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

Tom Gilb
Tom@Gilb.com

Abstract. Risk management must be fully integrated into all the implementation (development, production and delivery) and operational 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 'Planguage' (a planning language and set of system engineering methods) contributes to handling risks.

DEFINITION OF 'RISK'

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

A 'risk' is anything that can lead to results that deviate negatively from the stakeholders' 'real'[1] requirements for a project.

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 that 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 performance and resource requirements must be identified and quantified numerically
2. Maximize profit, not minimize risk: Focus on achieving the maximum benefits within budget and timescales 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, it is a necessary cost to use conscious backup redundancy for outmaneuvering risks.
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, 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.

Now, let's consider, each of these principles in turn and describe some (but not all!) of the roles that the Planguage methods play in risk management.

However, first here is an outline sketch of the Planguage methods:

- **Planguage Specification**: Requirement, design and project management specification using the Planguage language, which insists on quantified values.
- **Impact Estimation (IE)**: An analysis tool (a table) allowing evaluation of the likelihood of achieving

[1] 'Real' requirements are all the requirements of all the stakeholder types, both internal and external to the system. Such requirements may, or may not, have been identified and specified. To check the validity of the requirements, several tactics should be used: reviewing requirements with stakeholders, Inspection of requirements, design analysis (Impact Estimation) and evolutionary delivery (See later discussion in this paper).

Only during the process of evolutionary delivery will all the 'real' requirements be identified. 'Real' requirements are likely to change during the course of a project in response to both the delivered results and, to changes in the business environment.

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)**: This method is based on the work by the quality gurus, Deming and Juran. It is 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.

• **Specification Quality Control (SQC) (also known as Inspection)**: A technique for measuring and improving technical document quality. Technical documents are evaluated against their source documents and any prevailing standards by 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.

Project Risk	Planguage Method
Project Fit to the Stakeholder	Requirement Specification / Qualifiers: Specific qualified parameter levels for each stakeholder and market. Goal [Market=UK]: 66%.
Customer Perception	Evo: Test perception through frequent, early result deliveries.
Political Influences	Requirement Specification: Use of Source, Authority, Rationale, Depends On parameters.
Organizational Stability	Standards: By capturing best practice in standards and demanding that work is carried out to them.
User Involvement	Requirement Specification and Evo: Capturing specific User requirements and involving user during early delivery.
Conflicting Goals	Requirement Specification: Capture detailed stakeholder information within requirements and carry out conflict resolution
Conflict over Resource Allocation	IE: Using IE to establish benefit-to-cost ratios for design ideas and carrying out engineering tradeoff.

Table 1: A List of some Project Risks and the selected Planguage Methods to use to tackle them.

Readers wanting a more detailed explanation of the Planguage methods should look at (Gilb 2002).

Principle 1: Quantify requirements: All critical performance 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 statements 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* performance[2] and resource *attributes* of the system. In practice, this could be ten or more critical performance attributes (e.g. availability) and, five or more critical resource attributes (e.g. operational costs). The critical attributes will vary from project to project and must be identified locally by each project.

• Define exactly how to understand *variation* in each attribute by specifying a scale of measure, e.g. 'Scale: Probability of being fully operational throughout the office day' and 'Scale: Total of all monetary operational expenses including long term decommissioning costs'.

• For each attribute, state one or more benchmark levels: 'Past' levels. Of special importance is the recent level. The risk involved in attaining any required level has to be assessed in relation to the level the system currently achieves.

• 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 three important categories: 'Survival', 'Fail' and 'Goal'. A 'Survival' level defines the system survival level. A 'Fail' level defines the system failure level. A 'Goal' level defines a formal success level. Only when all the specified Goal levels for all the attributes have been met for a defined system, can the system formally be declared a total success.

For risk management, there are several performance and cost parameters which have different priority:

First two constraint parameters – which have higher priority than target parameters,

• *Survival* levels define the minimum and maximum

[2] Performance consists of quality, workload capacity and resource savings attributes.

required levels for system survival. A value for any performance attribute outside its Survival levels means (formally) total system failure, or possibly contractual failure

• *Fail* levels define the levels at which some types and degrees of associated system fault are expected to occur. For example, failure conditions can describe *safety problems, operator discomfort, customer discomfort, loss of value, loss of market share etc.*

Then *target* parameters,

• *Goal* levels define the levels for success

• *Stretch* levels define desired, but not promised levels. Their prime motivation is to encourage and challenge the system designers to attempt to go beyond the Goal levels

• *Wish* levels state stakeholder requirements that are not yet budgeted. They are independent of considerations of cost or technical feasibility. Wish levels are captured so that they will not be forgotten if at some future date a possibility opens up to achieve them.

• For all levels, define additional qualifying information. We call this using 'qualifiers'. You are basically defining time, place and event conditions, i.e. **when** it is critical for you to achieve a certain level of an attribute, **where** it is critical and under what **events**. For example:

Goal [2004, Europe, If the Euro is primary currency]: 99.98%.

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

Goal [2004, UK, If Euro is used in Norway & UK]: 60% ±20%.

In other words, allowing for an error margin, the range of results 40% to 80% is an acceptable upper and lower limit for the Goal level, but below 40% is unacceptable. Here is a more complete example:

Usability: Type: Performance.Quality Requirement. Scale: Mean time to learn defined [Task] to minimum proficiency. Fail [Timescale = Release 2.0, Language Variant = English, Task = Modifying Files]: 10 minutes. Rationale: will be beaten by competition. Goal [Release 2.0, English, Task = Modifying Files]: 7 minutes. Goal [Release 3.0, English, Task = Modifying Files]: 5 minutes. Goal [Release 3.0, French & Dutch, Task = Finding a File by Content]: 5 minutes.

In this example, the most critical risk is the Fail 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:

Goal: 60-80. "*Specification of a range.*" Goal: 60±30. "*Specification of an upper and lower limit.*" Goal: 60 → 90. Goal: 60? "*Expressing that the value is in doubt.*" Goal: 60?? "*Expressing that the value is in serious doubt.*" Goal: 60 ← A wild guess. "*Using the source of the information to show the doubt.*" Goal: 60 ← A.N. Other. "*Depends on A.N. Other's credibility in setting this value.*" Goal: <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 *uncertainty specification* rather than intimidate people from using it.

Goal [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' if the level is below 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.

Strategy03 [If hunger famine in a country and 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 timescales rather than on attempting to eliminate all risk.

Elimination of all risk is not practical, not necessary and, not even desirable. To eliminate all risk would lead to infinite costs. At some point as you approach no risk, you would eliminate necessary profit or incur costs that were too high.

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 requirements. My focus is always on achieving the requirements *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 requirements and, across the top of the table should be the proposed strategies. For the requirements, assuming you have expressed them using Planguage, it is usually a question of listing all the performance 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 Fail and Goal levels. Then, against each attribute, you state the current level and the Goal level for your chosen date (If you are especially risk averse you would use the Fail or Survival 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 Goal level?' First you state the actual value, on your defined Scale, you would expect and *then* you convert this into a percentage of the total 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.

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 sliding scale of 0.0 (no supporting evidence) to 1.0 (completely relevant supporting evidence) because the credibility rating can then be used later to modify estimates in a conservative direction by multiplication (For example, 0.5 modifies an impact to half its original estimated value).

Risk Analysis using 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 limitation 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.

The risk analysis calculations are as follows:

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

• **Impact of a Strategy**: For each individual strategy, sum all the percentage impacts it achieves vertically across all the performance attributes to get an estimate of its overall effectiveness in delivering the required performance. The resulting estimates can be used to help select amongst the strategies. It is a case of selecting the strategy with the highest estimated value and the best fit across all the critical performance 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.

In addition to looking at the effectiveness of the

	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%

Table 2: An example of a simple Impact Estimation (IE) table. Adding the percentage impacts for a set of strategies on a single performance attribute or cost can give some impression of how the strategies are contributing overall to the requirements. Note Strategies A, B and C are independent and complementary.

Performance Attribute	Past <-> Goal	New Idea
Reliability	900 <-> 1,000 hours MTBF	50%+/-10%
Maintainability	10 minutes to fix <-> 5 minutes to fix.	100%+/-50%
Total estimate of total effect of New Idea on all goals		150%+/-60%

Table 3: A measure of the effectiveness of strategy 'New Idea' can be found by adding together its percentage impacts for all the performance attributes.

individual strategies in impacting the performance attributes, the cost of the individual strategies also needs to be considered (See next).

• **Benefit-to-Cost Ratio**: For each individual strategy, calculate the benefit-to-cost ratio. For benefit, 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* use 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 performance attributes), we can calculate the average credibility of each strategy and, the average credibility of achieving each performance attribute. 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 performance or resource attribute where the sum of the % Impacts of all the proposed strategies does not total 200% (A 100% safety factor has been mandated, 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 requirements are unrealistic and they may be modified instead.

Within software engineering, the art of designing a system to meet multiple performance and cost targets, is almost unknown (Gilb 1988). 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 different strategies.

Principle 4: Design-in redundancy: When planning and implementing projects, it is necessary to use conscious backup redundancy for outmaneuvering risks.

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 numerous backup computer systems for manned space missions. The additional redundancy cost is always weighed against the consequential cost of failed systems. We don't build 'superfluous' redundancy into a system.

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 Principle).

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.

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[1]) 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 4, which is a hypothetical example of how an evolutionary project can be planned and controlled and risks understood using an IE table. 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 the areas at risk in the 'deviation' numbers.

The beauty of this, compared to conventional risk estimation methods (Hall 1998) is as follows:

- it is based on *real* systems and *real* users (not estimates and speculation *before* practical experience)
- it is *early* in the *investment* process
- it 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, 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 Specification Quality Control (SQC), also known as Inspection, as a third useful method.

SQC is a direct weapon for risk reduction (Gilb 1993,

[1] Practice (For example, academic studies on Hewlett Packard Evo practice over 14 years.) has shown that a 2% to 5% range (or 1-2 weeks) is realistic. The Evo step size is a function of the level of risk you want to take, which in turn is a function of things such as the technology, the market and the profitability. A project manager can take any step size they wish, but will find a step size of between 2% and 5% of a project's total time and financial budget is practical and challenging.

2000). Early SQC performed 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 SQC measures risks!

Many people think that the main benefit from SQC 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 SQC 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 SQC is not what happens at the level of a single document, but in teaching the people and the organization (Gilb 1993, 2000). The real effects of SQC include:

- *teaching* individual system engineers exactly how often they violate best practice rules
- *motivating* the system engineers to take rules seriously (<u>really</u> avoid injecting major defects)
- *regulating flow of specification defects*, so that high major defect project specifications and user documents can neither exit their initial process, nor enter adjacent work processes.

Staff involved in SQC meetings learn very quickly how to stop injecting defects. Typically, the defects introduced by an author reduce at the rate of about 50% less injection every time a new document is written and inspected using SQC. For example, using SQC methods, Raytheon reduced 'rework' costs, as a % of development costs, from 43% to 5% in an eight year period (Dion 1995).

Sampling. One other little-appreciated aspect of SQC 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

Step-> Attribute	**STEP1** plan % Impact	actual % Impact	deviation %	**STEP2** to **STEP20** plan	plan cumulated to here	**STEP21** [CA,NV, WA] plan	plan cumulated to here	**STEP22** [all others] 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

Table 4: A hypothetical example. This shows the expected percentage impacts (plan % Impact) for an Evo plan consisting of a series of Evo steps, STEP1 to STEP22, on a series of critical performance and resource budget attributes. Each Evo step comprises one or more design ideas (solutions). STEP1 has been implemented and the feedback results are shown (actual % Impact). The feedback is compared to the estimated impact, to see if the project is progressing as expected (deviation %). The plan is updated to include the estimates (and later the actual results), in the future cumulated figures.

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 using new/improved source documents. This is generally cheaper than trying to clean up the entire document using defect removal SQC or testing. You can calculate what pays off.

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 SQC 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 in SQC. The more effort that is put into making rules more effective and efficient, by incorporating feedback from SQCs, the more productive the SQCs are, and the greater the reduction in risk.

Even more benefit can be achieved if what is learnt from SQC 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 SQC itself. Their software defect rate went down by a factor of three (Dion 1995).

Using SQC defect and cost data , analysis of the identified defects to find process improvements is carried out in the Defect Prevention Process (DPP, part of SQC). 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 concept in getting DPP to work, compared to earlier failed efforts within IBM (Fagan's Inspection, 10 years earlier tried to use statistics to improve process – but was more successful in defect removal), 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 that 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 (Hall 1998) and a handful of my clients are reporting success at doing so. But, most are still working on other disciplines like defect detection SQC alone (not DPP) and others mentioned 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 defect searching roles, or process improvement roles within SQCs).

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 effects of contracting out a risk include:

- you have removed the risk in some sense, 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,
- the supplier will be motivated to tell you exactly what *you* have to do to avoid being hit by risks,
- the supplier might come up with a more realistic bid and time plan to cope with the risks.

You might wonder if a supplier will voluntarily accept contracts with in-built risk guarantees. My experience is that contractors will always accept reasonable risks to ensure they get the business. Contractors should only be made responsible for risks which they, not us, have knowledge and control over. In many respects, we are defining responsibility *before* a lawsuit situation, rather than after. A buyer has great power, but usually fails to use it to maximum advantage, thus allowing greater risk exposure. Relating the payment mechanism to the results is a key means of transferring risk. All critical success factors in the contract should be defined with Scales and target and constraint levels. For performance attributes, below Survival level means *no* payment, below Fail level means *partial* payment and reaching Goal level, within stated conditions, means 100% payment.

Specifying the use of Evolutionary Project Management within contracts is another key risk reduction mechanism. If a contractor fails to meet early deliverable levels, then you have early warning of the problem.

PRACTICAL APPLICATIONS OF PLANGUAGE APPROACHES TO RISK MANAGEMENT

There are extensive individual case studies carried out by the Author's clients (For example, Hewlett Packard,

Ericsson and Intel) on the various elements of the Planguage approach to risk management (Gilb 2002). The most well-studied aspects are in the areas of SQC (Inspection) and Evolutionary Project Management. See specifically, the case study by Dick Holland for an integrated example of three of the risk management methods (SQC, Evo and Planguage quantified requirement specification).

SUMMARY

Risks can be handled in many ways and at many levels. The need to *fully integrate* risk management into all implemenation and operational processes is clear.

I have tried to point out some risk management methods which are not so well known, or well treated, in existing literature (See Pennock 2002 for more conventional risk management thinking).

The Planguage approach to risk management includes, in summary:

• Within the Planguage specification language:

- quantification of benchmark and target levels for all performance and resource attributes. This aims to obtain a clear, unambiguous statement of the current and required levels for all the critical factors. This reduces the risk of misunderstandings or overlooking a critical requirement and, means that success can be measured.

- explicit specification of the constraint levels for all performance and resource attributes. This clearly defines the minimum and maximum acceptable limits.

- the use of qualifier conditions [when, where, event] to reduce generalization and identify specific dimensions for requirements. This helps eliminate risk by reducing the spread of requirements down to the precise areas of interest.

• Using SQC to sample the quality of specifications against best practice standards. By allowing process exit only when the level of remaining defects is acceptable, the risk of downstream pollution of the system engineering process is reduced.

• Using Impact Estimation (IE) to evaluate the estimated quantitiative impacts of designs and architectures on all the critical requirements. The risk of inadequate design is reduced by consideration of the evidence and credibility of design impacts and, by involving safety factors.

• Using Evolutionary Project Management (EVO) to obtain early delivered results and early feedback. By aiming for results at frequent intervals (say 2% of project resource budgets), there is more immediate feedback on how the project plans work in reality and how the customer perceives the results. If changes are required, there is resource budget to take corrective action. By implementing changes in small steps, risk is distributed and, it is easier to identify where any problems exist.

Tables 5 and 6 recap the ideas presented in this paper. Table 5 is a set of policies for risk management. See (Gilb 2002) for more detail. Table 6 contains 'Twelve Tough Questions' that you should ask when assessing risk.

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 and cost attributes.

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

Numeric Expectation Specification
The benchmark, constraint and expected target levels of all performance 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.

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 SQC method (Gilb 1993). 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 (300 words) or less per hour).

Complete Design Specification and Impact Estimation Specified
A complete *set* of designs or strategies for meeting the complete *set* of performance and cost targets will be specified. They will be validated against all specified

performance 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.

Evolutionary Proof-of-Concept Priorities
The Evolutionary Project Management method (Gilb 2002) will be used to sense and control risk in mid-project. The dominant paradigms will include:

- 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 5: Policy Ideas for Risk Management

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 and budgets, 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 requirements 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?

Table 6: Twelve Tough Questions

REFERENCES

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

Dion, Raymond and Haley, Tom and Ireland, Blake and Wojtaszek, Ed, "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.

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

Gilb, Tom and Graham, Dorothy, *Software Inspection*, Addison-Wesley, 1993, ISBN 0-201-63181-4, 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 (SQC), with hundreds of larger and smaller improvements over initial Inspection practices.)

Gilb, Tom 2000
"Planning to get the most out of Inspection" in
Software Quality Professional, ASQ (American Society for Quality
Volume Two, Issue Two, March 2000
Pages 7-19 , http://sqp.asq.org
Published (2001) in book
"Fundamental Concepts for the Software Quality"
Engineer, Order ASQ 800 248 1946, Quality Press
Online http://qualitypress.asq.org

Gilb, Tom, http://www.Gilb.com/, 2002. Various papers and manuscripts. These include:
. *Competitive Engineering* (CE) in draft
. *Evolutionary Project Management*. 1996 draft. Also slides
. "Requirements Engineering Language." Slides only.

Hall, Elaine M., "Managing Risk: Methods for Software Systems Development". *SEI Series in Software Engineering*, Addison Wesley Longman, USA, 1998, ISBN 0-201-25592-8, 374 pages.

May, Elaine L. and Zimmer, Barbara A., "The Evolutionary Development Model for Software", *Hewlett-Packard Journal*, August 1996, Vol. 47, No. 4, pages 39-45.

Pennock, Michael J. and Haimes, Yacov Y., "Principles and Guidelines for Project Risk Management in Systems Engineering". *The Journal of INCOSE*. Vol 5, No. 2, 2002. Pages 89 – 107.

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 in the Radice IBM

CMM version directly, and later indirectly 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.

Edit note this version was edited by Tom after Lindsey's rewrite July 8th. Changes to LB are tracked but not on screen.

Further information can be found at http://www.Gilb.com/.

E-mail: Tom@Gilb.com

This paper was edited by Lindsey Brodie, lindseybrodie@BTopenworld.com.