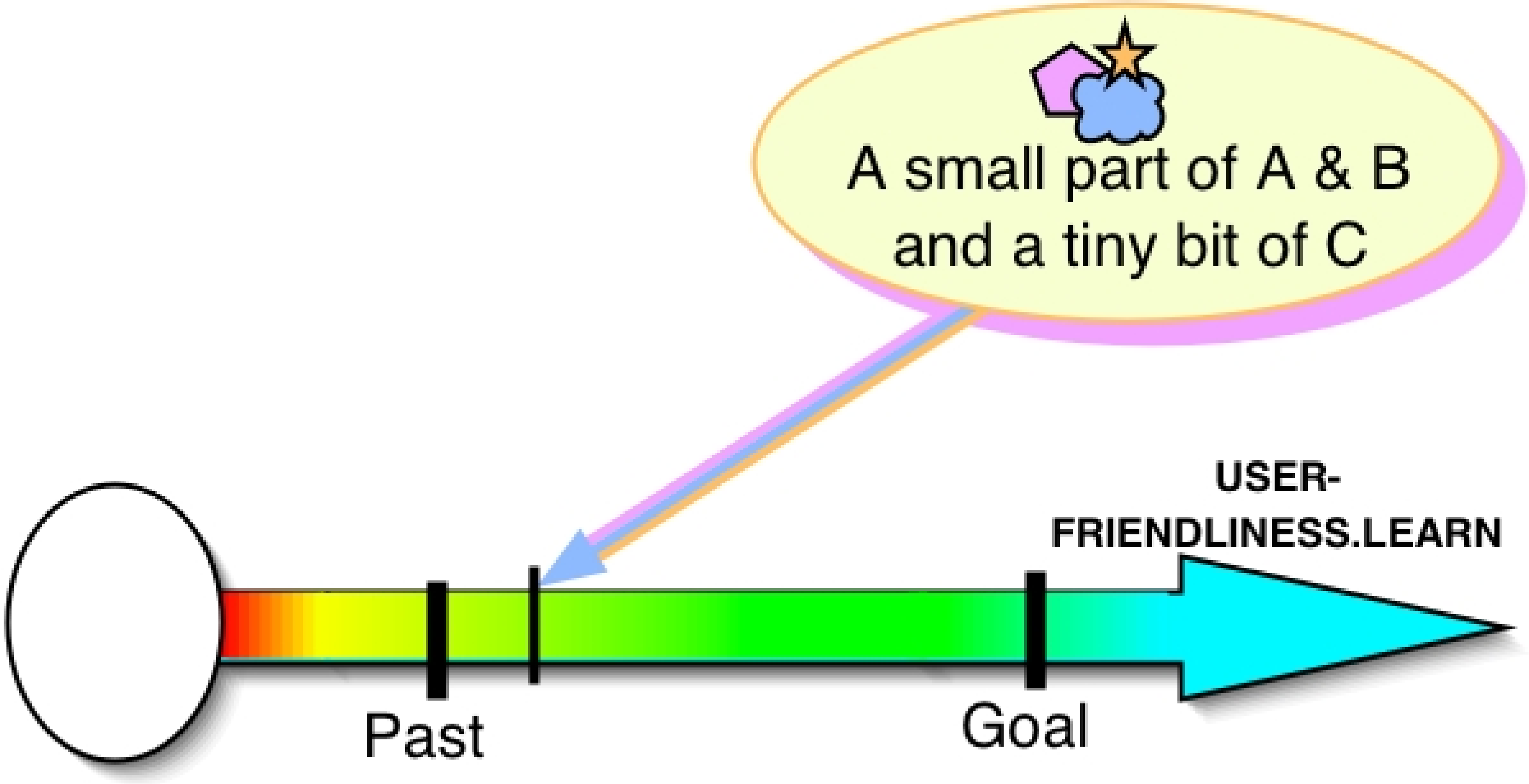


Each Evolutionary Step uses a constrained budget of Resources



Evolutionary Steps

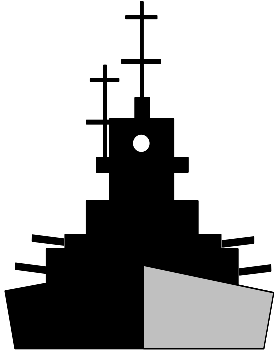
usually must contain everything that is necessary to improve towards the Goal levels.



Basic Principles of Evo Delivery

1. Any Project can be managed better using Evo control.
2. Any project can be delivered as a series of smaller steps.
3. No person knows all the results of a design, in advance.
4. No person can know what all the goals should be, in advance.
5. You must be prepared to ‘compromise intelligently’ (change requirements and design during project) with reality (Evo results).
6. Early delivery means early payback.
7. The customer is always right, even when they change their goals.
8. There is no ‘real end’ to a project, if we have competition.
9. You cannot foresee every change, but you can foresee change itself. (need open ended architecture)
10. ‘Useful results’ are your only justification for existence.
11. It is never too late to implement an Evo process!

The Naval Weapons System. Slide 1 of 7

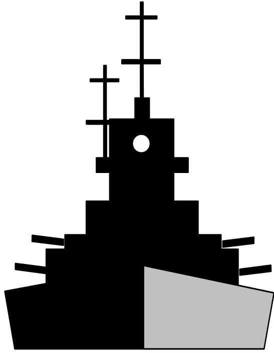


Once, when holding a public course
on the EVO method in London,
a participant came to me in the first break
and said he did not think he could use this early
Evolutionary method.

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".

The Naval Weapons System. Slide 2 of 7



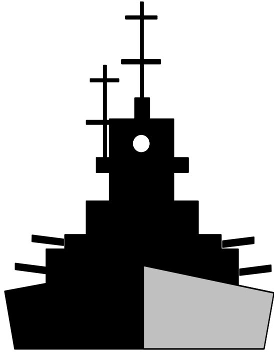
Faith:

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 Naval Weapons System. Slide 3 of 7



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.

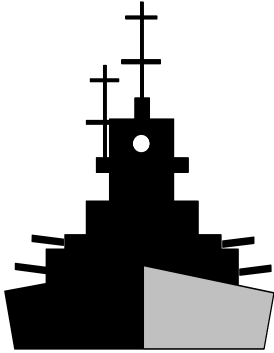
"Does your 'box' work more or less, now, in your labs?", I ventured. (Because if it did, that opened for immediate use of some kind)

"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.

The Naval Weapons System. Slide 4 of 7



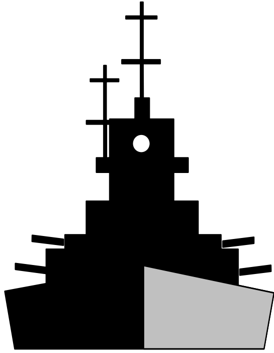
"You know, Tom", he said after five minutes of silent contemplation, "the thing that really amazes me, is that not one person at our research labs has ever dared think that thought!".

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.

The Naval Weapons System. Slide 5 of 7



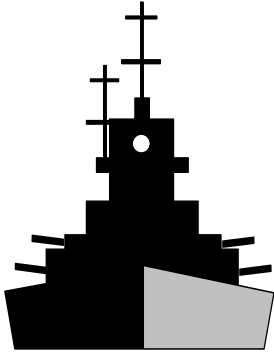
Notice the “method” emerging from this example:

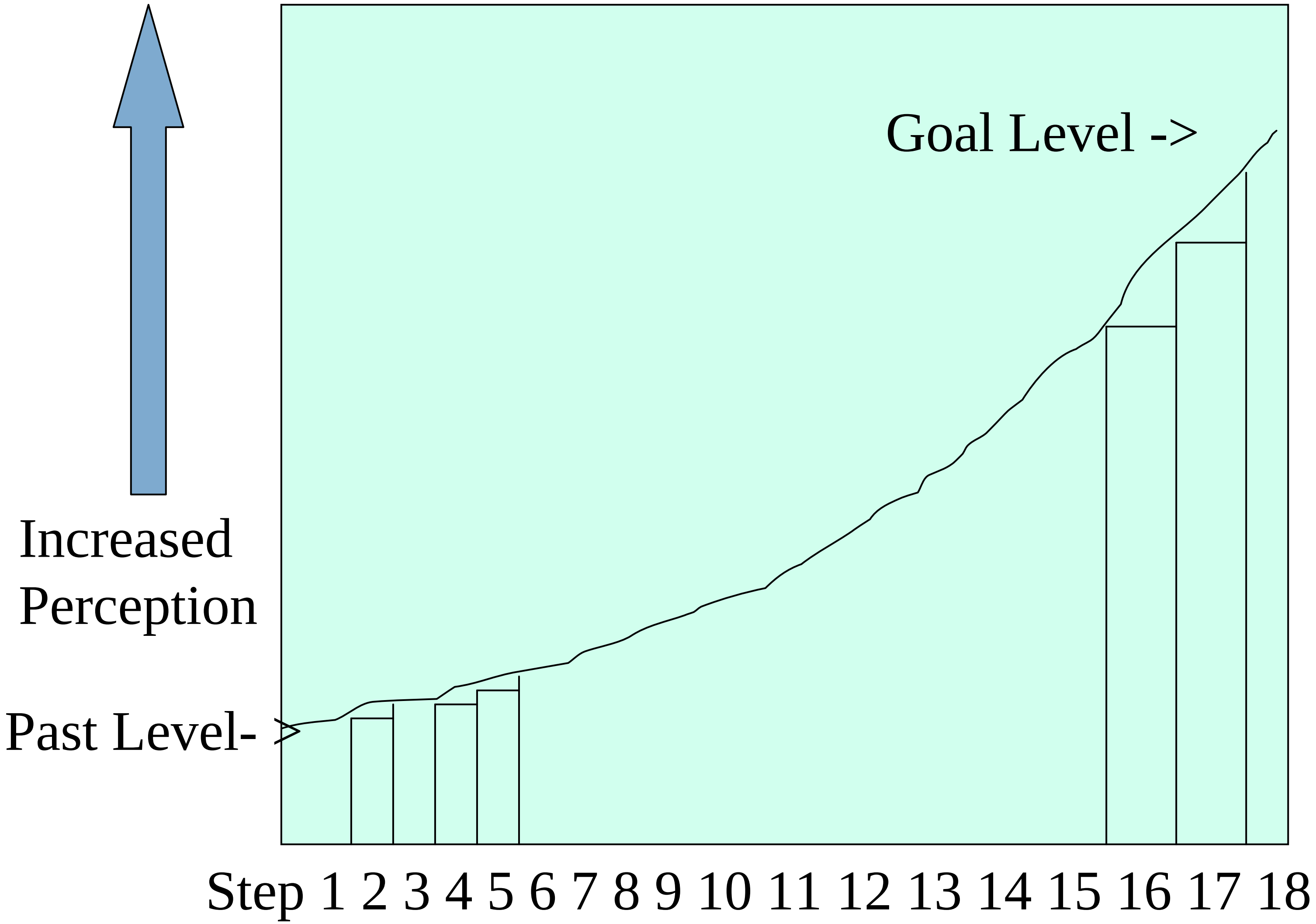
1. Identify the real customer,
   and plan to deliver results to them.
2. Identify the real improvement results
   and focus on delivering those results to the real customer.

in other words:

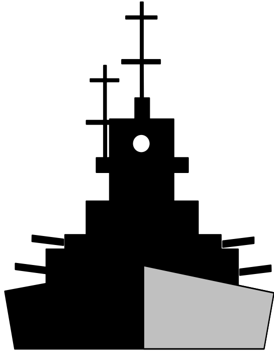
1. Do not get distracted by intermediaries (the new ship)
   think “The Royal Navy” or even “The Western Alliance”.
2. Do not get distracted by the perceived project product (the new radar device for the new ship):
   think “increased accuracy of perception”.

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





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

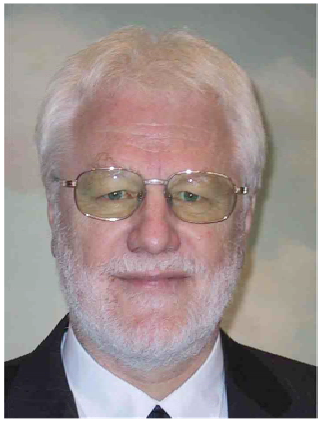


Evolutionary Projects are not normal thinking even amongst well educated engineers.

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 customer (The Royal Navy, not a ship)



Thank you

For more information,

including the book manuscripts:

Evo, Evolutionary Project Management

and,

Competitive Engineering

www.Gilb.com

Contact me at:

Tom@Gilb.com