THE MANAGEMENT OF RISK IN SYSTEM DEVELOPMENT: 'PROJECT SP' AND THE 'NEW SPIRAL MODEL'

J Gerard Wolff

School of Electronic Engineering Science, University of Wales, Dean Street, Bangor, Gwynedd, LL57 1UT, UK. Telephone: +44 248 351151 ext 2691. Janet: g.wolff@uk.ac.bangor.complab.vaxa.

Issue 1.1

5 January 1989

In the *Software Engineering Journal 4(3),* 134-142, May 1989.

1 ABSTRACT

The article discusses the development of complex products, with a particular emphasis on software, and focusses on the problem of how to manage the risks in the development process.

A number of models of the development process are described including Boehm's Spiral Model, which has risk management as a central theme.

An example is described where Boehm's Spiral Model has been tried. The strengths and weaknesses of the model are discussed in the light of this experience.

In the last part of the article, a notation called 'Project SP' is presented as a means of recording the progressively growing knowledge base of a project and the areas of uncertainty and associated risk.

Also described is the New Spiral Model, derived from Boehm's model and designed to be used in conjunction with Project SP. The New Spiral Model appears to preserve the advantages of the previous model and appears to remedy its weaknesses.

Associated issues are described and discussed.

2 INTRODUCTION

Many products of modern technology - cars, aeroplanes, computing systems - have proved to be very useful. But the development of products like these is complicated - and there are risks [11]. There are many examples - from Concord to Nimrod - to illustrate the hazards in the development of new complex systems. In the software industry - which is what I know best - cost overruns and the wasted effort represented by projects which are abandoned at a late stage, or whose deliverables are never used, are notorious.

This article considers the process of designing and developing complicated systems, especially software, and discusses how the risks in development and the overall cost of development can be minimised.

In many ways, software provides a paradigm for all kinds of design and development. Software is pure information and, as such, it captures the essence of all kinds of design. Software systems are often very complicated (although the same is increasingly true of integrated circuits). The main difference between software and hardware systems is that, with software, there is no significant process of 'production'. Once the design is complete, multiple copies may be made very easily. This difference has some bearing on the design process - and this will be briefly discussed. But the model of design and development proposed in this article is general enough to cover all kinds of system.

In the next section I briefly examine a small range of models of design and development giving special attention to Boehm's 'Spiral Model' and its approach to the management of risk. Then I describe and discuss a project at Praxis (where I was employed until recently) where the Spiral Model has been tried. In the following section, the New Spiral Model is presented. It is derived from the Spiral Model and includes a notation, called 'Project SP', for tracking the status of a project from start to finish. The article concludes with a discussion of some related issues.

3 MODELS OF SYSTEM DEVELOPMENT

In order to manage any kind of system development, including small projects, it is necessary to have some kind of model of the development process, preferably one which is explicit and precise. In this section, I will first briefly review some of the more popular models and then I will describe and discuss Boehm's Spiral Model at more length.

- **The Waterfall Model**. This model of system development (see, for example, [4]) is one of the oldest and is still popular. In this model, a project proceeds in an essentially fixed sequence of 'stages'. A typical sequence is:

  1. Project inception: defining objectives and constraints on the project.
  2. Planning.
  3. Specification of requirements for the system.

4 Design meaning definition of the software at some high level of abstraction.

5 Coding, meaning the production of an executable program.

6 Testing and integration.

7 Release of the system to the client.

At any stage (except the first) it is possible to return to the previous stage and rework it. More radical backtracking is discouraged because of the management problems which that entails.

The Waterfall Model marries naturally with the principle of *top-down design*. In top-down design, there is a more or less fixed progression in the design process from the definition of the largest abstract components of the design to the progressively more detailed elaboration of each part and its component parts. As with the Waterfall Model, some backtracking is accommodated, provided it is not too radical.

- **The Two-Legged Model**. In this model (see, for example, [7]) the process of design and development is divided into two 'legs': 'abstraction' and 'reification'. Abstraction leads from users' requirements to a formal specification of a system while reification is a process of progressively translating a specification into some kind of runnable 'implementation'. This translation process is sometimes called 'refinement'.

  Associated with the concepts of abstraction and reification are the concepts of 'validation' and 'verification'. The former means establishing that the specification conforms to the users' requirements while the latter means proving that the implementation conforms to the specification. There has been a marked tendency in discussions of this model and in its applications for abstraction and validation to receive much less attention than refinement and verification.

- **The Prototyping Model**. This model and variants of it are described in [6]. One of the main motivations for this model is the recognition that prospective users of a system are rarely able to define their requirements fully in one operation. Users also often find it difficult to define what they want in abstract or verbal terms independent of some working system: "I'll know it when I see it" (IKIWISI). A prototype provides a means for users to say more precisely what they do or do not want. The general idea, then, is to construct a series of prototypes, to allow prospective users to examine each one, and say what changes they want in the next one.

  Prototyping can also provide a means for system designers to clarify other aspects of a system: how it should be structured for easy modification or maintenance; how best to optimize the performance of the system; and so on.

In the Throw-it-away variant, all prototypes are discarded; the delivered system is the last system in the sequence. In the Evolutionary and Incremental variants, software is carried forward from stage to stage and the functionality of the system is progressively refined and increased. The Evolutionary Model allows deletion and changes from stage to stage while the Incremental Model allows only additions.

There is a natural affinity between the Evolutionary and Incremental Models and object oriented design (see, for example, [1], [5]). The mechanism of *inheritance* in such languages as Simula, Smalltalk and LOOPS provides a streamlined way of integrating new software with old. Because it supports the creation of 'well structured' designs, object orientation facilitates the process of changing designs.

3.1 Boehm's Spiral Model

Boehm [2] claims, with some justice, that the Spiral Model embraces most other models as special cases. It is a more general view of the process of design and development than other models and can apparently be applied to a wide range of types of project. But it is not so general as to be vacuous: it provides useful disciplines and constraints on the way development is done.

According to the model, a project should comprise a series of **cycles** or **rounds**. The steps in each cycle are broadly these:

1. Define **objectives** for the cycle.
2. Identify **constraints**, eg budgetary constraints and timescales.
3. Identify **alternative** means of meeting objectives (eg design A, design B, reuse, buy etc).
4. Evaluate alternatives with respect to objectives and constraints and, for each alternative, identify areas of uncertainty and the corresponding **risks**.
5. Decide how to **resolve** risks (eg construct a prototype, consult an expert, do a simulation, administer a user questionnaire, build system) and then do the chosen task.
6. Gather and review the **results** of the risk resolution exercise.
7. **Plan** the next cycle.
8. Review the plan and make a **commitment** to it.

The main differences between this model and the more traditional approaches are these:

- There is explicit recognition for alternative means of meeting the objectives of a project.
- The identification of risks associated with each alternative, and the ways in which those risks may be resolved, are brought centre stage. With traditional approaches, the easy bits of a project may be done early and the areas of uncertainty left till later - and this can give a spurious impression of progress. A risk driven approach to development more readily avoids this pitfall.

- The division of a project into cycles with a ‘commit’ step at the end of each cycle, means that there is explicit provision for changes in the direction of a project or the termination of a project, at any stage, in the light of what has been learned since the start of the project. By contrast with ‘big bang’ approaches to system development, the cyclic approach enables a limit to be placed on the risks which have to be accepted at any time.
- The model accommodates types of activity (eg consulting an expert or library research) which are often valuable in reaching the objectives of a project but which have no place in other models.

4 THE PCIS PROJECT

In Praxis we have tried using the Spiral Model in an internal project to develop the "Praxis Company Information System" (PCIS). The experience that we gained with the Spiral Model seems to be useful and worth reporting.

The development of business information systems is, of course, fairly well understood and the project is not intrinsically risky. Nevertheless we found that the framework provided by the Spiral Model has been valuable and has enabled us to retreat from potential pitfalls before any strong commitment was made.

What has been done maps fairly well to the model although there have been some areas of uncertainty as we learned to interpret and apply the model. As we used the model, some weaknesses became apparent and these are discussed below.

Our application of the Spiral Model included the Evolutionary Model as a subset. We felt that it would be rash to try to gather all requirements before producing any working system because of the IKIWISI phenomenon and because the needs of a company like Praxis, which is growing fast, do change as time goes by. We envisaged that several of the cycles of the Spiral Model would each be largely concerned with the development of a ‘prototype’ or ‘version’, to be evaluated by prospective users and, in most cases, carried forward into the next cycle for refinement and enhancement.

4.1 The First Cycle

The objective defined for the project at the outset, which applied throughout the project, was to "develop an information system to meet the needs of Praxis management".

It seemed necessary to start the project off with a fairly conventional gathering of requirements from prospective users of the system but, because we envisaged evolutionary development, this first gathering of requirements was not done in exhaustive detail. An outline data model was constructed together with an outline description and analysis of the functions to be performed and a description of the ‘non-functional’ attributes required in the system.

This first gathering of requirements was regarded as part of the process of defining the objectives and constraints on the project.

The next four steps in the model were covered by a study called "Analysis of Options and Risks". In this study, we identified a set of alternative means of meeting the objectives:

- Sub-contract the work to another systems house. Since we are, ourselves, a systems house, this option did not look very plausible. But we felt it would be useful, nevertheless, to weigh the pros and cons.
- Do the work ourselves. On this basis, we identified four alternative means of developing the system:
    - Use software DBMS "A" and its 4GL tools.
    - Use software DBMS "B" and its 4GL tools.
    - Use database machine "C" and its 4GL tools.
    - Assemble a collection of packages to serve the various functions of the company.

Criteria for evaluation were derived from the initial study of requirements and the options were investigated and evaluated against them. Examples of criteria for evaluating the proposed vehicles for the system include: cost of the vehicle, ease of use of the development tools (and corresponding cost of development effort), the quality of the user interface, the existence or otherwise of mechanisms to preserve the integrity of data in the face of hardware failures, and so on. The criteria may be seen as ‘risks’: if an option does not meet one or more of the criteria then the final system is likely to be unsatisfactory.

The best option appeared to be to do the work ourselves using the database machine "C", perhaps after some further investigation of the machine.

However, there were constraints which dictated a slightly different course. We already had a licence for software DBMS "A" and, for budgetary reasons, there was little prospect of getting the database machine soon. There was also a need to gain experience with DBMS "A".

Consequently, we planned to do the first one or two versions of the PCIS with software DBMS "A", and also do some further investigation of "C". If, after further investigation, "C" was still looking good, we would re-write the system on it and continue with it in subsequent cycles.

This longer-term plan was the basis of the shorter term plan prepared for the next cycle: to develop a first version of the PCIS to serve the needs of the company in the areas of marketing and sales.

The plan was reviewed and a commitment made to it.

4.2 The Second Cycle

The objective of the second cycle was the development of the first version of the PCIS under the constraint that it should use DBMS "A".

No significant alternatives presented themselves in this cycle and, with a qualification described below, we did not see significant areas of risk requiring investigation. The cycle constituted a fairly straightforward development of the first version of the PCIS. Detailed requirements were gathered from relevant staff, a running system was developed using the 4GL tools, and the system was carefully evaluated by the prospective users of the system. This evaluation led to the definition of changes required in future versions and new features needed.

The evaluation of the first version of the PCIS and the definition of changes required in future versions may be regarded as the first part of the Third Cycle: these activities are part of the process of defining the next set of objectives in the project.

The idea that there was no apparent risk in this cycle should perhaps be qualified by the thought that the development of the first version of the PCIS may, itself, be seen as an exercise in the reduction of uncertainty and the resolution of risk. Before the development is undertaken, there is a degree of uncertainty about what form the system will take and about how well it will work; this uncertainty is reduced or removed by constructing and evaluating the system. In this spirit, the cycle - and the project - were reviewed.

The main conclusion of the review of ‘results’ was that DBMS "A" is much less satisfactory than we had anticipated. Its main defect is unacceptably slow response times on our equipment even with very little data in it and few users. A new and bigger machine could, of course, be bought for it but this option needs to be set against the apparently preferable option of buying dedicated hardware - the database machine.

The slow response times with DBMS "A" could, of course, have been discovered without actually developing a version of the PCIS. In retrospect, it might have been wiser to conduct some tests on the DBMS as a risk resolution exercise prior to building the first version of the system. However, the development of the first version of the PCIS was not wasted effort because it sharpened our understanding of users’ requirements in several other areas, especially the user interface.

On the strength of the review, the Third Cycle was planned to take in a further short investigation of the database machine and its tools, and a decision on its purchase. In this Third Cycle, a new version of the PCIS is to be developed using the new system, drawing on the knowledge gained in developing and evaluating the first version. The definition of these objectives, like the definition of changes required in future versions of the PCIS, may be regarded as the first stage in the Third Cycle.

4.3 Discussion

To date, the Spiral Model seems to have been a reasonably good framework for the PCIS project. But there is a need for more clarity in some areas, or changes in the model. These will now be described.

1 There is a need to define more clearly how longer term objectives and plans may be carried forward from cycle to cycle and integrated with the shorter term objectives and plans. An example of a long term intention which needed to be carried forward into later cycles is our early decision in the PCIS project to migrate to database machine "C" after one or two versions of the PCIS produced using software DBMS "A", provided that further evaluation of the database machine was satisfactory.

2 There seems to be a need to provide for hierarchical relationships and for ‘spirals within spirals’:

- Within a cycle, the process of resolving risks may be regarded as a spiral in its own right: there should be objectives and there is likely to be alternative means of meeting the objectives, each with risks which may need to be resolved. The model needs to be applied recursively.
- Similar remarks apply to the planning activity within each cycle. That planning may itself be modelled on a spiral model has been noted and discussed by Boehm and Belz [3]. Other points about how planning relates to the Spiral Model are made below.
- Alternatives may easily demand a hierarchical structure. The alternatives shown in section 4.1 are one example. As another example, one of the alternative means of resolving risks - "consult an expert" - may itself be broken down into a set of alternatives: "consult Susan", "consult Joe" etc.

3 As we saw in section 4.2, there may not be obvious major alternatives in a cycle. And it is not very obvious that developing a version of a system is ‘risk resolution’ in quite the same sense as investigating alternative vehicles for the system.

4 Not only may the planning activity within each cycle of the Spiral Model be modelled on a spiral model but several of the other activities in each cycle - defining objectives, identifying constraints, identifying alternative ways of meeting objectives, identifying risks and deciding how to resolve risks - may themselves be regarded as planning activities. Moreover, the Spiral Model may itself be regarded as a skeleton plan for a project. There is, in general, a need to define more clearly how any planning done within the model relates to the model itself.

5 ‘PROJECT SP’ AND THE ‘NEW SPIRAL MODEL’

In this section a notation and a new version of the Spiral Model are described. They seem to meet most of the problems with the Spiral Model which were noted above.

5.1 ‘Project SP’: a Notation for Project Management

This section introduces a simple notation called ‘Project SP’ which is an informal variant of the SP computer language, described elsewhere [10].

Project SP, which is similar to BNF, may be used for recording the progressively growing **knowledge base** for a project. This includes the objectives of the project, constraints on the project, alternative means of meeting objectives, the areas of risk and uncertainty, planned activities on the project and the information gathered to reduce uncertainty and resolve risks. The last-mentioned category includes information about the design of the system being developed. Boehm and Belz [3] have also proposed using a data store in conjunction with the Spiral Model.

Here is the notation:

- (...) - A sequence of items or 'ordered AND object' (OAO). This includes the normal sequencing of words in English sentences.
- [...] - An 'unordered AND object' (UAO): a group of items where order is not defined.

  {...} - An 'OR object' (ORO): the items between the curly brackets represent alternatives.

  Each option may be marked with a 'per cent' ('%') or 'weight' showing the apparent strength of the case for choosing that option: 0% means that the option should never be chosen while 100% means that it should always be chosen.

  If the weights in an ORO total to 100% then it represents an **exclusive** OR relation. If weights within an ORO total more than 100% then it represents an **inclusive** OR relation. If all the weights within an ORO are, individually, 100% then the structure is equivalent to a UAO.
- * - A star placed after an item shows that the item may be repeated.
- ? - Question marks show where more information appears to be needed, the strength of the need being shown by the number of question marks.
- Indendation may be used to improve readability.

5.2 An Example

As an example to introduce the uses of Project SP, Figure 1 shows how the notation may be used to describe the knowledge base for the PCIS project at the time of writing. This and other examples are discussed in the following sections. In the Figure, sets of three dots ("...") are used to show where information has been left out to save space.

Please insert Figure 1 about here

5.3 The Uses of Project SP

The ways in which Project SP may be used are described here with illustrations from Figure 1 and other examples.

5.3.1 Groupings of Objects

OAOs and UAOs both serve to group their constituent objects. UAOs are used where there is no spacial or temporal ordering of the constituents. One example in Figure 1 is the evaluation criteria for the vehicle for the PCIS. Another is the grouping of any object with its label or identifier (see below).

OAOs are used where there is some kind of spacial or temporal ordering of constituents. Examples in Figure 1 are the ordering of activities in a project plan and the order of words in English text. The relevance of Project SP to planning is discussed more fully below.

5.3.2 Identifiers and References

The notion of 'identifier' has no formal significance in Project SP. Indeed, any object may be identified by any of its constituents which is sufficiently distinctive. However, as a matter of psychology, it is often convenient to regard one of the constituents of an object as being primarily an identifier or label for that object.

The relationship between an identifier or label and what it identifies seems to be best represented as an unordered AND relationship. In Project SP, any object may be labelled by bracketing it with its label within a UAO, eg [Fred (...)]. If the object being labelled is itself a UAO then the inner brackets may be dropped.

Since the identifier and what it identifies are constituents of a UAO, they may be written in any order. However, for the sake of readability and easy comprehension, it is convenient to put the identifier first. Examples in Figure 1 of the labelling of objects in this way include [DBMS-A ...] and [objectives-1 ...].

A reference to any object may be created by using its identifier as a constituent by itself within a UAO. Examples in Figure 1 include [objectives-1] and [UI]. The latter example is a reference to the user interface, one of the criteria for evaluation listed elsewhere in the structure.

References may be used to avoid recording a structure more than once when it appears in more than one context. The examples in the last paragraph illustrate this use in Figure 1.

References may also be used when incorporation of a large data object at a given point would be cumbersome and would make the organization of the whole structure less easy to see. Examples of this second use of references are the numbers in square brackets, eg [42.1-1.0], which are reference numbers for Praxis documents. The text of any such document is not incorporated directly in the main structure.

5.3.3 Uncertainty in the Knowledge Base

OROs in Project SP provide a means of representing uncertainty in a project. As described earlier, the weights on items within a ORO (figures with '%') show the apparent strength of the case for choosing that option. Whenever weights are less than 100% and more than 0% there is a corresponding uncertainty in choosing.

An example in Figure 1 of the use of an ORO in this way is the choice between doing the development work in-house and contracting it out to another systems house. In this example, the in-house option is itself broken down into a number of subsidiary options.

Question marks in Project SP provide a second means of representing uncertainty or ignorance within a knowledge base. Of course, we can never be sure that our knowledge of any one thing is complete: there is a metaphorical question mark attaching to every object in every knowledge base. The use of question marks in Project SP is intended to show the perceived strength of the need to find out more in this or that part of the knowledge base. Exactly how one makes such judgements is itself a significant question mark in our understanding of system development.

5.3.4 'Deletion' or 'Replacement' of Objects

The intention with Project SP is that information should never be destroyed within the structure - unless it is a simple error in editing the structure or if it is redundant. In the latter case, where an object has been replicated in more than one context, all instances except one 'master' copy may be replaced by references to that master copy.

If one object supercedes another in the knowledge base then the two objects should be formed into an ORO, with the later item marked as 100% and the superceded item marked as 0%. If, at some later stage in the project, the old item seems useful again, then the weights may be adjusted to show this.

If an object is to be 'deleted' and nothing put in its place then it should be formed into a selection where its alternative is a UAO which is empty except for its 100% weight, eg [...{[0% Fred (...)],[100%]}...]. An acceptable shorthand is simply adding a 0% weight to the object which is to be demoted - without incorporating it in an ORO.

To preserve all information in the knowledge base even when it is out of date may seem cumbersome but it is no different from what has been normal practice for many years with conventional, paper-based filing systems. Of course, many computerised databases are designed for the deletion and overwriting of information but this is likely to change as storage technology improves and the advantages of non-overwriting systems like ADAM [10] are seen.

A main reason for never destroying information in the knowledge base is that one can always backtrack if necessary. Another important reason is that one can always construct an 'audit trail' for a project. This can be useful in project debriefing and in case of dispute if that unfortunate contingency arises.

5.3.5 Representing the Structure of Software Systems

Project SP is an informal version of the SP computer language, described in [12]. The SP language is intended as a ‘broad spectrum’ language with a wide range of uses including data storage and retrieval, software specification and design, representation of rules for expert systems, logic programming and others. If the potential of SP is realised - and a research programme is needed to establish how and how far one simple language can serve such a wide range of applications - it may be possible to represent the whole structure of a developing software system with one notation which is the same as is used for other parts of a project knowledge base.

Figure 1 shows information about the PCIS in only the barest outline, with references out to Praxis documents where the real meat of the information about the structure of the system is stored. It would take us too far afield to discuss fully how SP or Project SP could be used more directly to represent the structure of the PCIS (there is relevant discussion in [12]) but some brief indication of the possibilities is warrented here:

- Project SP embraces the concepts of ‘sequence’ (OAO), ‘selection’ (ORO) and ‘iteration’ (‘*’) which are widely recognised as basic organizing principles in software (see, for example, [7]). Iteration does not feature in SP but an equivalent effect may be achieved by using recursion. The UAO construct may provide a means of representing ‘concurrency’ in software systems.
- SP and Project SP apparently lend themselves to the efficient representation of versions or variants of software systems. In other words, they have potential to facilitate configuration management. The mechanisms which will serve the ‘object oriented’ concepts of classes, sub-classes and inheritance of attributes (see [12]) will also serve to keep track of the several versions of a software system which usually arise in system development. A simple example appears in Figure 1 where the versions of the PCIS are represented as constituents of an ORO - alternative realisations of the PCIS.

5.3.6 Representation of Plans

In keeping with the ‘wide spectrum’ remarks in the last section, there is potential in Project SP to represent plans directly in the knowledge base. The main constructs recognised in project planning also appear in the notation:

- A sequence of activities in a plan may be represented using an OAO.
- A UAO may be used to represent activities which are independent of each other in much the same way that it may be used to represent concurrency in software. Independence of activities corresponds to the slightly innaccurate use of the term *parallel* in project planning. ‘Parallel’ activities may be performed in parallel or they may be performed in some arbitrary sequence, depending on the resources available and the required timescales.

- An ORO may be used to represent activities which are **alternatives** in a plan. This is not very common in ordinary projects but the concept is recognised in the term ‘contingency planning’. Where there are alternatives in a plan there is a need to know how to choose between the options. This can be achieved by associating each option with a condition which must be satisfied before that option can be chosen. In Project SP, each alternative course of action should be enclosed within an OAO together with its corresponding condition or conditions:

```
{
    (condition-1 action-1)
    (condition-2 action-2)
    (condition-3 action-3)
    ...
}
```

The similarity between the organizing principles recognised in project planning and those recognized in software design is interesting in its own right. It also reinforces Osterweil’s argument [9] that "Software processes are software too". If project planning is to be seen as "process programming" then it will be convenient and elegant if the notation used to describe a software product can also be used to describe the process by which that product is created. Whether or not that can be achieved with SP or Project SP remains to be seen.

5.4 The New Spiral Model

Given the use of Project SP to record the growing knowledge base of a project, Boehm’s model may be recast in a modified form.

In the New Spiral Model (NSM), there are two fundamental operations or activities: (1) gathering new knowledge and adding it to the knowledge base; (2) reviewing, analysing and rationalising what is in the knowledge base. There will usually also be the **execution** of plans described within the knowledge base. Here is a fuller description:

1 Gather new knowledge and add it to the knowledge base.

"New knowledge" in this context can mean any or all of the following kinds of knowledge:

- Knowledge about the **objectives** of a project or activity.
- Knowledge about **ignorance** and corresponding **risks**. This is not as paradoxical as it sounds. For example, anyone with experience with computers and similar technology knows that, for any proposed new piece of equipment, one ought to find out how reliable it is, the callout time for maintenance, the cost of maintenance, the availability of spares, and so on. Knowing what one needs to find out is part of the skill of system development.

Defining gaps in one's knowledge is closely related to the definition of **objectives** for an activity or project. A requirement to "support the production and maintenance of company accounts" is a gap waiting to be filled by a suitable system. The gaps in one's knowledge about any proposed new piece of equipment are objectives for corresponding activities to plug those gaps: "find out about reliability", "find out about callout times", etc.

- Knowledge gleaned from various sources about **constraints** on a project or subsidiary activity, eg budgets and timescales.

- Knowledge about **methods** of meeting objectives within the given constraints; where there is more than one method for an objective these represent the kind of **alternatives** which feature in Boehm's Spiral Model.

  A method, together with such information as start and end dates and staff assignments, is the kind of information which is normally represented in some kind of network of **activities** or PERT chart. As indicated above, SP or Project SP may also provide a means of representing this kind of knowledge, including sequences of activities (OAOs), independent or 'parallel' activities (UAOs) and alternatives or contingencies (OROs).

  Gathering information about methods and corresponding activities, incorporating it in the knowledge base, reviewing, analysing and rationalising it (see below) is what is normally meant by **planning**.

  Knowledge about methods includes such hum-drum things as how to make a journey to visit a client, how to obtain information from libraries, and so on. Knowledge about methods also includes knowledge about the several models of system development - including the NSM itself!

  If knowledge about methods is supplied from the experience of members of the project team, as it often will be, it is not, in that sense, 'new'. However, in terms of the project's knowledge base it is new knowledge. Unless it is trivial, everyday knowledge, it needs to be recorded explicitly regardless of whether it comes from some external 'expert' or from within the team.

- Knowledge about the developing system. For most projects this is the main 'deliverable' of the project. It includes such things as high level design, low level design, source code, maintenance documentation, user guides and so on.

- Records of any commitments made to project plans.

2 Review, analyse and rationalise the knowledge base. This activity includes any or all of the following kinds of activity:

- Look for redundancy in the knowledge base and reduce it where possible. Where objects are replicated, references may be used to reduce redundancy, as described earlier. Examples of searching for redundancy in a knowledge base and reducing it include the process of 'normalising' data models and the processes needed to achieve 'good structure' in the design of software (see [12]).
- Review objectives against the results of system development to see whether the objectives have been met.
- In the light of any new knowledge gained since the previous cycle, review the weights attaching to the constituents of OROs.
- Identify areas of uncertainty, ignorance and risk and decide where further investigation is needed.

3 **Execute** any plans which are ready to go. A plan is ready to go if it has been created and reviewed, if a commitment has been made to it and if all pre-conditions attaching to it are satisfied. Pre-conditions include such things as the completion of previous activities on which the plan depends (eg the delivery of equipment), the availability of relevant staff, reaching the start date, and so on. 'Execute' in this context is analogous to 'eval' in Lisp.

Figure 2 represents the main points in foregoing description of the NSM using the Project SP notation.

Please insert Figure 2 about here

The 'activities' object in Figure 2 describes the two basic operations in each cycle of gathering new knowledge and then reviewing, analysing and rationalising it. Under each heading there is an ORO describing the options. One or more of these options may be chosen on each cycle.

The [define [activities]] object is the 'planning' or 'process programming' operation. It means fleshing out the bare bones of the basic [activities ...] object in the NSM with the more detailed descriptions of activities needed to make a project go, using any available knowledge about methods.

As already noted, knowledge about methods can include the NSM itself. Thus the NSM may reappear within itself as, for example, when a major project has been planned to contain one or more sub-projects. The model can be applied at any level.

Notice that the application of the NSM within itself means that it can be used in the creation of project plans. In other words, it can be included within the [define[activities]] object. Boehm and

Belz [3] have, in a similar way, proposed using the Spiral Model in the development of project plans.

With or without this kind of recursive application of the model, Project SP accommodates an hierarchical structuring of activities within activities, with sequence, selection, concurrency and iteration at any level.

5.5 Discussion

To quote a well-known phrase, "forewarned in forearmed": the reduction of risk is closely related to the reduction of ignorance or the building of knowledge.

The idea behind the use of Project SP and the NSM is to treat a development project as a progression from relative ignorance to relative knowledge. The design of a computing system (or anything else) is a form of knowledge and the process of creating a design may be seen as a process of knowledge accretion. Likewise, all the other aspects of a development project - the model of development, objectives, constraints, user's requirements, project plans, and knowledge about equipment and facilities - are forms of knowledge which typically grow as the project proceeds.

Areas of uncertainty in a development project take two forms in Project SP: alternatives in a selection, or straightforward gaps in knowledge (represented by question marks). Uncertainty is reduced when an alternative is chosen or when a gap in the structure is plugged.

Project SP and the NSM together meet the points made in section 4.3:

1. Short and long term objectives and plans may be recorded in the knowledge base and carried forward from cycle to cycle.
2. Hierarchical relationships are naturally accommodated in the notation. Selections within selections are shown in Figure 1. The NSM is itself one of the candidate 'methods' which may be used in any part of a structure of planned activities and at any level. In other words, the NSM can be applied recursively to create 'spirals within spirals'.
3. Risk reduction may be achieved by any kind of gain in relevant knowledge. Sometimes this relates to alternatives but it may equally well be the plugging of gaps in knowledge - and that includes gaining knowledge by building a system.
4. The relationship between 'planning' and the NSM is this:
    - The NSM is itself a skeleton plan.
    - The NSM replaces three activities in Boehm's model with a single 'process programming' activity for fleshing out the skeleton with the more detailed descriptions of activities needed to make the plan practical. The three activities which are replaced are: the identification of alternative means of meeting objectives, deciding how to resolve risks, and 'planning'.

6 RELATED ISSUES

In this section, I pick up some loose ends from what I have said and discuss them briefly.

6.1 The Spiral Models and Iteration in Design

As we have seen, the idea of **iteration** features not only in the spiral models but in the prototyping model as well. In the spiral models iteration serves a management need to limit the risk to which a project is exposed at any time. In the prototyping model, iteration is probably serving the rather different needs of the design process:

- Iteration seems to suit end users best. A major reason is IKIWISI but there is the related reason that potential users of a system naturally examine and re-examine their requirements in the process of thinking them out.
- In a similar way, it is psychologically natural for designers to iterate in the process of thinking out the best organization for a design. Drafting and re-drafting has always been required in all kinds of design - from graphic art to the writing of articles and books.

Whatever the reasons for iteration, it is accommodated very well within the spiral models.

6.2 Should One Always Concentrate on High Risk Areas First?

As we have seen, there are good reasons why a project should focus on areas of ignorance and risk. However, in the design process, it sometimes seems better to design the best understood parts first. For example, in object oriented design it is recognised (see [1]) that one should first identify and define the most salient classes in the system and should later add the less salient classes, whether they be at higher or lower levels of abstraction.

I believe this contradiction is more apparent than real. There is a difference between recording things which you already know and working out something new. As a result of discussions with users, a designer will know things about the form which the system must take and these things need to be recorded. Recording them explicitly in a partial design is quite consistent with the principle of seeking out areas of uncertainty. One cannot see the areas of uncertainty clearly unless one has recorded what one already knows.

6.3 The Design of Other Kinds of System

The ideas which have been presented seem to have a fairly wide scope and should prove useful in the development of systems where there are significant components which are not software.

The main difference between hardware and software is that hardware systems are usually less ‘malleable’ than software systems. A significant commitment of resources is often required in the construction of hardware systems.

This feature has been a reason for the traditional notion that ‘design’ always precedes ‘construction’. When the software industry first developed, it was natural to borrow this idea. And it flourished because, until recently, significant resources were required in the creation of software systems and there was a consequent need to get the 'design' right before embarking on 'implementation'.

Modern tools and the increasing performance of computers for a given price is progressively reducing the resources required in the creation of a working software system. There is, consequently, more scope to adopt new methods of working which take advantage of this new flexibility.

Of course, it was never entirely true in the hardware world that design necessarily preceded construction. The creation of prototypes has been a part of engineering methodology from the earliest days. Brunel built prototypes where there was uncertainty about design. And the construction of any one bridge, for example, may be regarded as a trial run for the design and construction of bridges which come later. The eighteenth century bridge at Pontypridd, which at the time was the longest single span in the world, fell down three times during construction before a successful design was found.

These observations confirm the validity of the iterative principle in the world of hardware development and suggest that the spiral models can indeed apply to the development of systems other than software.

7 CONCLUSION

In this article, I have tried to identify key concepts in the management of system development which will reduce risks, reduce the cost of failure and thus reduce the overall cost of developing complex systems.

I hope that other people will try out the ideas which have I described and report their experiences and new thinking in the future.

8 ACKNOWLEDGEMENTS

I am grateful to all those involved in the PCIS project, in development work, reviewing, or as prospective users of the system - Dave Allen, David Bean, Tim Huckvale, George May, Jane Northcote, Martyn Ould, Stephen Robertson, Tony Voss - for cooperation in the application of a new management model and for constructive comments on the model and earlier drafts of this article. I am also grateful for very useful comments and suggestions from an anonymous referee.

9 REFERENCES

1 BIRTWISTLE G. M., DAHL O-J., MYHRHAUG B. and NYGAARD K.: *Simula Begin* (Van Nostrand Reinhold, 1979)

2 BOEHM B. W.: 'A spiral model of software development and enhancement', *ACM Sigsoft Software Engineering Notes*, 1986, **11**, (4), pp. 14 - 24

3 BOEHM B. W. and BELZ F.: 'Applying process programming to the Spiral Model'. Proceedings of the IEEE Fourth Software Process Workshop, Devon, England, May 1988.

4 BUCKLE J. K.: 'Software Configuration Management' (MacMillan, 1982)

5 COOK S.: 'Languages and object-oriented programming'. *Software Engineering Journal*, 1986, **1**, (2), pp. 73-80

6 HEKMATPOUR S. and INCE D.: 'Rapid software prototyping'. Open University Technical Report 86/4. (The Open University, 1986)

7 JACKSON M. A.: *Principles of Program Design*. (Academic Press, 1975)

8 LEHMAN M. M.: 'A further model of coherent programming processes'. Proceedings of the IEEE Software Process Workshop, Feb. 1984, pp. 27-33

9 OSTERWEIL L.: 'Software processes are software too'. Proceedings of the Ninth International Conference on Software Engineering, Monterey, 1987.

10 PEELING N. E., MORISON J. D. and WHITING E. V.: 'ADAM: an abstract database machine'. RSRE Report No. 84007 (Royal Signals and Radar Establishment, 1984)

11 WINGROVE A.: 'The problems of managing software projects'. *Software Engineering Journal*, 1986, **1**, (1), pp. 3 - 6

12 WOLFF J. G.: 'Simplicity and power: some unifying ideas in computing'. To appear in *The Computer Journal*

10 FIGURE CAPTIONS

Figure 1. A representation in Project SP of the knowledge base of the PCIS project at the time of writing

Figure 2. The New Spiral Model represented using Project SP