Decomposition

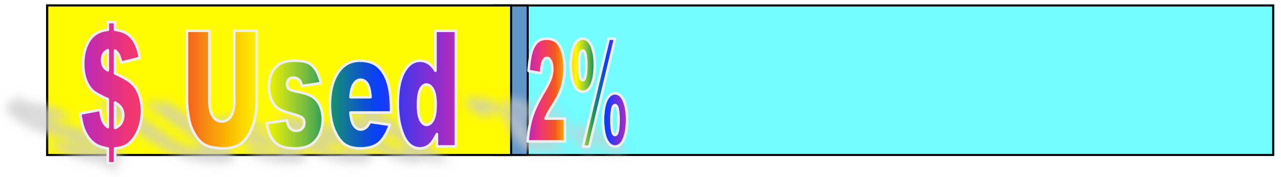
Tom@Gilb.com

http://www.gilb.com/tiki-download_file.php?fileId=350

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

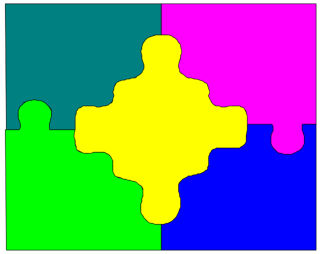
Evo Project Planning Policy



- PP1.(Budget) No Evo cycle shall exceed 2% of total budget before delivering measurable results to a ‘real’ environment.
- PP2. (Deadline) No Evo cycle will exceed 2% of total project time (=one week, for a year's projects) before it demonstrates practical measurable improvement, of the kind you targeted.
- PP3.(Priority) Evo cycles which deliver the most ‘planned value to stakeholders, for the ‘resources they claim’, shall be delivered first, to the stakeholder. Do the juicy bits first!

Why do people have difficulty finding smaller delivery steps?

- people have almost no formal experience, or teaching, on *how* to sub-divide
- they assume falsely, that we are going to sub-divide their *‘design idea’*
  - (we divide *results*)
- they don't realize we are going to sub-divide the *result delivery*
  - (increments of user pleasure)
- they have *little faith* that there is a solution, and give up easily
  - (there is always a way to divide)
- they have no *managers who insist* on an Evolutionary solution,
  - and who then ‘give their people help’ to do it.



Here are some principles of dividing up a project into smaller steps: (Solving the problem of decomposition at all)

- Divide the result/Value delivered, not the system. Go see NavY CASE.
- Get some real improvements now, don't wait for final goals.
- Ask your customer/stakeholder what they want most improved.
- Don't be afraid to use the "old system" as a temporary crutch.
- Don't hesitate to use some temporary "scaffolding" to get results now.
- Make unusual agreements with suppliers, like 'pay as you actually use'.
- Don't give up. Call it a ‘challenge’. Try other minds. There is a way.
- Invent new design ideas,
  - better for incrementing than those initially specified.
- Go to the "customer" to get permission to change constraints.
- Use the policy (2% resource cycles, biggest bang for buck first)

Practical Tip . Advice on finding smaller implementation cycles when it seems difficult or impossible to achieve. 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!
- Focus on some usefulness for the user or customer, 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 customer, tomorrow, one interesting improvement.
  - When that succeeds, multiply it.

Practical Tip . Advice on finding smaller implementation cycles when it seems difficult or impossible to achieve. PART 2 of 4

- Focus on the results (which you should have defined in your goals, moving toward PLAN 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 customer 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!

Practical Tip . Advice on finding smaller implementation cycles when it seems difficult or impossible to achieve. PART 3 of 4

- If you focus on helping your customer in practice, now, where they really need it, you will be forgiven a lot of "sins"!
- You can understand things much better, by getting some practical experience (and removing some of your fears).
- 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.

Practical Tip . Advice on finding smaller implementation cycles when it seems difficult or impossible to achieve. 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 "customer" 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.

to learn as much as possible

- Pick high ‘unknown’ areas first
  - New markets
  - New stakeholders
  - New technology
  - New combinations of technology
  - Areas which are critical for your performance 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

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 ± uncertainty (> ± 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 factors before the high value factors.

To deliver value to stakeholders

- Decompose and schedule high value to cost steps first
  - Recognize the Internal Stakeholders are usually opportunity for early value at low risk
- Use Impact Estimation tables to compare value alternatives
- Use Evo step templates which force you to estimate value and cost.
  - See next 2 slides
    - General template with hints
    - Real example
- Have a clear Evo planning policy:
  - 'Evo steps will, after high risks are tested,
    - primarily select and deliver steps
    - based on overall numeric value delivery to performance requirements
    - in relation to overall cost in terms of resource budgets.

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 step is actually implemented.

Quick Action, Quick Correction

- "I think it is very important for you to do two things:
  - Act on your temporary conviction as if it was a real conviction;
  - and if you realize that you are wrong, correct your course very quickly."
  - Source: Andy Grove, co-founder and chairman of Intel Corporation. Born 1936, Hungary

Value, 'Sometime' -> High Value Early (every month from first month)

We have to move from a situation where project value is

'alluded to' vaguely *(next slide for examples in our project)* (TO)

defining our project values in top-level, critical, quantified statements

(drafts for this project are available already, if this seems strange).

Quantified Goals, deadline driven, with named responsible sponsor and Business Owners.

We need to tear away the juicy bits,

the high value changes,

from the infinite mass of 'nice to have sometime',

and DELIVER THEM to the organization.

We need to make sure, motivate, reward, and punish, to make sure that the intended value is actually firmly in place

(the head count is really reduced, the new business has really arrived - no excuses) before the 'job is considered done'.

Before anybody is paid a bonus for the change

thanks Mark R, for being so clear on this idea in your interview 3-oct-07!

Advantages of an Evo Result Delivery Approach to Projects

- You can dive in very early, and deliver some real results
  - value is delivered to the business
  - Credibility of IT team is raised
  - Motivation to go with your project is raised
  - Communication with users/Business is much better both ways (what can be done, what we need)
- You choke off inefficient projects - No Cure No Funding
  - they cannot deliver
  - or cannot do so profitably,
    - even the smallest practical thing
- You avoid throwing good money after bad projects
  - Real time reprioritization of projects, During a project
- You are not dependent on theory and estimates and flaky ideas
  - Everything is rapidly tested on the battlefield of business reality

Evo Wisdom

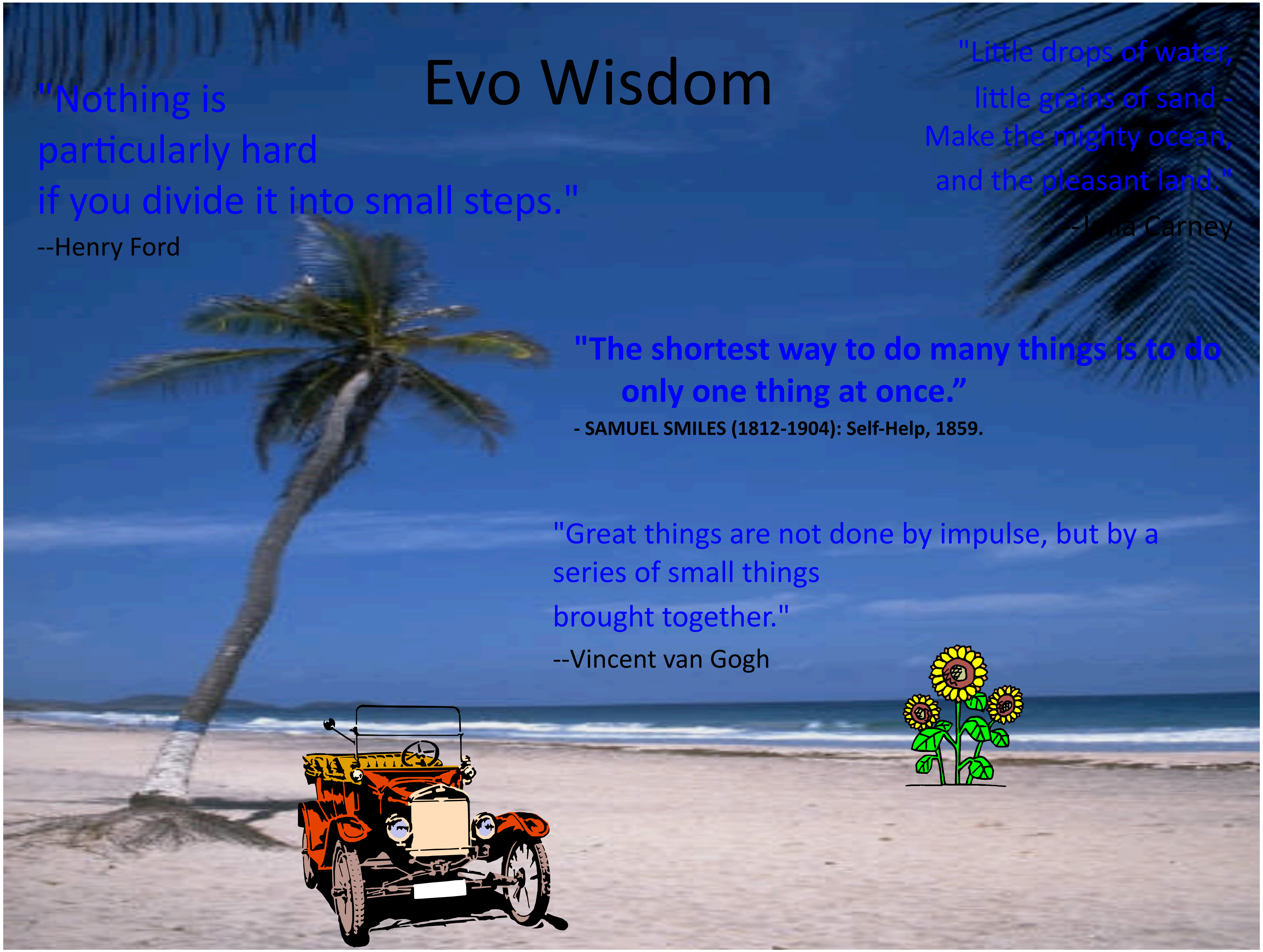
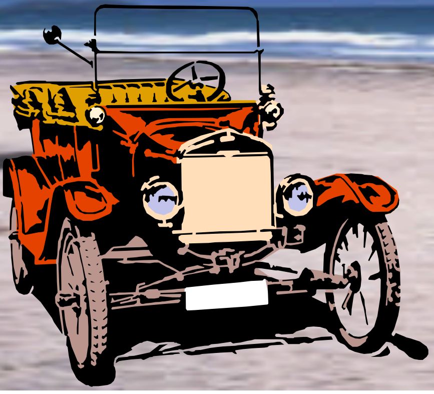
"Nothing is
particularly hard
if you divide it into small steps."
--Henry Ford

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

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

- SAMUEL SMILES (1812-1904): Self-Help, 1859.

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

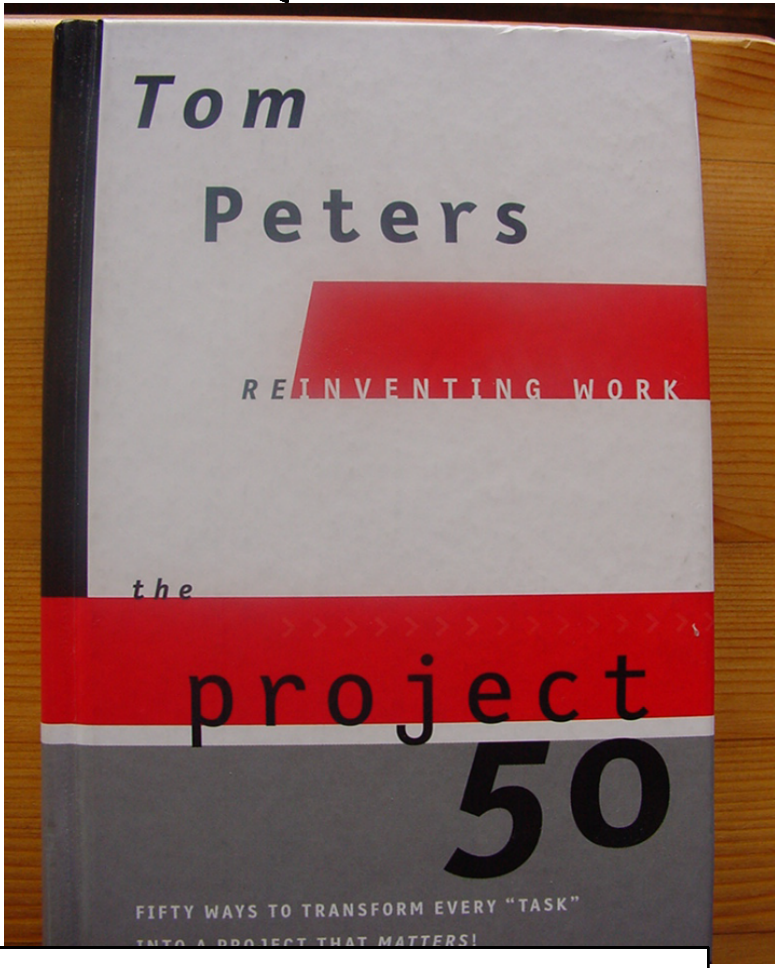



Petroski: Form Follows Failure

As the engineering historian Henry Petroski suggested in his 1992 book The Evolution of Useful Things,

- continual refinement is the usual rule in technology.
- Engineers constantly notice shortcomings in their designs and fix them little by little, (a process Petroski wryly described as "form follows failure.")
- As a result, products incrementally improve.

Quick Prototyping á la Peters


Tom
Peters
REINVENTING WORK
the
project
50
FIFTY WAYS TO TRANSFORM EVERY "TASK"
INTO A PROJECT THAT MATTERS!

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

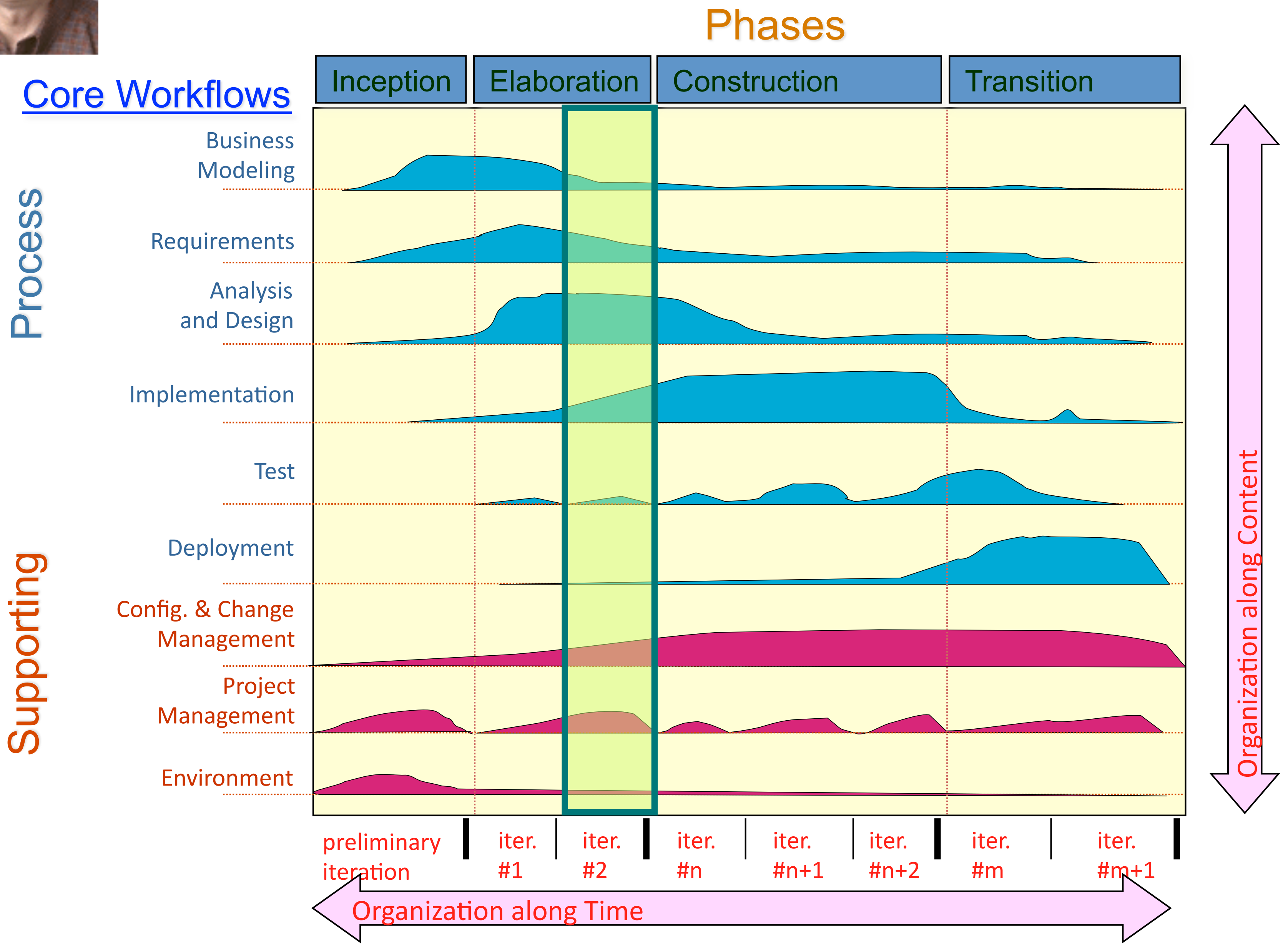
Tom Peters

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.

The Unified Software Development Process/ RUP SE
I just wanted to show that RUP Does try to do Evo !



<u>Source:</u> *P. Kruchten, The Rational Unified Process, Addison-Wesley, Reading, MA, 1998.*

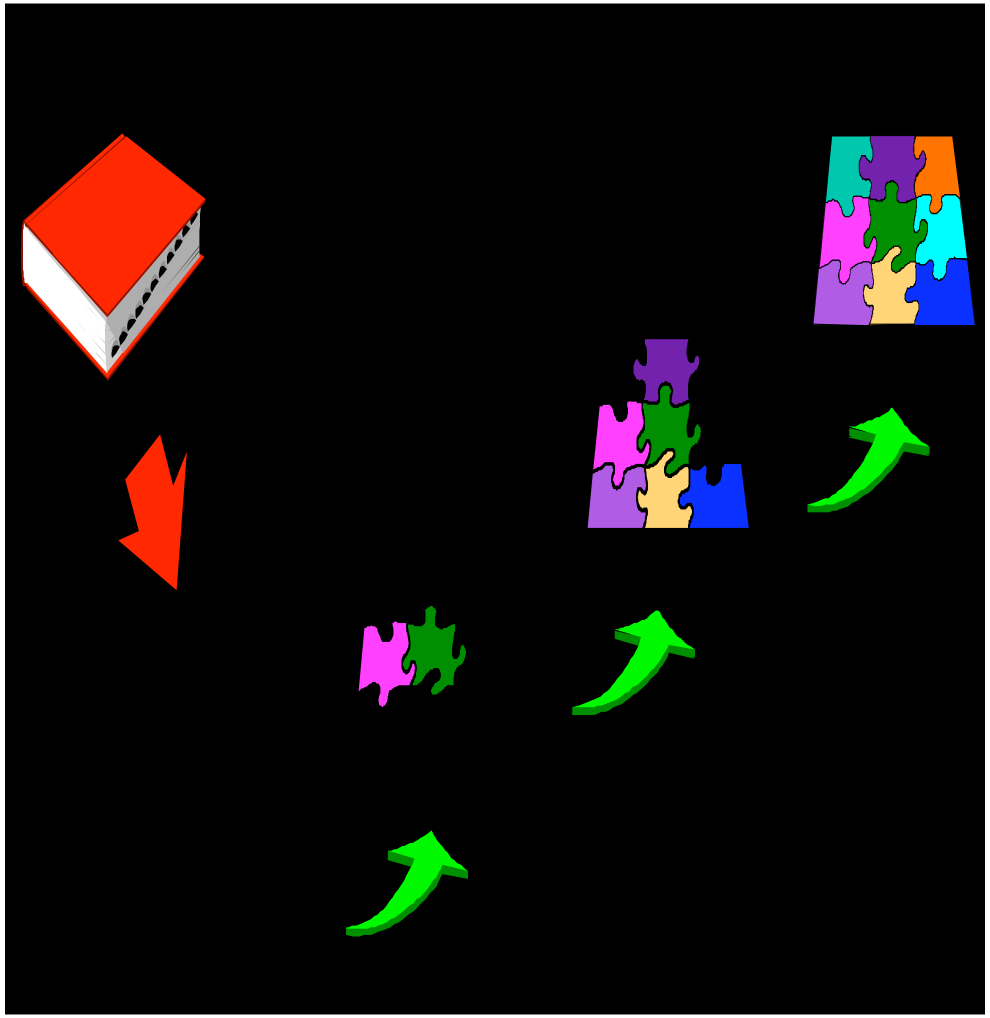
The Good: Benefits of Iterative Development <- Philippe Kruchten, Rational

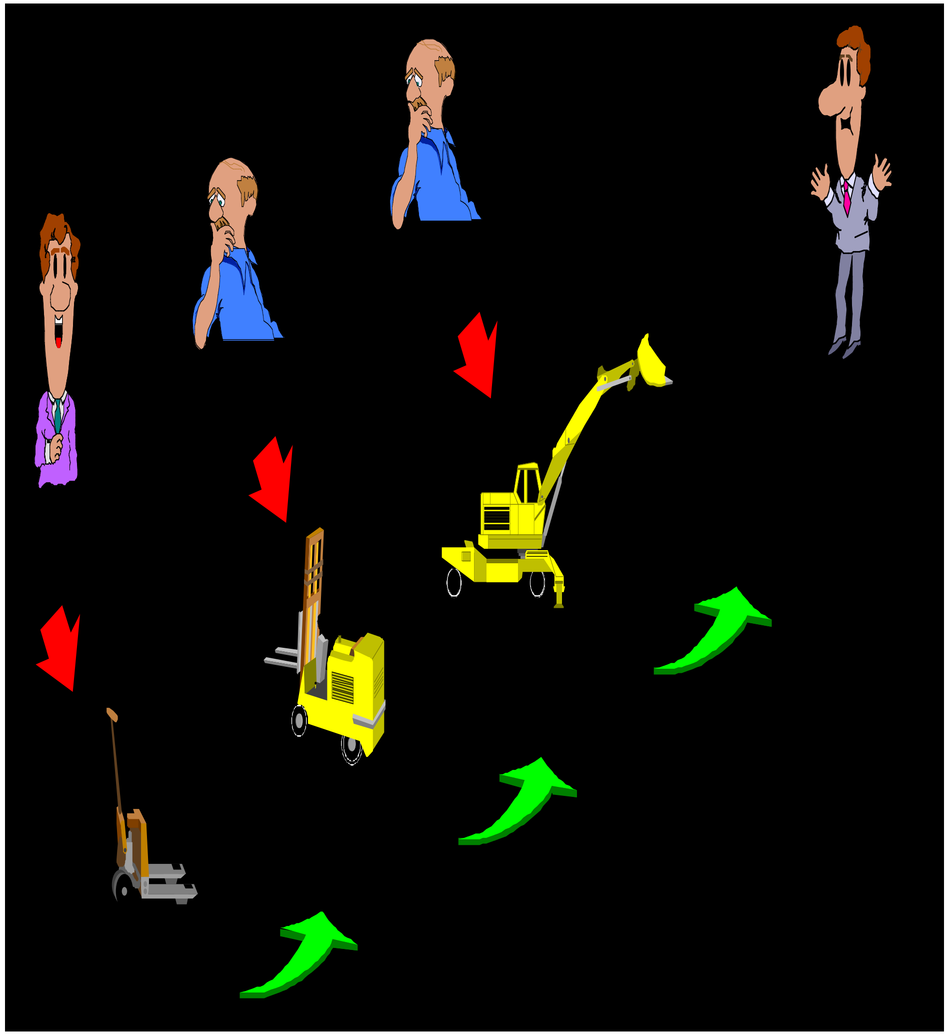
Compared with the traditional waterfall process, the iterative process has many advantages.

1. Serious misunderstandings are made evident early in the lifecycle, when it's possible to react to them.
2. It enables and encourages user feedback, so as to elicit the system's real requirements.
3. The development team is forced to focus on those issues that are most critical to the project, and team members are shielded from those issues that distract them from the project's real risks.
4. Continuous, iterative testing enables an objective assessment of the project's status.
5. Inconsistencies among requirements, designs, and implementations are detected early.
6. The workload of the team, especially the testing team, is spread out more evenly throughout the lifecycle.
7. This approach enables the team to leverage lessons learned, and therefore to continuously improve the process.
8. Stakeholders in the project can be given concrete evidence of the project's status throughout the lifecycle.

How Does This Relate to Scrum?

Incremental and Evolutionary





- Source: A Strategy for Acquiring Large and Complex Systems
  Dr. Helmut Hummel, Bonn September 23 2002, see note for paper
- Email: hummel@iabg.de

How does Evo differ from Incremental?

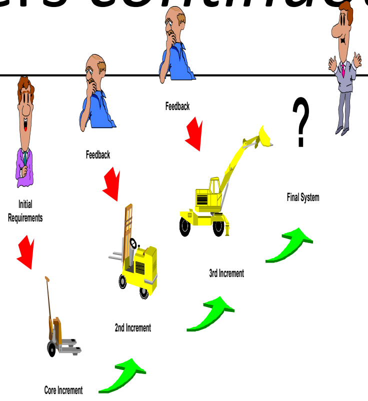
Evo

Focus on *business value*

Ability to *learn* rapidly

Quantified value tracking

Cooperation with users *continuous*



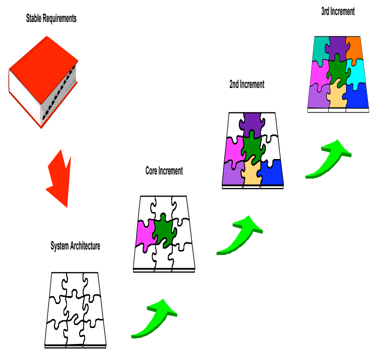
Incremental

Focus on *construction*

No intent to learn or change plans

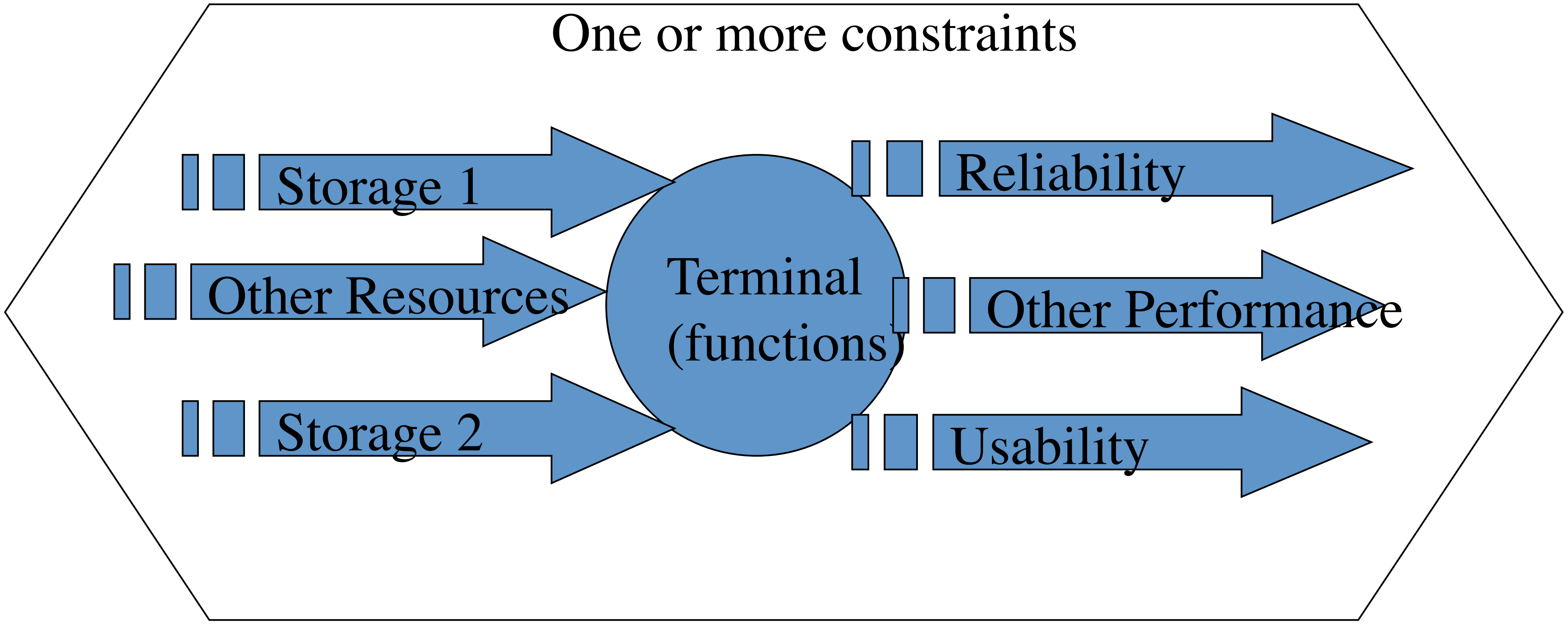
No value tracking

No plan to cooperate with users



Evo and Requirements, Conceptually

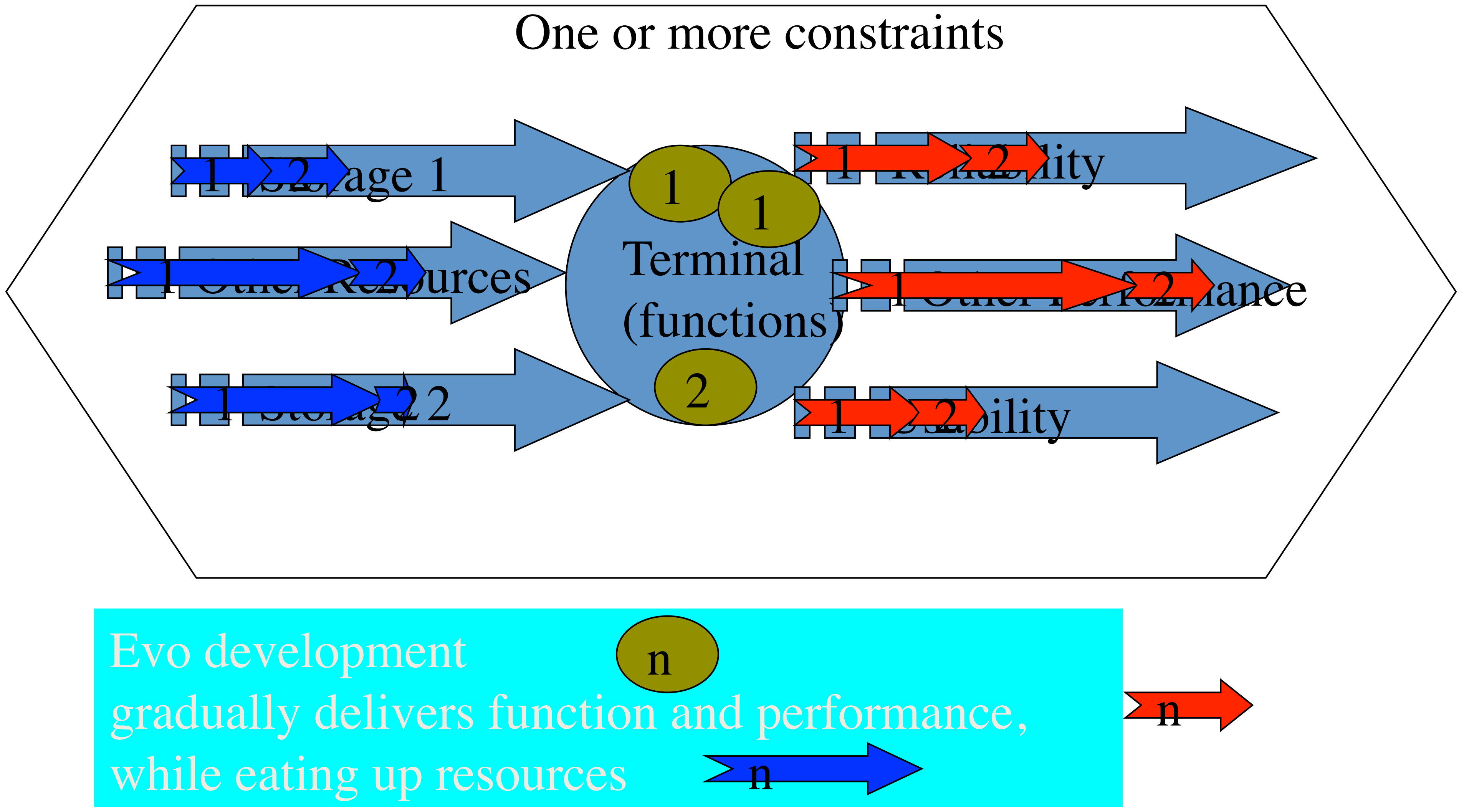
Requirements are the framework for Evo development



Basic requirements model:
We need to meet performance and function requirements,
Within available/planned resources and within constraints.

Evo and Requirements, Conceptually

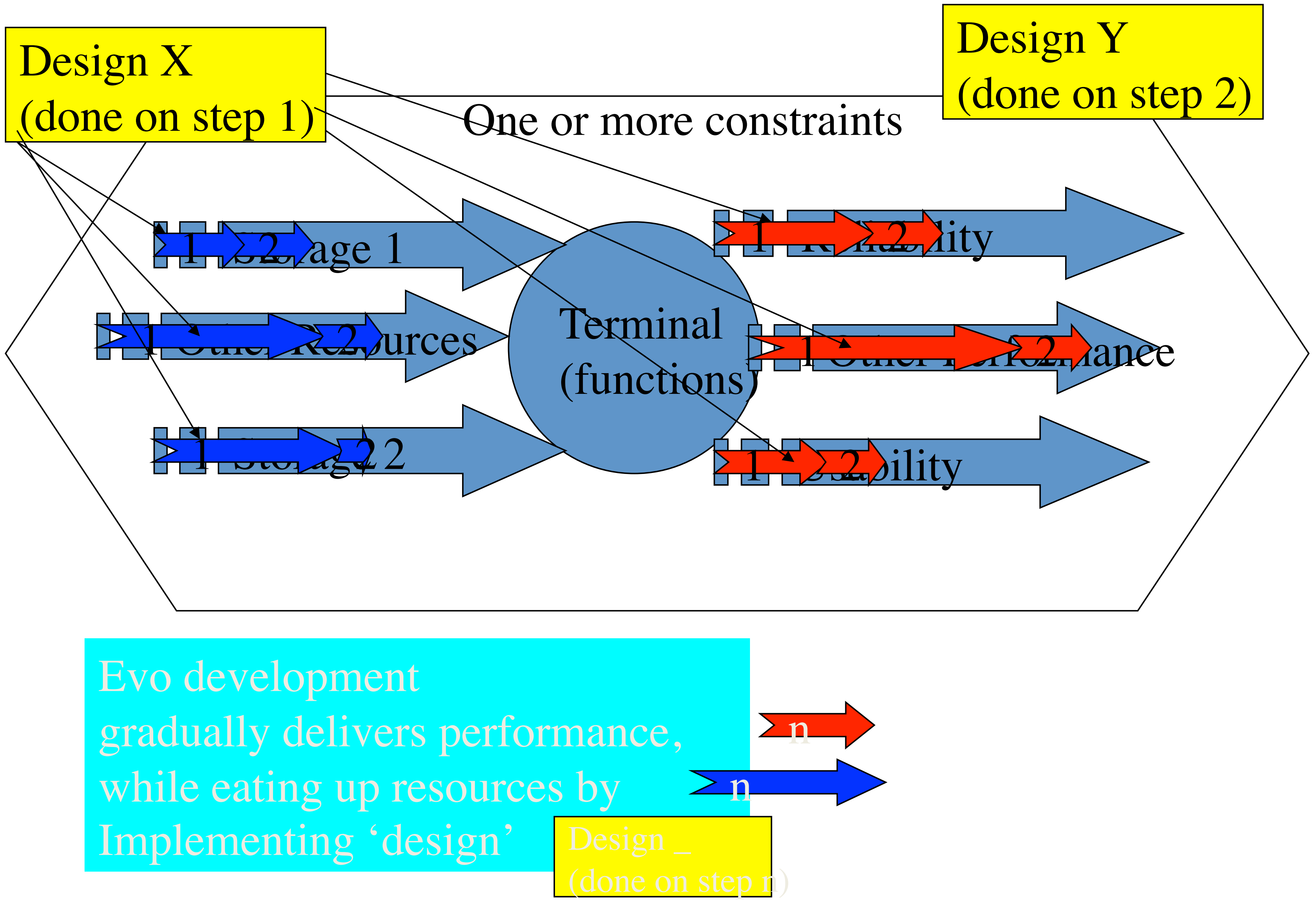
Evo steps deliver partial requirements



Evo development
gradually delivers function and performance,
while eating up resources

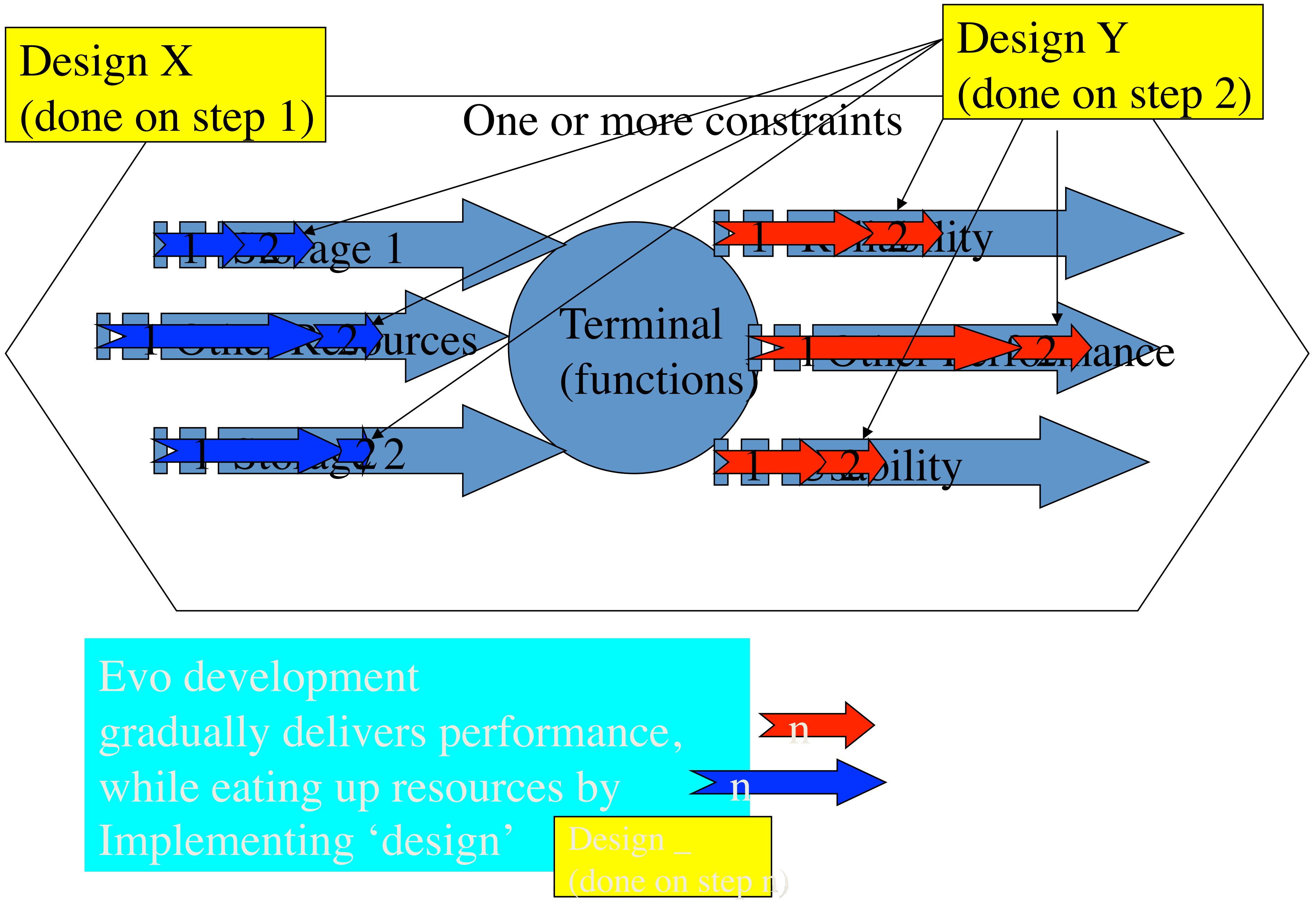
Evo and Requirements, Conceptually

‘Design’ is what delivers performance, and costs resource



Evo and Requirements, Conceptually

'Design' is what delivers performance, and costs resource



Simplified Evo Process: Implement Evo Steps (Process)

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 four 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. O*n a single page,* summarize the *progress to date* towards achieving the goals and the costs incurred.

Evo seen as hierarchical steps

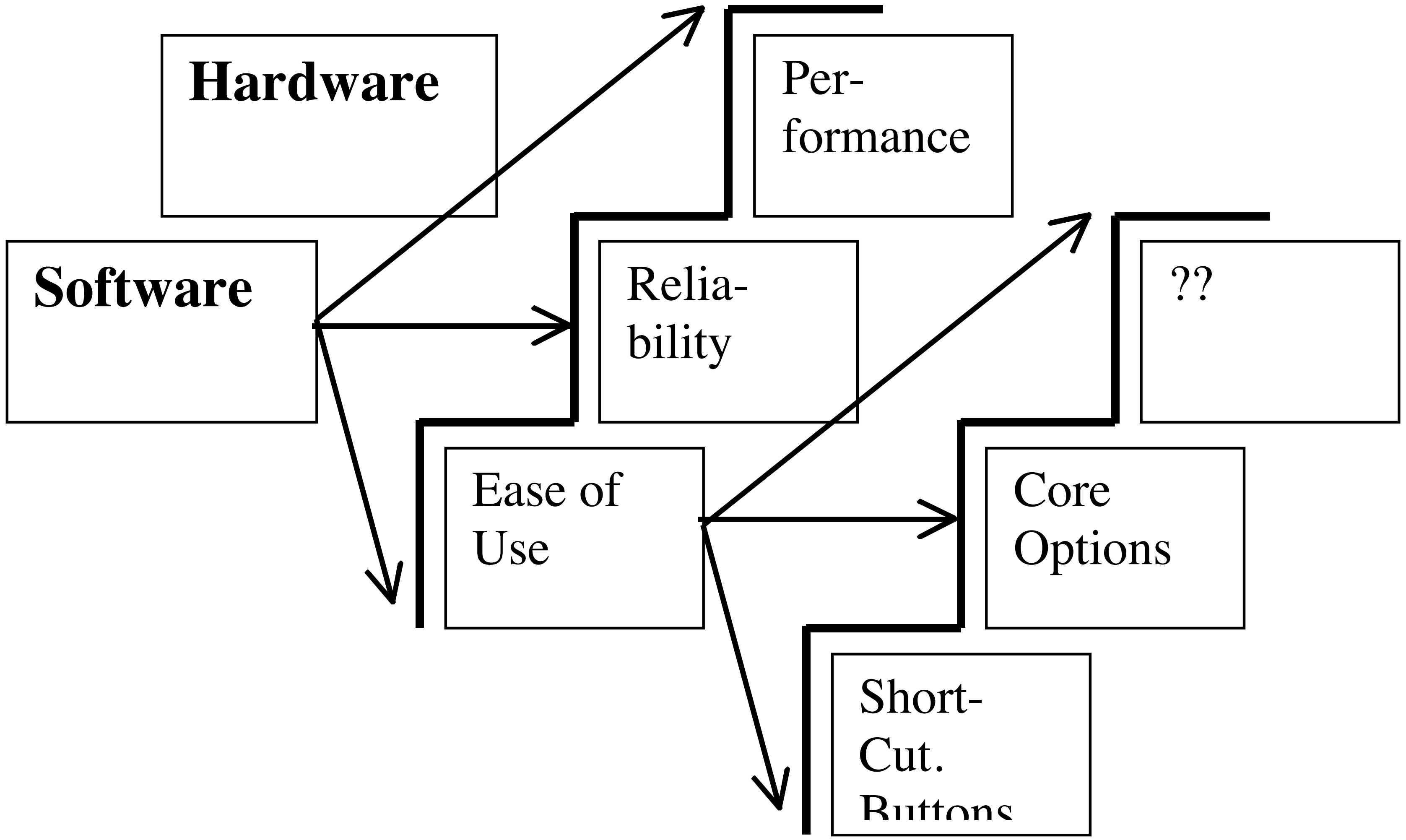


Diagram courtesy of Kai Thomas Gilb, from his manuscript 'Whirlwind', @ www.result-planning.com December 4th 2001

Basic Principles of Evo Delivery

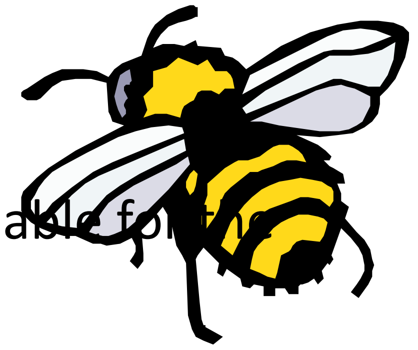
1. Any Project can be managed better using Evo control.
2. Any project can be delivered as a series of smaller steps.
3. No person knows all the results of a design, in advance.
4. No person can know what all the goals should be, in advance.
5. You must be prepared to ‘compromise intelligently’ (change 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!

Tao Te Ching (500BC)

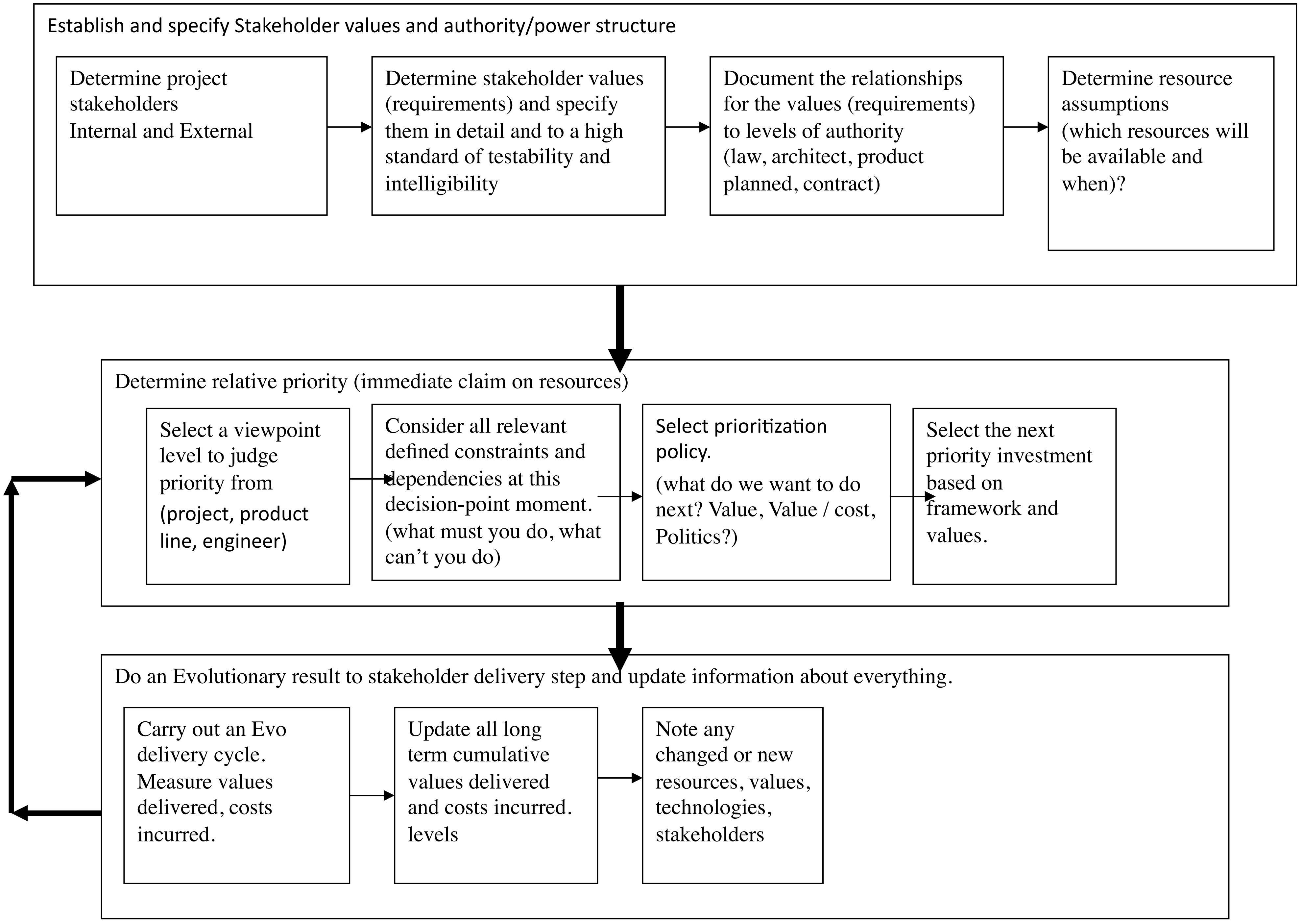
- That which remains quiet, is easy to handle.
- That which is not yet developed is easy to manage.
- That which is weak is easy to control.
- That which is still small is easy to direct.
- Deal with little troubles before they become big.
- Attend to little problems before they get out of hand.
- For 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 (also quoted in Gilb, Principles of Software Engineering Management page 96), Penguin book

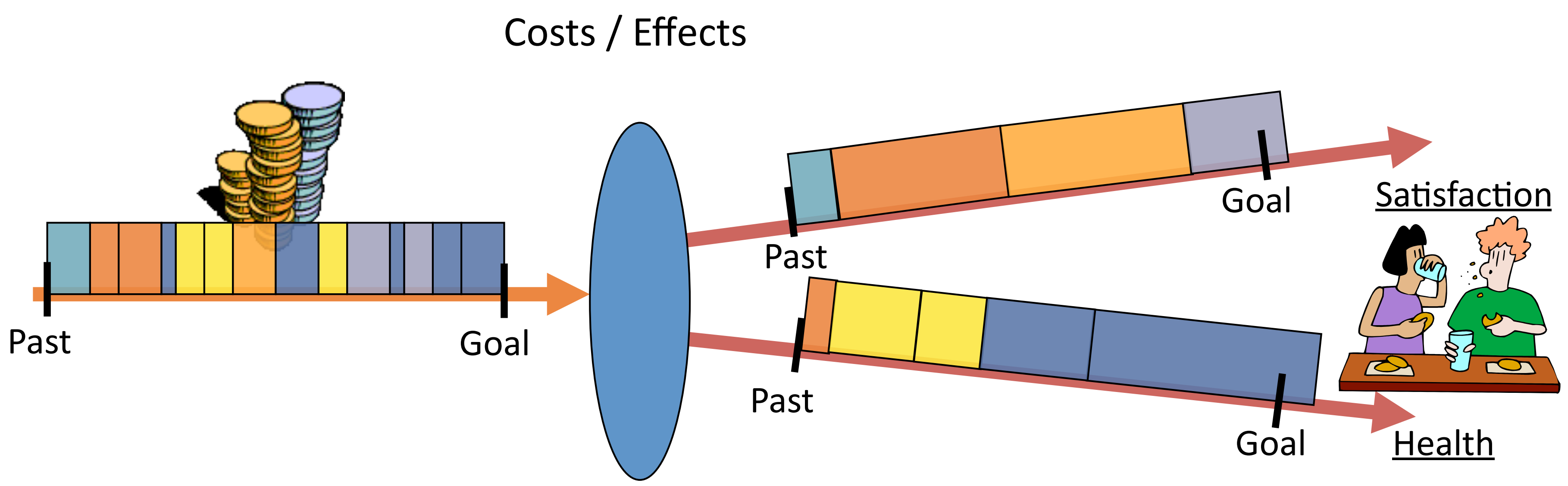
Evo as a ‘Process Improvement’ tactic?

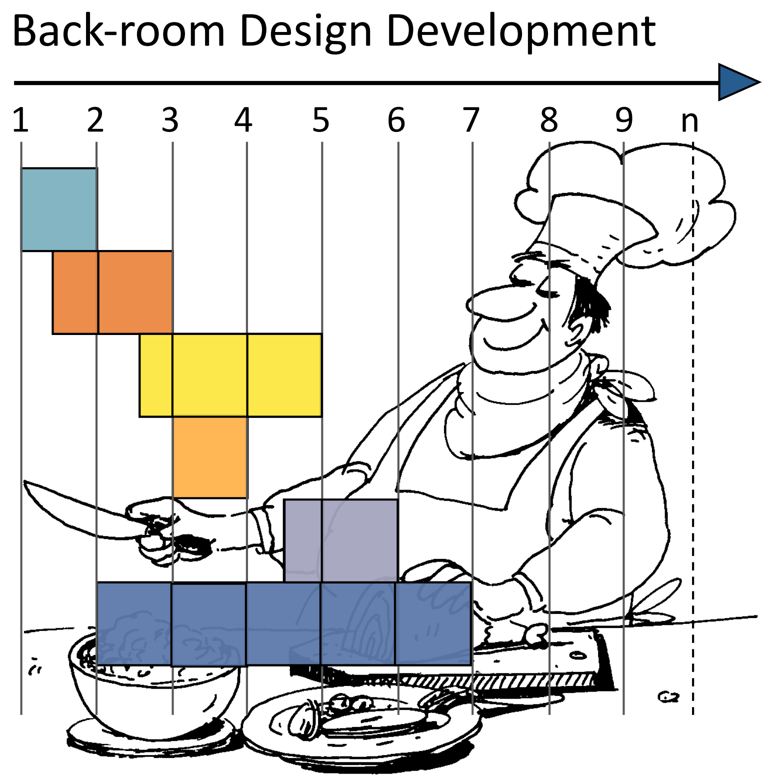
- It can be hard to get co-operation to improve engineering processes ‘in general’
- People are so busy meeting their project deadlines!
- Evo can be used for process improvement management within a project
- People have time for that
  - because it immediately benefits them
- Successful project improvements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

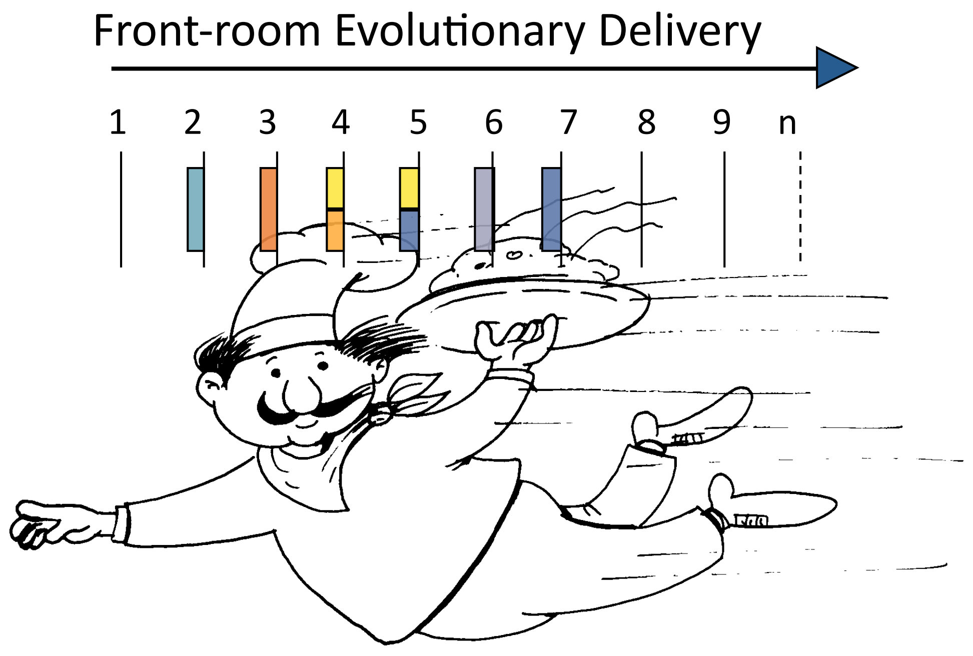


Priority Determination Process: in Evo

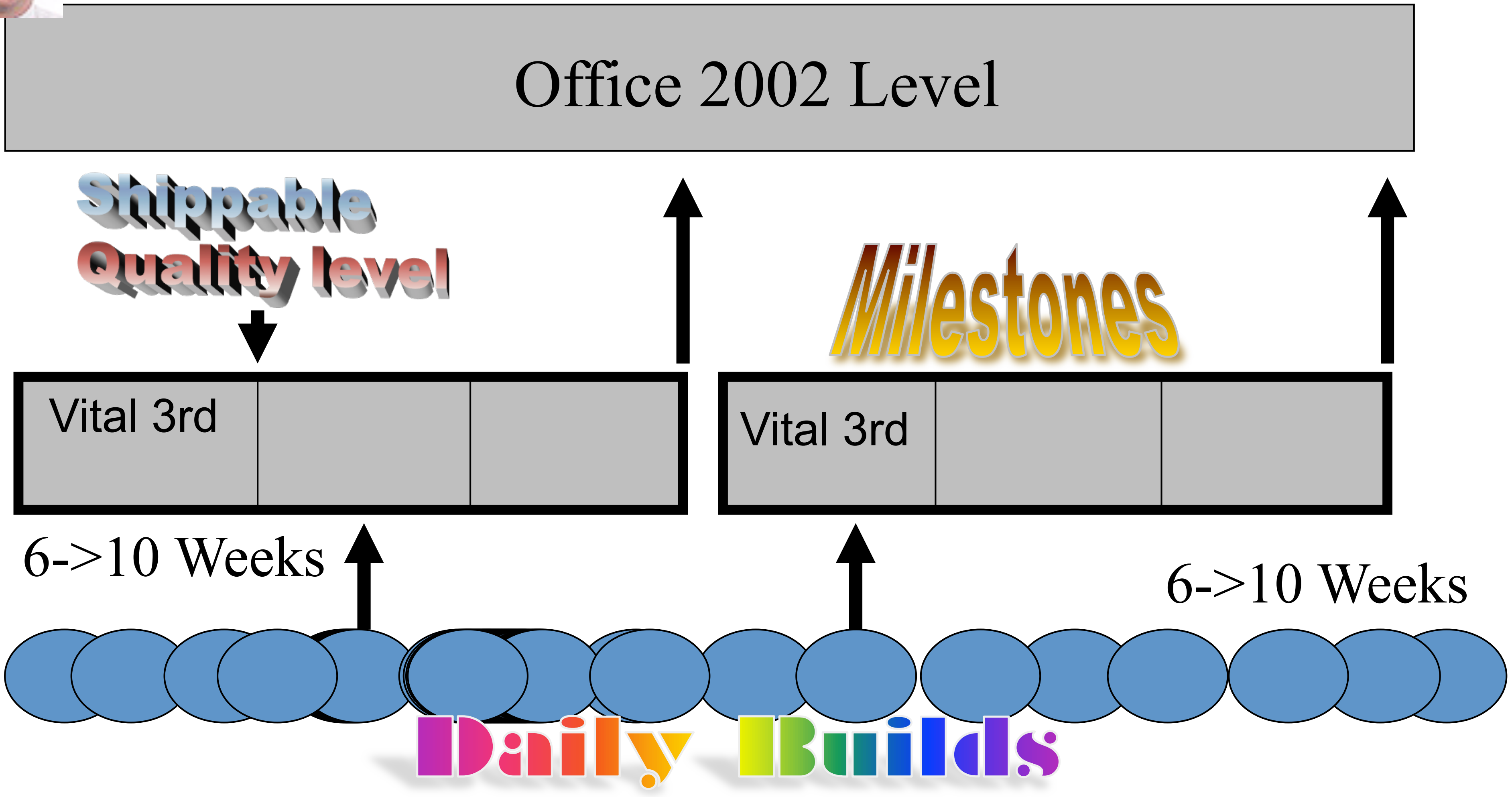

Establish and specify Stakeholder values and authority/power structure
Determine project stakeholders Internal and External
Determine stakeholder values (requirements) and specify them in detail and to a high standard of testability and intelligibility
Document the relationships for the values (requirements) to levels of authority (law, architect, product planned, contract)
Determine resource assumptions (which resources will be available and when)?
Determine relative priority (immediate claim on resources)
Select a viewpoint level to judge priority from (project, product line, engineer)
Consider all relevant defined constraints and dependencies at this decision-point moment. (what must you do, what can't you do)
Select prioritization policy. (what do we want to do next? Value, Value / cost, Politics?)
Select the next priority investment based on framework and values.
Do an Evolutionary result to stakeholder delivery step and update information about everything.
Carry out an Evo delivery cycle. Measure values delivered, costs incurred.
Update all long term cumulative values delivered and costs incurred. levels
Note any changed or new resources, values, technologies, stakeholders


Costs / Effects
Past
Goal
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

Multiple Test Levels of Microsoft Evo



Reference: Cusomano: Microsoft Secrets. *Drawing by TG*

See reference [MacCormack2001, Cusomano 2003] in PowerPoint notes for this slide.

SWI

Cycle Length: short! (HP)

- The general rule of thumb is to keep the cycle length as short as possible. Within Hewlett-Packard, projects have used a cycle length as short as one week and as long as four weeks. The typical cycle time is two weeks.
- The primary factor in determining the cycle length is how often management wants insight into the project’s progress and how often they want the opportunity to adjust the project plan, product, and process.
- Since it is more likely that a team will lengthen their cycle time than shorten it, it is best to start with as short a cycle as possible.
- [COTTON96]

hp invent

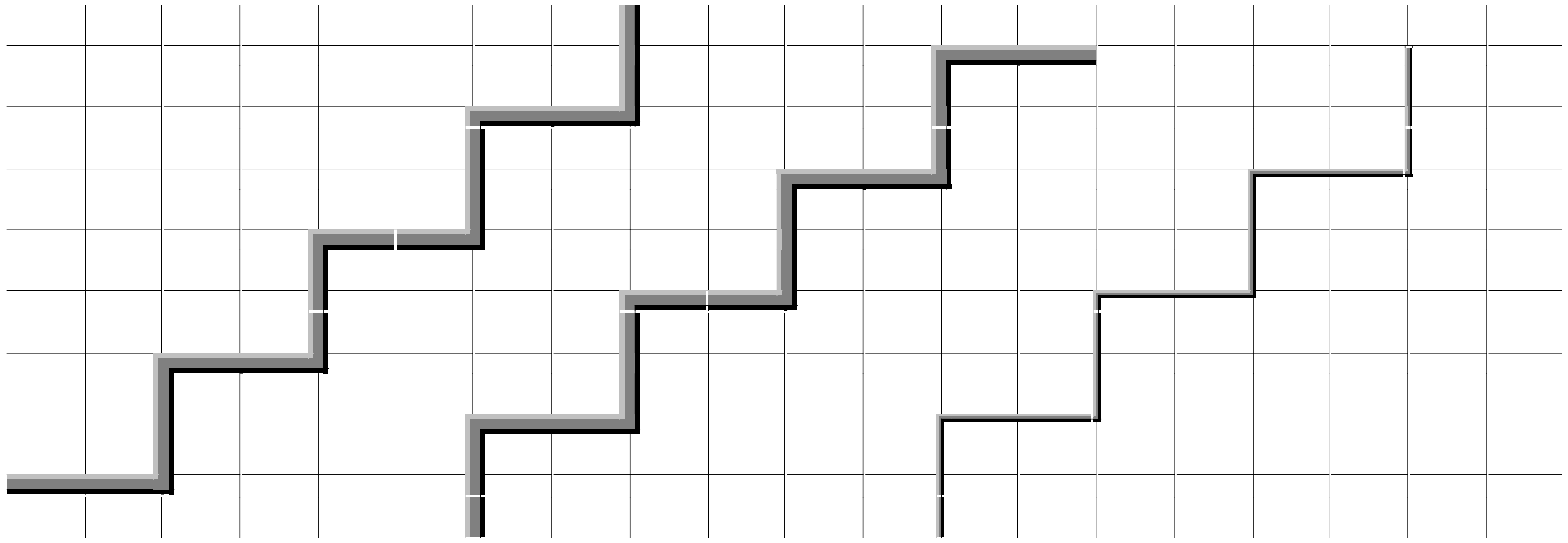
SWI Schedule Risk Steps Early (HP)

- “Some of the criteria commonly used in setting priorities during this initial planning activity are …:
  - Features with greatest risk.
- The most common criterion used for prioritizing the development phase implementation cycles is risk.
- When adopting [new] technology, many teams are concerned that the system performance will not be adequate.
- Ease-of-use is another common risk for a project.
- The [step content] that will provide the best insight into areas of greatest risk should be scheduled for implementation as early as possible.” [COTTON96]

Prioritizing Essentials (Ericsson)

- “Customers do not like disappointments and broken promises.
- Promising the earth and then failing to deliver is not a good way of treating your customers.
- It means you have to negotiate constant upgrades in order, some time in the future, to achieve what was originally promised.
- It is far better to deliver something which just about meets the specification, and then sell enhancement packages, i. e.. additions.
- When every bit costs money, you only add what is really needed. At the center of this discussion … lies the importance of [us] understanding which features and functions are essential for our customers to be able to do what they intend to with our products.
- In the final analysis it is to make money.
- No amount of [product simplification] should ever threaten this.
- This is precisely the main reason why we must always, always be able to deliver on time.
- And exactly the same reason why we must always put basic functionality before special features.
- Even to the extent that special features should often be excluded altogether since otherwise there Is a risk that they may threaten the product’s basic functionality.
- Few people want to admit that this is in fact often the case.”
- [Ericsson94], page 49-50, Jack Järkvik, in the context of building mobile telephone base stations

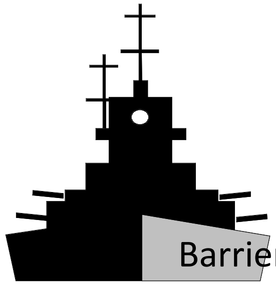
The Step Plan Template



A Simple way to start doing Evo Use this recipe for starters

- **Decide on one required quality objective which the recipient would most value improvement in (example 'Reliability').**
- **Decide on an interesting minimal increment for the recipient. Use this as Must level.**
- **Decide what might be accomplished by the best technology you can insert in the next cycle. Use this as a Plan level. This technology will normally be extracted from the system architecture specification.**
- **Specify the technology you believe will get you the increase (in the Step specification).**
- **Estimate that resources needed to implement that technology. Put this in the step plan.**
- **If the resources exceed those available or permitted by step planning policy, go to step 2 and 'adjust'.**
- **Estimate the impact of the step technology on all other qualities. Document your estimates. Example: in an Impact Table.**

Case 2. The Naval Weapons System. Slide 1 of 3



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

- British Naval Weapons System case: 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."
- 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.
- 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.
- 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.

- Prey 360 deg vision, hunter binocular vision <- Joseph_m_saur@keane.com

Case 2. The Naval Weapons System. Slide 2 of 3

"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!".

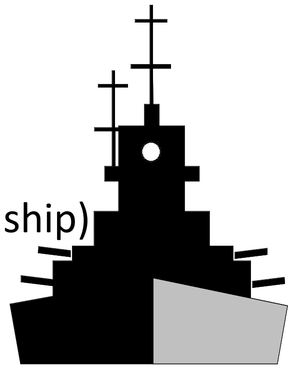
The thing to notice here was that the customer was not the new ship, and that the project was not to put the electronics box on the new ship. The project was to give increased perception to the real customer, The Royal Navy.

Notice the "method" emerging from this example:

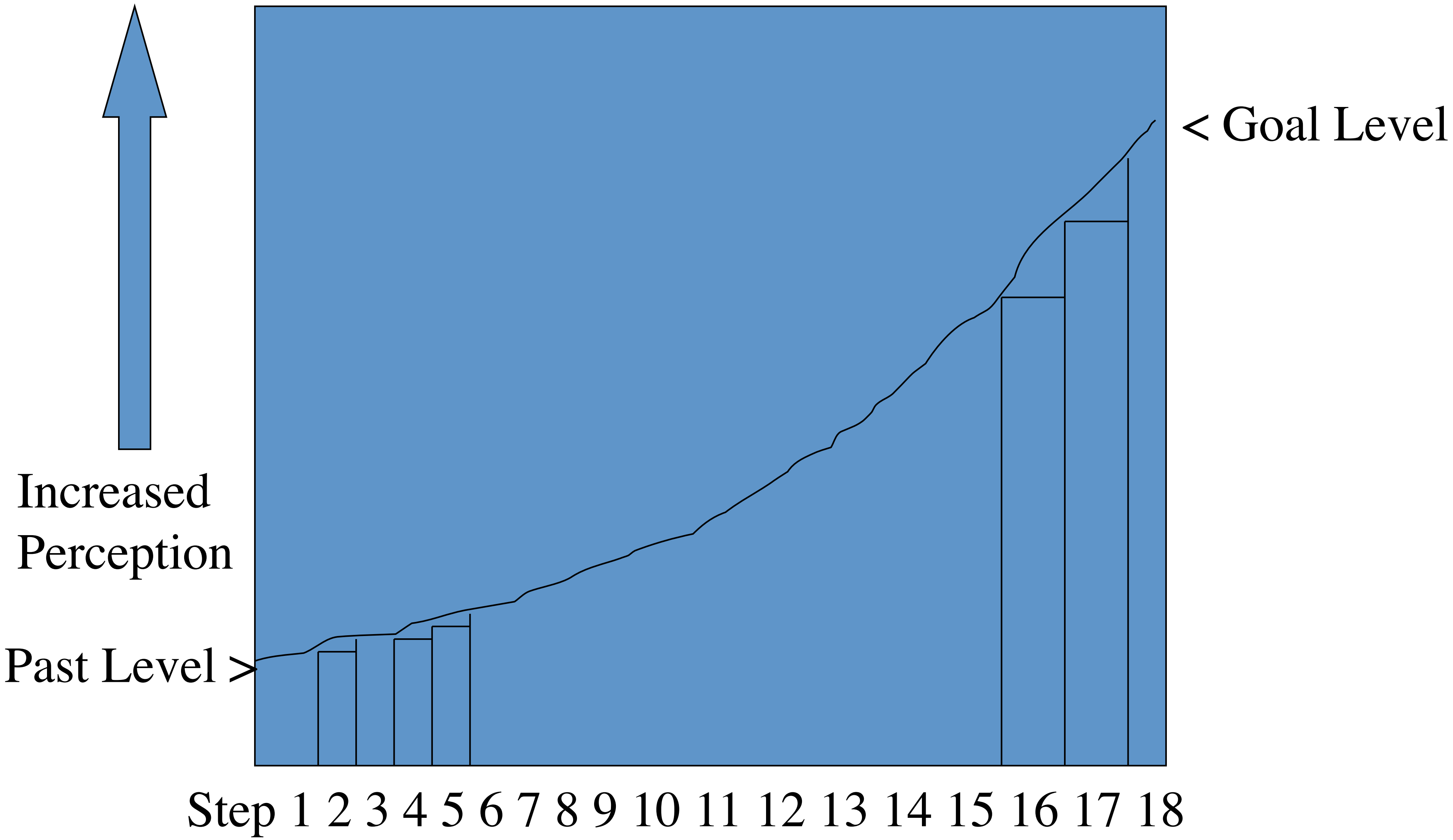
1. Identify the real customer, and plan to deliver results to them.
2. Identify the real improvement results and focus on delivering those results to the real customer.

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".

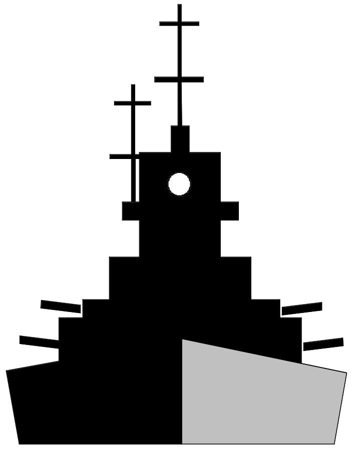


Evo increase of Perception



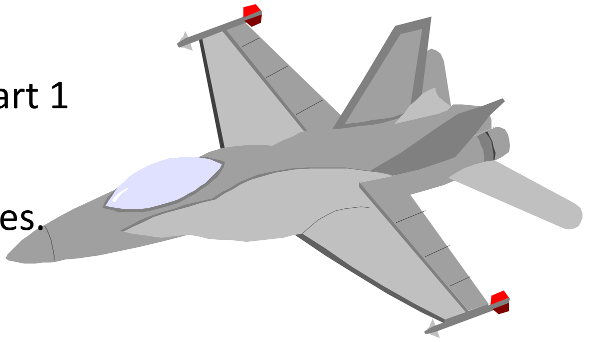
Lessons Learned: Naval Weapons Case (3 of 3)

- **Evolutionary Projects are not normal thinking even amongst well educated engineers.**
- **Evo is a systems method not 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)**



Case 3.
The Air Traffic Control System. part 1

Detail here. See next slide for summary headlines.



• Barrier: "Our customer, and the contract we have made with them, would never permit it."

Air Traffic Control System: In 1986-7 a Swedish client was building an air traffic control training system for another European government.

It was admittedly late. I was called in to help. I suggested we re-plan the project into a series of Evolutionary cycles, delivering the most vital ones earliest.

Fine, they said. One problem though. **The client won't have it**. They stick to their "all at once" delivery contract fanatically, like good bureaucrats.

"What would you want, if you were the customer?", I tried. The assembled executive team said they, of course, were rational people, and would prefer to get the system incrementally. But, they were sure the customer would have none of that nonsense.

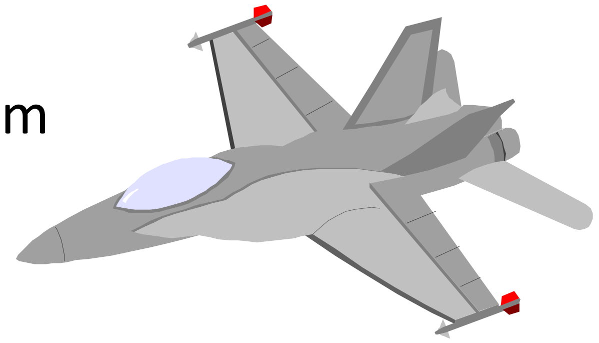
Well, I had their attention for the rest of the day, and they were paying me. So, I asked if they would play the game of making an Evolutionary result delivery plan for the project with me, the way they would like it, if they were customers. The customer had a training building specially built. The hardware was ready, the "software" was late.

Our rough sketch included the design that the early stages should deliver enough of the system so that the **instructors could begin to prepare a course for their students**, and then, that simple air traffic situations were working before more exotic ones. They worked out a ten basic Evolutionary cycle plan that afternoon.

At the end of the day, I said: "Let's do it". They said. "Tom, you don't understand the mentality of our customer. There is no way they will accept this. They only think in terms of the contract as it is.

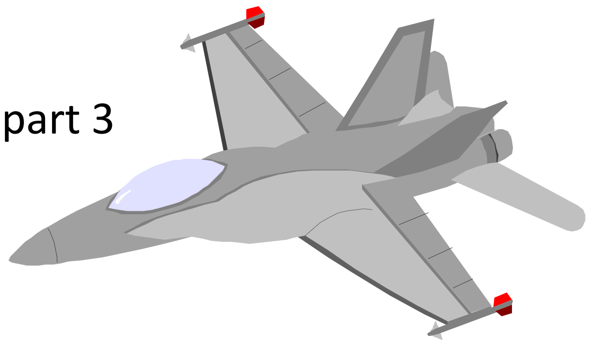
I asked them **what it would cost them to phone the customer and ask what the customer wanted to do**. They promised to try, no promises about results, after I left.

Case 'A-SIM'.
Part 2, The Air Traffic Control System
(HEADLINES SUMMARY SLIDE)



- **Software project, Sweden for Germany Air Traffic Control Training system**
- **Size: 80,000 pages of logical specs to be coded**
  - **Inspection found 20 logical spec defects per page using managers**
- **German contract demanded all or nothing: no Evo there**
- **We (Managers & TG) planned 10 Evo steps; in one day**
  - **Step one: deliver courseware for building new training courses**
- **Customer project manager accepted the Evo plan**
  - **Much to the amazement of the supplier.**
- **Project finally delivered (as opposed to total failure!) by another Company (Brofors).**

Case 3. The Air Traffic Control System. part 3



What happened in later reality?

A few weeks later, I was received back with almost a hero's welcome.

The customer, to everyone's surprise, had accepted the evolutionary plan.

The terrible deadline pressure was "off".

The project could now concentrate on doing a quality job on the Evo cycles.

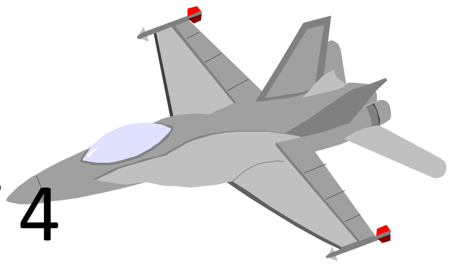
They finally did deliver, and the project was considered "not unsuccessful".

Years later I met the main customer representative who made that decision, on my course in Berlin. At a lunch at a Yugoslavian restaurant he told me: "Of course we accepted the evolutionary plan. Do they think we are stupid?"

Lesson: don't under-evaluate the customer's survival-intelligence, especially if you talk to the appropriate level of manager.

Another lesson: evolutionary results delivery can be used to save large projects even after much initial effort has been placed in the wrong direction.

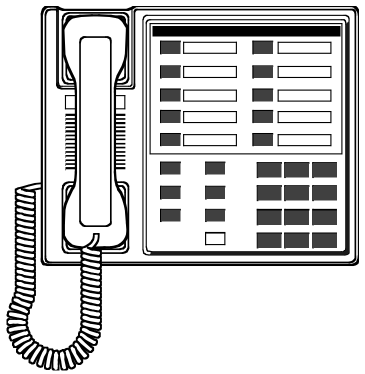
Lessons Learned: Air Traffic Control Case. Part 4 of 4



- **Do not believe all the local experts,**
  - **they are in trouble because they do not 'believe' the answer.**
- **Get to the <u>right level of management</u> to get change decisions.**
- **Do not be afraid to use *common sense*:**
  - **'what would I want if I were the customer?'**

Case 5. The German Telecommunications Company. (see next slide for simplification)

- Barrier: "It is too late, we have already invested so much the old way, that we just have to see it through".
- At a large German telecommunications business in December 1984, about 985 software engineers had been working for three years on a major new world-market product. December 1984 was the deadline for delivery of the product, but their 40,000 node PERT chart, the Financial Director told me, estimated that they had 2 or 3 years more effort left.
- Corporate top management had given them one more year, to December 1985. Deliver, or forget the whole market, which by that time would be taken over by competitors.
- As usual, I suggested re-planning the project in smaller and critical increments first. They told me that this was unthinkable. The software was already written, they claimed, only testing remained. They also had a rather long list of other reasons why Evo would not work for them.
- Using common sense we worked out a basic Evolutionary scheme. The small-model software first (there were 35 signed contracts for it, none for the medium and large systems). Then fundamental telephone services before advanced fancy stuff.
- After what seemed like seven management layers of "you must present this to my boss", we ended up in the office of Herr Raab, The Project Director. He thought it was all good common sense, and stared coldly at his (cowardly, cautious?) sub-ordinates as he asked: "Can you do it this way?", (assenting nods) "Then do it!".
- They did too. By November 1985, on a return visit, they told me that the small systems had been operating for over six weeks with several real customers, with no problems whatsoever. Note, three months before the impossible deadline!
- As in many other cases, I had to spell out the basic steps myself. I had to make them obvious clear simple steps. I could not merely suggest the method. And then, in spite of the obviousness, I had to get to the right person to make a decision.

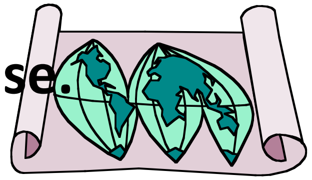


Case. The German Telecommunications Company. HICOM

- 900+ software engineers, had worked 2 years
- Deadline had arrived
  - 35 customer contracts
- 40,000 node PERT plan used for project control
  - Project Finance Manager: ‘we are 2-3 years late’
- 1 day to re-plan EVO, ten steps, critical function first
- 1 days to change critical objectives
  - From Max. 0.5 bugs /KLOC to a numeric ‘availability’ level
- 2 days to sell it to management (Herr Raab, Project director) - who kindly permitted me to report our experiences
- 7.5 months later, first successful deliveries
  - Reported by Dr. Bernhard Falkenberg to me, in München.

See previous slide for more detail.

Case 8. The Swedish Map-making Institute case.



Barrier : "Our system is so small and simple that we don't need many cycles, we'll have it all done shortly."

A Norwegian systems house had contracted a fixed price ($80,000), and fixed deadline for making a map-drawing system, including computer hardware, for the Swedish Government Fixed-Property Directorate.

There were only two employees, Gunnar and Anne-Lise, experienced in making a similar system, and they were assigned to the task.

As an experiment, their top management asked me to spend a day with them.

On that first day, we set quantified goals. One goal in particular was to dominate. The "maintainability" of the system by the customer, at the local customer site had to be very easy. This goal immediately led us to find suitable design ideas (software tools for error analysis and testing). The result of this was that it became clear that we had twice as much work to do, than the salesperson had envisaged, when making the fixed price and deadline commitment.

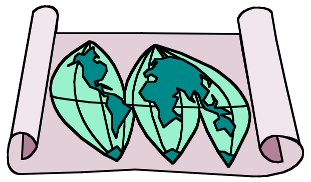
That first day, we developed a ten-cycle Evolutionary plan with the two project people, over, hopefully, the deadlined three-month period.

We planned to get the basic map-making system up and running, in the first cycle or so. We had to use a crude temporary link to their main computer, but it already existed, and cost nothing to make use of, until we in a later cycle built a customized link. Fancy polygon drawing features were also put off until later cycles. All things like maintenance and training documentation were in the last cycles.

After about two, of the projected three, months a small crisis erupted. Both Gunnar and Anne-Lise had been off the job ill. Both had been forced to take time away from the project to service emergencies with previous projects.

So, after two months, three Evolutionary cycles had been delivered to Sweden, and it became painfully clear that the last seven cycles were never going to be completed by the original deadline.

"Hat in hand" Gunnar went to the customer to ask advice about what best to do about his "late" project. "LATE!?" the customer replied (and I was later to hear him and Gunnar both personally tell the tale at a conference in Sweden, where they specifically asked the audience to take my talk about Evolutionary delivery and 'RDM' seriously, as that was the key to their success). LATE!? We are already happily using the system weeks before the promised deadline. We don't yet need the missing features. Please do not be tempted to do "crash program" actions which might threaten the quality of our system. Take your time. Do a good job. And we will only speak of the only system we have experienced, which was actually delivered BEFORE the deadline.



Case . The Swedish Map-making Institute case

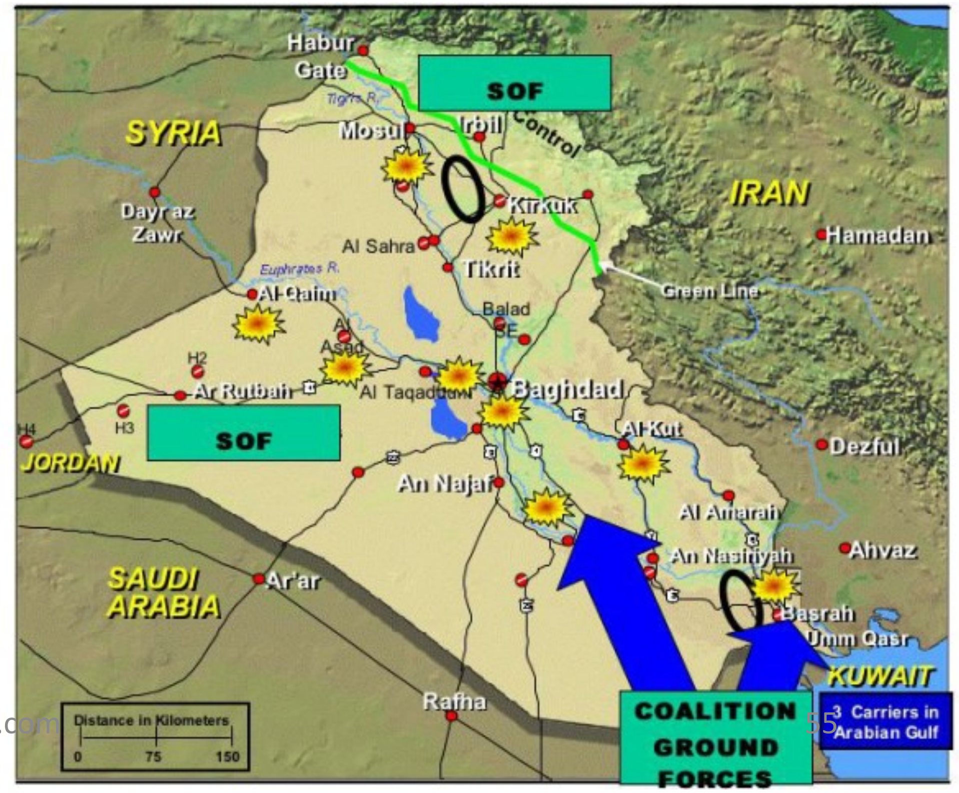
- Small Project: 2 people, 1500-3000 lines of code, 3 months
- Ten Evolutionary steps planned first day
- Both people 1. Got sick a month, 2 had other fires to put out
- Basic and critical increments were successfully delivered to customer satisfaction BEFORE the 3 month deadline in contract. Customer stated in public (conference) they were delighted with (only instance of) ‘Early Delivery’

Case 8. The British Life Assurance Computerized System cancelled by the Board.

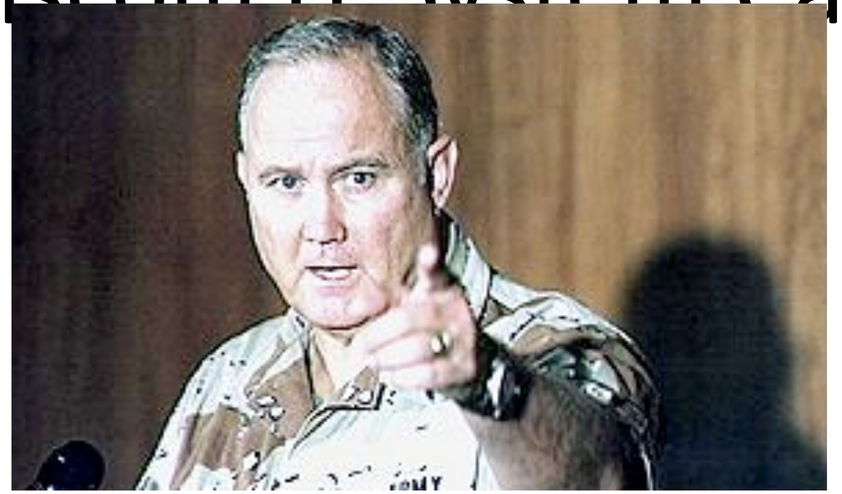
- Spring 1993, nn
- I was asked to analyze the situation at a well known British Life Assurance company. They had spent about £70 million on an "advanced" computerization project. But I found no useful objective was served for anybody. They were building it because advanced stuff was a good idea for a large company.
- Their goal was to have a more-flexible advanced system (not much better defined than that either) - no mention of precisely which improvements, like in operating costs, were going to appear, and when. I believe that it was partly this total lack of clearly-defined goals which permitted the project to wander on, without any warning signals, without any commitment to specific result delivery in small early increments.
- Finally the huge expenditure, and total lack of any planned or previous result delivery to anybody in the insurance company, got to the Board of Directors. They closed down the 200-person project about a month later. It was reported in the trade weeklies.
- So was, a similar "Building Society" (Savings and Loan) system and a Stock Exchange reported, at the same time, with the same magnitude of losses.
- There were plenty of "managers" involved at the assurance company. But they were not managing.
- They did not, in my opinion understand how to define useful results.
- And I had to document this thoroughly for them. I fear that the culture of poor definition of results is so prevalent that my message did not change their practice.

And Now A True War Story

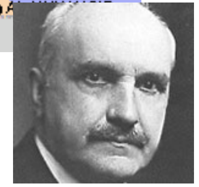
- About Why Bad IT Requirements
  - Can lose a war in Iraq
  - Or at least make it drag on for years



The Persinscom IT System Case



He who does not learn from history is doomed to repeat it



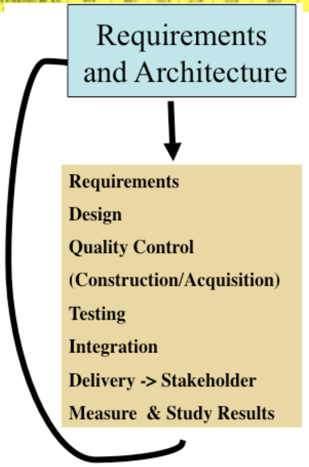
A Man Who understood that “a bird in the hand is worth two in the Bush” <-tsg

The Evo Planning Week at DoD

- **Monday**
  - Define top Ten critical objectives, quantitatively
  - Agree that thee are the main points of the effort/project
- **Tuesday**
  - Define roughly the top ten most powerful strategies,
  - for enabling us to reach our Goals on Time
- **Wednesday**
  - Make an Impact Estimation Table for Objectives/Strategies
  - Sanity Test: do we seem to have enough powerful strategies to get to our Goals, with a reasonable safety margin?
- **Thursday**
  - Divide into rough delivery steps (annual, quarterly)
  - Derive a delivery step for 'Next Week'
- **Friday**
  - Present these plans to approval manager (Brigadier General Palicci)
  - get approval to deliver next week

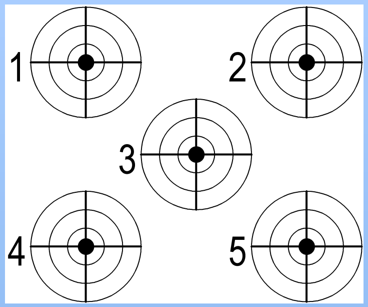
US Army Example: PERSINSCOM

[illegible]



US Army Example: PERSINSCOM: Personnel System

STRATEGIES ➔ OBJECTIVES
Customer Service ?➔0 Violation of agreement
Availability 90% ➔ 99.5% Up time
Usability 200 ➔ 60 Requests by Users
Responsiveness 70% ➔ ECP's on time
Productivity 3:1 Return on Investment
Morale 72 ➔ 60 per mo. Sick Leave
Data Integrity 88% ➔ 97% Data Error %
Technology Adaptability 75% Adapt Technology
Requirement Adaptability ? ➔ 2.6% Adapt to Change
Resource Adaptability 2.1M ➔ ? Resource Change
Cost Reduction FADS ➔ 30% Total Funding



Monday

←The Top Ten Critical Objectives Were decided

Sample of Objectives/Strategy definitions
US Army Example: PERSINSCOM: Personnel System

- *Example of one of the Objectives:*

Customer Service:

Type: Critical Top level Systems Objective

Gist: Improve customer perception of quality of service provided.

Scale: Violations of Customer Agreement per Month.

Meter: Log of Violations.

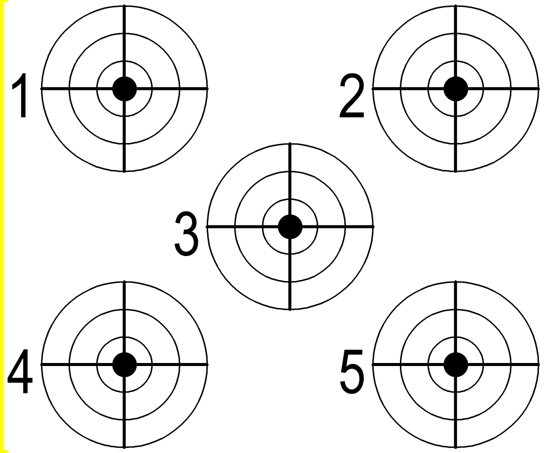
Past [Last Year] Unknown Number ←State of PERSCOM Management Review

Record [NARDAC] 0 ? ← NARDAC Reports Last Year

Fail : <must be better than Past, Unknown number> ←CG

Goal [This Year, PERSINCOM] 0 "Go for the Record" ← Group SWAG

.



US Army Example: PERSINSCOM: Personnel System

STRATEGIES ➔ OBJECTIVES	Technology Investment	Business Practices	People	Empow-erment	*Principles of IMA Management*	Business Process Re-engineering	SUM
Customer Service ?➔0 Violation of agreement							
Availability 90% ➔ 99.5% Up time							
Usability 200 ➔ 60 Requests by Users							
Responsiveness 70% ➔ ECP's on time							
Productivity 3:1 Return on Investment							
Morale 72 ➔ 60 per mo. Sick Leave							
Data Integrity 88% ➔ 97% Data Error %							
Technology Adaptability 75% Adapt Technology							
Requirement Adaptability ? ➔ 2.6% Adapt to Change							
Resource Adaptability 2.1M ➔ ? Resource Change							
Cost Reduction FADS ➔ 30% Total Funding							

Tuesday
The Top Ten
Critical Strategies
For reaching the
←objectives
Were decided

Sample of Objectives/Strategy definitions
US Army Example: PERSINSCOM: Personnel System

A Strategy (Top Level of Detail)

<u>Technology Investment</u>:

Gist: Exploit investment in high return technology.

Impacts: productivity, customer service and conserves resources.

A Sample Objective in 'Planguage'

Customer Service:

Gist: Improve customer perception of quality of service provided.

Scale: Violations of Customer Agreement per Month.

Meter: Log of Violations.

Past [1991] Unknown Number ← State of PERSCOM Management Review

Record [NARDAC] 0 ? ← NARDAC Reports 1991

Fail : <better than Past, Unknown number> ← CG

Goal [1991, PERSINCOM] 0 "Go for the Record" ← Group SWAG

More info on 'Planguage' see www.gilb.com, especially 'Competitive Engineering'

The Persinscom System: Day 3 Impact Estimation Table

	STRATEGIES ➔ **OBJECTIVES**	Technology Investment	Business Practices	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**	

What was the 'next week' Evo step?

- The one week change
  - Change the system to allow tagging the requestor with rank (General Schwartzkopf)
  - Move higher ranks to head of the request queue
- Result:
  - The 'General' would get faster response from the system.
    - The 'Responsiveness' objective.
  - One major objective would be partly achieved

Next weeks Evo Step??

- “You won’t believe we never thought of this, Tom!’
- The step:
  - When the Top General Signs in
  - Move him to the head of the queue
    - Of all people inquiring on the system.

The Step Definition: Template
Simple version with <hints>

Step Name: <a tag> [more detail like <which product>, <which area of application>]

Step owner: <who has control over this step and responsibility for it?>

Stakeholder: <who are you going to give value to??>

Implementor: <who is in charge of implementing this step>

Step Content:

{Functions and Designs}

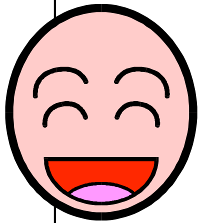
Tasks

<list of step tasks>

Step Value:

<numeric or rough estimate of value to stakeholder in terms of formal objectives planned level and scales>,

at least value on scale 0 (none) to 9 (highest)



Step Cost:

<Estimates of time and other costs (engineering hours) which are budgeted or constrained by the Evo 2% policy>

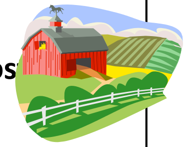
At least cost on scale 0 (dirt cheap) to 9 (high and unpredictable)

Step Constraints:

< any legal, political, economic, security constraints imposed on implements>

Step Dependencies:

<anything which must be in place, finished, working properly, for us to be able to start this Evos to complete it> <--<who says this is true?>



Version March 18 2002

The Step Definition: Template
Simple blank space version

Step Name:

Type: Evo step specification.

Step Version:

Step Owner:

Step Stakeholder(s):

Step Implementor:

Step Content:

Step Values:

Step Costs:

Step Constraints:

Step Dependencies:

Step Risks:

Version March 18 2002 tg

EVO STEP TEMPLATE <with hints>

Tag:
Type: 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 this is true?)>.
Quality Level: <Maximum remaining major defects/page, sample size, SQC date>.

Stakeholders: <Who are you going to give value to? >.
Owner: <Who is taking responsibility for the step in terms of specification>.
Implementers: <Who is in charge of implementing this step>.

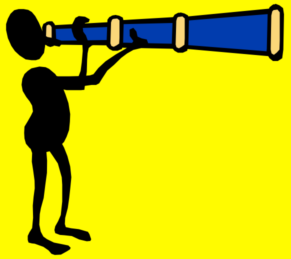
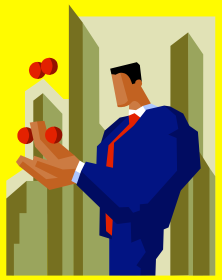
Step Content Summary: <Brief description>.
Description: <Give a detailed, unambiguous description of the step, or a Tag reference to a place where it is described. Remember to include definitions of any local terms>.
Implementation Details: "Includes relevant details, such as <which product>, <which area of application system>."
Step Content: <Step Elements: {Design Ideas, Functions, Tasks, Re-used Step definitions}>.

=========== 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>.
Risks: <Any risks that need to be taken into account>.
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?)>.
Priority:
<Name, using Tags, any system elements, which this function can clearly be done *after* or must clearly be done *before*. Give any relevant reasons>.

=========== 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?)>.

=========== 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>..

CE book Figure 10.9: A possible specification template for a one page Evo step plan. VersionApril 26 2002

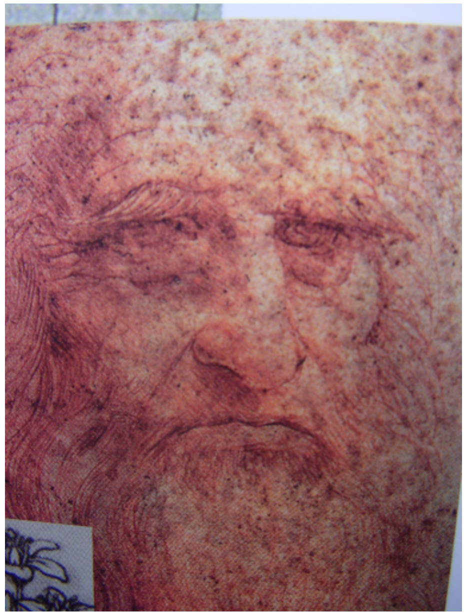



The Step Definition: EXAMPLE 2
OF FILLING OUT THE FORM DEFINING A STEP

- **Step Name: Tutorial [Product XX, Basic]**
  - **Stakeholder: Marketing, Timo M (<agreed, Next Friday>)**
  - **Implementor: <Mika>**
  - **Step Content:**
    - **HCTD :<Hard Copy Text document> <-- Can do 1 week Mika**
      - **Basic minimal functions.**
        - » **Step by Step Instructions, in English**
        - » **Focus on sales aspects, not how to do it (not yet, in this step)**
        - » **Go to specific web sites**
        - » **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**
        - » **Intended audience: Marketing Guy**
        - » **Questionnaire for Stakeholder**
        - » **Process for Testing with stakeholder (example observation, times)**
      - **No illustrations, just text.**
  - **Step Value: (to Timo, Saleability) : <some possibility of value>,**
    - **Stakeholder Developers: value of feedback on a tutorial.**
  - **Step Cost: 10 hours per page, < 10 hours <--Mika**
  - **Step Constraints: must be deliverable within 1 calendar week.**
    - **At Least 3 hours of Timo's time for input and trial feedback**
  - **Step Dependencies:**
    - **<feature list of XXX and Product XX WAP Browser> <--Mika**

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



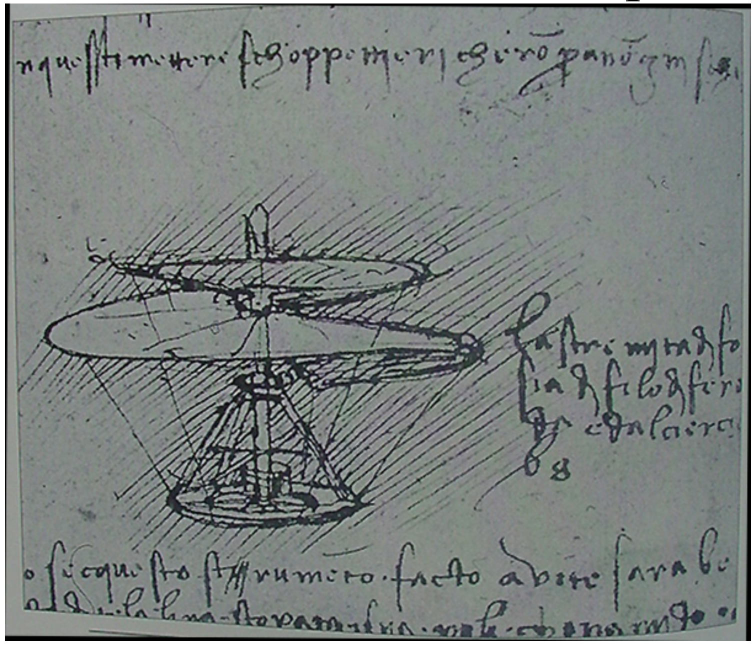
Da Vinci on Practical Feedback

- Leonardo, proudly described himself as:
  - **Uomo senza lettre** (man without letters)
  - **Discepolo delle esperienza** (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

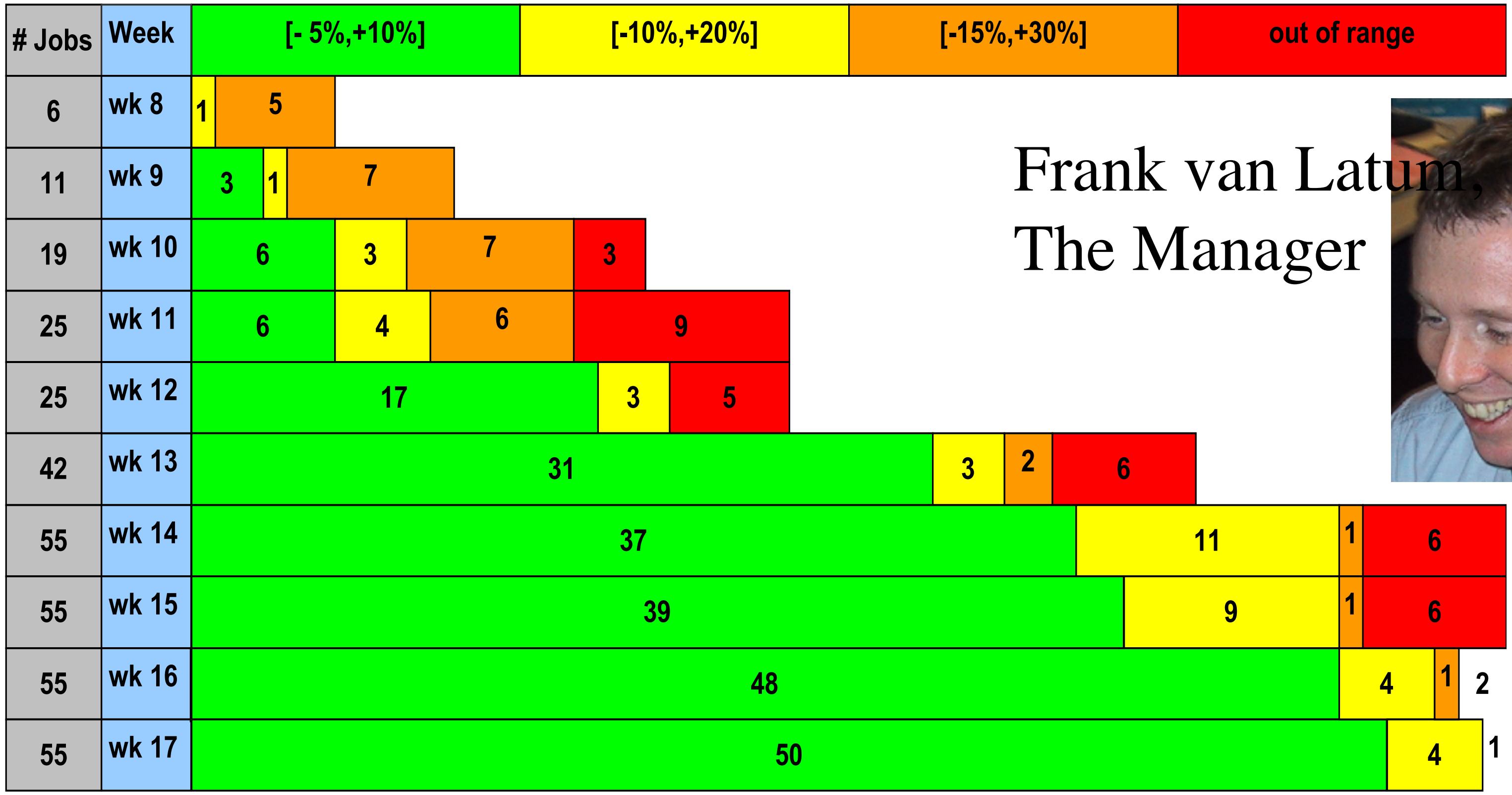
Leonardo's persistence

- "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



Philips Evo Pilot May 2001



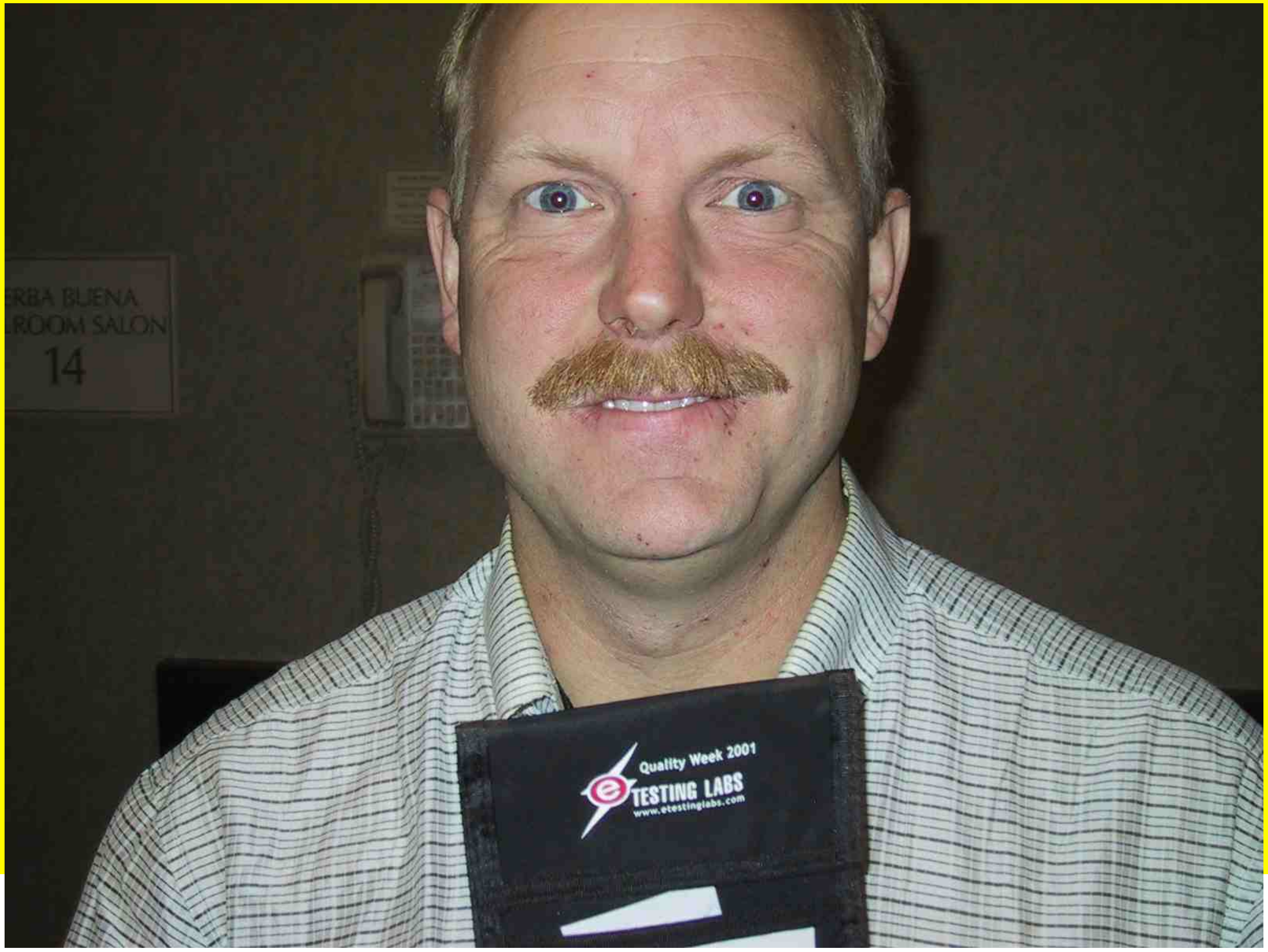
Frank van Latum,
The Manager

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

Kent on XP

•



ERBA BUENA
ROOM SALON
14
Quality Week 2001
e TESTING LABS
www.etestinglabs.com

Overlaps between Evo and XP

(BLUE)

Planning

- **User stories are written**
- **Release planning creates the schedule**
- Make frequent small releases
- The Project Velocity is measured
- The project is divided into iterations
- Iteration planning starts each iteration
- **Move people around**
- **A stand-up meeting starts each day**
- Fix XP when it breaks

Designing

- Simplicity
- **Choose a system metaphor**
- **Use CRC cards for design sessions**
- Create spike solutions to reduce risk
- No functionality is added early
- Refactor whenever and wherever possible

Coding

- **The customer is always available.**
- Code must be written to agreed standards.
- **Code the unit test first.**
- **All production code is pair programmed.**
- **Only one pair integrates code at a time.**
- Integrate often.
- **Use collective code ownership.**
- **Leave optimization till last.**
- No overtime.

Testing

- **All code must have unit tests.**
- **All code must pass all unit tests before it**
- **can be released.**
- **When a bug is found tests are created.**
- Acceptance tests are run often and the score is published.

Courtesy Niels Malotaux July 2002, Niels Malotaux <niels@malotaux.nl>

Differences between Evo and XP

EVO

- Suited for large & small Systems & Software Development
- Results Centric
- Stakeholder focus
- Works with anybody
- Numeric
  - specification of (strategic) objectives
  - prioritization (impact tables)
  - progress tracking

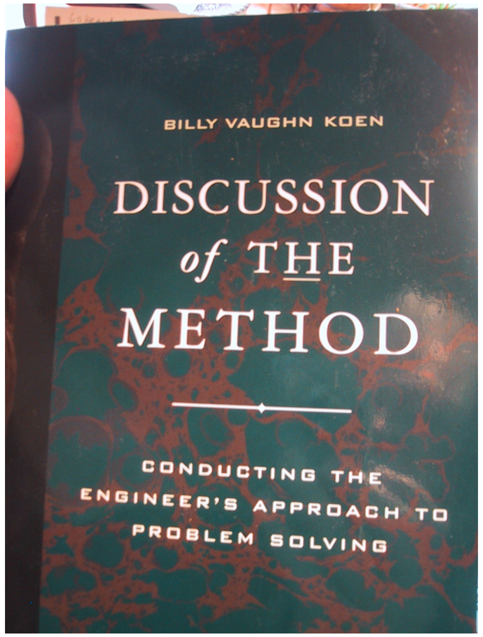
XP

- Suited for small Software Development only
- Code Centric
- Developers focus above Process focus
- Need seasoned programmers
- NO numeric specification of objectives, prioritization nor tracking

Courtesy Niels Malotaux July 2002, Niels Malotaux <niels@malotaux.nl>

Koen on Risk Control

- 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



Evo from legacy system

- Why we fail to exploit the old system
  - Nobody wants to deal with the messy old system
  - Young technologists want to play with new technology to save the world
    - But that have 50% total project failure rates!
- Advantages of doing it
  - Customers/Stakeholders are already there
  - They might pay for short term improvements
  - They might not leave you for competitors
  - You can focus on real valued improvements
    - Not 95% effort to recreate a system which already works!
      - And risk introducing new problems, not in the old system
  - You can still, eventually introduce the newest technology
    - But in the long run
    - And sometimes in the short run (ex Reuters PC : Old data).