

Version 3.0 Sept. 27 1998

"Risk Management : A practical toolkit for identifying, analyzing and coping with project risks"

**By Tom Gilb,
Senior Partner, Result Planning Limited
www.Result-Planning.com**

Paper Summary

Risk management must be fully integrated into all the development and maintenance processes for systems. It involves more than applying risk assessment methods to identify and evaluate system risks.

To explain this broad approach to risk management, this paper discusses the way in which Requirements Driven Management (RDM) methods contribute to handling risks.

Definition of ‘Risk’

Risk is an abstract concept expressing the possibility of unwanted outcomes.

**A ‘risk’ is anything which can lead to results
which deviate from the ‘real’ project Stakeholder requirements.**

It is in the nature of risk that the probability of risks actually occurring, and their actual impact when they do so, can only be predicted to varying degrees of accuracy. Not all risks can be identified in advance.

Risk Management is any activity which identifies risks, and takes action to remove, reduce or control ‘negative results’ (deviations from the requirements).

Principles of Risk Management

In my view, the fundamental principles of risk management include:

1. Quantify requirements

All critical quality and resource requirements must be identified and quantified numerically.

2. Maximize profit, not minimize risk

Focus on achieving the maximum benefits within budget and time-scales rather than on attempting to eliminate all risk.

3. Design out unacceptable risk

Unacceptable risk needs to be ‘designed out’ of the system consciously at all stages, at all levels in all areas, e.g. architecture, purchasing, contracting, development, maintenance and human factors.

4. Design in redundancy

When planning and implementing projects, conscious backup redundancy for outmaneuvering risks is a necessary cost.

5. Monitor reality

Early, frequent and measurable feedback from reality must be planned into your development and maintenance processes, to identify and assess risks before they become dangerous.

6. Reduce risk exposure

The total level of risk exposure at any one time should be consciously reduced to between 2% and 5% of total budget.

7. Communicate about risk

There must be no unexpected surprises. If people have followed guidelines and are open about what work they have done, then others have the opportunity to comment constructively. Where there are risks, then share the information.

8. Reuse what you learn about risk

Standards, rules and guidance must capture and assist good practice. Continuous process improvement is also needed.

9. Delegate personal responsibility for risk

People must be give personal responsibility in their sector for identification and mitigation of risks.

10. Contract out risk

Make vendors contractually responsible for risks, they will give you better advice and services as a result.

Let’s now consider, each of these principles in turn and describe some (not all!) of the roles that the RDM methods play in risk management. However, first here is an outline sketch of the RDM methods:

- **Planguage**; a requirements specification language insisting on quantified values.
- **Impact Estimation (IE)**; an analysis tool (a table) allowing evaluation of the likelihood of achieving requirements and, the evaluation and comparison of different designs (strategies). A strength of IE is that it also helps identify new designs and uncover previously unstated requirements.
- **Evolutionary Delivery (Evo)**; based on the work by the quality gurus Deming and Juran, a way of working that focuses on evolutionary delivery of early, measurable, system benefits to the customers. A system is developed, by small risk steps, in a series of plan, develop, deliver and evaluate cycles.
- **Inspection**; a technique for measuring and improving technical document quality. Technical documents are evaluated against their source documents and any prevailing standards by Inspection teams consisting of individuals with specially assigned roles. The overall aims are to identify defects, to identify patterns in the introduction of defects (leading to process improvement), to help train individuals to avoid creating defects and, to assist team-building.

Readers wanting a more detailed explanation of these methods should look in the References.

Principle 1. Quantify requirements

All critical quality and resource requirements must be identified and quantified numerically.

Risk is negative deviation from requirements. So, if we are going to understand risk, we must have some way of specifying exactly what we want. If we use vague ways like “State of the Art, World Class, Competitor-Beating Levels of Quality”, we cannot understand and assess risk.

Planguage helps because it demands numerically quantified requirements. Using Planguage, we must go through the following steps:

- Identify *all critical* quality and resource *attributes* of the system. In practice, this could be ten or more critical qualities (e.g. availability) and, five or more critical resources (e.g. operational costs).
- Define exactly how to *understand variation* in each attribute by specifying a scale of measure, e.g. ‘Scale: Probability of being fully operational during the office day’ and ‘Scale: Total of all monetary operational expenses including long term decommissioning costs’.
- For each attribute, define one or more critical points on the defined scale of measure which are needed for the system to function properly and profitably. There are two important categories: ‘Must’ and ‘Plan’. A ‘Must’ level defines the system survival level. A ‘Plan’ level defines the planned point for success. For risk management, ‘Must’ is the first level and ‘Plan’ is the second level for risk determination. A value for any attribute less than its required Must level means total system failure. Only when all Plan levels for all the attributes have been met can a system be declared a success.
- For all the Must and Plan levels, define additional qualifying information. We call this using ‘qualifiers’. You are basically defining time, place and event, i.e. when it is critical for you to achieve a certain level of an attribute, where it is critical and under what conditions. For example,

Plan [1999,Europe,IF European Monetary Union implemented anywhere] 99.98%

We can even give direct expression to the amount of risk we are prepared to take by a statement such as :

Must [2001, UK, IF Euro is used in Norway & UK] 60% \pm 20%

In other words the range of results 40% to 80% is an acceptable upper and lower limit, but below 40% is unacceptable.

Here is a more complete example:

Usability:
 Scale: Mean time to learn [defined tasks] to minimum proficiency.
 Must [Release 2.0, English Version, Task: Modifying Files] 10 minutes.
 Plan [Release 2.0, English Version, Task: Modifying Files] 7 minutes.
 Plan [Release 3.0, English Version, Task: Modifying Files] 5 minutes.
 Plan [Release 3.0, French & Dutch Versions, Task: Finding a File by Content] 5 minutes.

In the example, the most critical (failure of system) risk is the Must level. The other statements are only of secondary risk; they indicate the levels required to declare success.

It should be obvious that the degree of risk can be expressed in terms of the deviation from the target levels. For example,

Method A can sometimes result in a learning time of 10 minutes, while method B can never result in a learning time exceeding 4 minutes.

This means that for the specified requirements, method A poses a real risk, but method B does not.

A template specification of risk levels

In addition to the basic statements described above, it should be noted that there are a wide variety of ways within Planguage to indicate that the information contains some element of risk. Here are some examples:

Plan 60-80	Specification of a range
Plan 60 \pm 30	Specification of an upper and lower limit
Plan 60 \rightarrow 90	
Plan 60?	Expressing that the value is in doubt
Plan 60??	Expressing that the value is in serious doubt
Plan 60 \leftarrow A wild guess	Using the source of the information to show the doubt
Plan 60 \leftarrow A.N. Other	Depends on A.N. Other’s credibility in setting this value
Plan <60>	Fuzzy brackets indicate data needing improvement

All of the above signals can be used to warn of potential risk. Of course, the culture must encourage such specification rather than intimidate people from using it.

Plan [IF Euro is used in UK] 99%

The above is an example where the risk is controlled by making the specification totally dependent on the IF condition. There is no risk that anyone will plan to achieve 99% if the condition is false. However, they are warned to plan to achieve 99% should the condition turn true.

Note, you can also use IF qualifiers to constrain the use of a strategy (a means for achieving a goal). This reduces the risk that an expensive strategy is applied under inappropriate conditions.

Strategy99 [IF hunger famine in a country, IF road and rail transport unavailable] Aerial Supply of Food.

Principle 2. Maximize profit, not minimize risk

Focus on achieving the maximum benefits within budget and time-scales rather than on attempting to eliminate all risk.

Elimination of all risk is not practical, not necessary and, not even desirable. All risk has to be controlled and balanced against the potential benefits. In some cases, it is appropriate to decide to use (and manage) a strategy with higher benefits and higher risks. I use Impact Estimation (IE) to help me assess the set of strategies I need to ensure I meet the required objectives. My focus is always on achieving the objectives *in spite of* the risks.

Outline Description of Impact Estimation (IE)

The basic IE idea is simple: estimate quantitatively how much your design ideas impact all critical requirements. This is achieved by completing an IE table. The left-hand column of the table should contain the objectives and, across the top of the table should be the proposed strategies. For the objectives, assuming you have expressed them using Planguage, it is a question of listing down all the quality and resource attributes you wish to consider. You need next to decide on a future date you want to use. This should be a system ‘milestone’; a date for which you have specified Must and Plan levels. Then, against each attribute, you state the current level and the Plan level for your chosen date. (If you are especially risk averse you would use the Must level!) For the strategies, you simply list them across the top of the IE table.

You then fill in the table, for each cell you answer the question, ‘How does this strategy move the attribute from its current level towards the Plan level?’ First you state the actual value you would expect and then you convert this into a percentage of the amount of required change.

For example, Training Time for Task A is currently 15 minutes and you require it to be 10 minutes within six months. You estimate Strategy B will reduce Training Time for Task A to 12 minutes. In other words, Strategy B will get you 60% of the way to meeting your objective. See Table 1.

TABLE 1

	Strategy B	
	Real Impact	% Impact
Training Time Past = 15 minutes in June 1998 Plan = 10 minutes by end of Dec. 1998	12 minutes	60%
Resource = Development Budget Plan = \$2000 up to end Dec. 1998	\$1,000	50%

Further improvements to specifying the impacts

There are a number of improvements to this basic idea, which make it more communicative and credible. Here is a brief summary of them :

Uncertainty of Impact: you can specify a range of values rather than a single value.

Evidence for Impact Assertion: you can state the basis for making your estimate. For example: "Strategy B was used for 5 projects last year in our company, and the percentage improvement for Training Times was always 60% to 80%".

Source of Evidence for Impact Assertion: Of course, some skeptic might like to check your assertion and evidence out, so you should give them a source reference, e.g. "Company Research Report ABR-017, pages 23-24."

Credibility Rating of the Impact Assertion: We have found it very useful to establish a numeric 'credibility' for an estimate, based on the credibility of the evidence and the source. We use a scale of 0.0 to 1.0 (because it can then be used later to modify estimates in a conservative direction). See Table 2.

TABLE 2

Credibility Rating	Meaning
0.0	wild guess, no credibility
0.1	we know it has been done somewhere
0.2	we have one measurement somewhere
0.3	there are several measurements in the estimated range
0.4	the measurements are relevant to our case
0.5	the method of measurement is considered reliable
0.6	we have used the method in-house
0.7	we have reliable measurements in-house
0.8	reliable in-house measurements correlate to independent external measurements
0.9	we have used the idea on this project and measured it
1.0	perfect credibility, we have rock solid, contract-guaranteed, long-term, credible experience with this idea on this project and, the results are unlikely to disappear

Further Analysis of the IE data

Once you have completed filling in all the impacts, there are a number of calculations, using the percentage impact estimates (%Impact), that help you understand the risks involved with your proposed solution.

Let me stress that these are only rough, practical calculations. Adding impacts of different independent estimates for different strategies, which are part of the same overall architecture, is dubious in terms of accuracy. But, as long as this is understood, you will find them very powerful when considering such matters as whether a specific quality goal is likely to be met or which is the most effective strategy. The insights gained are frequently of use in generating new strategies.

Impact on a Quality: For each individual quality or resource attribute, sum all the percentage impacts for the different strategies. This gives us an understanding of whether we are likely to make the planned level for each quality or cost. Very small quality impact sums like '4%' indicate high risk that the architecture is probably not capable of meeting the goals. Large numbers like 400% indicate that we might have enough design, or even a 'safety margin'.

TABLE 3

Example: Adding the percentage impacts for a set of strategies on a single quality or cost can give some impression of how the strategies are contributing overall to the objectives. Note Strategies A, B and C are independent and complementary.

	Strategy A	Strategy B	Strategy C	Sum of Strategy Impacts	Sum Uncertainty
Reliability 900->1000 hours MTBF	0+/-10%	10+/-20%	50+/-40%	60%	+/-70%

Impact of a Strategy: For each individual strategy, sum all the percentage impacts it achieves across all the qualities to get an estimate of its overall effectiveness in delivering the qualities. The resulting estimates can be used to help select amongst the strategies. It is a case of selecting the strategy with the highest estimate value and the best fit across all the critical quality requirements. If the design ideas are complementary then the aim is to choose which strategies to implement first. If the strategies are alternatives, then you are simply looking to determine which one to pick.

TABLE 4

A measure of the effectiveness of strategy ‘Big Idea’ can be found by adding together its percentage impacts across all the qualities

QUALITY	PAST-PLAN	Big Idea	
Reliability	900->1,000 hours MTBF	50% +/-10%	

Maintainability	10 min. fix to 5 min. to fix.	100%+/-50%	
		150%+/-60%	Estimate of total effect of Big Idea on all goals

In addition to looking at the effectiveness of the individual strategies in impacting the qualities, the cost of the individual strategies also needs to be considered, see next section.

Quality to Cost Ratio: For each individual strategy, calculate the quality-to-cost ratio (also known as the benefit-to-cost ratio). For quality, use the estimate calculated in the previous section. For cost, use the percentage drain on the overall budget of the strategy or use the actual cost.

The overall cost figure used should take into account both the cost of developing or acquiring the strategy and, the cost of operationally running the strategy over the chosen time scale. Sometimes, specific aspects of resource utilization also need to be taken into account. For example, maybe staff utilization is a critical factor and therefore a strategy that doesn't utilize scarce programming skills becomes much more attractive.

My experience is that comparison of the 'bang for the buck' of strategies often wakes people up dramatically to ideas they have previously under- or over-valued.

Average Credibility / Risk Analysis: Once we have all the credibility data (i.e. the credibility's for all the estimates of the impacts of all the strategies on all the qualities), we can calculate the average credibility of each strategy and, the average credibility of achieving each quality. This information is very powerful, because it helps us understand the risk involved. For example, "the average credibility, quality controlled, for this alternative strategy is 0.8". Sounds good! This approach also saves executive meeting time for those who hold the purse strings.

Principle 3. Design out unacceptable risk

Unacceptable risk needs to be ‘designed out’ of the system consciously at all stages, at all levels in all areas, e.g. architecture, purchasing, contracting, development, maintenance and human factors.

Once you have the completed initial IE table, you are in a position to identify the unacceptable risks and design them out of the system. Unacceptable risks include:

- Any quality or resource attribute where the sum of the %Impacts of all the proposed strategies does not total 200%. (A 100% safety factor has been assumed to reduce the risk of failure.)
- Any strategy providing i) a low total for the sum of its %Impacts, ii) very low credibility or iii) low benefit-to-cost ratio.

New strategies will have to be found that reduce these risks. In some cases, it may be decided that the levels set for the objectives are unrealistic and they may be modified instead.

Within software engineering, the art of designing a system to meet multiple quality and cost targets, is almost unknown [GILB88]. However, I have no doubt that there is great potential in conscious design to reduce risks. For example, it is a hallowed engineering principle to be conservative and use known technology. However, this concept has not quite caught on in software engineering technology, where ‘new is good’, even if we do not know much about its risks. At least, with the use of an IE table there is a chance of expressing and comparing the risk involved in following the differing strategies.

Principle 4. Design in redundancy

When planning and implementing projects, conscious backup redundancy for outmaneuvering risks is a necessary cost.

Under Principle 3, we have discussed finding new strategies. Principle 4, takes this idea a step further. Actively look for strategies that provide backup. An extreme example of this practice is NASA’s use of backup computer systems for manned space missions.

Principle 5. Monitor reality

Early, frequent and measurable feedback from reality must be planned into your development and maintenance processes to identify and assess risks before they become dangerous.

I expect the IE information only be used as an initial, rough indicator to help designers spot potential problems or select strategies. Any real estimation of the impact of many strategies needs to be made by real tests (Ideally, by measuring the results of early evolutionary steps in the field). Evolutionary Delivery (Evo) is the method to use to achieve this (See next Principal).

Principle 6. Reduce risk exposure

The total level of risk exposure at any one time should be consciously reduced to between 2% and 5% of total budget.

IE can also be used to support Evolutionary Delivery (Evo) as a budgeting and feedback mechanism during project building and installation of partial deliveries [GILB98, MAY96].

The Evolutionary Delivery (Evo) method typically means that live systems are delivered step by step to user communities for trial often (e.g. weekly) and early (e.g. 2nd week of project).

One of the major objectives of Evo is to reduce and control risk of deviation from plans. This is achieved by:

- getting realistic feedback after small investments
- allowing for change in requirements and designs as we learn during the project
- investing minimum amounts at any one time (2% to 5% of project time or money) so that total loss is limited if a delivery step totally fails.

IE is of use in helping to plan the sequencing of Evo steps. IE tables also provide a suitable format for presenting the results of Evo steps. See Table 5.

TABLE 5

Step-> Attribute	STEP1 plan %	actual %	Devia- tion %	STEP2 to STEP20 plan	plan cumul- ated to here	STEP21 [CA,NV,WA] plan	plan cumul- ated to here	STEP22 [all others] plan	plan cumul- ated to here
QUAL-1	5	3	-2	40	43	40	83	-20	63
QUAL-2	10	12	+2	50	62	30	92	60	152
QUAL-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

Table 5 is a hypothetical example of how an evolutionary project can be planned and controlled and risks understood. The ‘deviation’ between what you planned and what you actually measured in practice is a good indicator of risk. The larger the deviation, the less you were able to correctly predict about even a small step. Consequently there is a direct measure of areas of risk in the ‘deviation’ numbers.

The beauty of this, compared to conventional risk estimation methods [HALL98] is that it:

- is based on *real* systems and *real* users (not estimates and speculation *before* practical experience)
- is *early* in the *investment* process
- is based on the results of *small* system increments, and the *cause* of the risk is easier to spot, and perhaps to eliminate, or to modify, so as to avoid the risk.

Evolutionary Project management does not ask what the risks *might* be. It asks what risks have shown up in practice. But it does so at such an *early* stage, that we have a fair chance to do something about the problems.

Principle 7. Communicate about risk

There must be no unexpected surprises. If people have followed guidelines and are open about what work they have done, then others have the opportunity to comment constructively. Where there are risks, then share the information.

Hopefully, readers will by now have begun to understand that Planguage and IE are good means of communicating risk. Let me now introduce Inspection as a third useful method.

Inspection is a direct weapon for risk reduction. [GILB93]. Early Inspections on all written specifications is a powerful way to measure, identify and reduce risk of bad plans becoming bad investments. The key idea is that Major defects are measured, removed, and that people learn to avoid them, by getting detailed feedback from colleagues. A *defect* is a violation of a ‘best practice’ rule. A *Major* defect is defined as a defect which can have substantial economic effect ‘downstream’ (in practice, in ‘test’ phases and in the field). By this definition, a Major defect is a ‘*risk*’. So Inspection measures risks!

Many people think that the main benefit from Inspection is in identifying and removing Major defects early (e.g. before source code reaches test phases). This is not the case. (My experience is that Inspection is as bad as testing in % defect-removal effectiveness. In very rough terms half of every defect present is not identified or removed.) The really important economic effect of Inspection is not what happens at the level of a single document, but in teaching the people and the organization. The real effect of Inspection is in:

- • teaching individual engineers exactly how often they violate best practice rules
- • motivating the engineers to take rules seriously (really avoid injecting Major defects)
- • regulating flow of documentation, so that high Major defect documents can neither exit nor enter adjacent work processes.

Staff involved in Inspections learn very quickly how to stop injecting defects. Typically, the defects introduced by an author reduce at the rate of about 50% every time a new document is written and Inspected. For example, using Inspection, Raytheon reduced ‘rework’ costs, as a % of development costs, from 43% to 5% in an eight year period [DION95].

Sampling

One other little-appreciated aspect of Inspection is that you can use it by *sampling* a small section of a large document, rather than trying to ‘clean up’ the entire document. If the sample shows a high Major defect density (say more than one Major/Page) then the document is probably ‘polluted’ and action can be taken to analyze the defect sources. A complete rewrite may be necessary using appropriate specification rules or new/improved source documents. This is generally cheaper than trying to clean up the entire document using defect removal Inspection or testing.

Principle 8. Reuse what you learn about risk

*Standards, rules and guidance must capture and assist good practice.
Continuous process improvement is also needed.*

In the previous section, the importance of Inspection was discussed and rules were highlighted as one of the essentials required to support it. It is worth emphasizing the aspect of reuse that is occurring here. The more effort that is put into making rules more effective and efficient by incorporating feedback from Inspections, the more productive the Inspections and the greater the reduction in risk.

Even more benefit can be achieved if what is learnt from Inspection is used to modify the processes that are causing the defects; Continuous Process Improvement has been shown to have a major influence on risk. For example, Raytheon has achieved zero deviation from plans and budgets over several years. They used a \$1million/year (for 1,000 software engineers) for 8 years to do continuous software process improvement. They report that the return on this investment was \$7.70 per \$1 invested on improving processes such as requirements, testing and Inspection itself. Their software defect rate went down by a factor of three [DION95].

Using Inspection, analysis of the identified defects to find process improvements is carried out in the Defect Prevention Process (DPP). DPP was developed from 1983 at IBM by Robert Mays and Carole Jones and, is today recognized as the basis for SEI CMM Level Five. The breakthrough in getting DPP to work, compared to earlier failed efforts within IBM, was probably in the decentralization of analysis activity to many smaller groups, rather than one ‘Lab Wide’ effort by a Quality Manager. This follows what the quality Guru Dr. W Edwards Deming taught the Japanese; factory workers must analyze their own statistics and be empowered to improve their own work processes.

Analysis of ‘root causes’ of defects is very much a risk analysis effort [HALL98] and a handful of my clients are reporting success at doing so. But, most are still working on other disciplines like Inspection and others elsewhere in this paper.

Principle 9. Delegate personal responsibility for risk

People must be give personal responsibility in their sector for identification and mitigation of risks.

To back up communicating about risk, people must be given ownership of the risks in their sector (e.g. allocating ownership/sign off of IE tables and giving people specific roles within Inspections).

Principle 10. Contract out risk

Make vendors contractually responsible for risks, they will give you better advice and services as a result.

I would like to point out that contracting for products and services gives great opportunity to legally and financially control risks by squarely putting them on someone else’s shoulders.

The effect of contracting a risk to someone else is that:

- you have gotten rid of the risk in some senses, but if they fail, you will still be affected!
- the supplier (assuming they get the risk) will be more motivated to take steps to eliminate the risks,
- be motivated to tell you exactly what you have to do to avoid being hit by risks,
- might come up with a more realistic bid and time plan to cope with the risks.

Summary

Risks can be handled in many ways and at many levels. I have tried to point out some risk management methods which are not so well known or well treated in existing literature. Hopefully, the need to fully integrate risk management into all the development and maintenance processes is clear.

Table 6 and Table 7 recap the ideas presented in this paper. Table 6 is a set of policies for risk management [See GILB98 for more detail]. Table 7 contains ‘Twelve Tough Questions’ to ask when assessing risk.

TABLE 6: Policy Ideas for Risk Management

- **EXPLICIT RISK SPECIFICATION**

All managers/planners/engineers/testers/quality assurance people shall immediately in writing, integrated in the main plan, specify any uncertainty, and any special conditions which can imaginably lead to a risk of deviation from defined target levels of system performance.

- **NUMERIC EXPECTATION SPECIFICATION**

The expected levels of all quality and cost attributes of the system shall be specified in a numeric way, using defined scales of measure, and at least an outline of one or more appropriate ‘Meters’ (test or measuring instruments for determining where we are on a scale).

- **CONDITIONS SPECIFIED**

The requirements levels shall be qualified with regard to when where and under which conditions the targets apply, so there is no risk of us inadvertently applying them inappropriately.

- **COMPLETE REQUIREMENT SPECIFICATION**

A *complete* set of all critical quality and cost aspects shall be specified, avoiding the risk of failing to consider a single critical attribute.

- **COMPLETE DESIGN SPECIFICATION and IMPACT ESTIMATION**

A complete set of designs or strategies for meeting the complete set of quality and cost targets will be specified. They will be validated against all specified quality and cost targets (using Impact Estimation Tables). They will meet a reasonable level of safety margin. They will then be evolutionarily validated in practice before major investment is made. The Evo steps will be made at a rate of maximum 2% of budget, and 2% of ‘project time’, per ‘incremental trial’ (Evo step) of designs or strategies.

- **SPECIFICATION QUALITY CONTROL NUMERICALLY EXITED**

All requirements, design, impact estimation and Evolutionary project plans, as well as all other related critical documents such as contracts, management plans, contract modifications, marketing plans, shall be ‘quality controlled’ using the Inspection method [GILB93]. A normal process Exit level shall be *that ‘no more than 0.2 Major Defects per page maximum, can be calculated to remain, as a function of those found and fixed before release, when checking is done properly’* (e.g. at optimum checking rates of 1 logical page or less per hour).

7. EVOLUTIONARY PROOF-OF-CONCEPT PRIORITIES

The Evolutionary Project Management method [GILB98, GILB88] will be used to sense and control risk in mid-project. The dominant paradigms will be

- 2% steps,
- high value to cost with regard to risk delivered first.
- high risk strategies tested ‘offline to customer delivery’, in the Backroom of development process, or at cost-to-vendor, or with ‘research funds’ as opposed to project budget.

TABLE 7: Twelve Tough Questions

1.	Why isn't the improvement quantified?
2.	What is degree of the risk or uncertainty and why?
3.	Are you sure? If not, why not?
4.	Where did you get that from? How can I check it out?
5.	How does your idea affect my goals, measurably?
6.	Did we forget anything critical to survival?
7.	How do you know it works that way? Did it before?
8.	Have we got a complete solution? Are all objectives satisfied?
9.	Are we planning to do the 'profitable things' first?
10.	Who is responsible for failure or success?
11.	How can we be sure the plan is working, during the project, early?
12.	Is it 'no cure, no pay' in a contract? Why not?

References

DION93: Raymond Dion, "Process Improvement and the Corporate Balance Sheet", IEEE Software, July 1993, Pages 28-35.

DION95: Raymond Dion, Tom Haley, Blake Ireland and Ed Wojtaszek of Raytheon Electronic Systems, “The Raytheon Report: Raytheon Electronic Systems Experience in Software Process Improvement”, November 1995, SEI web-site, <http://www.sei.cmu.edu/products/publications/95.reports/95.tr.017.html/>. This is an important update of earlier reports.

GILB88: Tom Gilb, “Principles of Software Engineering Management”, Addison-Wesley, 1988, 442 pages. ISBN 0-201-19246-2. See particularly Chapter 6, Estimating the Risk (reproduced in Boehm, Software Risk Management, IEEE CS Press, 1989 page 53).

GILB93: Tom Gilb and Dorothy Graham, “Software Inspection”, Addison-Wesley, 1993, ISBN 0-201-63181-4, 5TH printing 1998, 471 pages. This book covers the Defect Detection Process and the Defect Prevention Process, as well as giving sample Rules to check by, defined processes and a well defined set of Glossary terms to aid quantification and comparison. It is a next-generation Inspection, with hundreds of larger and smaller improvements over initial Inspection practices.

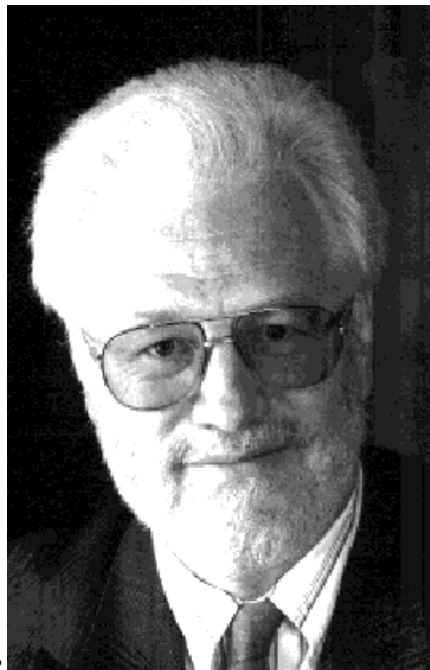
GILB98: Tom Gilb, Various papers and manuscripts on

<http://www.Result-Planning.com/>. The manuscripts include:

- . ‘Requirements-Driven Management using Planguage’ (1995-6)
- . ‘Evolutionary Project Management’ (1997)
- . ‘Requirements Engineering Language’ (1998).

HALL98: Elaine M. Hall, “Managing Risk: Methods for Software Systems Development. SEI Series in Software Engineering”, Addison Wesley Longman, USA. (enq.orders@awl.co.uk) , £31.95, 1998, ISBN 0-201-25592-8, 374 pages. This book is impressive and contains a lot of useful detail and original thought. Anyone interested in risk will enjoy and learn from the book as I did. It does not however deal with most of the subjects in this paper {specification languages, impact estimation, inspections, evolutionary delivery}. This in no way detracts from the book’s favorable recommendation. It does tackle ‘quantified objectives’ much better than other texts.

MAY96: Elaine L. May and Barbara A. Zimmer, “The Evolutionary Development Model for Software”, Hewlett-Packard Journal, August 1996, Vol. 47, No. 4, pages 39-45. *The author was at HP in 1989 on a project team who were taught early versions of the Planguage method. The article is full of practical advice and case studies gleaned from ten major projects in eight HP divisions. It must be strongly recommended to anyone interested in the practical implementation of Evo in a project and especially in an organization for many projects. See also article by Todd Cotton in same issue on Evo.* HP Journal subscription free to qualified individuals: Write Distribution Manager, HP Journal, M/S 20BH, 3000 Hanover Street, Palo Alto, CA, USA-94304, or Email: **hp_journal@hp_paloalto-gen13.om.hp.com**. HP Journal is available on World Wide Web at “**<http://www.hp.com/hpj/Journal.html>**”. (*Warning HP may have moved this site, but you can get to new site from here*)



Author Biography

Tom Gilb is the author of “Principles of Software Engineering Management” (1988) and “Software Inspection” (1993). His book “Software Metrics” (1976) coined the term and, was used as the basis for the Software Engineering Institute Capability Maturity Model Level Four [SEI CMM Level 4]. His most recent interests are development of true software engineering and systems engineering methods. His sons, Kai and Tor, now work with him.

Tom Gilb was born in Pasadena CA in 1940. He moved to England in 1956, then two years later he joined IBM in Norway. Since 1963, he has been an independent consultant and author.

This paper was edited by Lindsey Brodie, lindsey@brodie.source.co.uk

-----End of Paper-----