Chapter
10

Evolutionary Project Management

How to Manage Project Benefits and Costs

Glossary Concepts

Evolutionary Project Management
Gap
Step (or Evo Step)
Result Cycle
Backroom
Frontroom
Before
After
Dependency

10.1 Introduction to Evolutionary Project Management

In 1994, the US Department of Defense issued a temporary military standard, MIL-STD-498, which explicitly supported the use of evolutionary project management (Evo). It also supported the related concept that projects do not initially have the 'final and correct user requirements' specified. This standard has now been evolved into civil standards (such as IEEE standards) and is continuing to influence new standards. Such recognition for Evo is deserved as it has probably the best track record of any known project management method (Larman and Basili 2003).

Practical Experience with Evolutionary Project Management

Surprisingly, many project cultures have little formal knowledge of Evo, even though it has been in use since the 1960s. The first documented large-scale industrial use of Evo was from 1970 to 1980 and on, within IBM Federal Systems Division (IBM FSD, later owned by Loral and Lockheed Martin). Working within the military and space sector, they had complex 'high-tech' projects requiring *state-of-the-art* performance with *fixed* financial budgets and *fixed* deadlines. These extreme requirements drove them into developing methods known as 'Cleanroom', which included using Evo. Harlan Mills reported on these early IBM FSD experiences as follows:

> Ten years ago general management expected the worst from software projects – cost overruns, late deliveries, unreliable and incomplete software. Today, management has learned to expect on-time, within budget deliveries of high-quality software. LAMPS . . . a 4 year . . . 200 person-years (project was delivered) in 45 incremental deliveries. Every one of those deliveries was on time and under budget. [The] NASA space program . . . 7,000 person-years software development . . . few late or overrun [budgets] . . . in . . . [a] . . . decade, and none at all in the past four years.
>
> *Harlan Mills (Mills 1980: reprinted (IBMSJ 1999))*

Harlan Mills told me that it was precisely the 'fixed deadline' and the cost situation of 'lowest bidder wins', which led to the development of Evo. He also told me that their model for Evo was the way in which intelligent military and civil rockets move towards their targets using feedback and control mechanisms.

//INTEGRA/ELS/PAGINATION/ELSEVIER UK/OMP/3B2/FINALS/0750665076-CH010.3D – 291 – [291–320/30] 29.6.2005
12:44PM

More recently, Hewlett-Packard has also publicly documented the benefits of Evo (Cotton 1996; May and Zimmer 1996; Bronson 1999; Upadhyayula 2001; MacCormack 2001). Evo has been in use within the organization since at least 1988.[1]

> The evolutionary development methodology has become a significant asset for Hewlett-Packard software developers. Its most salient, consistent benefits have been the ability to get early, accurate, well-formed feedback from users and the ability to respond to that feedback.
>
> *Elaine May and Barbara Zimmer, Hewlett-Packard (May and Zimmer 1996, Page 44).*

Microsoft has also been documented as using Evo extensively (MacCormack 2001; Cusumano and Selby 1995). The Open Source Methods (like Linux) (Maier and Rechtin 2002) and Agile Software Development methods (Cockburn 2002; Abrahamsson et al. 2002) have also clearly demonstrated the power of Evo in delivering good software rapidly.

The use of Evo is also proven within engineering processes. For example in 1988, this author consulted with over 25 projects (about 120 aircraft design engineers) at Douglas Aircraft. Of course, new aircraft did not fly the next week (most of the projects were modifications and upgrades, or integrating new components). But, real results capable of giving useful feedback were delivered to real stakeholders in weekly increments. Management approved each Evo step in advance. They found the method was practical, low risk and they could not resist seeing results fast.

Underlying Principles of Evolutionary Project Management

The underlying principle of Evo is the Plan-Do-Study-Act cycle (PDSA cycle). In other words, the 'process control cycle' as taught by Walter Shewhart of AT&T from the 1920s onwards and, by his pupils, W. Edwards Deming from the late 1940s to the 1990s (Deming 1986) and Joseph Juran (1974). It is one of nature's great laws; learn, adapt and survive.

Evo expands on the Statistical Process Control 'Plan-Do-Study-Act' (PDSA) cycle concepts since it demands:

[1] In 1988, the author taught Evo to an HP project team, which included Todd Cotton, who later went on the spread the method widely at HP (Cotton 1996), (May and Zimmer 1996).

- *Early* delivery of project results to stakeholders (for example, 'next week'!)
- *Frequent* releases to stakeholders (for example, 'every Friday')
- *Small* increments ('steps') (for example, no more than 2% of total project)
- *Useful*-to-stakeholder steps (benefit delivered, value experienced)
- Selection and sequencing of steps according to degree of stakeholder benefit; usually but not always, high-profit steps first (using dynamic priority determination).

Who could be against such an idea? It is a powerful competitive weapon. In practice, the main problem for project management is usually 'how?' How is a major project divided up into a succession of say, monthly improvements to be delivered into the hands of the users? Some people don't see any difficulty. Many, however, are unable to envision such small step decomposition for their projects, and usually claim it is impossible. In my experience, there are *always* ways of achieving such decomposition. It is a question of training, being determined to find the answer and having the right technical knowledge and/or sufficient insights into the stakeholder environment.

As our entire political, technological and economic world now has a greater rate of change and is much more unpredictable and complex than ever before, adaptive methods, such as Evo, must

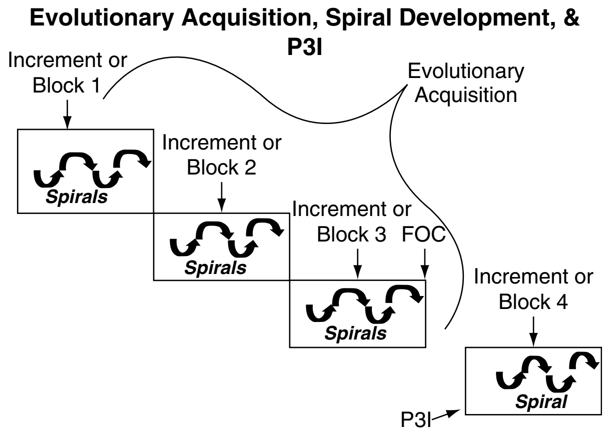

Figure 10.1
Illustration from letter to senior staff in the US Department of Defense from Under Secretary of Defense E. C. Aldridge, Jr., April 12, 2002. ''Since the publication of DoD Directive 5000.1 and DoD Instruction 5000.2, in which the Department established a preference for the use of evolutionary acquisition strategies. . .'' See http://www.acq.osd.mil/dpap/Docs/ar/1_multipart_xF8FF_2_EA%20SD%20Definitions%20final.pdf. This illustration is included mainly to show that evolutionary methods have been accepted at top levels within US Government.

become the norm to manage projects, rather than the exception. (It is only in the case of predictable, low-risk, low-turbulence, low-competition situations that there is little need for an evolutionary method.)

A Basic Evolutionary Planning Policy

1. **Financial Control**: No project cycle can exceed 2% of total initial financial budget before delivering some measurable, required results to stakeholders.
2. **Deadline Control**: No project cycle can exceed 2% of total project time (For example, one week for a one year project) before delivering some measurable, required results to stakeholders.
3. **Value Control**: The next step should always be the one that delivers best stakeholder value for its costs.

Policy Example: Formulating an Evo policy is the first stage of deciding how to do Evo. I frequently recommend this Evo policy to senior managers who must implement and support an Evo project management policy. You can adjust the level of detail to suit your environment.

10.2 Practical Example: Evolutionary Project Management

I have used this following example on numerous occasions. It stands such repetition as it gives such good insight into one common perceived barrier to Evo: "Delivery cannot be done until the new thing (such as a building, organization or IT system) is *ready* in some years time."

The Naval Radar System

Once, when holding a public course on Evo in London, a participant came to me in the first break and said he did not think he could use this early-incremental method. Why? "Because my system is contracted 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 my confidence that there is *always* a solution to making a project evolutionary. So, I 'bet' that we could find an Evo solution during our lunch break. He sportingly accepted.

//INTEGRA/ELS/PAGINATION/ELSEVIER UK/OMP/3B2/FINALS/0750665076-CH010.3D – 291 – [291–320/30] 29.6.2005
12:44PM

At lunch, he started by explaining that his research team made a radar device that had two antennas instead of the usual one (the dual signal sources were analyzed by a computer, which presented their data). It was for monitoring the ship- and air-traffic surrounding the ship it was on. This, I understood, was similar to having two eyes, instead of being a cyclops.

I then made a stab at identifying the 'results' he was delivering and who his stakeholders were (two vital insights for making Evo plans). "May I assume that the main result you provide is 'increased accuracy of perception', not just a black box, and that your major stakeholder is 'The Royal Navy', not primarily the ship (also one of the many stakeholders) itself?" "Correct", he replied. (I'm simplifying a bit, but the point to note is that identifying the primary real requirements and stakeholders, gives a 'wider playing field'.)

"Does your 'black box' work, more or less, now, in your *labs* (another stakeholder)?" I ventured. (Because, if it did, that opened for early use of some kind.) "Yes", he replied. "*Then what is to prevent you from putting it aboard one of Her Majesty's current ships* (yet other stakeholders!)? Initially running in parallel with conventional radar. Then ironing out any problems in practice. Enhancing it. Possibly giving that ship itself immediate increased capability, in a potential sudden real war? Then, when your new ship is launched, your system will be far more mature and safe to use", I tried innocently. (Actually, he got these points before I said anything!)

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

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

Notice the 'method' emerging from this example:

1. Identify the primary stakeholders. Do not get distracted by secondary stakeholders. The primary stakeholder was not the 'new ship'. It was the 'Royal Navy' or even 'The Western Alliance.'
2. Look for the primary and real performance objective. Do not get distracted by the *perceived* project 'product' or secondary and supporting requirements or designs (like the radar system with two sources). Keep asking 'why?' until you find the primary objectives.

   The real objective was not 'to put the electronics box on the new ship'. It was 'an increased accuracy of perception.' In other words, 'an improvement in a performance aspect ("perception": a quality) of the radar function.'

The moment you have converted the result into such a 'scalar requirement,' then *evolving* towards your targets along that scale seems relatively easy to plan. This is one reason why we emphasize quantifying performance requirements in Planguage for use in Evo.

3. *Focus* your activity on delivering the required results to the stakeholders. Plan to deliver the results in increments or 'steps' to stakeholders. One reason is in order to get feedback from stakeholders on the way. There are several practical tactics you can use for identifying potential Evo steps, see Figure 10.6.
4. Don't take the formal contract too literally! *Early useful* results are *always* welcome, even when not demanded in a contract or requirements. (Check! Does the contract or requirements specification actually say: "We refuse to accept early delivery of any useful, partial results"?)
5. Don't assume that your project staff is thinking along these lines. Evo is not yet 'normal thinking' even amongst well-trained engineers.
6. Avoid Narrow Distractions. Think Big. Think System-wide.

10.3 Language Core: Evolutionary Step Specification

Step Content

An evolutionary step ('step' or 'Evo step') is a package of one or more design ideas.

A step has to be capable of being delivered as an organic whole. A step should stand 'on its own' as a complete deliverable.

A step also will have a defined size constraint. From the point of view of risk control, any step should only consume between 2% and 5% of the total project budget for time and financial cost. This has implications for the decomposition of design ideas, function changes and the scope covered by a step.

Delivery of a step is always intended in some defined way to move a real system (very occasionally, a trial system) in *the direction of its specified requirements*. A step might modify a system's function attributes, its performance attributes and/or its resource attributes. (Of course, when implemented, a step might not produce the expected results. It could, for example, have unintended, unexpected and undesired side effects.)

At some stage during the design process (maybe when initially specified, maybe later), a step is identified for delivery at some specific time, place and on some defined event conditions for a system. The conditions are specified using *qualifiers (for example, [Europe, Next Year, If a Competitive Product is on the Market].* 'Place' qualifiers can be any useful combination of system users, system locations, system components and system functions (*for example, [{Marketing Staff, Accountants}, {Europe, North America}, {Software Products, Training Products}, {Accounting, Marketing, Analysing Monthly Reports}]*).

Step Name

For communication purposes, a step may be named after its dominant content. A unique step-reference name is useful for specification of reuse of a step. Step names can have qualifiers (for example, Step A [Europe]).

EXAMPLE

Web Plan [Europe]:
Type: Evo Plan.
Consists Of: Step {Handbook, On-line Help, French Language Variant, Remove Spelling Mistakes, Provide For Customer Feedback}.

Step Dependency

Delivery of some steps might be dependent on other steps having already been delivered. Any step dependency must be explicitly stated in a step specification. In some cases, step dependency will be total and it will be impossible to deliver unless the dependency is met. In others, the step will be *capable* of delivery, but its impact on the requirements, such as meeting goals, will be significantly less.

EXAMPLE

Web Plan [Europe]:
Type: Evo Plan.
Consists Of: Step {Handbook, On-line Help, French Language Variant, Remove Spelling Mistakes, Provide for Customer Feedback}.
Dependency: Remove Spelling Mistakes Before Provide for Customer Feedback.

Step Sequencing

An Evo plan is a set of sequenced and/or a set of yet-to-be-sequenced steps. The current planned sequence of delivery of any of the steps should be reconsidered after each step has actually been delivered and the feedback has been analyzed. Many factors, internal and external, can cause a re-sequencing of steps and/or the insertion of previously unplanned additional step(s) and/or the deletion of some step(s).

It is the identification of the *next* step for delivery that should be our focus for detailed practical planning. After all, at the extreme, other planned steps may never be implemented in practice. So, the minimum information, initially needed for a step, is that needed to support the decisions for step sequencing.

When determining step sequencing, there are several factors including:

1. *Step dependencies* with other potential steps.
2. The *value that stakeholders will obtain if a step is delivered*. This is the key factor. Ask your stakeholders what unfulfilled requirements would be of most value to them and ask them for evidence supporting their choices. Different stakeholders might well choose different requirements. It could be that a requirement 'wins' due to its aggregated value to several stakeholders.
3. The *value to cost ratios* and the *performance to cost ratios* for steps (generally, the step with the highest value to cost ratio is implemented *earliest*. The performance to cost ratios are also another consideration).
4. The *gaps* between benchmark levels, or currently delivered levels, and the goals ('the biggest' gap or 'the toughest' gap is highest priority). The requirement with the biggest gap is the requirement that is least satisfied (that is, the lowest percentage of the way from the baseline towards the target has been achieved). A requirement, which is considered very tough, might also be a priority to start work on.
5. Stakeholder *opinion* (which can overrule any other logic). ''I want this now!''

Specification using Planguage

There are many ways to express Evo plans. You can use any style that suits you. Here are some examples showing how Evo plans and steps can be specified using Planguage.

EXAMPLE

Early Adopters:
Type: Step [Base = Current Product].
Consists Of: {DIF: Function [Country = USA]: Get Geographical Data, DIA, DIB}.

Note:

i) DIA, DIB and Get Geographical Data will have been previously specified and defined elsewhere. The design idea, DIF is defined here.
ii) 'Early Adopters' is a step, which consists of a package of three design ideas, DIF, DIA and DIB.

iii) Future steps are incremental additions to an existing system. We can define the system prior to delivery of a step by using the parameter 'Base'.
iv) In the 'Early Adopters' step, we explicitly declare the specification to be 'Type': (<- that's the 'explicit' part) 'Step.' 'Function' is used to tell the reader the type of Get Geographical Data.
v) The function Get Geographical Data is limited to one specific Country, USA.

Here is an example of stating an Evo plan.

EXAMPLE Product Plan:
Type: Evo Plan.
Includes: Step {SP [Before Rest], Rest: {SM, Step = SV, SX [After SM], SZ}}.

This example shows that an entire Evo plan can be summarized, at a high level, in a single Planguage statement. An Evo plan consists of a series of steps. In this example, there is a set of steps {SP, SM, SV, SX, SZ} which make up the Evo plan, named 'Product Plan.' 'Rest' is defined as being part of an Evo plan and consists of four defined steps. 'SP' is defined as the step to do 'before' the steps in Rest. The other steps have not had their sequence determined. There are no restrictions on doing them in parallel, or in arbitrarily convenient sequences, except that SX must be done 'after' SM.

Qualifiers specify 'where' a step is to be implemented. Here are two examples showing system location and system function being stated:

EXAMPLE Step 22:
Type: Step.
Consists Of: SR [State = {CA, NV, WA}].
'Step22' implements 'SR' in three US states: CA, NV and WA.

EXAMPLE Step 1:
Type: Step.
Consists Of: SP [Country = North America = {USA, CAN, MEX}, FX [State = {WA, GA, FL}]].

'Step 1' implements 'SP' in three countries. The function, FX within SP, is however restricted to just three US states. Note, the geographical concept or market area, 'North America' is defined here locally, for intelligibility or future reuse.

//INTEGRA/ELS/PAGINATION/ELSEVIER UK/OMP/3B2/FINALS/0750665076-CH010.3D – 291 – [291–320/30] 29.6.2005 12:44PM

10.4 Rules: Evolutionary Project Management

Tag: Rules.EVO.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Gist: Rules for Evo Plan Specification.

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

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

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

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

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

R5: **Priority**: The 'next step', at any point in the project, should ideally be selected using an Impact Estimation table to evaluate step options. Steps that you estimate to deliver the greatest stakeholder benefits, performance improvements (Sum of Percentage Impacts) to stakeholders, or that have the best performance to cost ratio, shall generally be done earliest, wherever logically possible, and when 'other considerations' (such as a customer contract or request) do not have higher priority. Any specific priority factors, which override going for the greatest stakeholder benefits first, shall be clearly documented.

There must be some *specified* clear rationale, policy or rule behind prioritizing steps differently from this rule. This could be some estimate of value of a step, which is outside the scope of the specific Impact Estimation table, which might have priority.

EXAMPLE Step 44:
Type: Step.
Consists Of: ABC [UK]: <- Contract Requirement 6.4.
Rationale: The contract demands we deliver this step at this point.

Optionally, there can be a project-defined constraint of a step having to achieve a minimum estimated value (financial growth or saving), overall performance improvement or performance to cost ratio before being considered for implementation at all.

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

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

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

R9: **Completeness**: *All* the specified design ideas for a system, implemented or not, must be represented *somewhere* on an Evo plan. (Remember, you can use tags and you can declare a large set of designs with a single tag. For example, A: Defined As: {B, C. D, E, F}.)

Rationale: This is because failure to include all the specified design ideas somewhere on the Evo plan causes confusion. It leaves us to wonder:

- *Was it forgotten inadvertently?*
- *Why is it specified, if it is planned never to be implemented? (If you are just keeping the idea in reserve, be specific.)*

10.5 Process Description: Evolutionary Project Management

These Evo processes are generalized. Modification to suit individual circumstances might well be required.

See also Figures 1.3 and 1.7 in Chapter 1.

Process: Strategic Management Cycle ('The Head')

Tag: Process.SM.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Note: Process.DC (Delivery Cycle) is a separate process defined below.

Entry Conditions: Entry.SM

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

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

E3: The design idea specifications have been evaluated using IE and, the IE table has exited from SQC.

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

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

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

Procedure: Procedure.SM

P1: Plan:

1. Modify if necessary top-level project requirements and design ideas.

//INTEGRA/ELS/PAGINATION/ELSEVIER UK/OMP/3B2/FINALS/0750665076-CH010.3D – 291 – [291–320/30] 29.6.2005
12:44PM

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

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

P2: Do:

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

P3: Study:

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

P4: Act:

Adopt the change, or *abandon* it (revert to previous state before step implementation). Or, decide to run through the cycle again, but possibly under changed conditions *(paraphrased from W.E. Deming 1986).*

Go to P1 (that is, *continue cycling*), unless Exit Conditions are met.

Exit Conditions: Exit.SM

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

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

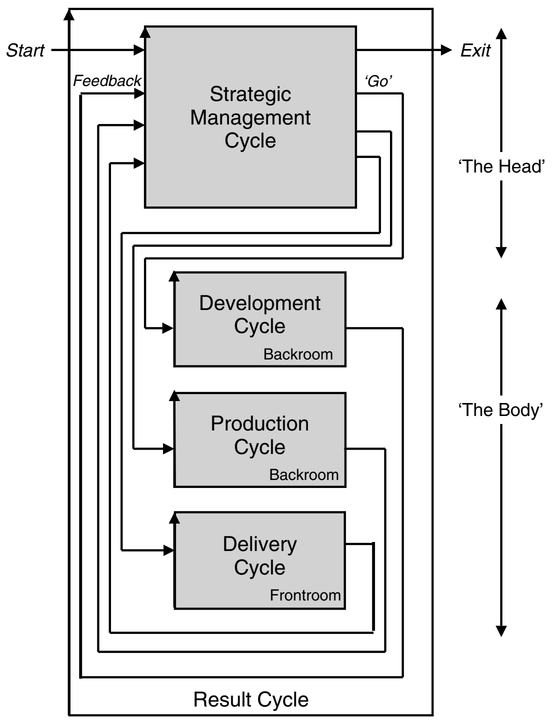

Figure 10.2
The result cycle for an Evo Step.

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

Tag: Process.DC.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

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

Entry Conditions: Entry.DC [Step n]

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

E2: The numeric feedback results from any previously completed steps must be available to the design team and must have been studied. *(You may want to re-do the previous step before proceeding.)*

Procedure: Procedure.DC [Step n]

P1: Plan:

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

P2: Do:

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

P3: Study:

1. Determine the results of delivering the step. Obtain any relevant measurements: test, measure and sample, to establish the new performance levels and the new operational cost levels. Compare results to the short-term and long-term targets.
2. Analyze the data and produce a feedback report for management.

For example, use an Impact Estimation table as a tool to do this study task.

P4: Act:

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

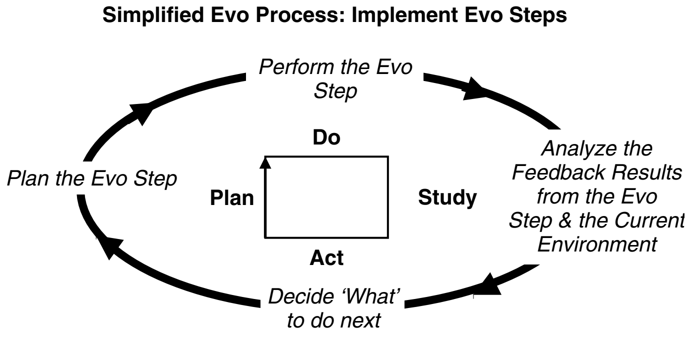
Simplified Evo Process: Implement Evo Steps
Perform the Evo Step
Do
Plan the Evo Step
Plan
Study
Analyze the Feedback Results from the Evo Step & the Current Environment
Act
Decide 'What' to do next

Figure 10.3
A simplified Evo process: implementing Evo steps.

Exit Conditions: Exit.DC [Step n]

X1: Step completed, or dropped. Exit a step only when all step performance levels and function requirements are reached (or wavered formally). Give up if reaching planned requirements is impractical, or if you run out of resources.

Note: Process Descriptions for the Development Cycle and the Production Cycle are not given in this text.

A Simplified Evo Process

Background: A simplified version of the Evo process to use on small projects. It also serves to help understand the larger, full-scale Evo process.
Tag: Simplified Evo.
Version: October 7, 2004.
Owner: TG.
Status: Draft.

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

Policy

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

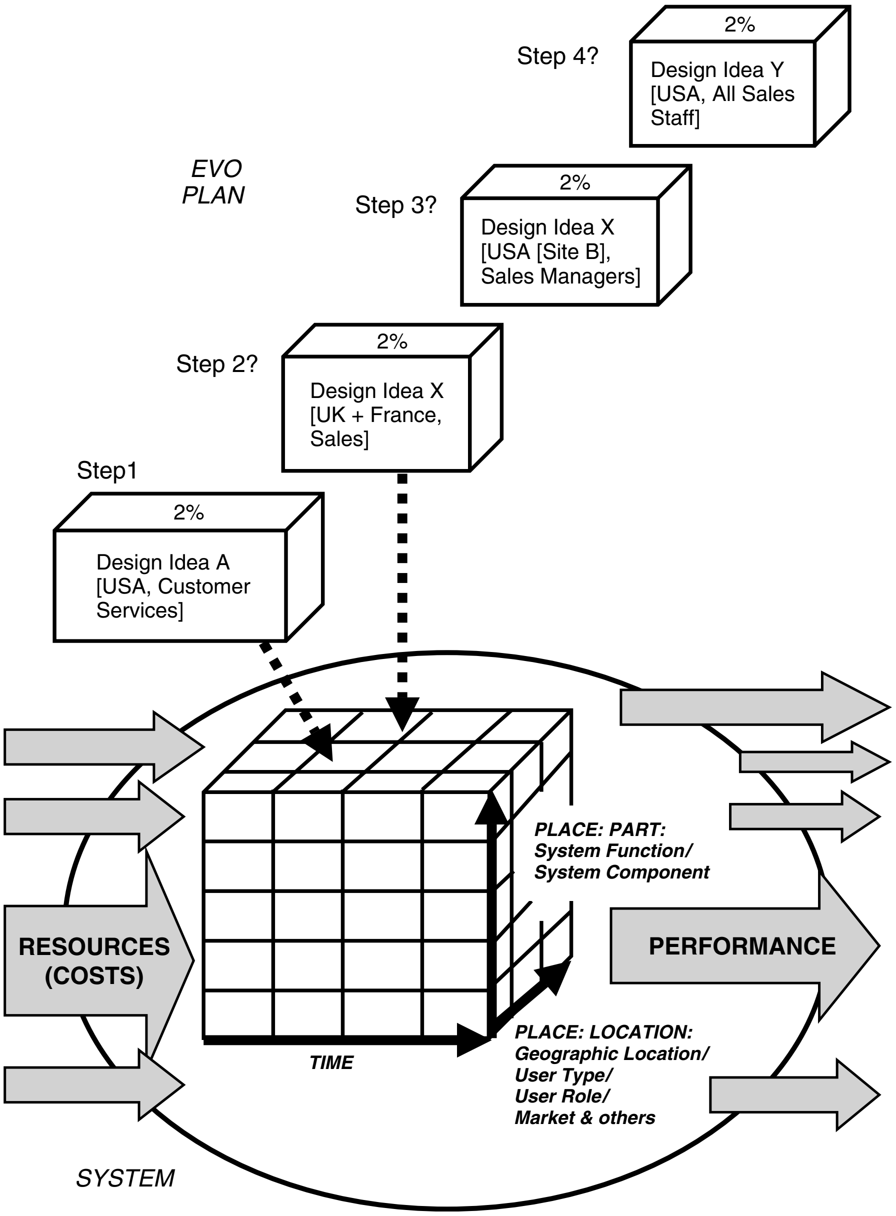
2%
Step 4?
Design Idea Y [USA, All Sales Staff]
EVO PLAN
2%
Step 3?
Design Idea X [USA [Site B], Sales Managers]
2%
Step 2?
Design Idea X [UK + France, Sales]
Step1
2%
Design Idea A [USA, Customer Services]
PLACE: PART: System Function/ System Component
RESOURCES (COSTS)
PERFORMANCE
PLACE: LOCATION: Geographic Location/ User Type/ User Role/ Market & others
TIME
SYSTEM

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

10.6 Principles: Evolutionary Project Management

1. **The Principle of 'Capablanca's next move'**
   There is only one move that really counts, the next one.
2. **The Principle of 'Do the juicy bits first'**
   Do whatever gives the biggest gains. Don't let the other stuff distract you!
3. **The Principle of 'Better the devil you know'**
   Successful visionaries start from where they *are*, what they *have* and what their *customers* have.
4. **The Principle of 'You eat an elephant one bite at a time'**
   System stakeholders need to digest new systems in small increments.
5. **The Principle of 'Cause and Effect'**
   If you change in small stages, the causes of effects are clearer and easier to correct.
6. **The Principle of 'The early bird catches the worm'**
   Your customers will be happier with an early long-term stream of their priority improvements, than years of promises, culminating in late disaster.
7. **The Principle of 'Strike early, while the iron is still hot'**
   Install small steps quickly with people who are most interested and motivated.
8. **The Principle of 'A bird in the hand is worth two in the bush'**
   Your next step should give the best result you can get now.
9. **The Principle of 'No plan survives first contact with the enemy'**[2]
   A little practical experience beats a lot of committee meetings.
10. **The Principle of 'Adaptive Architecture'**
    Since you cannot be sure where or when you are going, your first priority is to equip yourself to go almost anywhere, anytime.

[2] This saying is attributed to Prussian general staff and the elder Von Moltke: "They did not expect a plan of operations to survive beyond the first contact with the enemy. They set only the broadest of objectives and emphasized seizing unforeseen opportunities as they arose . . . Strategy was not a lengthy action plan. It was the evolution of a central idea through continually changing circumstances" (From Von Clausewitz in his 'On War', quoted by General Electric's CEO, Jack Welch in a speech December 8, 1981, in Slater, 2000: 194).

//INTEGRA/ELS/PAGINATION/ELSEVIER UK/OMP/3B2/FINALS/0750665076-CH010.3D – 291 – [291–320/30] 29.6.2005
12:44PM

The Principles of Tao Teh Ching (500 BC)

That which remains quiet, is easy to handle.
That which is not yet developed is easy to manage.
That which is weak is easy to control.
That which is still small is easy to direct.
Deal with little troubles before they become big.
Attend to little problems before they get out of hand.
For the largest tree was once a sprout, the tallest tower started with the first brick, and the longest journey started with the first step.[3]

10.7 Additional Ideas: Evolutionary Project Management

Backroom/Frontroom

Some step components will inevitably have a longer development and/ or production elapsed time. This can be due to a variety of reasons, for example, lead time for purchasing. In such cases, 'backroom' activities will have to be underway well before the decision about which step to deliver next is made. There will have to be parallel step component development and production cycles.

The stakeholder in the frontroom, who receives the delivery step, is unaware of the backroom work. A useful analogy is a restaurant kitchen (backroom) and the customers in the restaurant (frontroom): if all goes well, the food is delivered at frequent intervals and all the preparation, cooking time and co-ordination of dishes for a specific table are invisible to the customers.

In fact, a skillful project manager will probably aim to have more than one potential delivery 'stockpiled,' so that there is choice over the next delivery, and leeway if any major problem effects step development.

Using IE Tables for Evo Plans

The steps of an Evo plan can be analyzed using an IE table. See Table 10.1. In this case steps (which consist of design ideas) are specified. Steps are estimated and (after deployment) measured for impact on requirements (performance and resource attributes), in relation to targets.

[3] From Lao Tzu, in Bahn (1980).

Table 10.1 This is a conceptual example. Three goals (performance targets) and two resource targets are having the real impacts on them tracked, as steps are delivered. The same IE table is also being used to specify the impact estimates for the future planned steps. So at each step, the project can learn from the reality of the step's deviation from its estimates. Plans and estimates can then be adjusted and improved from an early stage of the project.

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

An Evo Step Specification

Evo Step: Tutorial [Model 1234, Basic].

Stakeholders: {Marketing, Department XX}.
Implementers: Department XX.
Intended Audience: Marketing.

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

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

Notes [HCTD12]:
- Can write the basic minimal functions, MMM, in 1 week. <-GF.
- Provide step by step instructions, in English.
- Questionnaire for Stakeholders.
- Intended audience: Marketing.
- Focus on <sales aspects>, not how to identify information in detail (not yet, in this step).
- Go to <specific web sites>.
- Process for Testing with Stakeholder (for example, observation, times).
- Pinpoint some characteristics of what we see on the terminal compared with what we see on a <PC or other terminal>.
- What instructions should be on the terminal to begin?
- No illustrations to be provided, just text.

Questionnaire: Defined As: Questionnaire to walkthrough with stakeholders.

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

Constraint: Step must be deliverable within one calendar week.

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

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

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

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

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

Figure 10.5
An example of using the specification template for an Evo step.

Notes:
1. *New user-defined types of 'Questionnaire' and 'Intended Audience' have been locally declared.*
2. *There is a brainstorming and note-taking atmosphere here. Do not expect to understand the internal language of my client in an isolated first draft teamwork example*!
3. *This illustration is mainly given to show an example of a set of parameters describing the attributes of a step. You should feel free to design your own set of useful step specification parameters.*

Using Templates for Specifying Evo Steps

Figure 10.5 shows an example of a filled-in template for specifying an Evo step (*see also Figure 10.8 in Section 10.9 for an outline template*).

How to decompose systems into small evolutionary steps: (a list of practical tips)

1. Believe there is a way to do it, you just have not found it yet![4]
2. Identify obstacles, but don't use them as excuses: use your imagination to get rid of them!
3. Focus on some usefulness for the stakeholders: users, salesperson, installer, testers or customer. However small the positive contribution, something is better than nothing.
4. Do *not* focus on the design ideas themselves, they are distracting, especially for small initial cycles. Sometimes you have to ignore them entirely in the short term!
5. Think one stakeholder. Think 'tomorrow' or 'next week.' Think of one interesting improvement.
6. Focus on the results. (You should have them defined in your targets. Focus on moving *towards* the goal and budget levels.)
7. Don't be afraid to use temporary-scaffolding designs. Their cost must be seen in the light of the value of making some progress, and getting practical experience.
8. Don't be worried that your design is inelegant; it is results that count, not style.
9. Don't be afraid that the stakeholders won't like it. If you are focusing on the results they want, then by definition, they should like it. If you are not, then do!
10. Don't get so worried about "what might happen afterwards" that you can make no practical progress.
11. You cannot foresee everything. Don't even think about it!
12. If you focus on helping your stakeholder in practice, now, where they really need it, you will be forgiven a lot of 'sins'!
13. You can understand things much better, by getting some practical experience (and removing some of your fears).
14. Do early cycles, on *willing local mature* parts of your user/stakeholder community.
15. When some cycles, like a purchase-order cycle, take a long time, initiate them early (in the 'Backroom'), and do other useful cycles while you wait.
16. If something seems to need to wait for 'the big new system', ask if you cannot usefully do it with the 'awful old system', so as to pilot it realistically, and perhaps alleviate some 'pain' in the old system.
17. If something seems too costly to buy, for limited initial use, see if you can negotiate some kind of 'pay as you really use' contract. Most suppliers would like to do this to get your patronage, and to avoid competitors making the same deal.
18. If you can't think of some useful small cycles, then talk directly with the real 'customer', stakeholders, or end user. They probably have dozens of suggestions.
19. Talk with end users and other stakeholders in any case, they have insights you need.
20. Don't be afraid to use the old system and the old 'culture' as a launching platform for the radical new system. There is a lot of merit in this, and many people overlook it.

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

Figure 10.6
Ideas to assist identifying steps.

10.8 Further Example/Case Study: The German Telecommunications Company

Here is an example of another perceived barrier preventing use of the evolutionary method: "It is too late, we have already invested so much in the old way, that we just have to see it through."

At a large German telecommunications business, almost 1,000 software engineers had been working for three years on a major new world-market product. The hardware was ready, but the software was late. I was told by the Financial Director for the project that the next month, December, was the actual deadline for product delivery, but that their 40,000 node PERT chart (really!) estimated they had two or three years more software effort left. Corporate marketing management had given them one more year, until December next year. They had to deliver, or forget the whole market, which by that time would be taken over by competitors.

I suggested re-planning the project into smaller steps with critical increments first. They told me that this was unthinkable: the software was 'already written' and they claimed that only testing remained. They also had a rather long list of other reasons why evolutionary delivery would not work for them.

Using common sense, we worked out a basic evolutionary plan: we used a day to plan and a second day to sell the idea. We decided we ought to aim to deliver the small-system software first (there were 35 signed contracts for it and, none for the medium and large systems). Then we decided to select for Evo delivery the fundamental telephone services before any advanced complex stuff.

After moving through what seemed like seven management layers (there were probably only four) with "You must present this to my boss," we ended up in the office of Herr R., the Project Director. He thought it was all good common sense, and stared coldly at his (cowardly, cautious?) subordinates as he asked: "Can you do it this way?" When they gave assenting nods, he merely said, "Then do it!"

They did! On a return visit in the November of the next year, they told me that the small systems had been operating for over six weeks with several real customers and with no problems whatsoever. Note, three months *before* the impossible deadline!

The lessons to learn include:

1. You must *look* for Evo steps, don't assume that others have done so,
2. Evo steps are usually clear, simple and found by using product knowledge and stakeholder requirements, and
3. You have to get the right person to make the decision to 'go' with Evo, usually a senior manager, who is focused on delivering business benefit.

The product remained a major successful product on the market for many years. Herr R. correctly concluded that unless the organization changed its mode of thinking, the same type of project problem would continue to recur. So he took steps to improve the organization. Prevention is better than cure!

10.9 Diagrams/Icons: Evolutionary Project Management

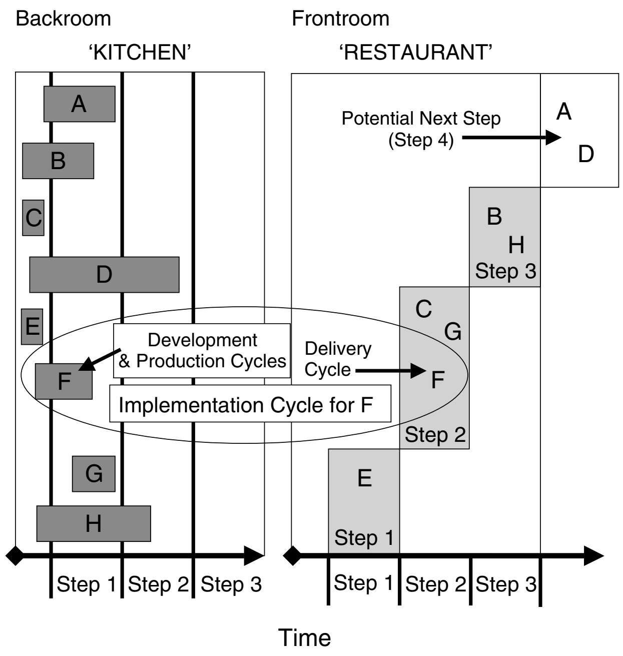

Figure 10.7
Backroom and frontroom activities: diagram showing the relationship between backroom and frontroom activities. The step components are developed and assembled in the backroom and then delivered in steps to the frontroom. A frequency of step delivery is maintained. Step 4 is actually ready ahead of its delivery time and is held back. Note, given when the system components were ready for delivery, there were several choices about the delivered step content. The step time lines in the backroom show when the corresponding frontroom steps were done. For example, G but not H, was complete by the beginning of Step 2, and G was therefore available for delivery if we decided to do that.

A Template For EVO Step Specification

Tag: <Tag name for the step>.
Type: Evo Step.

=========================== Basic Information ==========================

Version: <Date or version of last update to step specification>.
Status: <{Specification Stage [{Draft, SQC Exited, Approved}], In Evo Plan, Scheduled Next, Under Implementation, Delivered awaiting Feedback, Feedback Obtained}, date> <- <Source (who says 'Status' is true?)>.
Quality Level: <Maximum remaining major defects/page, sample size, SQC date>.
Owner: <Who is taking responsibility for the step in terms of specification>.
Stakeholders: <Who are you going to deliver requirements to? >.
Implementers: <Who is in charge of implementing this step>.
Gist: <Brief description of the main idea of this step>.
Description: <Give a detailed, unambiguous description of the step, or a tag reference to a place where it is described. Remember to include definitions of any local terms>.

Implementation Details: "Includes relevant details, such as <which product>, <which area of application system>."
Evo Plan: <Tag of the Evo Plan that this step is associated with>.
Step Content: <Step Elements: {Design Ideas, Functions, Tasks, re-used step definitions}>.

============================= Measurement ===========================

Test: <Refer to tags of any test plan and/or test cases, which apply to this step>.
Step Validation/Feedback:
Specification Quality Control (SQC): <outcome, date>,
Pre-Delivery Test: <outcome, date>,
Post Delivery Results: <{problems, stakeholder feedback}, date>,
Certification Specification: <refer to the certification plans>.

======================= Priority and Risk Management ======================

Constraints:
<Any legal, political, economic, security or other constraints imposed on implementation>
<- <Source (who says this is true?)>.
Assumptions: <Any assumptions that have been made>.
Dependencies:
<Anything which must be in place, finished, working properly, for us to be able to start this Evo step or to complete it> <- <Source (who says this is true?)>.
Risks: <Any risks that need to be taken into account>.
Priority:
<Name, using tags, any system elements, which must clearly be done *after* or must clearly be done *before*. Give any relevant reasons>.
Issues: <Unresolved concerns or problems in the step specification or the system>.

========================== Benefits and Costs =========================

Rationale: <Justify the existence of this step>.
Step Value:
<Real measurements or estimates of numeric value to stakeholders>. "Value in terms of meeting the requirements. At least, the value on scale 0 (none) to 9 (highest)."
<- <Source (who says this is true?)>.
Step Cost:
<Budgets or real costs>. "For example, financial costs and engineering hours. These must be constrained by the Evo 2% policy. At least, the value on scale 0 (very cheap) to 9 (high and unpredictable)." <- <Source (who says this is true?)>.

Figure 10.8
A possible specification template for a one-page Evo step. Notice that the parameters are designed to give you enough information to decide on the order for step sequencing in an Evo plan.

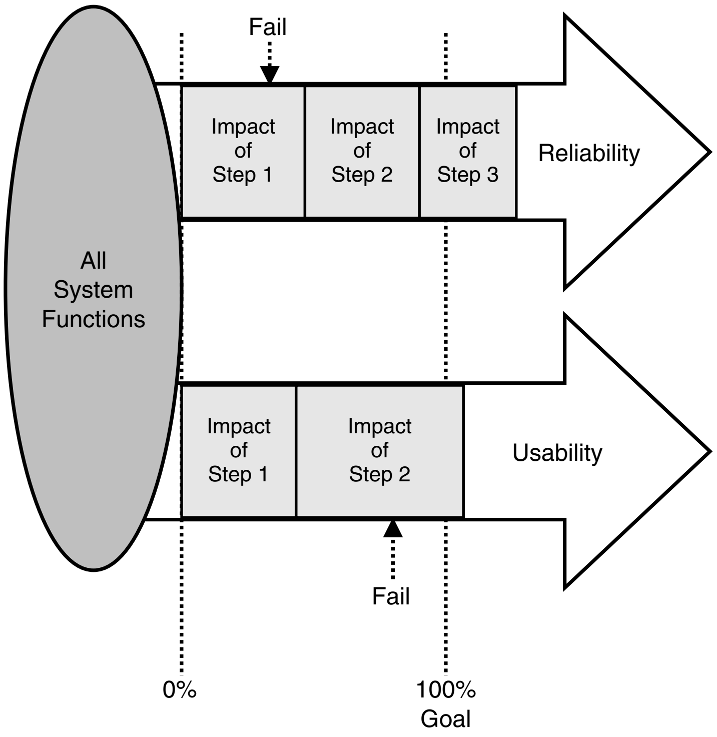

Figure 10.9
Dynamic priority: 0% is the baseline level – before we start further evolution of the system. After Step 1, usability has priority because it is not at an acceptable level (that is, not better than the specified fail level) yet. Reliability is above the fail level (and thus in the 'acceptable area'). After Step 2, reliability now has priority because it has not reached goal level yet. After Step 3, both reliability and usability have reached 100% of their respective goal targets. Consequently the project is 'finished' – no more performance gaps.

10.10 Summary: Evolutionary Project Management

Dr. Deming had a charming understated way of expressing the outcome of a venture: "Survival is not compulsory." Sadly, far too many projects demonstrate the truth of this and, in the process, waste years and large sums of money, and deliver nothing except weakened economy and reputation. Professor Peter Morris in his book, *The Management of Projects*, identifies that none of the existing well-known project management methods really enable sufficiently good control of projects (Morris 1994). He also outlines that the way forward must incorporate evolutionary methods. In fact, Evo exists and already has a track record of success; it is just not widely known and practiced within the systems engineering and other engineering communities.

Hopefully, this chapter has taught the basic concepts of Evo:

- Evo is, above all, the application of the Shewhart process control cycle, 'Plan-Do-Study-Act'. It is learning from doing and acting on that learning. It is adapting to the complex and changing realities of a project. Evo is systematic engineering work (of the type described by Koen (Koen 1984).
- Evo is primarily guided by well defined, quantified, *but not necessarily static*, multiple requirements for performance and cost. These requirements represent, at least indirectly, the value system of the stakeholders. Deviation from the path to reaching the requirements is corrected with minimum loss of resource. We are always open to corrections of our requirements. Such corrections are caused either because the world has changed or because we better understand how to formulate our 'values' in terms of requirements (Keeney 1992).
- Evo is concerned with *controlling risks*. By insisting on small steps, you learn early about the project's capability to deliver, and about the users, other stakeholders and their system environment. You are in a position to adapt to what you learn; and also to incorporate any additional changes requested.
- Evo demands early delivery of the high priority improvements. This gains credibility for the project and should attract resources to continue to do so.

Don't be fooled by the term 'evolutionary' into thinking Evo means 'slow and small change'. If you want change, even revolutionary change, then Evo project management:

- will give you better results
- will get you faster to market
- will help you meet your deadlines.