Abstract Project Model: A Description Of A Generic Software Project

By Mark Johnston B.Sc.

School of Computer Applications

Dublin City University

Glasnevin

Dublin 9

Supervisor: Professor J.A. Moynihan

A thesis submitted for the degree of

Master of Science

May 1999

Declaratıon

I hereby certıfy that thıs matenal, whıch I now submıt for assessment on the programme of study leadıng to the award of Master of Scıence ın Computer Applıcatıons ıs entırely my own work and has not been taken from the work of others save and to the extent that such work has been cıted and acknowledged wıthın the text of my work

Sıgned: ______________ **ID No.:** 97970638

Mark Johnston

Date: 13-5-1999

Acknowledgements

The funny thing is that when I began this research I thought, this should not be too difficult and I probably will not need too much help In fact to be more wrong would have been difficult In true Oscar winning style I am aware that this thesis could not have been written without the help of a number of splendid characters

Thanks to Robert Cochran for allowing me so much flexibility with my 9-5 life and also for his financial support throughout my two years at Catalyst Software I thank Marty Sanders for being a consistent reality check on the application of this research to a real software tool A big thank you to Jackie Berkery from the CSE for providing me with a validation source for this research when there seemed to be no data points left Thanks to Stephen Brown for being as thorough as possible with the examination of this research This has given me faith in this piece of work Thank you to Renaat Verbruggen for identifying some of the weaker aspects of this research at an early stage Finally, thank you to Tony Moynihan who supervised this research providing words of wisdom when my tender years showed

This research took place within the framework of the P3 project I worked with a number of top class individuals throughout this project especially Brian McCarthy, Rory O'Connor, Eamon Gaffney and Sylvia Kelly, all of whom made me angry, made me laugh or made me see sense I thank you all for a superb 18 months

Abstract Project Model: A Representation of a Generic Software Development Project

Abstract

This thesis records the research component involved when constructing a set of templates that describe the typical software development project This set of templates is a major component in the ***Prompter*** tool The ***Prompter*** tool was developed by a consortium of software developers including Dublin City University, Catalyst Software and Objectif Technologie The development of ***Prompter*** has been partially assisted under the ESSI fourth framework Its goal is to provide decision support to the user in the field of Software Project Planning

In the ***Prompter*** tool the user provides a detailed description of the starting point of their project This detailed description is used to provide recommendations by a built-in critiquing system This value of this advice is based upon the accuracy and relevance of the data that the user provides

This research had two main objectives The first of which was to consolidate a representation mechanism that is flexible enough to provide ***Prompter*** with a detailed description of a user project The second objective was to research the state of the average software project and attempt to categorise these typical software projects according to size and complexity If these projects are deemed to overlap sufficiently, these common characteristics can be added to a model describing all software projects that are operating within these constraints These characteristics can then be added to the tool using the description mechanism described above

Table of Contents

Chapter 4: CURRENT INDUSTRY PRACTICES AND SOURCES OF THE APM VALUES 42

Chapter 5: VALIDATION OF RESULTS 58

List of Figures

List of Acronyms

APM	Abstract Project Model
BS	British Standard
CASE	Computer Aided Software Engineering
CMM	Capability Maturity Model
COCOMO	Constructive Cost Model
CORBA	Corporate Object Request Broker Architecture
EC	European Commission
ESI	European Software Institute
ESSI	European Systems and Software Initiative
ESPRIT	European Software Programme for Research in Information Technologies
EU	European Union
GUI	Graphical User Interface
IDL	Interface Definition Language
IEE	Institute of Electrical Engineers
IEEE	Institute of Electrical and Electronic Engineers
IPM	Instantiated Project Model
ISO	International Standards Organisation
IT	Information Technology
JDK	Java Development Kit
KLOC	Kilo (1000) Lines Of Code
MIS	Management Information Systems
MS	Microsoft
NATO	North Atlantic Treaty Organisation
OMG	Object Management Group
ORB	Object Request Broker
PIE	Process Improvement Experiment
RPM	Refined Project Model
SME	Small and Medium-Sized Enterprises
SEI	Software Engineering Institute
SPICE	ISO TR 15504 Software Process Assessment and Improvement
VASIE	Value Added Software Information for Europe
WWW	World Wide Web

Chapter 1: Introduction

1.1 Introduction

This research was performed in conjunction with the development of a software product known as ***Prompter***   This chapter intends to give an overview of the ***Prompter*** tool as a precedent to describing the research performed   This will be achieved by providing an insight into the framework within which the ***Prompter*** tool has been developed   This framework was the P3 Project   The target users of the end product are described which provides an insight into the type of customer that was targeted   An architectural overview is provided which briefly describes the various components of the ***Prompter*** tool   A functional walkthrough is provided which describes a possible usage session of this tool   Finally, a brief overview of the thesis is supplied

1.2 The P3 Project

The P3 Project was a European Commission sponsored project under the ESPRIT programme involving partners from France, Greece and Ireland   This project had a 30 month duration and was intended to be completed by March 1999   At the end of month 30 there was a packaged software tool that allowed software project managers to obtain decision support with respect to project planning   Decision support in the context of the ***Prompter*** tool refers to providing the software project planner with recommendations based on the project scenario that has been described to the tool by the user   The tool delivered by the project was in fact an operational prototype with some outstanding issues to be resolved before making it available outside the project team   There were a number of interim deliverables during the project schedule which were subject to external evaluation under the control of a EC appointed project officer and his peer review group

The development team consisted of one Irish software company, Catalyst Software Ltd, a French software company, Objectif Technologie and an Irish research institute, Dublin City University In addition, there were two user partners that provided feedback to the development team in relation to the various interim deliverables   These user partners were Intrakom SA, Greece and Schneider Electric, France   The user partners also provided feedback in relation to each of the prototypes that were incrementally delivered   The four prototypes intended to incrementally add architecture, look and feel, functionality, and finally, advice to the user All EC sponsored project support was dependent upon the acceptance of formal deliverables which occurred at six month breaks throughout the project

1.3 Target Users

Prompter is a MS Windows based software product intended for the project manager who wishes to employ best practices while managing a software development project In order to minimise cost, reduce development time and maximise customer satisfaction it is necessary to control many facets of a software project This is ***Prompter's*** area of expertise – best practice within the context of a specific software project The tool is primarily targeted at the manager of an exclusively software project who wishes to follow the roadmap of known quality standards such as CMM, ISO, SPICE etc Because ***Prompter*** is a training tool, the target user is the novice project manager or even student user in *training mode* The user typically wishes to examine *what-if* situations and make the best possible decision at any stage in the project In this way a project manager can improve practices within the framework of a real project This tool is primarily suited to the manager of a level 1 CMM [1]organisation striving to introduce basic management to achieve a stable process with a repeatable level of statistical control

1.4 Architectural Overview

An architectural overview is provided in this section Figure 1 1 on the following page shows the principle architectural components of the ***Prompter*** tool at a high level ***Prompter*** is composed of three major components GUI, Kernel and Daemons The **GUI** component manages user interactions with the tool This part of the tool processes all user input and selections and passes them to the Kernel The **Kernel** is responsible for managing the storage of a user's project description and assembles advice to be returned to the user The **Daemons** analyse the user's project description and creates advice based upon the state of the user project The Daemons use their critiquing system to analyse the project description constructively These three components are described in greater detail below Not only are these components distinctly unique in their functionality they were also developed by different organisations within the P3 project These three components were connected using an interfacing standard known as CORBA The three components, GUI, Kernel

[1] Level 1 CMM is a classification of organisation maturity identified by the SEI

and Daemons were implemented using the Java programming language These technologies are described below

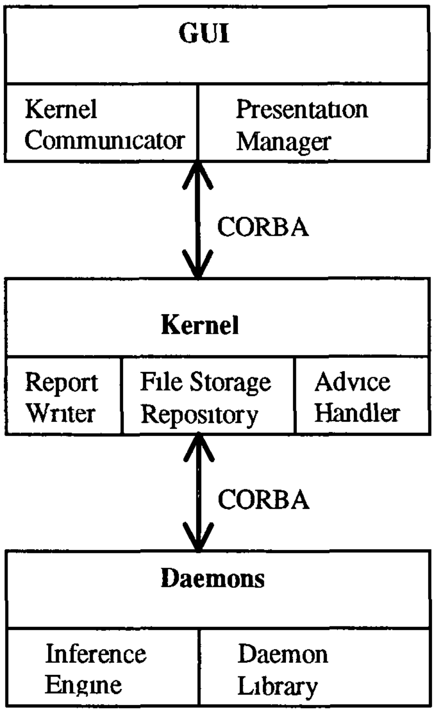


Figure 1.1: The above diagram illustrates a view of the architecture of the ***Prompter*** tool at a high level

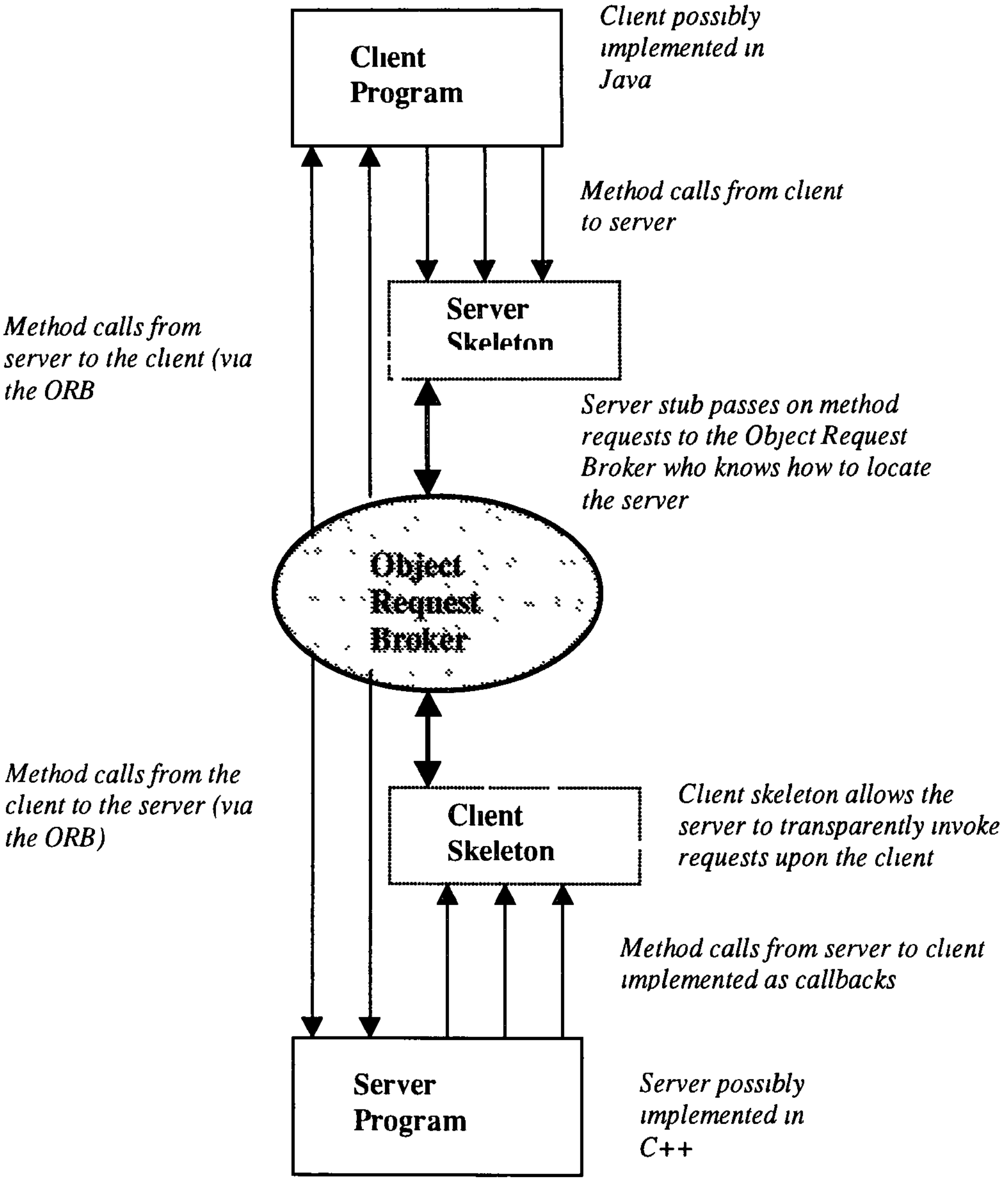
1.4.1 The CORBA Standard

The Corporate Object Request Broker Architecture (CORBA) is a standard which is managed by the Object Management Group (OMG) The OMG is composed of more than 500 software organisations concerned about standardising distributed object communication CORBA is essentially a software layer that allows possibly remote software components to communicate (Iona, 1997) This communication is achieved via an Object Request Broker (ORB) This ORB is a middleware software component that acts as an intermediate between clients and servers The CORBA architecture is in fact an extension of the traditional client-server approach but allows this form of communication with the following benefits

- Clients and Servers may be distributed outside of the local network
- Clients and Servers may be executing on different hardware platforms
- Clients and Servers may be executing on different software platforms

- Chents and Servers may be ımplemented ın dıfferent programmıng languages

In order to develop software usıng CORBA, ıt ıs necessary to employ an ORB to perform all necessary transactıons between chents and servers Propnetary and non-propnetary ORBs are avaılable for commonly used programmıng languages such as C, C++, Vısual Basıc, etc The next step ınvolves defimng an ınterface between the chent and server



Chent Program
Chent possıbly ımplemented ın Java
Method calls from chent to server
Server Skeleton
Method calls from server to the chent (vıa the ORB
Server stub passes on method requests to the Object Request Broker who knows how to locate the server
Object Request Broker
Method calls from the chent to the server (vıa the ORB)
Chent Skeleton
Chent skeleton allows the server to transparently ınvoke requests upon the chent
Method calls from server to chent ımplemented as callbacks
Server Program
Server possıbly ımplemented ın C++

Fıgure 1.2 : The above dıagram shows the topology of how a system usıng CORBA communıcates.

CORBA allows these clients and servers to interface using the IDL (Interface Definition Language) These interfaces are described in a hardware and software neutral format This format allows the initial definition of the interfaces between components followed by the development of each side of the interface independently This provides the developers with a great deal of freedom as the interface can be constructed independent of hardware or software nuances This interface can be used in conjunction with the ORB to create a number of *stubs* which serve as communication agents between the developer written code and the ORB

In Figure 1 2 on the previous page, the definition of an interface using the IDL leads to the creation of both server and client stubs (also known as server and client skeletons) A set of programs, one for client and one for server are written Any requests from the client to the server or vice versa are forwarded to the Object Request Broker by the client/server skeleton The ORB knows where to locate the server or client and hence forwards the requests to the relevant party The ORB is responsible for the following (Iona, 1997)

- Registration of servers
- Management of operating system resources
- Underlying communications and synchronisation
- Error detection
- Faithful transmission of requests

1.4.2 The Java Programming Language

Java is emerging as one of the most important development platforms is use today In a survey conducted by EXE Magazine (Bennett, 1998b), 16% of the 311 respondents said that they are using Java as a development language This is behind C++, Basic and Pascal, all languages that have been available as development languages for the last decade or more Java was initially conceived by researchers at Sun Microsystems but really only came to the fore as a usable software development tool in 1997 This was because version 1 1 of the JDK included functionality that made Java more usable as a commercial development tool The language builds upon the object oriented paradigm and employs a similar syntax to C++ which allows developers to make the move to Java quite gracefully However, in principle the main selling point of Java is its platform independence This means that software can be developed in Java on one machine and may run on any other machine that is equipped with the Java virtual machine. This Java virtual machine is a runtime environment that interprets 'bytecodes'[2] *on the fly* instead of running native executable code which is platform specific. Java initially appeared useful only for multimedia applications running within web browsers. This is partly because the first release of the JDK restricted the applet in its ability to access the local machine. This image has also been largely as a result of the reduced execution speed of Java due to the interpreted bytecodes. However, with the development of optimised compilation tools and proprietary libraries offering efficient solutions, the Java development platform is appearing more attractive to organisations who wish to do more than deploy their applications across the internet.

1.4.3 GUI

It is desirable to provide desktop software tools with a user interface that is both intuitive and familiar. Familiarity with the user interface was achieved by following recommendations for MS Windows 95. The user interface of ***Prompter*** is designed to allow the user to interact with the tool using the typical commands that an MS Windows product would provide. The GUI is in fact responsible for capturing all user data and requests, formatting these and passing all information to the Kernel. The user interface also performs preliminary checking upon the data to ensure that invalid values are not dispatched to the Kernel.

1.4.4 Kernel

The Kernel provides a mechanism for maintaining the user's project description both in dynamic form while a session with ***Prompter*** is in progress, and in static form, committing the project description to persistent storage. The Kernel is also responsible for loading configuration information for the GUI at startup. The Kernel is instrumental in providing the communication mechanism between the GUI and the Daemons. Any requests made by the user via the GUI for processing are passed initially to the Kernel and forwarded to the Daemons if appropriate. Because the Daemons component of the tool is responsible for creating advice, the user's advice requests need to be transmitted to the Daemons and the returned advice needs to be forwarded to the user via the GUI. In this way, the Kernel provides a level of indirection between the GUI and the Daemons.

[2] Bytecodes are an architecturally neutral form of binary code which allows instructions to be defined universally and translated at runtime into a machine -specific format.

1.4.5 Daemons

The Daemons in ***Prompter*** consist of a number of advice agents which are each able to critique a certain aspect of the user's project description Each one of these agents is a mini-expert capable of providing advice about a certain aspect of project management (Gaffney, 1999) There are daemons responsible for the areas of

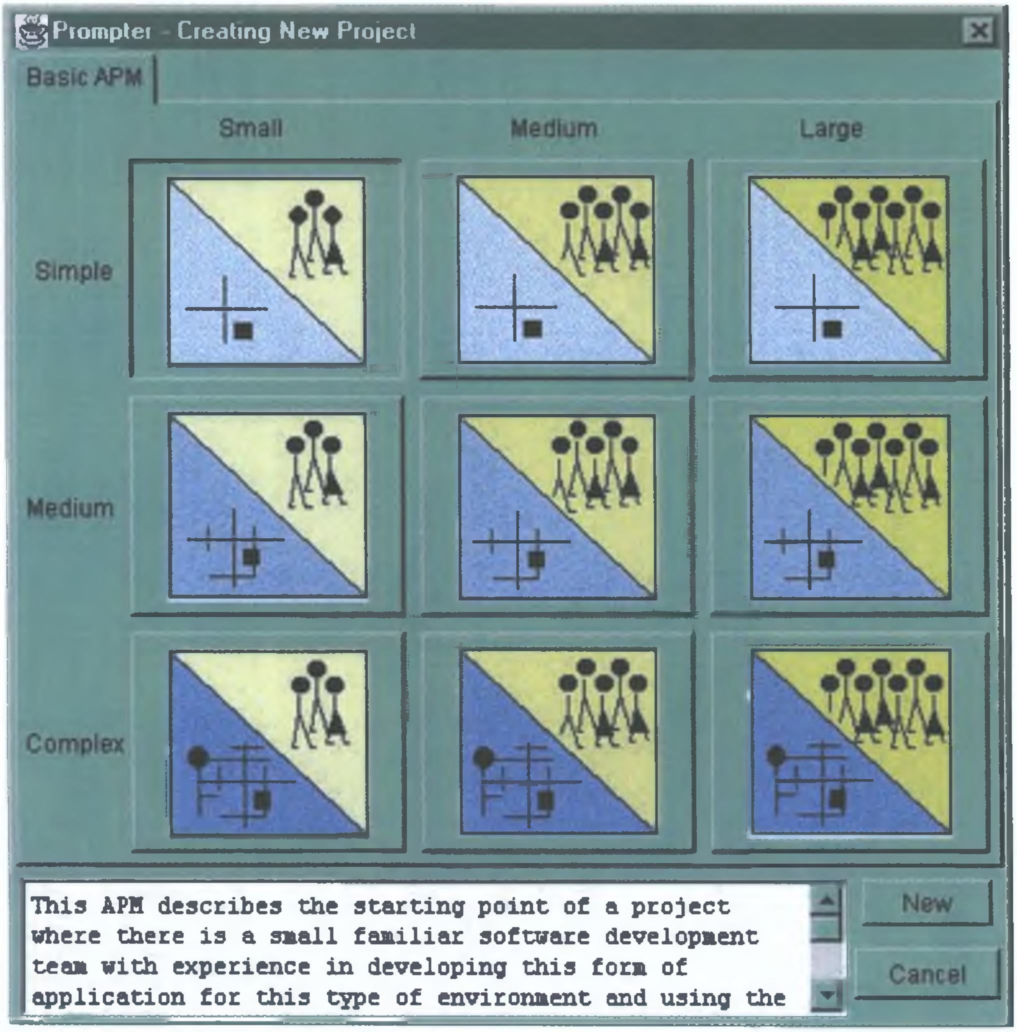
- Analysis and Planning
- Estimation
- Activity Planning
- Resource Allocation
- Project Re-planning
- Measurement
- Risk Management

These daemons have the ability to critique the project description provided by the user and may suggest a sensible alternative in the form of advice This occurs by the execution of a set of rules, in their most basic form as simple *if - then* statements These rules examine facets of the user's project description, which has been provided by the user at this point For further exploration of how the user provides this information see the following section

1.5 Functional Overview

This section is intended to give a walkthrough of the functionality of the ***Prompter*** tool There are a myriad of possible use cases for this tool as there are different classes of users for this tool as well as different project descriptions and advice formats that can be provided for each project A generic use case is emphasised here to allow the reader to understand what a session with ***Prompter*** involves A more detailed set of use cases is provided in Appendix B This set of use cases was included in the User Requirements document for the P3 Project

The user begins by either opening a project that is in progress or by creating a new project Consider the user who is new to ***Prompter*** and has not previously created a project using this type of decision support tool beforehand When the tool begins, the user is presented with a set of models from which to select the one most appropriate to their particular project This is in fact the scope of this research, to create these starting point models for the user The user's selection criteria may be based on project size and project complexity as can be seen in the screen shot shown in Figure 1.3. It is important for the user to understand what these two parameters imply in order to make a suitable selection.



This APM describes the starting point of a project where there is a small familiar software development team with experience in developing this form of application for this type of environment and using the

Figure 1.3: The above screen shot shows the APM window that allows the user to select an appropriate model for their project.

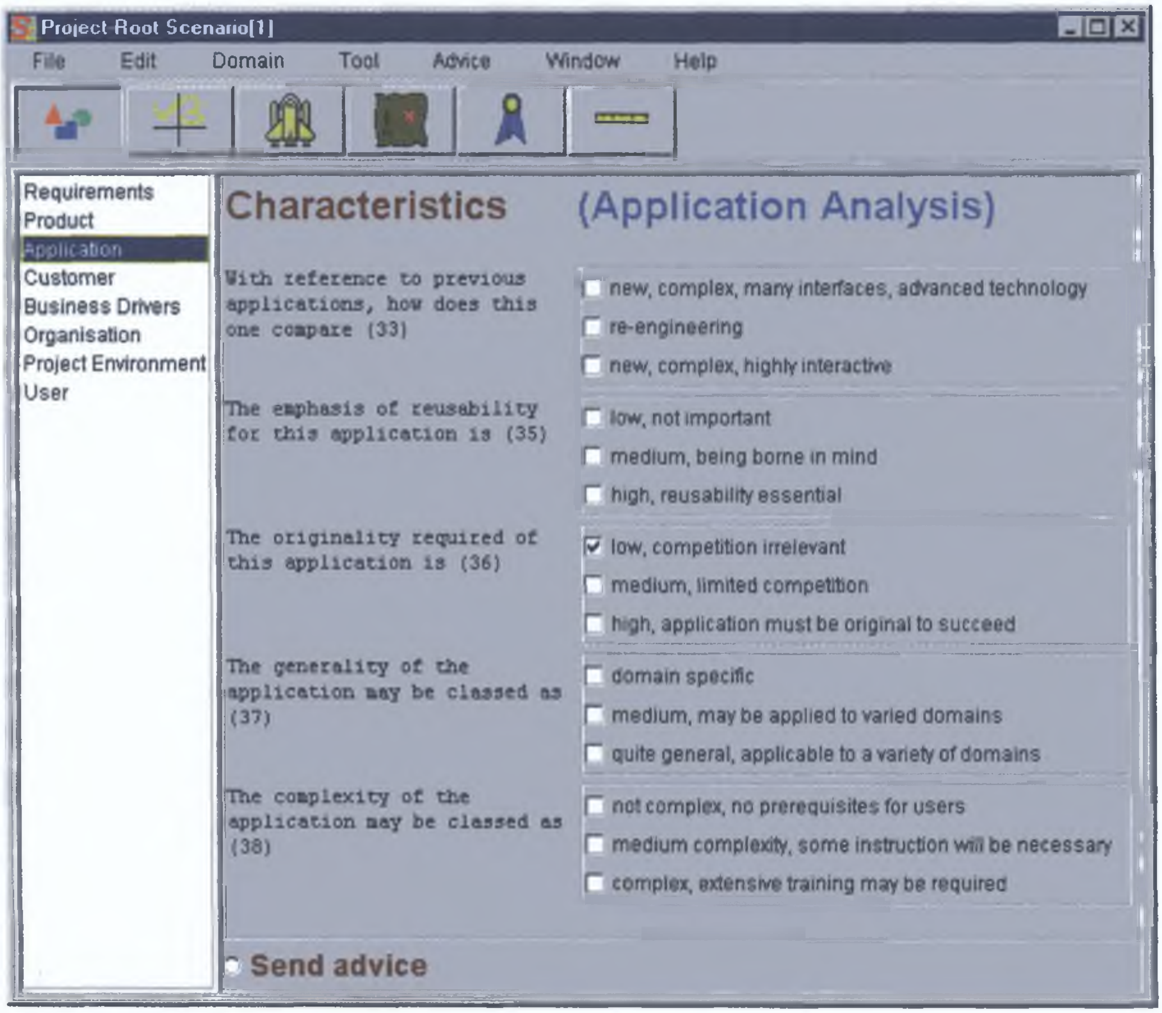
Figure 1.3 displays nine distinct models from which the user selects the project description most appropriate to their project. The suitability of each model can be evaluated by selecting the relevant model and reading the description provided in the dialog box at the bottom of the window in Figure 1.3. Having made this choice of starting point for the project the user now can add some project specific data to this template. As more information becomes available to the user about their project, the initial description becomes refined by the subsequent addition of data as it becomes available to the project planner. The user adds to the project description by answering a series of questions posed by the tool.

Domain	Subdomain	Token ID	Question
Characteristics	Requirements	12	How do the requirements compare in relation to what we are accustomed to
		13	How much change are product requirements likely to be subject to during the course of development
	Product	9	Evaluate the interfaces between the product and other software/hardware components
		84	What is the level of portability expected of the product
	Business Drivers	59	What is the level of market competition facing this product
	Customer	92	What is the level of software usage experience at the customer organisation
		26	How well does the client understand the requirements
	Application	36	How important is reusability for this application
Project	Physical Resources	48	Evaluate the equipment at the disposal of the development team
	Human Resources	52	Describe the teams experience of professional software development
	Estimating	78	Describe the standard of estimating at your organisation
Quality		61	Is your organisation capable of CMM level 2 compliance
		60	Is your organisation capable of ISO 9001 compliance
		72	Describe the standard of quality expected of subcontractors

Table 1.1: The above table shows an example of the type of questions posed by the *Prompter* tool when obtaining a user's project description.

These questions are classified into domains such as characteristics, project, quality, metrics, etc. These domains are further subdivided into sub-domains which contain a set of questions about a specific aspect of project planning. To give a feel for the type of information this questions and answers session intends to elicit, Table 1.1 on the previous page provides a suitable illustration.

In Table 1.1 above, the domain column is a top-level categorisation of concepts which may be associated with project planning. Each domain may be broken down into a number of sub-domains which contain specific questions about project planning. The user answers the questions posed by the tool by selecting the most appropriate option from a short list of qualitative terms such as [high, medium, low]. A screen shot in Figure 1.4 on the following page shows, the Scenario window of the ***Prompter*** tool with the user answering questions about a particular aspect of their project.



With reference to previous applications, how does this one compare (33)

□ new, complex, many interfaces, advanced technology
□ re-engineering
□ new, complex, highly interactive

The emphasis of reusability for this application is (35)

□ low, not important
□ medium, being borne in mind
□ high, reusability essential

The originality required of this application is (36)

☑ low, competition irrelevant
□ medium, limited competition
□ high, application must be original to succeed

The generality of the application may be classed as (37)

□ domain specific
□ medium, may be applied to varied domains
□ quite general, applicable to a variety of domains

The complexity of the application may be classed as (38)

□ not complex, no prerequisites for users
□ medium complexity, some instruction will be necessary
□ complex, extensive training may be required

Figure 1.4: Screen shot of the Scenario window of *Prompter* with the user answering questions within the Application sub-domain of the Characteristics domain.

The user now has the option of creating alternative descriptions for their project. For example, a project manager may wish to evaluate the impact of increasing the size of their project team. Rather than changing the size of the project team, the user may create a clone of their project description and change the team size in the clone. In this way the user does not lose the project description when seeking to evaluate alternatives. Because a large part of planning is impact evaluation and prediction, it is necessary to be able to observe the possible effects of such changes without committing resources at such an early stage. The ability to create a clone of your current project descnption (can be thought of as a snapshot) is provided by ***Prompter*** These cloned project descnptions are known as scenanos and often represent a decision pomt m a user project at which the user wishes to evaluate multiple outcomes of a decision These scenanos are arranged by the tool m a tree-like fashion Scenanos can be added or deleted by a user so that discarded or unwanted scenanos can be removed from the project descnption The scenanos that are maintained will represent the path to the refined project descnption In the Figure 1 5 below, it can be seen how scenanos created m the order A-H are related

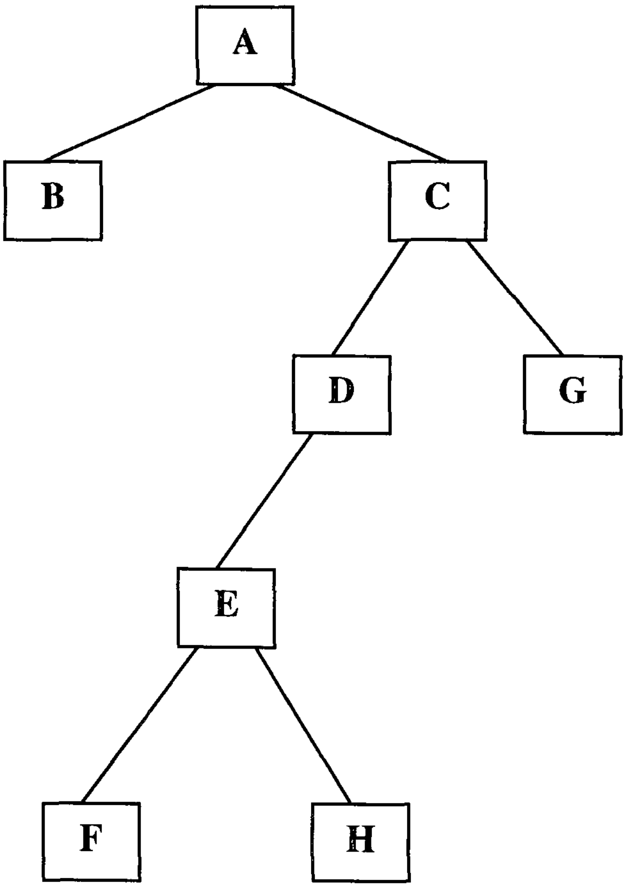


Figure 1.5: The above tree depicts the relationship between a number of Scenanos m *Prompter.*

The most recently created scenano can be considered scenano H If the user decides that some of the scenanos have become obsolete, these unneeded scenanos can be removed by *prunmg* the scenano tree This may give nse to the scenano tree m Figure 1 6 on the following page

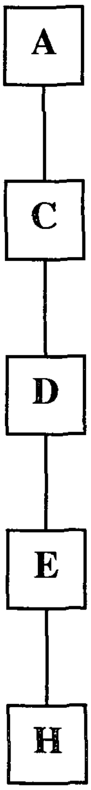
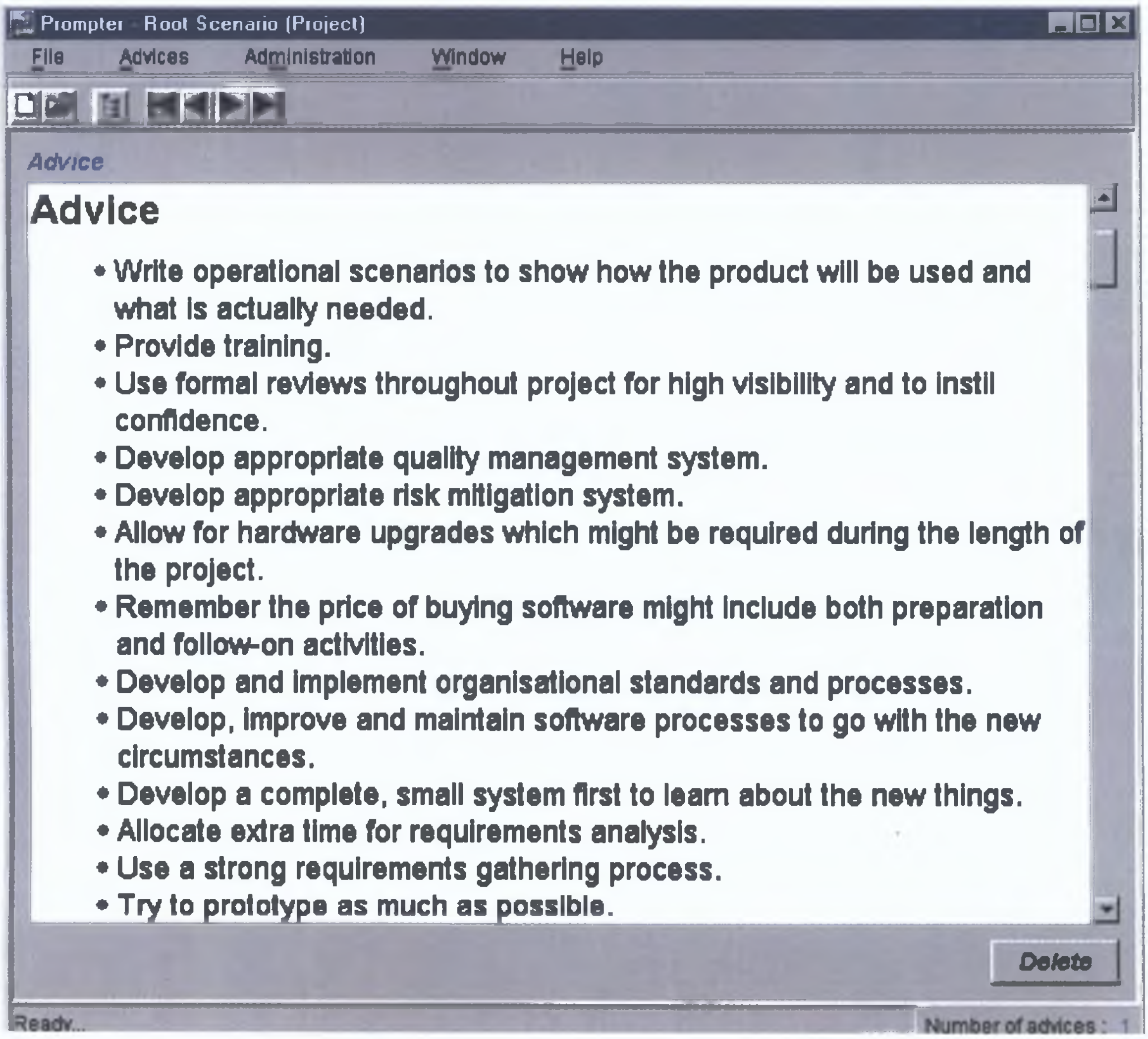


Figure 1.6: The above Scenario tree shows the result of a user pruning the tree in Figure 1.5.

Finally, advice can be requested by the user at any stage in the project A user can select general advice that is based upon documented best practices that is not related to any particular project Some of the advice is taken from sources of accepted best practices in software project management literature However, the larger part of the advice is taken from knowledge held by members of the project with over twenty years experience in software project management The second form of advice is based on the project description that has been provided by the user This advice is provided by the Daemons section of the tool which analyse the user's project description and make suggestions based upon this (Gaffney, 1999) This is why the user provides such a detailed project description to the tool the richer the project description, the more useful and relevant the advice will be The user will subsequently accept or reject the advice provided by the Daemons If the user accepts the advice, changes may be made to the user's project description in light of the Daemon's recommendations This is how advice is beneficial to the user The screen shot provided below shows the Advice window of ***Prompter*** This window shows a set of advice provided by the tool according the user's project description. This advice appears a little superficial but the user has the ability to obtain a justification for this advice. The functionality allows the user to find out why such recommendations were made by the tool. This is an important feature of the tool - not only are suggestions made to the user but also an explanation is provided as to why these suggestions were considered necessary.



Advice

- Write operational scenarios to show how the product will be used and what is actually needed.
- Provide training.
- Use formal reviews throughout project for high visibility and to instil confidence.
- Develop appropriate quality management system.
- Develop appropriate risk mitigation system.
- Allow for hardware upgrades which might be required during the length of the project.
- Remember the price of buying software might include both preparation and follow-on activities.
- Develop and implement organisational standards and processes.
- Develop, improve and maintain software processes to go with the new circumstances.
- Develop a complete, small system first to learn about the new things.
- Allocate extra time for requirements analysis.
- Use a strong requirements gathering process.
- Try to prototype as much as possible.

Figure 1.7: The Advice Window of *Prompter* is shown above. This advice has been offered by the Daemons component of the tool as a result of the project description provided by the user.

1.6 This Research in the Context of the P3 Project

At this point, a description of the ***Prompter*** tool and a typical walkthrough has been provided As the tool provides decision support to software project managers which is based upon a description of a user project, it is necessary to obtain a project description in as clean and faithful a manner as possible This means that there is a need to create a model in the tool by which this project description can be represented This leads on to my first responsibility in the P3 project - to construct such an initial model known as the project description template and refine this model in parallel with the development of ***Prompter*** This model is a description mechanism for the characteristics of a user's software project Creating such a model requires identifying the type of information to be represented, the ideal representation mechanism and finally establishing its worthiness

Prompter aims to provide decision support in the area of software project planning Such decision support is provided to the user in the form of textual advice that appears dynamically when using the tool In order to provide such advice in a project-sensitive manner, it is necessary to obtain a large amount of information from the user describing their project This is a time-consuming task and led to fears that the *pain vs gain* ratio would be such that the user would find the tool difficult and uncomfortable to use For the purposes of ergonomics, it was conceived that this tool should provide a starting point description that diminishes the responsibility upon the user to enter information that could be reasonably inferred This starting point is in fact a description of a generic software development project When this idea was first conceptualised, there had been no research to verify the feasibility of constructing such a generic project description This initial wish-list item in the system requirements of the ***Prompter*** tool became an integral component of the tool through my research into this area From this concept, I constructed a set of Abstract Project Models that characterise the starting point of the typical software development project The Abstract Project Model (APM) will be described Chapter 3

1.7 Overview of Thesis

The ***Prompter*** tool has now been described and the context of this research within the development of this product has been revealed The remainder of the thesis will deal with the following

Chapter 2 describes the market place and the its need for a product such as ***Prompter*** This involves accurately describing the category of software tool which ***Prompter*** competes with This analysis of the domain of the ***Prompter*** tool culminates in the revelation that there is no direct competitor to ***Prompter*** in the marketplace at present

Chapter 3 describes the concept of the Abstract Project Model and it's design This involves identifying the type of data that the APM seeks to model and the way in which this information can be represented The initial research to create the *token* data type is also described

Chapter 4 aims to provide a description of the seven primary sources examined when performing this study The model by which a user may describe their project is outlined in Chapter 3 The data which must be added to this model is introduced in this chapter Sources identified by this research are also described m Chapter 4 The validity of each source is evaluated and any problems encountered when performing this research are stated

Chapter 5 describes the process by which this research is validated This involves a description of how the Project Description Template and the APM set were evaluated

Chapter 6 provides the conclusions arrived at following this research.

Appendix A presents the Token Data Dictionary This document is a formal deliverable from the P3 Project which embodies the work performed to create the project description template

Appendix B provides the use cases which were a component of the User Requirements document of the P3 Project

Chapter 2: The Argument for *Prompter* and the Abstract Project Model

2.1 Introduction

This chapter seeks to illustrate that there was a need for a tool to provide decision support to software project managers and that there was also an opportunity for the success of such a tool. There shall be a logical explanation as to why software vendors must streamline their process in order to remain competitive. The manner in which the ***Prompter*** tool assists with software process improvement is described. The scope of the tool is then revealed which shows the area towards which ***Prompter*** was aimed. The use of CASE tools by software developers is then investigated. This is followed by a brief description of tools which provided functionality in areas related to ***Prompter***. This shows that despite the presence of various tools there was no existing package that provided the functionality of ***Prompter*** and that the features of the APM were unique.

2.2 Increasing Complexity of Software Production

The application of computer technology in every aspect of life has become a norm. The technological revolution involving computer systems began in earnest in the 1950s (Boehm, 1981). Since then, computer systems have become an essential part of everyday life and perform many of the mundane tasks that were once considered both trivial and time consuming. This has been reflected by a growth of reliance on computers and software as shown in Figure 2.1 (Boehm, 1981). It is accepted that some of the references may appear to be out of date but a search for more recent data points was fruitless.

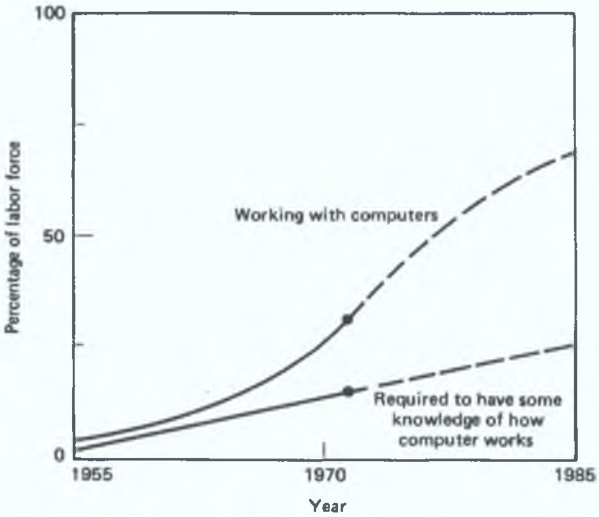


Figure 2.1: The above diagram shows the dramatic increase in the use of computer systems between 1955 and 1985 (Boehm, 1981).

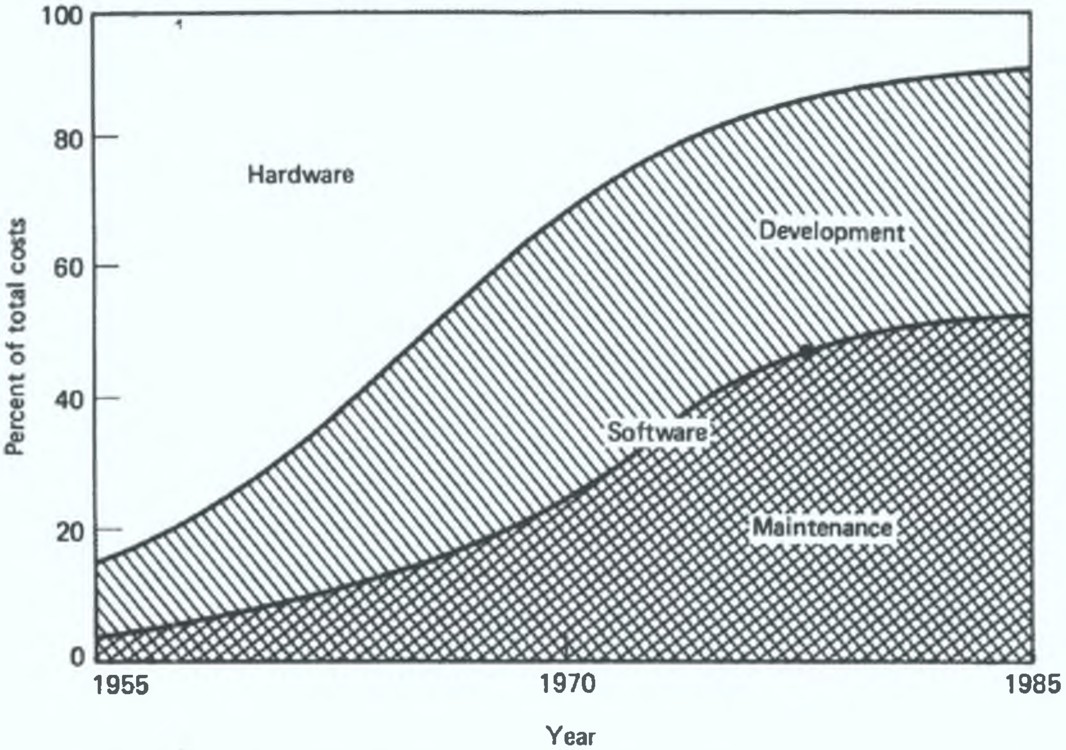


Figure 2.2: The above diagram indicates the changing profile of software: hardware costs also implying the increasing costs of maintenance to development (Boehm, 1981).

Not only has the dependence upon computer systems in general increased, the profile of the demand on computer software in relation to computer hardware has changed dramatically. This change has followed the typical pareto profile changing from 80:20 to 20:80 for the cost of hardware to software. See Figure 2.2 above (Boehm, 1981).

With the advent of silicon technology hardware prices have plummeted further aiding the demand for computer systems. An example of the changing cost of computer hardware is that in 1962 a typical mini-computer cost about $US 20,000. A typical PC today would cost little more than $US 1000 (History, 1997). The cost of software in turn has soared due to the rising complexity and size of software solutions. It was not until the software crisis was identified by the NATO Science Committee in 1968 (Schach, 1990) that the complexity of creating large scale software systems was taken seriously. As systems grew, the search for a solution to the software problem appeared to become more earnest. This seemingly endless search culminated in the belief that there is *no silver bullet* for the difficulties of software development (Brooks, 1986) Having placed faith in the promises of new technology after new technology it finally became clear that a set of 'best practices' offered the only realistic way forward (Wasserman, 1996) This implies that no one particular practice will overcome these age old issues but an adoption of a set of practices and activities that address individual problems associated with software development (McConnell, 1997)

This realisation that streamlining software development activities according to recommended best practices has really been a 90s phenomenon (Yourdon, 1996) It would be incorrect to assume that the awareness of software development hazards was non-existent until 1990, but it was not until this point that the community at large became concerned with process improvement This has led to the development of software specific standards[1] which define a business as having the capability to produce a reliable software component These accreditations such as CMM, ISO, BS, etc are standards that are recognised internationally Primarily, these standards establish that a software vendor has the capability to produce a reliable product However, they also seek to assist the vendor with their productivity This improved productivity is expected to emerge from more informed project management techniques that are recommended by these standards The better practices that result from pursuing such standards intend to allow the software developer to balance many conflicting interests such as

- Higher productivity
- Lower costs
- Higher quality
- Higher maintainability
- Shorter time to market
- Maximising reusability

The realisation that Software Process Improvement is an effective way to abandon the chaos of disorganised software production is a contemporary issue

If a software organisation cannot balance the conflicting interests outlined above, the vendor will cease to remain competitive This implies that best practices programmes are an embodiment of

[1] In some cases accreditation for software development was specified for existing standards

the type of activities that an organisation should seek to perform in order to remain competitive (ESPITI, 1996)

As noted by Terry Rout at SPI '98[2], 'some organisations feel that such standards are too bureaucratic and restrictive to be used practically, particularly in SMEs The need to remain competitive is often a motivational factor for management to promote software process improvement Table 2 1 below strongly implies that there is a good business case for most companies to follow the SEI CMM process improvement approach

Category	Range	Median	No. of Orgs.
Years of effort	1-9	3 5	24
Process improvement cost $ / person	$490-$2,004	$1,375	4
Productivity gain / year	9%-67%	35%	4
Early defect detection gain / year	6%-25%	22%	3
Time to market gain / year	15%-23%	19%	2
Post-release defect reduction / year	10%-94%	39%	5
Savings / cost ratio	4 0-8 8	5 0	5

Table 2 1: The above table suggests that there is a good business argument in favour of software process improvement (Yourdon, 1996).

These results shown in Table 2 1 above are based upon a small number of organisations, all based m the US Peter Goodhew provided similar results for the European Software Industry which presented results from over 360 software development organisations throughout Europe (ESPITI, 1996) This survey of European organisations evaluated the productivity of the organisations against the software process maturity of the organisations The performance of the organisations varied dramatically The more mature organisations achieved development productivity in excess of 25 function points per person month and removed over 95% of defects before product delivery Their estimations were often consistent to within 10% of actual cost and duration of the project In contrast, the worst organisations had a development productivity below 5 function points per person month and remove less than 50% of defects before delivery Their projects often exceeded estimated by more than 40% These results show that organisations with a more mature software process can achieve higher levels of productivity

[2] SPI '98 was a conference on Software Process Improvement held in Monte Carlo, December 1998

2.3 *Prompter* and the Software Process Improvement Approach

Havmg outlmed the busmess advantage to makmg software process improvements it is now necessary to show how ***Prompter*** relates to this concept The ***Prompter*** tool provides decision support to software project managers In this way ***Prompter*** recommends that the user follow the recommendations of documented best practices and apply these practices to their projects ***Prompter*** uses its knowledge base to advise users about how to make decisions at the project level that should have an impact at a busmess level by streamlmmg the software production process Over the past decade, contributions have been made by a variety of domain experts to the software process improvement arena ***Prompter*** is equipped with a set of daemons (Gaffney, 1999) These daemons are used to give advice to a user at any time based upon the description of the project that has been provided by the user

The software producer is aware that the user demands the best from the developer and if this level of desired quality is not provided, the user may cancel their request or worse - offer their busmess to a competitor This is ***Prompter's*** niche assistmg a software producer to remain competitive by improving their software production process within the framework of real projects One of the primary user requirements of ***Prompter*** is to provide decision support to a project manager of a level 1 CMM organisation and to provide advice that will assist such a user to reach level 2 This requirement was mcluded in the User Requirements Document of the P3 Project which was a deliverable to the EC as part of the project contract It is logical to address this section of the market as approximately 80% of software development organisations are at the initial level (Yourdon, 1992)

2.4 Scope of the *Prompter* Tool

In order to define the scope of the ***Prompter*** tool, it is necessary to consider the full categorisation of software engineering tools As stated in the User Requirements Document of the P3 Project software engineering activities can be divided into four broad categories

1 The activities within the software development process (i e within the life cycle) - requirements, design, coding, etc

2 The support processes which are carried out in parallel to the development process - configuration management, resource management, quality assurance, etc

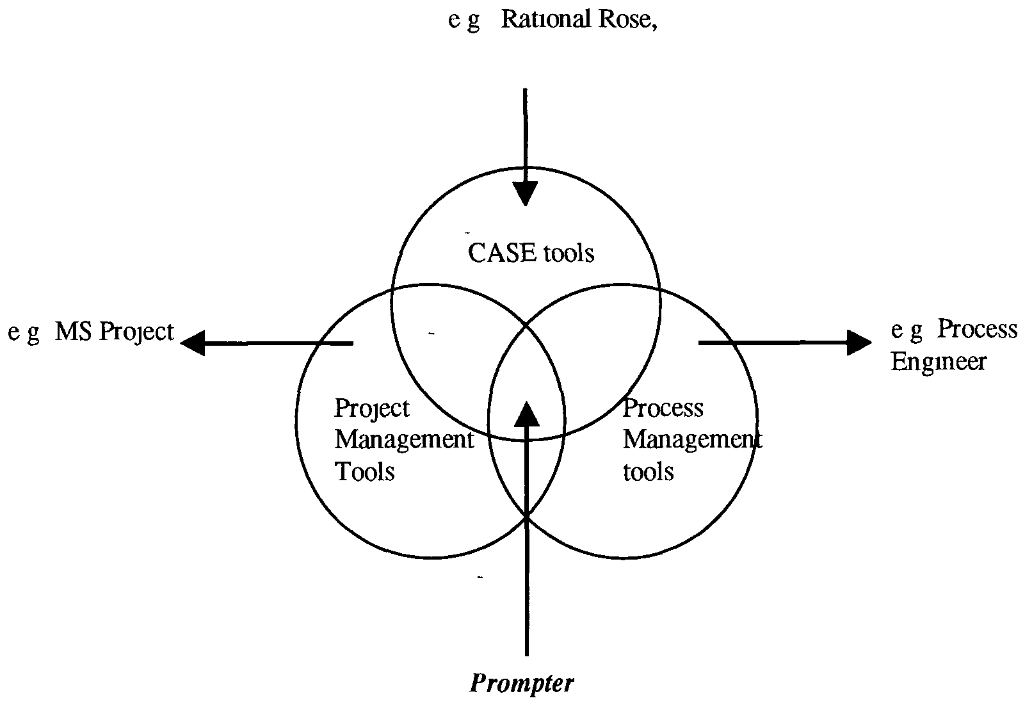


Figure 2.3 The above diagram illustrates the area of software engineering in which the *Prompter* tool operates.

3 Project management activities, which start before the development and support activities, continue on in parallel to them, and beyond them - scheduling, cost and effort allocation, project tracking, etc

4 Process management activities, which start even earlier and continue even longer than the project management activities Definition of and/or selection of the appropriate process, definition of tasks, roles, metrics and analysis of the process

Prompter is not concerned with the design or writing of code or providing automated assistance for supporting activities such as tracking and scheduling However, ***Prompter*** is directly concerned with decision support for process management and project management while not directly providing for these areas themselves Many tools are available that provide for activities such as quality, configuration management, project tracking, etc Examples of such tools are KnowledgePLAN from SPR, SLIM from QSM and ProjectView from Artemis No single tool offers the unique overlap shown in Figure 2 3 This has also been shown in the Technology Implementation Plan document which was a P3 Project deliverable to the European Commission

In Figure 2 3 above, the domain of software engineering tools encompassing CASE tools, project management tools and process management tools is illustrated Examples of project management and process management tools are provided in section 2 6 The examples of CASE tools used for the software development process are not described as such products are not in competition with ***Prompter*** These tools, Rational Rose from Rose Software and Borland C++ from Borland are both intended for the software development process and used for the direct production of software encompassing the areas of system design and implementation

Prompter is concerned with one aspect of the overlap between process and project management, namely decision support for the planning and potentially re-planning components within them It does not aim to cover all aspects of either project nor process management For example ***Prompter*** does not assist the user in creating a detailed project plan showing milestones as in MS Project or tracking resources as in Juggler from Catalyst Software In order to provide such decision support, it is however, necessary to overlap with aspects of project management and tracking Such an overlap involves ***Prompter*** providing recommendations for the type of skill mix a team should have This recommendation would be based upon the project description that the user has provided

2.5 The Use of CASE Tools to Streamline the Development Process

The use of CASE to support the development of software systems has become an essential part of a developers arsenal This view can be somehow misleading as over reliance on tools without the underlying process is a naive move that is littered with pitfalls that can add to costs and extend deadlines (Humphrey, 1989) CASE is not needed by level 1 organisations, they have more fundamental needs However, this warning has not stemmed the increasing dependence upon CASE In Chris Pickering's 1996 Survey of Advanced Technology, 52 6% of respondents use CASE to aid the development process In this survey, over 37% of respondents felt that the greatest factor preventing the use of CASE was that the benefits were not demonstrated (Pickering, 1996)

This interest in using CASE has also spilled into project management This is evident at a very general level in the number of MS Project and MS Schedule users among project managers There also exist some software-specific project management tools providing features which could be considered in competition with ***Prompter*** In Table 2 2, a classification is provided of the primary features found in project management tools This represents the functionality found in most software project management tools. It was deemed necessary by the EC to examine existing tools within this domain despite the fact that they are not direct competitors to ***Prompter***. It was felt that the interest in such tools indicated an awareness of the need for more rigorous software project planning. It also showed that there was no direct competitor to ***Prompter***. This was reported in the Technology Implementation Plan of the P3 Project.

2.6 Existing Tools

The tools that are currently in the marketplace that can be considered to provide functionality similar to that provided by ***Prompter*** are as follows:

- Open Plan by Welcom Software Technology
- Project Planner by Primavera
- Process Engineer by LBMS
- SELECT Process Mentor by SELECT Software
- Project Scheduler 7 by Scitor
- ProjectView by Artemis
- Risk+ by ProjectGear Inc
- KnowledgePLAN by SPR
- SLIM – Estimate by QSM
- IntraPlan by Intra2000

Many of these tools pride themselves on features that ***Prompter*** also offers. Almost all of the above offered

- Scheduling/Planning capability
- Customisation of the User Project
- Risk Analysis
- What if capabilities
- Reporting abilities
- Multi-user/groupware

These competitors do not offer Dynamic Advice based on the content of a user's project description or the ability to evaluate alternatives within a project via scenario analysis. From the point of view of this study it should also be mentioned that none of these competitors provide a starting point project description such as the APM. Both of which are two of ***Prompter***'s primary features and unsurprisingly are the features that distinguish ***Prompter*** from its competitors It can be concluded that there is no direct competitor to ***Prompter*** as none of these alternative tools possess the key features on which ***Prompter*** is based

Feature	**Description**
Scheduling	Scheduling is a key activity for project managers and the sophistication of the algorithm affects the usefulness of the product Aspects which should be taken into account include, task priorities, multiple schedules, fixed date, as soon as possible and as late as possible
Resource control	Initial and ongoing control of the resources applied to a project is a key element of project management Typically, tools assist with allocating and monitoring resources
Cost monitoring	Information regarding actual and estimated costs should be captured, such as, timesheets, committed costs, cash flows, borrowing needs, etc
Progress tracking	A wide variety of metrics are available for tracking the progress of a project against its plans Products normally support a variety of these types such as, percentage completion for time, cost or work, estimation of end date or cost and baseline comparison for time or work effort
Reporting features	A varied reporting mechanism is essential and should include a variety of reports such as, milestone report, variance report, status per task/team member, etc
Multiple projects	In many organisations, a project manager may be responsible for more than one project and will require software to handle aspects such as, prioritisation between projects, splitting projects, merging projects, staff/resource sharing and viewing consolidated information
Charts	A variety of charting mechanisms is desirable, such as, Gantt, Pert, Work Breakdown Structure, resource, etc
What-if capabilities	A common requirement for project managers is the ability to investigate the effects of potential changes in the situation of a project They may need to see the effects of adding or withdrawing a particular resource
Data import/export	In certain circumstances users may wish to import or export data to other packages

Help facilities	There are a number of aspects to help including, online tutorials, Internet support, on-screen context sensitive help
Networking	More and more organisations require packages to operate in a network environment and to allow for concurrent users

Table 2.2: The above table shows the various features expected of a project management support tool. This table is taken from the Technology Implementation Plan of the P3 Project.

Prompter has features that enable it to distinguish itself from other tools in the same area This was shown by the Technology Implementation Plan of the P3 Project This document was approved by the EC as a valid competitor analysis Again, it can be pointed out that none of the products evaluated above provide a baseline project description such as the APM which acts as a starting point to the user's project description This feature is therefore an important asset to the ***Prompter*** tool and further enables it to distinguish itself from the available tools described above This point not only indicates the value of the ***Prompter*** tool but also justifies the role of this research in conjunction with the P3 project

Chapter 3: The Concept of an APM

3.1 Introduction

The aim of this chapter is to define the APM (Abstract Project Model), discuss it's format and to show how it fits into the ***Prompter*** tool This is achieved by beginning with a high level description of the APM in the context of the ***Prompter*** tool The *raison d'etre* of the APM is approached with respect to the advantage to the user by employing the APM in ***Prompter*** The knowledge representation technique for the APM is illustrated by showing how the default values are accessed and used by the tool Finally, a description of how the APM will be used within the tool, both from a user perspective and from a functional perspective is provided

3.2 Research Framework

This research was carried out in parallel with the P3 project schedule This research began in October 1997 which was month 14 of the overall project duration Within the timeframe of the P3 project (30 months) there were four prototypes of the ***Prompter*** tool delivered The four prototypes were delivered as part of a Spiral lifecycle model This research involved a number of distinct activities The activities described below were carried out in a sequential manner Following the creation of an initial set of default values which were included in the second prototype of the ***Prompter*** tool there began a process of validating these default values using feedback from the user partners and internal review

Activity	Duration
Preparation - becoming familiar with the area of software quality and the ***Prompter*** tool specification	3 months
Data Collection - researching the default data for the APM set	4 months
Architectural Components - design and implementation of actual software to handle the APMs in the ***Prompter*** tool	2 months
Integration - adding the researched values to the tool	5 month
Documentation - documenting the actual research	4 months

Table 3.1 : This table summarises the activities involved and their associated durations when performing this research These durations are in calendar months and not person months.

Preparation involved examining the area of software quality and software process improvement This was essential as the ***Prompter*** tool seeks to assist software managers in making improvements in their technique of developing software During this time it was also important to become familiar with the ***Prompter*** tool's architectural components This was essential so that software could be developed to manage the APM set

Data Collection involved researching the characteristics of the typical software project This involved collecting surveys and texts containing previous research A large part of this time was spent working with the VASIE database described in Chapter 4 (ESI, 1998) Towards the end of this phase, the collected data was classified according to relevance

Architectural Components involved developing documentation and software components to manage the handling of the APM data values by the ***Prompter*** tool This code was written using Java and CORBA to take the APM values from an external data file and instantiate these values into a user's project upon the selection of a specific APM This activity was time consuming as no representation mechanism had been decided upon for the data in the APM set I had to make extensions to the design of ***Prompter*** to handle this This involved evaluating alternative representations for the APM data The Token object which is discussed in section 3 7 was the final representation decided upon

Integration involved the conversion of the selected characteristics identified by the Data Collection phase into a format acceptable to the tool

Documentation of this research (the writing of this thesis) commenced following the integration of the initial set of APM values into the ***Prompter*** tool The documentation phase continued in parallel with the final refinements which were made when delivering the two remaining prototypes of the ***Prompter*** tool

3.3 The concept of an APM

Before speaking about the APM itself in detail it is important to mention that the APM is a feature that is absent from any other tool in the same market area as ***Prompter*** The APM set was constructed by this research alone This involved bringing this concept from an initial verbose requirement to an actual component within the tool that has undergone a process of verification and validation incorporating any refinements that have been identified as necessary This requirement is shown in section 5 3 1

This section seeks to answer two important questions about the APM and ***Prompter*** These questions are

- What does the APM represent?
- How does the APM fit into what the tool plans to do?

3.3.1 What does the APM represent

The APM represents a starting point for a user's project This starting point is a generic representation of what a user project may look like It is perhaps surprising to note that nearly all software projects fail for the same reasons (Jones, 1996) The risks that lead to project failure are not localised or organisation-specific but have been documented on a global scale For this reason, it is possible to generalise quite liberally over the entire software development community and identify improvements that all organisations can make in order to achieve higher productivity If the necessary improvements can be identified, this means that the problems that these improvements seek to resolve can also be identified These problems can therefore be viewed as characteristics of a software project This is what an APM intends to represent - the characteristics of a software project Seen from another viewpoint, these are the characteristics that describe the starting point of a project The APM is thus a generic description for the starting point of the average software project This is possible because from an abstract viewpoint most projects appear similar An example of this similarity is that the most common type of software project is the small-to-medium size project developed in a familiar, in-house organic software development organisation (Tarek, 1991) These are the types of concepts that the APM seeks to model

Figure 3 1 on the following page illustrates the role of the APM in an abstract manner Inside the rectangle are three project descriptions An APM is depicted by a circle This circle represents a set composed of a number of default values applicable to a certain category of software project A level 1 CMM organisation conducting a MIS project is depicted by the oval labelled Project B Project A depicts a level 3 CMM organisation developing a software product for an embedded system that will perform life critical tasks. Within this rectangle and outside the three circles are all possible software projects (the universal set). The APM shown below has a large intersection area with Project B. Project A however has a small area of intersection with the APM.

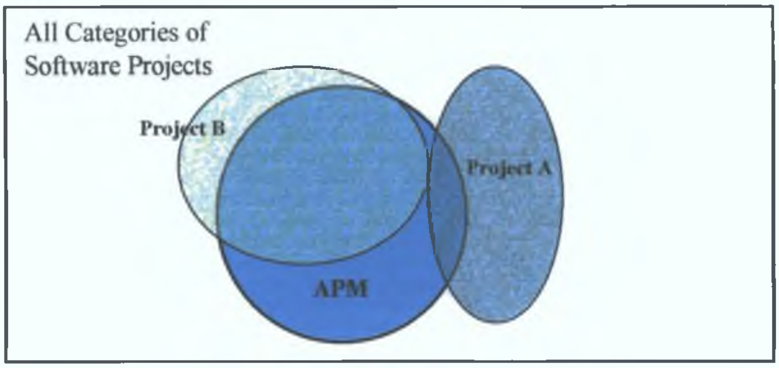


Figure 3.1: The above diagram shows the intersection area between two user projects and an APM.

This implies that there are a large number of features common to this APM and Project B but very few features common to the APM and Project A. It is the objective of the APM to have a large intersection area with as many software projects as possible so the user's starting point can be modelled by the ***Prompter*** tool.

3.3.2 How does the APM fit into what *Prompter* intends to do

To answer the second question, how does the APM fit into what the tool plans to do, it is necessary to understand the objective of ***Prompter***. ***Prompter*** aims to provide decision support to the project manager in the planning phase of a project. To provide the user with practical advice the tool must be provided with a description of the user's project. Otherwise the tool would only be facilitated to provide pointers and general guidelines to project managers. To provide sensible project based advice a project description must be available. To relieve the user from the time consuming nature of entering an extensive project description, it was conceived that the availability of a default project description would benefit the user. This can be considered loosely analogous to the concept of a letter template for an MS Word document. This APM may not always provide the ideal default description but it provides a starting point that is easily modified and extended.

3.4 Project Characteristics and the Project Description Template

As described above the APM describes a user project in a generic fashion This description thus needs a representation mechanism A set of descriptors are required that embody the concepts by which this starting point may be described At this point, the user has a mental picture of the starting point of their project A mapping is required from the user's mental model of their project to a form that can be used by ***Prompter*** This mapping is performed by a descriptor known as a token A token represents an atomic real world characteristic of a project plan Tokens may in fact model data which is concerned with features of a project or may alternatively describe facets of an organisation An example of such an atomic characteristic for a specific project is team stability This characteristic represents the stability of the software development team, or in other words the likelihood that there may be the loss of critical members during the project An organisational characteristic may model a concept that is invariant between projects An example of such an invariant characteristic is the organisation's attitude towards configuration management There are approximately 125 unique tokens used by the ***Prompter*** tool, each of which represents a different project or organisational characteristic which may or may not be known by a project manager during the planning phase (See Appendix A)

As explained above, a token represents a characteristic of project planning Therefore, each token is a variable of project planning For each token, a domain[1] over which the variable makes sense must be defined The definition of a domain of possible values allows the variation between projects to be modelled To illustrate this point, the example token described above, team stability has the possible values, low, medium or high Table 3 2 below illustrates what each of the elements in the range of possible values represents using a textual description

Each characteristic has a set of values as the previous table shows for team stability This set is the domain over which the token makes sense An entire set of such characteristics may provide a description of a project during the planning phase An example of such a description is provided in Table 3 3 on the following page

	Token: Team Stability
Value	Meaning
High	The project team is stable It is unlikely that critical members will be lost from the team before the end of the project
Medium	It is likely that members may leave the development team before the end of the project but this should not pose a risk to the success of the project
Low	The project team is unstable It is highly likely that critical members will leave the project team before the end of the project causing a risk to the success of the project

Table 3 2 The above table shows the descriptions which map to the values of low, medium and high for the token 'team stability'.

Project Name: Futile

Project Description:

Token Name	Token Range	Token Value
Project Size	[Small, Medium, Large]	Small
Requirements Complexity	[Low, Medium, High]	Medium
Team Development Experience	[Low, Medium, High]	Low
Team Skill Mix	[Low, Medium, High]	Medium
Project Budget	[0 ∞] Unit of Currency	350,000
Market Competition	[Low, Medium, High]	Low
Observed Standards	[Present, Not Present]	Not Present
Development Costs	[0 ∞] Unit of Currency	220,000
Project Duration	[0 ∞] Person Months	50
Project Life Cycle	[Waterfall, V, Spiral, Prototyping]	Waterfall

Table 3 3 The above table shows a set of example characteristics of a project in the form of tokens. The domain for the tokens are shown in the second column. The third column shows the instantiated values within the domain that make up a basic project description. The above is an example of a user's project, not an example of an APM.

The creation of a set of tokens is equivalent to creating a project description template This representation mechanism can be considered a project description language with a number of slots that may be filled The more slots that are filled, the more detailed and informative the description of the project It is precisely this **project description template** that allows the definition of an APM An APM is thus a set of instantiated tokens that the user may select as an appropriate starting point for their project. It is important to note that the APM will predict a set of token values appropriate to the user but will not contain a full set of tokens for the user's project. Only the tokens that can be reasonably justified by this study will be included in the APM.

3.5 The APM Set

Tokens are used in ***Prompter*** in two ways. The first way in which tokens are used is to create a project description. However, the token set can also be used to form an APM. This is because an APM is formed of a subset of the entire token set instantiated with particular values. The actual values that are allocated are the primary objective of this research. An APM is therefore composed of a number of such *default* token values. As described in the previous section, there are two distinct categories of token. These categories are those that model features specific to a project and secondly those that describe characteristics of an organisation that are independent of projects or in fact identical for all projects at a particular organisation. An APM can be constructed from these two token types because there is data available relating to both the typical characteristics of a software project as well as the typical characteristics of a software development organisation.

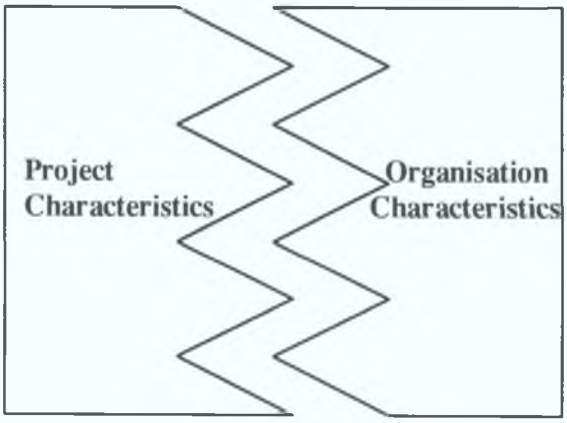


Figure 3.2: The APM is formed from a blend of organisational and project characteristics

As Figure 3.2 above indicates each APM is composed of a combination of characteristics particular to both project and organisation. The claim behind this thesis was that it was possible to create a description of the typical software development organisation and characterise this within the model such as the APM The ***Prompter*** tool intended to obtain a description of the organisation from the user This description would be then examined by the Daemons component of the tool and suggestions made in the form of decision support These suggestions would be largely based upon project characteristics rather than organisational characteristics however both are important to the tool

However, it is not quite as straightforward to create such a template for all software projects due to the variation between projects These variations are largely related to the size and complexity of the product Because project characteristics cannot be narrowed down to one particular model, a set of APMs were created to handle these variations These APMs are distinguished according to project size and complexity Before going any further it is necessary to clarify what should be understood by the terms project size and complexity in the context of the APM

3.5.1 Definition of Project Size and Complexity

Both size and complexity are terms that appear quite subjective due to their use in everyday conversation There is also the problem of familiarity with a particular baseline which acts as a reference point to which to compare all others in terms of size or complexity

Size in the sense of software development is measured by two particular techniques These two predominant techniques are known as Function Point analysis and Lines of Code measurement Both have merits which outweigh the other as a technique of measuring software size (Furey, 1997) A discussion of both lines of code and function points is beyond the scope of this research Size is described in ***Prompter*** as small, medium and large However, this measure of size is not based upon the LOC or Function Point metric but based on the size of the project team This is because neither of the measurement techniques described above are consistent enough across the software development industry

Small Size

- Small sized team working in a familiar environment
- Project team composed of around 15 members or less

Medium Size

- Medium sized team possibly divided into a number of sub-teams working on distinctive components
- Project team contains possibly more than 15 members but less than 50

Large Size

- Project team is large and distributed among a number of teams working on various components
- The project team may be geographically distributed
- The project team may be composed of more than 50 members across the various activities

The three intervals of size described above have actual numbers assigned to each These explicit sizes, 0-15, 15-50 and 50+ were decided by the project manager responsible for overseeing the delivery of the APM as a component within the P3 Project

Complexity can be described in simple terms as how difficult it is to produce a software component This is contributed to by many factors Some of these factors are related to the inability to cleanly allocate the requirements to a software design Other problems are related to the non-functional requirements of a system such as speed of execution or tight operating constraints In the context of the ***Prompter*** tool, complexity can be classified as low, medium and high The following classifications of complexity have been based on COCOMOs classification of complexity as organic, semi-detached, embedded (Boehm, 1981) These three terms are explained below

Low Complexity

- Familiar software development environment
- There is experience in developing related systems
- A small amount of communications overhead
- A stable set of requirements
- Stable development environment
- Low premium on early completion of the project

Medium Complexity

- Medium complexity represents an intermediate stage between low and high complexity with features of both present. For example, communication costs may be high but the data processing function may be managed by well documented or proven algorithms.
- Team members have an intermediate level of experience with related systems.
- The team may have a number of inexperienced members present.

High Complexity

- The software will operate within a coupled complex of hardware, software, regulations and operating procedures.
- The requirements are highly inflexible and the cost of making changes is high.
- The software is expected to conform strictly to the specifications.
- This type of project is usually working with unfamiliar software or hardware components.
- Changes to the project schedule are not usually negotiable.

The size of the project has been described in a qualitative manner above with respect to the number of team members. This is because of the absence of a reliable size metric in the software development industry. The most intuitive metric that can be used describes the size of the team required to deliver a software product.

Complexity is defined above in a verbose manner. No gauge of measurement is provided by which complexity can be estimated. Both of these points appear to be problems. This is not so however, as qualitative terms such as low, medium and high complexity and such large ranges for product size estimation suit a project manager's knowledge during the planning phase of a project. This is in fact one of the only means by which a planner is equipped to categorise their project at such an early stage.

3.5.2 An Example APM Set

From the definitions for complexity and size provided above, the next step is to define a set of APMs based upon these qualifiers. Both size and complexity defined above have a range of three possible values. This yields nine possible APMs if both project team size and complexity are used as discriminators. Some combinations are in fact redundant as it is highly unlikely that any software project that is large in size will be considered to be not complex Table 3 4 on the following page summarises the characteristics of each of these nine APMs

Estimated Product Complexity

Estimated Team Size	Low	Medium	High
Small	▪ Small team ▪ Familiar set of Requirements ▪ Flexible schedule	▪ Small team ▪ Interfaces may be complex ▪ Communications overhead ▪ Team may be unfamiliar	▪ Small team ▪ Highly complex requirements ▪ Team unfamiliar with this type of system ▪ Inflexible schedule
Medium	▪ Medium sized team ▪ Familiar set of Requirements ▪ Flexible schedule	▪ Medium sized team ▪ Communications overhead ▪ Team may be unfamiliar	▪ Medium sized team ▪ Highly complex requirements ▪ Team unfamiliar with this type of system ▪ Inflexible schedule
Large	▪ Large possibly distributed team ▪ Familiar set of Requirements ▪ Flexible schedule	▪ Large possibly distributed team ▪ Communications overhead ▪ Team may be unfamiliar ▪ Large number of interfaces	▪ Large possibly distributed team ▪ Highly complex requirements ▪ Team unfamiliar with this type of system ▪ Inflexible schedule

Table 3 4· The above table shows an example APM set for use in the *Prompter* tool. Each of the models is depicted by a box where the features of the APM are summarised. For a more detailed description see Chapter 5 which provides the APM set and their associated descriptions.

3.6 The Format of the APM

As explained in previous sections, the underlying data representation format in ***Prompter*** is the token A complete project description is made up of approximately 125 distinct tokens (See Appendix A) An APM however in made up of a much smaller subset (between 30 and 45 tokens) This is because the complete project description includes tokens that cannot be set by default Examples of such tokens that cannot be defaulted are relating to the duration of each of the stages of the project or the number of members on the project team or the training costs that will be incurred during the project These project specifics are added by the user of ***Prompter*** having selected the most appropriate APM for their project An APM is thus composed of a set of tokens initialised with a value which has been found by this study The composite set of tokens in an APM will be used as a starting point for the user's project description These tokens combined will give ***Prompter*** a starting point description of the user's project

The APM of ***Prompter*** is represented as a file external to the tool Each of the APM files is provided to the user by the installation of ***Prompter*** The installation of ***Prompter*** adds these APM files to a standard directory within the filespace of ***Prompter*** When the user requests the creation of a new project, a window is displayed which allows the user to select the model most appropriate to their situation See section 1 5 for a functional overview of using ***Prompter*** When the user has selected the most appropriate model, the tool opens a data file containing the default token values These token values are then used as a baseline to which the user will add their project specific data such as

- Project schedule information
- Team characteristics
- User environment
- Metrics data

3.7 Initial Research: Identification of a Base Token Set

This section describes the information that the token data type was required to represent in ***Prompter*** The technique by which this research was performed is then investigated and followed by a description of the Token Data Dictionary as a controlled document for Token Management

3.7.1 The Objective of the Token Data Type

The data type which represents characteristics of a user project and user organisation is known as the token Before characterising any default project models it was necessary to formulate this set of tokens into which the default values could be placed This set was intended not only to represent the APM of ***Prompter*** but also to represent generic project information in the tool When researching the token there were two main concerns

1 **Intuitive to the User:** A user of this tool must feel comfortable with the type of information being requested by the tool If the type of data required does not map to the user's concept of a software project it will be difficult to obtain a full or even partially valid project description This will render any project description useless

2. **Useful to the Daemons** Because ***Prompter***'s main aim is to provide decision support to software project managers, it is essential that the user project is represented appropriately The advisor components of ***Prompter*** need a rich and accurate project description in order to diagnose any problems or risks in a project This is necessary because the Daemons examine a number of tokens collectively and formulate advice based upon the conditions suggested by these token values aggregately (Gaffney, 1999) If the project description cannot convey this data, the daemons will be rendered useless

3.7.2 A Token Set for *Prompter*

As described above the token data type was required to be both intuitive to the user and also useful to the Daemons It was intended that ***Prompter*** supply the user with advice relating to a particular set of areas within the scope of software project management These areas are shown in Figure 3 3 on the following page

It was necessary that the token set provide sufficient information for the daemons to provide advice for these areas This required an analysis of these areas and an identification of the type of information about a software project that would allow the critique of a user's project The P3 Project Handbook which had been written prior to the identification of the token set provided a list of appropriate project characteristics that would be used by the Daemons These characteristics needed to be cast from a simple verbose description into the token data type as defined in Appendix A The Handbook format is illustrated on the following page in Table 3 5 which shows a number of suggested characteristics to represent variations in the project environment

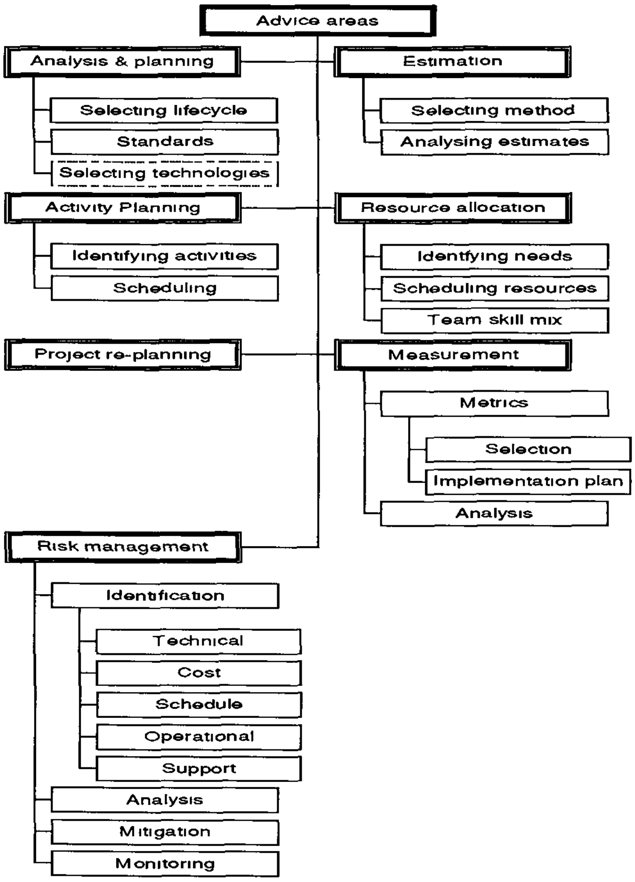


Figure 3.3: The diagram above illustrates the Advice Taxonomy of *Prompter*.

Not all of the characteristics in the handbook contributed to the Token Data Dictionary Other sources that were used were Boehm's USAF Risk Taxonomy (USAF, 1988) and the AMI Handbook (Pulford, 1996) These additional sources were intended to provide tokens appropriate to the areas of Measurement and of Risk Management I was not responsible for the creation of the Token Set for these areas but for the remaining areas shown in Figure 3 3 In summary these areas are

- Analysis and Planning
- Estimation
- Activity Planning
- Resource Allocation
- Project Re-planning

	Flexible and supported	Half-way in between?	Fixed parameters and not supported
Funding	Adequate and available	Partially or sporadically funded	Not adequate, not available
Equipment	Available and easy to support	Marginal	Not available or difficult to maintain
Software and tools	Available and adequate	Marginal	Not available, not adequate
Training	Not required	Some training required	Required
Schedule	Flexible	Modifiable	Fixed
Budget	Flexible	Modifiable	Fixed
Quality of product	Good	Better	Best
Functionality of product	Low	Medium	High
Productivity of programming effort	High level language, existing or purchased routines	High level language, no reusable code	Low level language, all original code
Estimated risk, based on project stability	Low	Medium	High
Software supplier or subcontractor required	No	Yes, for non-critical code	Yes, for critical code
Likelihood of change in scope or objective	Low	Possible	High
Ability to make changes in timely manner	High	Medium	Low
External requirements to provide data or information	Low	Medium	High
Concurrent development	No	Some, but not critical to project success	Yes, critical to project success
Development within systems engineering	No	Some, but not critical to project success	Yes, critical to project success

Table 3.5: The above table shows an example of a set of characteristics which were provided in the P3 Project Handbook. These characteristics were analysed and rejected or added to the Token Data Dictionary.

This work involved analysing the characteristic set provided by the Handbook and translating it into the format of the token The tokens that I identified are shown in Appendix A as part of the Token Data Dictionary The Token Data Dictionary is described in the next section

Following the identification of the token set that would be analysed by the Daemons, it was necessary to specify a presentation mechanism for the token set It was realised that many tokens would be used by more than one daemon in ***Prompter*** For example a token such as Requirements Complexity would be used to provide advice pertaining to more than one area in the Advice taxonomy (Gaffney, 1999) It was also identified that the tool would be provided with a number of domains and subdomains These domains and subdomains would be an organisation of the token set according to the similarity of the questions that the user is asked It was considered sensible to group related tokens for ergonomic resons whereby the user answers questions of a similar nature ***Prompter*** was provided with a specific set of domains and subdomains into which the tokens would be inserted The next task was to take the tokens set and distribute it among the various domains and subdomains These domains and subdomains and the allocation of the tokens throughout are provided in Appendix A

3.7.3 The Token Data Dictionary

Having created a project description template using the token set, the next step was to control this token set by creating a process for managing change Changes to the initial token set were expected for the following reasons

- Due to evolution of the ***Prompter*** tool through a series of incremental prototypes it was foreseen that modifications and refinements would be made to the token set
- Knowledge identification is the process of manipulating the token values in order to provide advice relating to a user project If the available token values were considered unsuitable for critiquing a user's project, it would be necessary to add tokens to model the missing project information
- Feedback from the users regarding the way in which project information is requested could result in modifications to the token set to increase usability

For this reason, a document known as the Token Data Dictionary containing the token set was to be controlled using a formal system of change control and configuration management This was essential as the developer partners suggested additions at different stages of the project and for varying reasons This document also keeps account of the token layout in the GUI Related tokens are grouped into domains and subdomains This layout is recorded in the Token Data Dictionary The most recent version of this document is provided in Appendix A

Chapter 4: Current Industry Practices and Sources of the APM Values

4.1 Introduction

The objective of this chapter is to describe the seven principal sources examined when creating the APM set These sources justified the allocation of default values to the appropriate tokens in the APM set Because the default values are allocated on the strength of these seven sources, there is a need to provide a description of each There is a description of the profile of the source, the sample set used is described and finally an evaluation of the validity of each source is provided The tokens that have been allocated default values as a result of each source is also provided A number of problems were encountered performing this study These problems are also discussed below as a preamble to the description of each source

4.2 Problems encountered when performing this literature study

A number of problems were encountered when collecting and examining the data necessary to create the APM set of ***Prompter*** These problems were caused by a number of factors that were unforeseen before beginning this study This section will discuss these problems

4.2.1 A General Absence of Metrics

One of the key points of the SEI CMM process maturity scale is that the use and application of metrics is not considered to be a key process area until level 4 is achieved. This does not imply that metrics cannot be collected at the lower levels of software process maturity, it does imply however, that there is no business gain to using metrics at these levels This is because the organisational maturity is not at an adequate level to apply these metrics accordingly (Humphrey, 1989) In light of this revelation it is no surprise to find out that there is a complete absence of quantitative data describing the activities of level 1 software organisations This is because this type of data is simply not recorded The net effect in terms of this study is that there is a shortage of useful data points for the APM set of ***Prompter*** The absence of valid metrics has proved the most serious problem facing this research Much of the useful data that has been located has been of a qualitative nature with *wordy* descriptions of how software organisations approach development practices

4.2.2 The Success Stories

Much of the data published in journals and magazines citing case studies of software organisations seem to contain bias For example, the case studies for software process improvements tend to highlight industry's success stories such as Raytheon, Motorola, Hughes Aircraft etc These producers of real-time embedded software have achieved levels of organisational maturity through the CMM process improvement approach These categories of software producer do not represent the typical software development organisation For the purpose of this study, this category of software development organisation falls outside the scope of those being considered

Other case studies tended to probe a specific aspect of improvement such as software reuse and ignore the rest of the supporting processes As a result, many such case studies failed to give an overall picture of software production at any particular organisation, which is the type of data which has been sought in this present study From all of the data collected (except for the VASIE database described below), there were no publications describing situations where an organisation attempted to make improvements and were not as successful as originally intended Many of the case study examples collected illustrated improvements at organisations which were CMM level 2 or higher To iterate the point made in earlier chapters, ***Prompter*** is predominantly aimed at the software organisation seeking to make small improvements which will increase productivity These organisations typically have no defined software process For this reason, it is about this type of organisation that this research seeks to collect data so that such a starting point can be represented by the APM Unfortunately, descriptions of this type of organisation are not as plentiful as those of the more mature developers described above

4.2.3 Politics

No organisation contributing to a case study or survey wishes to provide results that are used to exemplify mediocrity The contribution of valid data often requires the blessing of senior management Management is often concerned that competitors will use any published information against their organisation There is often concern that any involvement in case studies or surveys will result in a negative effect This concern means that organisations often decline to reveal productivity data or worse, provide results that have been tainted This scenario is hard to identify making spoiled data more difficult to isolate Due to these concerns, there is less data avaılable than expected descnbmg the ınternal workıngs of the typıcal software organısatıon

4.3 Sources of the APM Values

Each of the sources used ın thıs research ıs documented below A general comment about the source, the ıntended audıence and an evaluatıon of usefulness to the task of buıldıng the APM set ıs provıded The tokens that have been allocated default values as a result of the source are also lısted For a further descnptıon of the token and the concept ıt represents, see Appendıx A

4.3.1 The Description of the Average Software Organisation

The APM of ***Prompter*** seeks to model a software producer wıth a process maturıty level equıvalent to a CMM level 1 organısatıon Thıs categonsatıon of a software organısatıon as level 1 actually defines a number of charactenstıcs whıch are common to almost all level one organısatıons For thıs reason, these charactenstıcs can be ımplıed as present m any organısatıon wıshıng to use the APM of ***Prompter*** It ıs true that some organısatıons at the ınıtıal level perform actıvıtıes that are charactenstıc of a more mature organısatıon However, thıs ıs rare, as there are a number of charactenstıcs that are common to almost all level one organısatıons These common features are summansed below (Yourdon, 1996)

- ➢ Standards may be present but are generally ıgnored
- ➢ Endorsed methodologıes are practısed ınformally
- ➢ Tools may be present whıch are used on a haphazard basıs
- ➢ Estımatıon process ıs weak and often ınaccurate
- ➢ Faılure to track software sıze changes or code and test errors
- ➢ Schedules are often ınformal
- ➢ Programmers consıder themselves as artısts not subject to rules or procedures

The charactenstıcs descnbed above have been venfied by research 81% of software development organısatıons assessed by SEI up to 1992 were at the ınıtıal level No recent data has ındıcated that thıs figure has changed consıderably (CSE, 1998) Thıs ıs the justıficatıon for usıng the level one organısatıon as a baselıne for the APM set The next step ıs to ınvestıgate each of these charactenstıcs ın detaıl vıa the sources researched below

4.3.2 The European Software Institute

The European Software Institute (ESI) is an independent authority on software process improvement The ESI's principle aim is to act as a link between process improvement technologies and particular business needs primarily for European software development companies It is through their web site that the following two sources were employed in this research

4.3.2.1 Source 1: The VASIE Database

The VASIE (Value Added Software Information for Europe) database is maintained by the European Software Institute (ESI, 1998) Through the VASIE database, the ESI aims to *provide value added information for the European software best practice repository* and to permanently disseminate the validated PIE (software Process Improvement Experiment) results through the WWW All of the PIEs included in the VASIE database have been performed under the supervision of the ESSI (European Systems and Software Initiative) The ESSI is a body established by the EC to promote software best practice through support to organisations engaging in PIEs This database contains the final reports provided by organisations performing PIEs funded by the EC under the ESSI initiative These PIEs provide information reports of the experiences of software organisations making software process improvements These reports follow a set format describing

- ♦ Background including the starting scenario, work plan and expected outcomes
- ♦ Work performed
- ♦ Results and analysis from technical, business, cultural and organisational points of view
- ♦ Key lessons learnt from technical, business, cultural and organisational points of view
- ♦ Conclusions and future actions

The most important feature of these reports from the point of view of the APM was that there was a description of the starting scenario provided These starting scenarios were provided as a page of text containing statements describing the organisation and/or their projects such as

> *While software is quite well- designed from a modern technology point of view, documentation ethics tend to be low*
>
> *No formal methodology was in place to underwrite the quality of the requirements capture process*

For almost all of the PIEs analysed, the starting scenario described an organisation at the initial level This data helped describe the activities of a level one organisation before any process improvements were attempted As described in previous chapters this is the type of software organisation that the APM wishes to model

The VASIE database contained 141 software process improvement experiments that were available for use in this research Only 50 of the 141 PIEs were considered useful for the following reasons

- Some of the reports deviated from the report format rendering these reports difficult to use
- Some reports cited PIEs performed at stages of organisational maturity beyond the initial level addressed by ***Prompter***
- Many of these PIEs were documenting improvements of a specific aspect of the software process such as software maintenance

There were a number of positive features of using reports from the VASIE database The first of these positive features was that there appeared to be no bias The starting scenarios always appeared to portray the true situation before making any improvements From the description of such a starting scenario, the characteristics of a low maturity software organisation can be extracted Additionally, all of these PIEs were funded by the EU A requirement of such funding is that an appointed project officer would oversee the experiment ensuring that the report contains only what took place during the PIE This helped ensure the validity of the data For the reasons outlined above, the 50 PIEs obtained from the VASIE database proved to be the most useful source when creating the APM Table 4 1 on the following page lists the tokens that were allocated default values as a result of this source

Token ID	Token Name
24	TeamMix
36	ApplicationOriginality
48	ProjectEquipment
50	ProjectSubcontracting
52	TeamSoftwareDevelopmentExperience
59	MarketCompetition
60	StandardISO9001
66	OrganisationCodingStandard
67	OrganisationDocumentationStandard
68	OrganisationConfigManagementStandard
70	OrganisationInternalProductStandard
71	SubcontractorStandardRequired
72	SoftwareReuseStandard
78	EstimationStandard
90	IndependentVandV

Table 4.1: Tokens allocated default values as a result of the VASIE Database (ESI, 1998). For a description of each token, see Appendix A.

4.3.2.2 Source 2: ESI 1997 Software Best Practice Questionnaire - Results

A questionnaire was completed by organisations submitting project proposals to the European Commission during the ESSI call in 1997 (ESI, 1997) A total of 394 valid responses were obtained from 20 different countries and 37 different sectors The aim of the questionnaire was to collect data on widely recognised software management practices The questionnaire was made up of 42 questions divided into five sections

- Organisational issues
- Standards and procedures
- Metrics
- Control of the development process
- Tools and technology

The results were presented as a series of tables, one for each of the sections listed above Each table showed the question and the percentage (from the total responses) of positive responses to each question For example, the section on Metrics had the following entry

Management Practice	*Average Adoption Level*
Record and feedback of estimated versus actual efforts into estimation process	55%
Record and feedback of size into estimation process	21%

Table 4.2: The above is an example of the type of results that are provided by the 1997 Software Best Practice Questionnaire - Results (ESI, 1997).

The questionnaire was structured that only yes/no answers were permitted. This fact combined with the accompaniment of the questionnaire with an EC call for proposals could have led respondents to portray optimistic results. This point was documented in the survey report and not simply conjectured by my observations. This was certainly implied when the results of this experiment were compared with the results from examining similar organisations' final reports in the VASIE database. Another problem discovered when examining the results from the questionnaire was that this document sought to highlight aspects of key process adoption according to geographical location. The focus for the creation of the APM set was to be independent of region or country. Despite the problems outlined above, this source proved useful as many of the areas addressed within overlapped with the aims of the ***Prompter*** tool and hence the type of data modelled by the APM.

Token ID	Token Name
48	ProjectEquipment
78	EstimationStandard

Table 4.3: Tokens allocated default values as a result of the ESI 1997 Software Best Practice Questionnaire - Results. For a description of each token see Appendix A (ESI, 1997).

4.3.3 Source 3: Current Practice in Software Engineering: a survey

This survey was carried out between November 1995 and March 1996 and published in the IEE Journal, Computing and Control in August 1997 (Holt, 1997). The report was written by Dr. Jon

Holt of the University of Wales, Swansea, UK Fifty participants were obtained primarily as a result of a letter published in the IEE News and a web page containing the form for the survey Respondents ranged from single engineers, to small companies, major international companies and some academic institutions The main aim of the survey was to find out exactly who was using which methodologies, methods and standards, and their perception by the users The results of this survey also provided information about

- Frequency of lifecycle adoption
- Use of design methods and methodologies
- CASE tools
- Adoption level of various process improvement technologies

The results were provided as a series of paragraphs describing the respondents collated responses The questions that the respondents were asked were provided followed by the grouped results These results were coupled with a comment by the author which justified or suggested a reason for the result in question An example of such a set of results are illustrated in the Table 4 4 on the following page

Development Model	Percentage of Overall
Traditional Waterfall	30
V Model	24
Spiral	20
Other	4
No Model	22

Table 4 4 The above table shows the popularity of the various lifecycle models in use today taken from Jon Holt's survey of Current Practice is Software Engineering (Holt, 1997).

The results of this survey have proved useful to this research and have proved consistent with findings from other sources that have been documented in this chapter The source of the survey appeared to be reliable through the publication of the results by the IEE This reliability was also verified through actual contact with the author Of particular use from the results documented are description of the more popular lifecycle models in use and conformance to standards

Token ID	Token Name
60	StandardISO9001
89	ProjectLifeCycle

Table 4.5: Tokens allocated default values as a result of Jon Holt's 1997 Current Practice in Software Engineering Survey (Holt, 1997).

4.3.4 Source 4: 1998 Software Business Practices Survey

The 1998 Software Business Practices Survey represents the ninth annual survey of the business and operating practices of the US software industry (Price, 1998) The 1998 survey was completed by 716 of the 16,517 companies that were invited to participate The survey was conducted in January 1998 and was typically completed by respondent companies' chief executive officer Questions sought actual and projected information The questions in the survey sought information on the number of products, target markets, international activity and the number and assignment of employees Other questions sought information on revenue, profitability, capital-raising activities and demographic information Beyond these questions about general business practices, the 1998 survey focused primarily on customer support, pricing, marketing and distribution processes

The survey results were published as a series of questions and the percentage responses to each question of the overall survey respondent total For example the following chart provides an example of the results to the question *From the following list, please rank the top five issues of concern to your company and the top five issues of concern to the software industry*

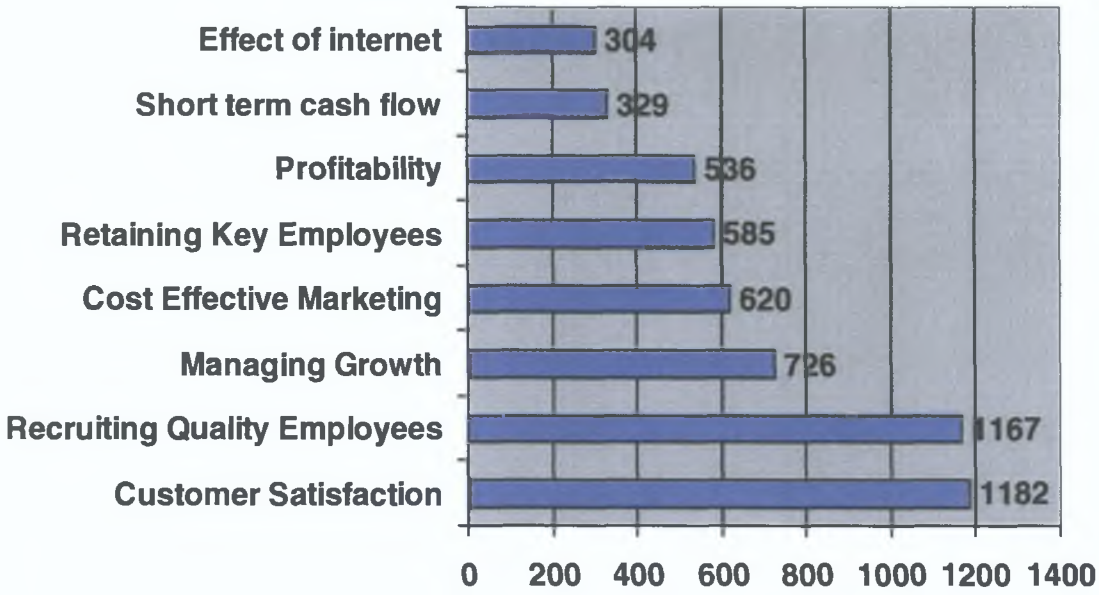


Figure 4.1 : The bar chart above shows the importance of customer satisfaction to software developers as provided in the 1998 Software Business Practices Survey (Price, 1998).

This survey was a valid source of business-related issues of use to this research. Information was supplied about application type and organisation profile. This survey provides a reliable source of data due to the extensive size of the sample set. This is reinforced by the acceptance of the survey for independent publication and also owing to the historical establishment of this annual report. Despite the validity of the data, much of the results proved unusable in the context of this research as the report was aimed predominantly towards the marketing and sales aspects of the software industry.

Token ID	Token Name
36	ApplicationOriginality
59	MarketCompetition

Table 4.6: Tokens allocated default values as a result of the 1998 Software Business Practices Survey (Price, 1998).

4.3.5 Source 5: EXE Magazine Surveys

EXE magazıne publıshed two separate surveys whıch provide usable ınformatıon about software development ın the UK Both of these surveys are summarısed below

4.3.5.1 Survey 1: What are you really worth?

The first of these surveys ıs ıntended to gıve software engıneers an ıdea of what to expect ın terms of salary and workıng condıtıons (Bennett, 1998a) Thıs survey targets certaın sectors and poınts to areas of hıgh fınancıal growth The survey was based upon 316 replıes to a questıonnaıre Also ıncluded was ınformatıon about

- Software development platform
- Hardware development platform
- Type of software beıng produced - bespoke, system, embedded, etc
- Percentage of ın-house users for developed software

Although the theme of thıs artıcle was not software process ımprovement or descrıptıon of software organısatıons many of the results proved useful for understandıng the type of product developed by the average Brıtısh software development organısatıon Thıs source can be consıdered to be useful due to the number of organısatıons partıcıpatıng ın the survey Addıtıonally, the results found from thıs survey were not contradıctory to any of the fındıngs made through the other sources Although the organısatıons used ın thıs survey were taken from the Brıtısh software development ındustry there ıs no reason to belıeve that these results would be any dıfferent ıf the organısatıons had been located elsewhere

4.3.5.2 Survey 2: Development Tools '98

The second set of survey results publıshed by EXE was ıntended to gıve a report of the followıng areas (Bennett, 1998b)

- Development envıronment - software and hardware platform
- Tools employed to aıd the development process
- Belıefs and opınıons about contemporary ıssues such as YR2K, CORBA, etc

Thıs survey gathered the results of 311 respondents who replıed to the questıonnaıre over telephone The respondents were chosen at random from EXE's readershıp The gatherıng of results was conducted over a two-week perıod between July and August 1998

As with the first EXE survey performed, the theme of the report was largely outside the scope of this research. Despite this, the survey touched off areas of interest providing information about software/hardware platform usage, team profile, application type and development methodologies. The results of this survey appeared equally valid to those taken from the first EXE survey however there is a suspicion that some of the same organisations were used as input to both surveys.

Platform	Percentage of responses for this platform (respondents can select more than one platform)
PC	95%
WorkStations	42%
Embedded	22%
Mini	12%
PSA	6%
Mainframe	5%
Games consoles	2%
Other	1%

Table 4.7: An example of the results from the Development Tools '98 survey published in EXE Magazine (Bennett, 1998b).

The format for both surveys described above was identical. Results were collated and percentages of responses were provided as a series of tables. An example from the second EXE survey, Development Tools '98 is provided in Table 4.7 above. These responses are to the question, which of the following software platforms do you develop software for?

Token ID	Token Name
24	TeamMix
36	ApplicationOriginality
48	ProjectEquipment
50	ProjectSubcontracting
52	TeamSoftwareDevelopmentExperience
59	MarketCompetition
60	StandardISO9001

66	OrganisationCodingStandard
67	OrganisationDocumentationStandard
68	OrganisationConfigManagementStandard
70	OrganisationInternalProductStandard
71	SubcontractorStandardRequired
72	SoftwareReuseStandard
78	EstimationStandard
90	IndependentVandV

Table 4.8: Tokens allocated default values from the Development Tools '98 (Bennett, 1998b) and What are you really worth (Bennett, 1998a) surveys published in EXE Magazine.

4.3.6 Source 6: Revision Labs

The highlights of a survey published on the web site of Revision Labs provides results obtained from 29 respondents (Revision Labs, 1997) The survey aimed to provide relevant information regarding current software testing and quality assurance practices as well as future trends in the use of third party resources The survey was posted on Revision Labs' website from April 1 to August 1997 The survey provides results regarding

- Lifecycle model used
- Quality practices used - e g formal testing, white box testing, test coverage analysis, etc
- Quality measurement techniques
- Amount of development subcontracted
- Amount of testing performed externally
- Application type
- Company size

The results from Revision Labs' survey were provided as a series of questions followed by a bar chart or table showing the distribution of the responses among the optional answers Table 4 9 on the following page shows the responses to the question, what is the most important way that you measure quality?

Quality Practice	Percentage of Overall
Total Defects (by severity)	39
Defects by KLOC	11
Defects per Function Point	4
Product Reviews	7
Measuring Quality	7
Customer Satisfaction Surveys	14
Customer Support Calls	18

Table 4.9: The above table shows the quality practices of most relevance to software developers from Revision Labs' 1997 survey (Revision Labs, 1997)

It is felt that the results from this survey, although interesting and within the scope of what the APM of ***Prompter*** seeks to model, there are discrepancies that cannot be overlooked The problems begin with the absence of details of who performed the survey and whether or not a technique of validating responses was used It is reasonable to be sceptical about a questionnaire that can be accessed via the web without respondent validation The sample set for the survey appears quite small and if there are invalid data present the error injected by this erroneous data will have greater effect There were no tokens that were allocated default values on the strength of this survey alone There were a number of tokens that had their default values collaborated by the results of this survey

Token ID	Token Name
36	ApplicationOriginality
50	ProjectSubcontracting
71	SubcontractorStandardRequired

Table 4.10: Tokens whose default values were collaborated by the Revision Labs' 1997 survey (Revision Labs, 1997).

4.3.7 Source 7: The Spire Handbook

The SPIRE Handbook (Centre for Software Engineering, 1998) was created with a view to assisting small software development organisations to achieve business benefits from employing software process improvement This handbook provides an explanation of the business and technical aspects of software process improvement A walkthrough of the various practices that should be associated with any improvement is also provided Included in the handbook are a number of case studies of software process improvement experiments that took place during the SPIRE project (Centre for Software Engineering, 1998) These six case studies provide a description of the organisational profile, the improvement actions taken and a record of the lessons learnt from the experiment These summarised PIE reports took a similar format to those taken from the VASIE database described in section 4 3 2 1 above This included a description of the starting point, the improvement project, lessons learned and plans for the future The starting point was the part of the case study that was useful This section provided a verbose description of the software organisation prior to any improvements Although the reports were not found to be very detailed, the scenario described appeared no different from any of the other organisations in the VASIE Database (source 1) before making such process improvements from a point of having no defined formal process This data therefore reinforced the characteristics identified through the use of the more detailed PIEs obtained from the VASIE database The default token values allocated as a result of source 1, the VASIE database were reinforced by the SPIRE Handbook Case Studies

The PIEs described in the SPIRE handbook can be considered to be valid descriptions of software organisations at level 1 The integrity of the handbook can be relied on for a number of reasons Primarily, these reports took the form of a summary of a PIE which took place within the context of the SPIRE project The handbook was also published by the Centre for Software Engineering in Dublin, an independent consulting organisation providing software process improvement training in Ireland and Europe

4.4 Summary

This chapter aimed to show the validity of each of the sources investigated in this study The validity of each source has been shown through a description of the sample set and an evaluation of its relevance to the study The most important source used has been the VASIE database provided by the ESI with over 50 PIEs used to create the APM set The validity of this source has been enhanced by the focus of software process ımprovement ın these reports This ınvolved the use of experıenced of software process ımprovement mentors and traıners who were responsıble for ensurıng that the reports created were accurate

Chapter 5: Validation of Results

5.1 Introduction

Tms chapter descnbes the vahdatıon process for the results of tms study   Tms vahdatıon begıns wıth an evaluatıon of the project descnptıon template as a representatıon mechanısm for a project descnptıon ın ***Prompter***   The process by whıch the APM set was vahdated by the project partners ıs then descnbed   Fınally, a vahdatıon usıng data from external organısatıons ıs descnbed

5.2 Validation of the Project Description Template

Tms sectıon wıll descnbe bnefly the project descnptıon template, the vahdatıon mechanısm ın place and any conclusıons to be made from the feedback obtaıned   Tms ınvolves examınıng the suıtabılıty of the project descnptıon template for ıts ıntended purpose

5.2.1 What ıs the Project Descnptıon Template

A bnef remınder of the project descnptıon template wıll be provıded ın tms sectıon   The project descnptıon template ıs the set of charactenstıcs by whıch a user may descnbe theır project   Tms concept was explaıned ın greater detaıl ın sectıon 3 4   Tms template ıs made up of a number of tokens each representıng a unıque concept ın software project plannıng   A full set of these may provıde a detaıled pıcture of the scenano ın place at a software development organısatıon when embarkıng upon a project   As Sectıon 3 7 1 descnbed, ıt ıs ımportant that tms format ıs suffıcıently

- Intuıtıve to the user
- Useful to the daemons

5.2.2 The Vahdatıon Process

The fırst step taken ın the vahdatıon process was to ensure that the project descnptıon template exısted ın a controlled manner wıth a defıned process for change and versıon control   A group was formed wıth a representatıve from each of the development organısatıons ın the P3 project Each representatıve was responsıble for vahdatıng changes to tms document and relayıng any changes to the rest of theır ınternal development team   Tms token management group was lead by the author of tms thesıs and was hence responsıble for all

- Configuration Management
- Change Control
- Version Control

The next step taken was to send the project description template to the user partners of the P3 project for validation The document was reviewed for a period of one month with a deadline proposed for all responses Both user partners had elected individuals responsible for disseminating the document among their internal teams at their organisations This document was reviewed by these internal teams The internal teams at the user organisations were typically composed of software project managers with a number of years experience of software project planning For this reason it was believed that these teams were equipped with the skills necessary to provide a valid critique of the project description template

The final step involved the analysis of the feedback from the user partners This feedback was made up of a number of general remarks about the document, a detailed description of the tokens that were deemed to be inaccurate, irrelevant or badly defined Finally, a section of 'missing tokens' was provided This section described tokens that the reviewers expected to see but felt were omitted This feedback was examined by the token management group within the project and also by senior project managers from the developer organisations

5.2.3 Conclusions from the Feedback Obtained

The user feedback was provided in a structured format with clear suggestions for changes, refinements and additions to the project description template The feedback from the users proved highly useful to the verification of the project description template This process of validation opened up an extra channel of communication with the user partners and provided essential feedback to the developer team about the essential components of the tool

5.3 P3 Project Validation of the Baseline APMs

The validation process for the baselme APM set of ***Prompter*** occurred at a number of stages most of which did not take place in parallel These stages are briefly

- Internal validation
- Validation by the EC
- Validation by user partners and field test

Each of these activities will now be described in detail

5.3.1 Internal Validation

The internal validation of the baseline APM set was by reference to the requirements described in the User and System Requirements deliverables for the P3 project The following excerpt from D3 1 System Requirements Document of the P3 project illustrates the relevant requirements

- **Process Selection**

FR-110	***Prompter*** shall provide the capability to select process models (both standard and company specific) from a repository of such models
FR-120	It shall provide the capability to instantiate the selected process with an initial set of suggested parameters

> "The tool shall have an underlying set of Abstract Project Models (APM), from which the user generates an Instantiated Project Model (IPM) appropriate to their situation This will be done by selecting one of the pre-defined standard process models - for example, the standard model for that organisation - although there will be a mechanism to generate a new template derived from the APM from scratch Some initial fine-tuning of the selected template can be done at this stage The output of that is the IPM "
>
> "This will then be refined and deepened (analysing several scenarios if required) with the assistance of advice from the daemons to form the final Refined Project Model (RPM) This last step can be repeated during the project by updating the key parameters arising from actual progress to date This can be illustrated thus (see Figure 5 1 on the following page) "

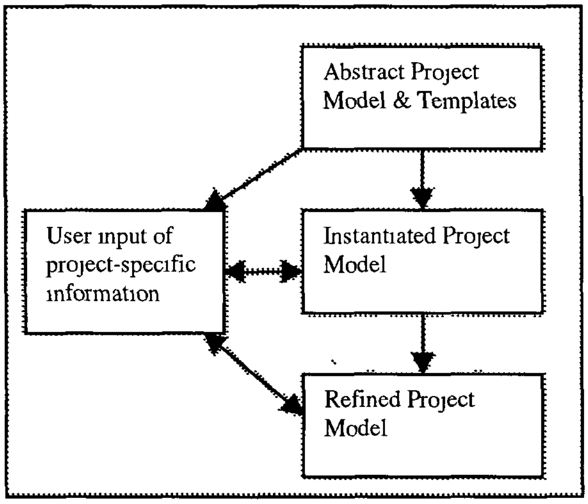

Abstract Project Model & Templates
User input of project-specific information
Instantiated Project Model
Refined Project Model

Figure 5.1. The diagram above shows the movement from APM to IPM to RPM.

"If the APM set conforms to these requirements, it will be deemed suitable for delivery with the final tool This will be decided by the Project Managers of the development organisations in the P3 project "

5.3.2 Validation by the EC

Throughout the duration of the P3 project there have been project reviews by an independent examining board representing the European Commission This group was responsible for ensuring that the project progressed according to plan and that all interim deliverables were achieved This meant that the project was obliged to deliver a pre-commercial prototype at the end of the 30 month project This tool was obliged to fulfil the system and user requirements unless otherwise agreed This group representing the EC were expected to identify any weaknesses or risks to the project A typical result of such weaknesses and risks would have been failure to deliver the product according to specification and schedule

There were five reviews throughout the duration of the project The 3rd and 4th project reviews involved presentations of both the tool and the APM as a component of this tool The APM was validated against the requirements described for this tool At both reviews the APM was deemed to be acceptable according to the requirements stated for ***Prompter*** The requirements for APM component of ***Prompter*** have been shown in section 5 3 1

5.3.3 Validation by User Partners and Field Test

The two user partners in the P3 consortium were responsible for verifying the suitability of the tool for its intended purpose If the user partners felt that certain aspects of the tool were not according to specification, it was their responsibility to draw attention to this The P3 project delivered four initial prototypes The final prototype was delivered at the end of the project which was a pre-commercial prototype Each delivered prototype incrementally added key functional aspects of the tool

Each of these prototypes was delivered at stages in the project illustrated in Table 5 1 below These prototypes were validated by the user partners according to the objective for each prototype The EC were provided with a copy of each prototype which was used to evaluate the progress of the project

Prototype	Delivery Date	Objective
1 (Noumea)	11/06/98	Look and feel of GUI components
2 (Salonika)	19/10/98	Functionality
3 (Burgundy)	18/12/98	Advice and knowledge provision
4 (Tipperary)	28/02/99	Complete tool - pre commercial prototype

Table 5.1: The above table shows the four prototypes which were constructed when building the *Prompter* tool

On the delivery of each prototype, the user partners examined the progress to date according to the functionality intended for the current prototype As the APM is an important component of the tool, the user partners have evaluated its suitability according to the user and system requirements described in section 5 3 1 Feedback regarding the APM was received for the Salonika and Burgundy prototypes Comments were received regarding the appearance of the APM in the GUI (see Figure 1 3) There were no negative comments regarding the default token values within the APM set

5.4 The Validation Process for the APM Set

A process of validation internal to the project for the APM set was described in the previous section This feedback was received in a *diluted* form in the sense that the reviewers were concerned with the entire tool and not just the APM set For this reason it was considered necessary to perform a more comprehensive validation of the APM set The following sections will give the results of this more comprehensive validation

5.4.1 The APM Set Arising from this Study

This study resulted in the creation of nine alternative models which characterise a generic software project during the planning phase The nine models are distinguished by project size[1] and complexity Both size and complexity have three possible values Size is described as small, medium and large Complexity is described as simple, medium and complex For a more detailed discussion of this topic see section 3 5 1 Each of the nine models created by this study is described in the following sections by a short narrative description There is some repetition across the models This repetition occurs because there are nine possible combinations of the characteristics complexity and size Each of these models is described below

5.4.1.1 Small Size - Low Complexity

This APM describes the starting point of a project where there is a small software development team that is familiar with working as a unit The team has experience in developing this form of application for this type of environment and using the relevant technologies There is not a great deal of communications overhead with the client The requirements are stable and not overly complex Changes can be negotiated with the customer The development team is composed of 15 members or less

5.4.1.2 Medium Size - Low Complexity

This APM describes the starting point of a project where there is a medium sized software development team with experience in developing this form of application for this type of environment and using the relevant technologies There is not a great deal of communications overhead with the client The requirements may be considered to be stable and not overly complex Changes to the project schedule can be negotiated with the customer There may be extra communications present as the team size is between 15 and 50 members

[1] Project size actually refers to project team size

5.4.1.3 Large Size - Low Complexity

This APM describes the starting point of a project where there is a large software development team with experience in developing this form of application for this type of environment and using the relevant technologies There is not a great deal of communications overhead with the client despite the size of the product The requirements are not overly complex and can be considered to be stable Changes to the project schedule can be negotiated with the customer The team is large with possibly more than 50 members and may be distributed geographically or between a number of organisations

5.4.1.4 Small Size - Medium Complexity

This APM describes the starting point of a software project of small size and medium complexity The requirements are reasonably complex, the team has an intermediate level of experience with related systems The requirements can be expected to change during the course of the project The user may be unwilling to accept changes to the project schedule Many of these projects display characteristics of complex and non-complex software projects This model is a level of indirection between the two There are some new technologies being employed The team size is small - around 15 members or less

5.4.1.5 Medium Size - Medium Complexity

This APM describes the starting point of a software project of medium size and complexity The project team is between 15 and 50 members and has an intermediate level of experience with related systems The requirements are reasonably complex and can be expected to change during the course of the project The user may be unwilling to accept changes to the project schedule Many of these projects display characteristics of complex and non-complex software projects This model is a level of indirection between the two There are some new technologies being employed

5.4.1.6 Large Size - Medium Complexity

This APM describes the starting point of a software project of large size and medium complexity The team is large with possibly more than 50 members and may be distributed geographically or between a number of organisations Internal communication linkages within the project may be difficult due to large team size The requirements are reasonably complex, the team has an intermediate level of experience with related systems There are some new technologies being employed The requirements can be expected to change during the course of the project The user may be unwilling to accept changes to the project schedule Many of these projects display characteristics of complex and non-complex software projects This model is a level of indirection between the two

5.4.1.7 Small Size - High Complexity

This APM describes the situation where there are tight constraints, a complex set of requirements that are not flexible to change The project team is small in size with around 15 members or less with possibly low communications overhead The requirements are not easily mapped or decomposed to software components and are likely to be highly volatile The product may be part of a systems software project or within a real-time or critical run-time environment The project team may not be familiar with developing this type of software and the deadline is not flexible The team is possibly unfamiliar with this type of product

5.4.1.8 Medium Size - High Complexity

This APM describes the situation where there are tight constraints, a complex set of requirements that are not flexible to change The project team is medium sized with between 15 and 50 members The requirements are not easily mapped or decomposed to software components and are likely to be highly volatile The product may be part of a systems software project or within a real-time or critical run-time environment The project team may not be familiar with developing this type of software and the deadline is expected to be inflexible

5.4.1.9 Large Size - High Complexity

This APM describes the situation where there are tight constraints, a complex set of requirements that are not flexible to change The project team is large in size with more than 50 members and may be distributed geographically or between a number of organisations Communication overheads will be high due to the project size and the complexity factor The requirements are not easily mapped or decomposed to software components and are likely to be highly volatile The product may be part of a systems software project or within a real-time or critical run-time environment. The project team may not be familiar with developing this type of software and the deadline is not flexible.

5.4.2 The Validation Process for the APM Set

In order to apply the APM set to real software projects it is necessary to ensure that the default values found by this study are accurate. One way of making such an evaluation is to compare the conjectured default values with data from actual software organisations running real software project under the constraints described above. These organisations must be representative of the organisation being modelled in the APM of ***Prompter***. Ideally, this validation should involve a comparison between each of these default token values and the validation data.

5.4.2.1 The Validation Problem

The APM seeks to model the planning phase of a software project of a level 1 CMM organisation. It would be ideal to compare the internal workings of a number of level 1 CMM software development organisations with the default values of the APM. This would provide the appropriate feedback to this study with which to evaluate the suitability of the conjectured default token values. Unfortunately, this sort of organisation is not as accessible as it would initially appear.

The ***Prompter*** tool addresses software development issues that are unheard of to the level 1 CMM organisation. Activities such as configuration management, subcontract management, estimation or metrics are frequently misunderstood by organisations lacking a formal software process. It is common for level 1 organisations to answer assessment questionnaires in an over-optimistic manner. This is caused by a misunderstanding of the complexity of the issues mentioned above. For example, configuration management according to the CMM is a key process area with ten individual activities assigned to it. Until such complexities are understood clearly, organisations are often under the illusion that they practise such activities in a manner that is actually characteristic of a more mature organisation.

The implication of this problem is that it is impossible to ensure the suitability of the default token values in the APM set by asking a software organisation lacking a software process. This is because this type of organisation is not equipped with a sufficient understanding of the key process areas which ***Prompter*** addresses Conversely, it is not possible to validate the default values by asking such questions of an organisation with an understanding of these issues This is because these are typically not CMM level 1 The problem is that those suitable to provide the answers are those who do not understand the questions and those who understand the questions do not provide the appropriate answers This *catch-22* situation implies that a technique of extracting the organisational profile of a level 1 organisation is required

In order to find an appropriate data set it was decided to consult the Centre for Software Engineering in Dublin This organisation performs software process improvement training and assessments for ISO, CMM and SPICE It was identified that an appropriate data set could be provided by using an organisational maturity assessment taking place at the beginning of a process improvement programme This source is described in section 5 4 3

5.4.2.2 Validation Scope

Each of the APMs in ***Prompter*** is composed of approximately 25 default token values These default token values have been allocated as a result of the work described in Chapter 4 Not all of these default token values require validation The reason for this discrimination between those that require validation and those that do not will be described in this section From the point of view of APM validation there are three distinct levels of default token values These are as follows

Project Level Tokens: The project tokens are those that are dependent upon the user's choice of APM The user of ***Prompter*** selects the particular APM having read a narrative describing the characteristics of what type of project the APM represents Figure 1 3 of Chapter 1 shows the user's selection of the APM These values do not require justification This is because the default token values are a result of the definition of the project For example, an APM defined as 'High Complexity' and 'Large Team Size' will have individual tokens to represent each of these project characteristics The user actually chooses these values through a verbose description of the APM so therefore these token values are not true defaults Instead, these values are a *tokenised* representation of the characteristics that the user has identified for their project For this reason, it was not necessary to use any sources in order to allocate or validate such defaults

Industry Level Tokens: These default token values represent specific features of the software industry at large rather than individual organisations. Additional justification is not required here because these default token values are based on industry surveys described in Chapter 4. These default token values are independent of organisational or project characteristics such as software process maturity or software organisation size. An Example of such a token is 'Team Volatility'. This token can be reasonably inferred because this factor is documented for the software industry at large, i.e: it is well documented that software personnel switch between employer frequently rendering project teams rather volatile. Another such token is 'Project Life Cycle' which defaults to *none* because there is no one lifecycle model that is used by the majority of software organisations - in this case, the value defaults to no lifecycle.

Organisational Level Tokens: These default token values require empirical validation. This is because these default token values describe aspects of the software development organisation or the way in which their projects are 'usually run'. Each of these default token values relates to the maturity of the organisation's software process - assumed in this study to be equivalent to level 1 CMM. These tokens represent characteristics such as the presence of a standard for documentation or configuration management. These tokens are in fact identical across the APM set. This is because these tokens describe characteristics that are independent of a particular project. These characteristics are usually independent of project complexity and size and are considered to be features of a particular organisation.

5.4.3 Validation of Organisational Level Default Token Values

The data that has been used to validate the APM set is described in this section. This took the form of results of a process maturity assessment of companies engaging in ISO 9001 training. This training was conducted at the Centre for Software Engineering, Dublin in 1998. For reasons of confidentiality, the companies involved in the training program cannot be named. The collated results from fourteen companies were used to verify the suitability of the organisational default values described in section 5.4.2.2 above.

The organisations involved in this training program appeared to be typical of a level 1 CMM organisation seeking to make improvements to their software process. For this reason the results of these assessments could be seen to represent the type of software organisation being modelled by the APM and addressed by the ***Prompter*** tool. Section 5.4.2.1 describes the problems associated with using data from software organisations lacking a knowledge of the key process associated with software process improvement The results from this assessment are partially affected by the problems described above This was confirmed by discussing this issue with the course trainer and co-ordinator However, there were some approaches taken to resolve this problem during the assessment Each of the areas to be evaluated was explained briefly to the organisations prior to the assessment It was also explained to the participating organisations that the results of the assessment would be private to the organisation and the Centre for Software Engineering In situations where such results are made public there is a tendency to portray a more optimistic scenario than the more realistic one With the absence of such a bias, the integrity of the results could be relied upon somewhat more

The pre-training programme assessment itself is composed of 143 individual questions The answer to each question is provided using the following three point scale

Rating	Meaning
S	**Sometimes/Never** Use this rating if the statement is never true, or sometimes true (i e is true less than one-third of the time) This value implies that practice is poor regarding the question being asked
U	**Usually/Often** Use this rating if the statement is usually true, or often true (i e between one third and two thirds of the time) This value implies that good practice is *sporadically implemented* where good practice is implemented on some projects but not at an organisation wide level
M	**Mostly/Always** Use this rating if the statement is usually true, or often true (i e between one third and two thirds of the time) This value implies that good practice is implemented regarding the particular question being asked

Table 5.2. The above table illustrates the options available to participating organisations when responding to the ISO for Small Companies self assessment.

The 143 questions are spread between 14 key-activities of software development These activities cover lifecycle activities, supporting activities and organisation level activities The activities are as follows

- User Requirements
- Software Requirements
- Architectural Design
- Production

- Transfer
- Maintenance
- Project Management
- Configuration Management
- Validation and Verification
- Quality Assurance
- Process Management
- Procurement
- Training
- Management Responsibility

From the 143 questions distributed among the 14 activities of which the answers to 45 can be considered of use to the validation process for the APM set of ***Prompter*** In some cases there is a one-to-one mapping between a question in the self-assessment and a particular token in the APM set of ***Prompter*** In other situations there are a number of questions when the results of which are combined an overall picture is given which can be used to justify a particular token For example, there are 21 questions for the activity of Configuration Management but there is only one token to represent the state of this activity in ***Prompter*** There is also a clear mapping between the range of possible responses in the self-assessment to the number of options available for instantiating each token value in ***Prompter*** The majority of the tokens in the APM set of ***Prompter*** have either a range of two or three possible options Often these options have a clear mapping to the technique illustrated in Table 5 2 above reflecting poor, mediocre or satisfactory practices

5.4.4 Validation

This section validates the actual tokens that have been allocated with default values in the APM set of ***Prompter*** This will involve a walkthrough of the organisational level tokens for which validation is required followed by the industrial level and project level tokens The three classes of token have been described in section 5 4 2 2 above Each of the tokens cited below is defined formally in Appendix A

5.4.4.1 Organisational Tokens Justified Using the Assessment Results

This section provides the validation of the organisational tokens in the APM set of ***Prompter***. The question (or possibly questions) provided in the assessment that maps to this token is provided followed by a summary of the collated responses received. The association between the default token and the results obtained from the pre-training assessment is then made.

Token: StandardISO9001

Token ID: 60

Explanation: This token can be set to either true or false. This token is set to true if the organisation is ISO 9001 for software compliant.

Default Value: False

Justification: From the fourteen organisations involved in completing the self assessments, none had ISO 9001 certification.

Token: StandardCMMLevel2

Token ID: 61

Explanation: This token can be set to either true or false. This token is set to true if the organisation achieves the standard laid out by the CMM level 2 assessment.

Default Value: False

Justification: From the fourteen organisations who completed the self assessments, 70% have no defined process for the estimation or scheduling of activities. These are typical characteristics of level one practices. This implies that the majority of these organisations have not got the capability to achieve level 2 with their current processes.

Tokens: StandardCMMLevel3,
StandardCMMLevel4,
StandardCMMLevel5

Token Ids: 62, 63, 64

Explanation: This token can be set to either true or false. This token is set to true if the organisation achieves the standard laid out by the CMM level 3 - level 5 assessments

Default Value: False

Justification: The justification for Token 60 - StandardCMMLevel2, shows that the default of non-conformance for CMM level 2 is justified If an organisation does not reach CMM level 2, the same organisation cannot possibly be at level 3, 4 or 5 as it is not possible to skip over levels

Token: StandardISO15504

Token ID: 65

Explanation: This token can be set to either true or false This token is set to true if the organisation is ISO 15504 (SPICE) for software compliant There is no assessment for ISO 15504 however there are a set of processes that must be followed

Default Value: False

Justification: There are no clear figures for the number of organisations who have adopted the SPICE standard However, it is reasonable to assume that the majority of software organisations are unaware of the presence of SPICE as a standard let alone using it as a guide to better practices The defaulting of this token is justified by the poor adoption of the alternative standards such as CMM and ISO 9001 both of which are clearly more popular than SPICE

Token: OrganisationCodingStandard

Token ID: 66

Explanation: This token can be set to either true or false This token is set to true if the organisation has a defined standard that is in place and followed for developing code

Default Value: False

Justification: From the self assessment described above only 14% of organisations participating claimed that they had a clear standard that was in place and documented and referenced in all design literature This shows that there is a low presence of formal coding standards that are followed strictly This justifies the false default value

Token: OrganisationDocumentationStandard

Token ID: 67

Explanation: This token can be set to either true or false This token is set to true if the organisation has a defined standard that is in place and followed for documentation of all project deliverables

Default Value: False

Justification: From the self assessment described above only 14% of organisations participating claimed that they had a clear standard that was in place and documented and referenced in all design literature This shows that there is a low presence of formal documentation standards that are followed strictly This justifies the false default value

Token: OrganisationConfigManagementStandard

Token ID: 68

Explanation: This token can be set to either true or false This token is set to true if the organisation has a defined standard that is in place and followed for controlling all items that should be subject to configuration management

Default Value: False

Justification: From the self assessment described above only 35% of organisations participating claimed to have a good standard of configuration management This figure is considered to be overly optimistic as many of the organisations were unsure about what configuration management really implies This result however still implies that the majority of the participating organisations do not have a configuration management process that is in place, defined and followed

Token: OrganisationInternalProductStandard

Token ID: 70

Explanation: This token can be set to either true or false This token is set to true if the organisation has a defined standard that is in place for evaluating the product that is being developed by the project in question

Default Value: False

Justification: From the self assessment described above six questions were included which evaluate the internal evaluation of a product before final delivery to the customer Only 26% of the responses received indicated that a defined standard is in place and applied to the evaluation of a product before final delivery

Token: SubcontractorStandardRequired

Token ID: 71

Explanation: This token can be set to either true or false This token is set to true if the organisation has a defined standard that is in place for subcontractors that are involved in delivering sub-components of the product being developed

Default Value: False

Justification: From the self assessment described above six questions were included which evaluate the processes that are in place to manage subcontractors A total of 84 responses were present to these questions, less than 10% of which indicated that an adequate process of subcontractor management was in place This indicates that the default value of such a standard being absent is a reasonable assumption

Token: SoftwareReuseStandard

Token ID: 72

Explanation: This token can be set to either true or false This token is set to true if the organisation has a defined standard that is in place for the evaluation of components that are reused or off-the-shelf

Default Value: False

Justification: From the self assessment described above, two questions were included which evaluate the processes that are in place to evaluate components that are designated for reuse or procured as an externally developed subcomponent A total of 27 responses were provided to these questions, less than 10% of which indicated that an adequate process of subcontractor management was in place This indicates that the default value of such a standard being absent is a reasonable assumption

Token: EstimationStandard

Token ID: 78

Explanation: This token can be set to either true or false This token is set to true if the organisation has a defined standard that is in place for the estimation of project schedule and duration

Default Value: False

Justification: The self assessment contains one particular question which asks if estimating of activities is performed in accordance with defined procedures 70% of responses indicated that there is no such process The remaining 30% was divided between those who claimed to have a defined process in place and practised and those who apply the standard on some projects

5.4.4.2 Industry Level Tokens

This section does not provide a further validation of the industry level tokens These values are based on surveys described in Chapter 4

Token: TeamVolatility

Token ID: 23

Explanation: This token represents the turnover of team membership during the course of the project These changes can be due to a team member leaving the organisation or being simply re-assigned to another project This token can be set to one of the following less than one-third, between one-third and two-thirds, more than two-thirds

Default Value: Between one-third and two-thirds

Justification: The default for this token may in fact be influenced by project characteristics Team member turnover may be caused by a variety of reasons such as political motivation, personality issues or complexity of the task at hand This token has instead been set as such as a function of the global profile of the software development industry The primary characteristics of which are the transience and shortage of skilled software development personnel which has been well documented throughout the industry This was actually documented in Software Project Dynamics (Tarek, 1991) which indicates that *the annual turnover rate is observed to be 25 1% but as high as 34% at some organisations*

Token: ProjectEquipment

Token ID: 48

Explanation: The token represents the availability of appropriate equipment to the software development team for the duration of the project The options for this token are less than adequate, adequate and more than adequate

Default Value: Adequate

Justification: The default for this token may in fact be influenced by organisational and project dependent characteristics However, the cost of equipment for the development of software has dropped considerably in latter years The cost of hardware and development tools has become considerably less than the cost of personnel This token has been set as a function of this global characteristic

Token: ProjectLifeCycle

Token ID: 89

Explanation: This token represents the lifecycle for the project in question This may be set to V Model, Waterfall, Incremental, Spiral, Prototype or 'no lifecycle chosen'

Default Value: No lifecycle chosen

Justification: Surveys quoted in Chapter 4 show that there is no lifecycle model in existence that is applied to the majority of software projects There is a wide variety of models that are in use including those that are tailored to a particular organisation

Token: Independent V and V

Token ID: 90

Explanation: This token represents the level of external verification and validation in this project The range of options for this token are as follows No external V and V, External V and V at some stage in the project, V and V at each milestone in the project

Default Value: No external V and V

Justification: External V and V is an activity that is usually only included for contractors involved in defence projects Such contracts are usually part of a real time system or safety critical component of such a system Level 1 CMM organisations do not have the resources or processes to perform an activity such as external V and V This is the category of organisation being represented in the APM set It has been defaulted so that the typical ***Prompter*** user is not required to provide a response to a question that can be reasonably conjectured

5.4.4.3 Project Tokens Exempt from Validation

There is also a class of tokens that do not require any kind of validation These token values although allocated as default have been chosen by the user This choice of token values is made by choosing the project description most appropriate to the starting scenario of the project Section 5 4 1 describes the nine APMs using a short narrative Each of the statements in this narrative map to a corresponding default token value These default token values simply translate the statements acknowledged by the user into actual token values that can be used sensibly by ***Prompter*** This is almost a mechanical process and therefore demands no validation apart from verification that these tokens have been translated correctly The concepts that are defaulted by this process are those such as

- Project team size
- Product requirements stability, complexity and volatility
- Team familiarity with the product being developed
- Customer schedule flexibility

5.5 Summary

This chapter has reviewed the validation process for the APM set that resulted from this study The chapter began by describing the validation process for the project description template Project based validation has shown that this component achieved all of its objectives The project based valıdatıon process for the APM set was then descrıbed Fınally, valıdatıon of the APM by use of process ımprovement assessments was descrıbed

Chapter 6: Conclusions

6.1 Objective of this study

There were two primary objectives of this study The first objective was to create a template that permits the description of a software project in the ***Prompter*** tool This was followed by the core component of this study which was to create a set of Abstract Project Models that characterise a set of typical software projects at the planning stage The outcome of both aspects of this study is recorded below

6.1.1 Description of the Generic Software Project

In order to describe the generic software project it was first necessary to construct the project description template which could represent this generic project description Construction of the project description template was the initial objective of this research The project description template would enable the definition of project characteristics and hence would act as a representation mechanism for both user projects and APMs m ***Prompter*** The process by which this model was created is described in Chapter 3 This model was created by examining the data that required representation and also identifying the complete set of use cases for this information within the ***Prompter*** tool These use cases have been taken from the User Requirements document of the P3 Project and are provided in Appendix B In Chapter 3, a suitable model was constructed that allows the description of a project using approximately 120 unique project characteristics

Having created the project description template it was necessary to describe the generic software project This involved characterising the starting point of the typical software development organisation and their software projects Organisational characteristics were found to be largely invariant of project type and most organisations displayed the same type of approach to software development at an organisational level Many of these common characteristics were based upon weaknesses identified in the software process of level 1 organisations Projects on the other hand have been minimised to a set containing nine models which are distinguished based upon size and complexity

In conclusion, it can be said that the creation of a template enabling the description of a user project was a success The creation of a generic project model required some distinction between projects in order to make the APM useful to the user This was as a result of the amount of variation between projects depending upon varying size and complexity The creation of a set of baseline models illustrated the need to distinguish between organisational characteristics and project characteristics The same organisational metrics appeared for all project types but the project related characteristics depended upon both team size and the complexity of the product being developed

6.1.2 The APM Set and the ***Prompter*** tool

The P3 project was intended to deliver a prototype decision support tool My task was to provide a project description template and a set of baseline models in the form of the nine APMs This feature was expected to free the user from the need to enter a considerable amount of information into the tool - information that could be reasonably predicted from industry surveys The project description template was expected to provide a mapping from the user's external view of their project to the critiquing system's view of the data which it uses to provide project related advice The creation of the project description template resulted in the introduction of a component known as the Token Data Dictionary which is a tightly controlled and managed document The Token Data Dictionary will undergo further refinements beyond the end of this research in parallel with the commercialisation of the ***Prompter*** tool

The creation of the APM set required a mapping from the sources investigated to the project description template created as the initial research Because I was responsible for the creation of both of these components, the evolution of the project description template and the APM set could be performed in parallel The way in which both the project description template and the APM set have been developed have proved appropriate to the original requirements as shown in Chapter 5 This has been confirmed by the user and developer partners review of both components from a project perspective This was shown in sections 5 2 and 5 3

The creation and addition of the APM set to ***Prompter*** was reviewed by the user partners in the P3 project as well as the EC The APM set was viewed as appropriate and an important component of the tool From the original requirements of the ***Prompter*** tool, the APM set was deemed to have more than sufficiently represented the starting point of a target user of ***Prompter*** There was also a need to validate the results of this study at another level This additional validation used a number of process maturity assessments used in a process improvement programme by the Centre for Software Engineering, Dublin. These assessments showed that the default values found actually represented the profile of a level 1 CMM organisation The validation process involved confirming the mapping between the observations made in this study and the resulting APM set It has been shown through feedback received that the APM set constructed is sufficient for the current version of the tool and hence this study achieved its primary objective - to characterise the project starting point of a level 1 CMM organisation in a model known as the APM

An ancillary objective of this research was to allow the exceptional user of ***Prompter*** to create a baseline model of their organisational specific starting point This functionality was added to the ***Prompter*** tool using the design evolved through this study and has been considered to be a useful extension to the tool The option of creating a user specific APM is described in the following section

6.2 Creation of a Customised APM

This section proposes the user option of creating a baseline model that is specific to an organisation This is an additional feature of the tool that is of use to user's not fitting the profile of the average software development organisation

The APM model appears quite useful to the typical user but is the exception to the rule isolated by this set of default models? The answer is yes The need to cater for the typical software organisation is a priority for the P3 project This implies that the atypical software project is not supported by the APM set within the tool This is because a business decision was made to aim this tool at the level 1 CMM organisation seeking to improve its software development process This is a sensible choice as this category of user is represented as approximately 80% of software developers However, the more mature software development organisation will be facilitated in a different manner Because software organisations beyond the initial chaotic level have reached a point where an aspect of repeatability has been introduced, it is frequent that processes are common to all projects within the organisation This implies that such organisations have a defined process in place that is reused from project to project It is necessary for such a user to be able to define their process

Prompter provides functionality for the user to create their own baseline model This functionality was provided by a *Save As* option in the tool The user begins by choosing to create a project description without employing an APM This is similar to creating an MS Word document without using a template of any kind The user goes through a questions and answers session providing descriptions of a generic software project This description involves questions about the following

- Product being developed
- Organisational characteristics
- Available resources
- Customer and user
- Business drivers
- Project environment
- Project plan and schedule

The user has the option of defaulting any of these token values If the user feels that any particular value may vary from project to project this token value can be left uninitialised This value may thus be initialised when the user refines the project description having applied the baseline APM

This project may subsequently be saved as an APM This description may be reused by other users as a starting point for their project In this way, a company specialising in products that require a process unique to that organisation may describe its activities once and reuse this description again and again This functionality is of particular use to large organisations where a predefined process is in place This functionality may be of benefit to a software developer at a level of organisational maturity higher than the chaotic initial level

Software project management has many concepts in common with other forms of management With a view to this point, the possibility of applying ***Prompter*** and the APM set to other domains has been discussed There appears to be no reason why a project description template could not be created for non-software projects An APM set could also be created if the relevant project data points are available through similar sources that were applied in this study For this reason it appears possible that a set of domain-specific APMs could also be created

6.3 Final Remark

The concept of the APM began as a user requirement for the ***Prompter*** decision support tool This requirement was evolved to a functional component in the ***Prompter*** tool by this research alone In parallel with the realisation of the APM set, the project description template was created The project description template was created which provided an appropriate representation mechanism for a project description in the ***Prompter*** tool Both the APM set and project description template were included in the final P3 Project Deliverable to the EC and remain core components of the ***Prompter*** tool of which version 1 1 was released to the market in June 1999 This research succeeded in its primary objective, to create an Abstract Project Model which was intended to be a description of a generic software project to be used in the ***Prompter*** tool This was achieved by examining a number of sources, revealed in Chapter 4, and creating default token values based upon the organisational characteristics revealed by these sources These characteristics were deemed to be indicative of the target user of the ***Prompter*** tool, the level 1 CMM organisation seeking to make process improvements through the use of recommended best practices

References

(Boehm, 1981) Boehm, Barry W , 1981 *Software Engineering Economics* New Jersey, United States Prentice Hall

(Brooks, 1975) Brooks, Frederick P Jr , 1975 *The Mythical Man Month* Philippines Addison -Wesley

(Brooks, 1987) Brooks, Frederick P Jr , 1987 No Silver Bullet Essence and Accidents of Software Engineering , IEEE Computer, (April 1987), 10-19

(CSE, 1998) Centre for Software Engineering, 1998 *The SPIRE Handbook* Dublin Centre for Software Engineering

(DeMarco, 1982) DeMarco, Tom, 1982 *Controlling Software Projects* New Jersey, Umted States Prentice Hall

(ESI, 1997) European Software Institute, 1997 *1997 Software Best Practice Questionnaire [online]* Available from http //www esi es/Publications/Reports/tr-sbpqaor3 html [Accessed 15 May 1998]

(ESI, 1998) European Software Institute, 1998 *VASIE PIE Database [online]* Available from http //www esi es, 1998 [Accessed 20 February 1998]

(ESPITI, 1996) ESPITI - ESPRIT Project 11000, 1996 *Training for SPI European Needs and Solutions,* Berlin

(Bennett, 1998a) Bennett, James, 1998 What are you really worth? *EXE The Software Developers Magazine*, 12 (11), 20-22

(Bennett, 1998b) Bennett, James, 1998 Development Tools '98 *EXE The Software Developers Magazine*, 13 (4), 38-39

(Furey, 1997) Furey, Sean, 1997 Why Should we use Function Points, *IEEE Software*, 14 (2), 28-31

(Gaffney, 1999) Gaffney, Eamon, 1999 *The Development of an Agent Based Critiquing System Architecture for a Project Management Tool Prompter*, Ireland Dublin City University

(History, 1997) History, 1997 *A History of the Computer Mini [online]* Available from http //www pbs org/nerds/timeline/mini html [Accessed 11 April 1998]

(Holt, 1997) Holt, Dr. Jon, 1997. Current Practice in Software Engineering: A Survey, *Computing and Control Engineering Journal*, 8 (4), 167-172.

(Humphrey, 1989) Humphrey, Watts S., 1989. *Managing the Software Process.* United States: Addison -Wesley.

(Iona, 1997) Iona Technologies, 1997. *OrbixWeb Programmer' s Guide.* Dublin, Ireland.

(Jones, 1992) Jones, Capers, 1992. *Assessment and Control of Software Risks.* New Jersey, United States: Prentice Hall.

(Jones, 1996) Jones, Capers, 1996. *Our Worst Development Practices, IEEE Software,* 12 (6), 102 -104.

(Mair, 1992) Mair, P., 1992. *CASE: A State of the Market Report*, Unicom.

(McConnell, 1997) McConnell, 1997, Software's Ten Essentials, *IEEE Software*, 14 (2), 144-145.

(OMG, 1998) OMG, 1998, *The Common Object Request Broker: Architecture and Specification,* MA USA, Object Management Group.

(Pickering, 1993) Pickering, Chris, 1993. *Survey of Advanced Technology.* Kansas, United States: Systems Development Inc.

(Pickering, 1996) Pickering, Chris, 1996. *Survey of Advanced Technology.* Kansas, United States: Systems Development Inc.

(Pressman, 1994) Pressman, Roger S., 1994. *Software Engineering: A Practitioners Approach.* (3rd ed) Maidenhead, England: McGraw-Hill.

(Price Waterhouse, 1998) Price Waterhouse LLP, 1998. *1998 Software Business Practices Survey.* (9th ed) Massachusetts, United States: Price Waterhouse.

(Pulford, 1996) Pulford, Kuntzmann-Combelles, Shirlaw, 1996. *A Quantitative Approach to Software Management.* Workingham, England: Addison Wesley.

(Rakos, 1991) Rakos, John J., 1991. *Software Project Management for Small to Medium Sized Projects.* New Jersey, United States: Prentice Hall.

(Revision Labs, 1997) Revision Labs, 1997. *Revision Labs Inc: Software Testing Survey Results [online].* Available from: http://www.revlabs.com/surresult.html [Accessed 10 June 1998]

(Rubin, 1992) Rubin, Howard, 1992. *The Software Engineer's Benchmark Handbook. United States:* Applied Computer Research Inc.

(Sanders, 1994) Sanders, Joc and Curran, Eugene, 1994 *Software Quality* Addison Wesley

(Schach, 1990) Schach, Stephen R , 1990 *Software Engineering*, Homewood, IL United States, Richard D Irwin, Inc , and Aksen Associates, Inc

(Sommerville, 1989) Sommerville, Ian, 1989 *Software Engineering* (3trd ed) Workingham, England Addison-Wesley

(Tarek, 1991) Tarek, Abdel-Hamid and Madnick, Stuart E , 1991 *Software Project Dynamics An Integrated Approach* New Jersey, United States Prentice Hall

(USAF, 1988) USAF, Department of the Air Force, 1988, *Software Risk Abatement* United States AFSC/AFLC Pamphlet 800-845

(Wasserman, 1996) Wasserman, Anthony I , 1996, Towards a Discipline for Software Engineering, *IEEE Software*, 13 (7), 23-31

(Yourdon, 1992) Yourdon, Edward, 1992 *Decline and Fall of the American Programmer* New Jersey, United States Prentice Hall

(Yourdon, 1996) Yourdon, Edward, 1996 *Rise and Resurrection of the American Programmer*, New Jersey, United States Prentice Hall

Appendix A: The Token Data Dictionary

1. Introduction

This appendix describes and specifies all the Tokens that will be stored in the project database and manipulated by the ***Prompter*** tool Each token described below is a variable of project planning that may be used by the ***Prompter*** tool Each of the tokens described in this appendix may have a number of optional token values It is the selection of a value from a number of possible values that allows a project description to be built up The instantiation of a token with a particular value is typically performed by the user The instantiation of a number of default tokens has been the primary objective of this research The tokens that store these default instantiations are listed in Chapter 5 The complete token set and a definition of each is provided below

The Token Data Dictionary had been created using the following headings

ID	Identification (index) number from 1 to N
Name	Given name of the token
Type	Datatype of token, int, boolean, etc or possible a more complex structure
Definition	An simple definition of what the token means
Values	Values the token may take on and the associated meaning
Source	This column describes any sources used to create this token This source has been one of three principal references These are as follows 1) The USAF Risk Taxonomy developed by Barry Boehm (USAF, 1988) 2) The P3 Project Handbook Vol II which describes a number of key characteristics which should be considered when planning a software project This document was an internal project deliverable 3) The AMI Handbook which assists the practical application of metrics at software organisations (Pulford, 1996)

2. Tokens

ID	Name	Type	Definition	Value	Source
1	ProjectTeamScale	Int	In relation to what we are accustomed to, the size of the project team is	3 = more than twice as big 2 = about the same 1 = smaller	USAF Risk Taxonomy (USAF, 1988)
2	ProjectSchedule	Int	In relation to what we are accustomed to, the duration of the project is	3 = more than twice the length 2 = about the same 1 = shorter	USAF Risk Taxonomy (USAF, 1988)
3	ProjectDevelopmentEnvironmentMaturity	Int	The development environment to be used is	3 = relatively novel/untested 2 = fairly mature/tested 1 = very mature/tested	USAF Risk Taxonomy (USAF, 1988)
4	ProjectTechnicalTargetMaturity	Int	What is the technical target environment (target environment = hardware/software environment that the product is running on)	3 = relatively novel/untested 2 = fairly mature/tested 1 = very mature/tested	USAF Risk Taxonomy (USAF, 1988)
5	ProjectExternalCommunicationsComplexity	Int	The communications linkages with any collaborators or subcontractors are	3 = complex 2 = not complex 1 = no external linkages	USAF Risk Taxonomy (USAF, 1988)
6	ProjectClientCommunications	Int	Communications linkages with the clients are	3 = complex 2 = not complex 1 = no client communications	USAF Risk Taxonomy (USAF, 1988)
7	ProjectImports	Int	Level of dependence of the project on 'risky' imports (eg reusable components, etc)	3 = critically dependant on imports 2 = somewhat dependant on imports 1 = not dependant on imports	USAF Risk Taxonomy (USAF, 1988)
8	ProductCohesiveness	Int	In relation to what we accustomed to the product is	3 = large or cannot be broken down into normal-size work packages 2 = medium sized or fairly easily broken down into normal-size work packages	USAF Risk Taxonomy (USAF, 1988)

				1 = small or easily broken down into normal-size work packages	
9	ProductOperationalInterface	Int	Interfaces between the product and other software and hardware components it must work with in the final user environment are	3 = badly defined or subject to uncontrolled change 2 = quite well defined and subject to tightly controlled change 1 = very well defined and subject only to tightly controlled change	USAF Risk Taxonomy (USAF, 1988)
10	ProductSupportabilityProcedures	Int	In terms of supportability there are	3 = nonexistant or inadequate supportability procedures 2 = some concerns about supportability 1 = procedures are in place and adequate	USAF Risk Taxonomy (USAF, 1988)
11	ProductSupportabilityPersonnel	Int	In terms of supportability personnel	3 = significant discipline is necessary, there is a mix of concerns 2 = minor discipline is necessary, there is a mix of concerns 1 = they are in place, sufficient and experienced	USAF Risk Taxonomy (USAF, 1988)
12	RequirementsComplexity	Int	In relation to what we are accustomed the requirements are	3 = complex and can be allocated to software only with difficulty 2 = not complex and easily allocated to software components/modules 1 = simple and easily allocated to software components/modules	USAF Risk Taxonomy (USAF, 1988)
13	RequirementsVolatility	Int	During the course of development, product requirements are likely to be subject to	3 = extensive revision 2 = some revision 1 = little or no revision	USAF Risk Taxonomy (USAF, 1988)
14	RequirementsInflexibility	Int	In relation to inflexibility of	3 = impossible to agree changes to	USAF Risk Taxonomy

			functional and other specification concerns, if we meet problems in development, it will be	functional and other specifications 2 = moderately difficult to agree changes to functional and other specifications 1 = not difficult to agree changes to functional and other specifications	(USAF, 1988)
15	ProjectTechnology	Int	The project technology can be described as	3 = new and little experience 2 = existent with some inhouse experience 1 = mature, existent, with in-house experience	USAF Risk Taxonomy (USAF, 1988)
16	RequirementsApplication	Int	The applications can be described as	3 = real-time embedded with strong interdependency 2 = embedded with some system interdependency 1 = non real-time with little system interdependency	USAF Risk Taxonomy (USAF, 1988)
17	ProjectTools	Int	The tools involved in the project are	3 = invalidated with major development required 2 = available, validated with some development required 1 = documented, validated and in place	USAF Risk Taxonomy (USAF, 1988)
18	TeamFamiliarity	Int	The project manager has a good knowledge of the skills and productivity of	3 = less than one-third 2 = between a third and two-thirds 1 = more than two thirds	USAF Risk Taxonomy (USAF, 1988)
19	TeamDesignerKnowledge	Int	In terms of application domain, the designer is considered to have	3 = little knowledge 2 = a good knowledge 1 = an excellent knowledge	USAF Risk Taxonomy (USAF, 1988)
20	TeamApplicationDomainExperience	Int	In terms of the technical tasks of developing software for this application domain	3 = nobody on the team has good experience 2 = a small proportion have good experience	USAF Risk Taxonomy (USAF, 1988)

				1 = a large proportion have good experience	
21	TeamDevelopersExperience	Int	With regard to experience of the development environment to be used, the team has	3 = no experience 2 = some experience 1 = extensive experience	USAF Risk Taxonomy (USAF, 1988)
22	TeamTechnicalTargetEnvironmentExperience	Int	The experience of team members of the technical target environment	3 = very few or only a small proportion have good experience 2 = a significant proportion have good experience 1 = most or all have good experience	USAF Risk Taxonomy (USAF, 1988)
23	TeamVolatility	Int	During the course of the project, turnover of team membership will probably be	3 = more than two-thirds 2 = between a third and two-thirds 1 = less than a third	USAF Risk Taxonomy (USAF, 1988)
24	TeamMix	Int	In terms of project team there is	3 = some disciplines not represented 2 = some disciplines inappropriately represented 1 = a good mix of software disciplines	USAF Risk Taxonomy (USAF, 1988)
25	TeamCriticalMemberLoss	Int	The loss of one or more critical team members during the project is	3 = very likely 2 = likely 1 = unlikely	USAF Risk Taxonomy (USAF, 1988)
26	ClientRequirementsUnderstanding	Int	In terms of understanding requirements the client has	3 = some understanding 2 = a good understanding 1 = an excellent understanding	USAF Risk Taxonomy (USAF, 1988)
27	ClientFamiliarity	Int	In terms of working with this client, the organisation has	3 = no experience 2 = moderate experience 1 = extensive experience	USAF Risk Taxonomy (USAF, 1988)
28	ConstraintsComputerResources	Int	The computer resources for the project can be described as	3 = new development, inflexible with no growth capacity 2 = available with some growth capacity	USAF Risk Taxonomy (USAF, 1988)

				1 = mature and flexible, growth capacity within the design	
29	ConstraintsPersonnel	Int	The project personnel are	3 = high turnover, little or no experience and not available 2 = available, not in place, some experience 1 = available, inplace, experienced and stable	USAF Risk Taxonomy (USAF, 1988)
30	ConstraintsStandards	Int	In terms of standards	3 = no tailoring required, none applied to contract 2 = some tailoring required, all not reviewed for applicability 1 = they are appropriately tailored for the application	USAF Risk Taxonomy (USAF, 1988)
31	UserProficiency	Int	Users level of proficiency	3 = Very demanding, large number of users 2 = Expert in field 1 = Not an expert	USAF Risk Taxonomy (USAF, 1988)
32	CustomerAccessibility	Int	How accessible is the customer	3 = Not easily available 2 = External, knowledgeable 1 = External available	USAF Risk Taxonomy (USAF, 1988)
33	ApplicationType	Int	With reference to previous applications, how does this one compare	3 = New, complex, highly interactive 2 =Re-Engineering 1 =New, complex, many interfaces, advanced tech	USAF Risk Taxonomy (USAF, 1988)
34	DevelopmentOrganisation	Int	Level of proficiency	3 = Some experience 2 = Good team, experienced 1 = Experienced in technology	USAF Risk Taxonomy (USAF, 1988)
35	ApplicationReuse	Int	The emphasis of reusability for this application is	3=High, reusability essential 2=Medium, being borne in mind 1=Low, not important	P3 Project Handbook Vol II - Internal Project Document
36	ApplicationOriginality	Int	The emphasis of originality	3=High, application must be	P3 Project Handbook Vol

			when developing this application is	original to succeed 2=Medium, limited competition 1=Low, competition irrelevant	II - Internal Project Document
37	ApplicationGenerality	Int	The generality of the application may be classed as	3= Quite general, applicable to a variety of domains 2=Medium, may be applied to varied domains 1= Domain specific	P3 Project Handbook Vol II - Internal Project Document
38	ApplicationComplexity	Int	The complexity of the application may be classed as	3=Complex, extensive training may be required 2=Medium complexity, instruction will be necessary 1=Not complex, no prerequisites for users	P3 Project Handbook Vol II - Internal Project Document
39	ProductFunctionality	Int	The functionality of the product is required to be	3= High, product must provide extensive functionality 2=Medium, an intermediate amount of functionality is required 1=Low, little emphasis on extensive functionality	P3 Project Handbook Vol II - Internal Project Document
40	ProductQuality	Int	The quality level of the product is required to be	3=High level of quality required 2=Medium 1=Low	P3 Project Handbook Vol II - Internal Project Document
41	ProductDependability	Int	The level of dependability required of the product is	3=High, failure unacceptable 2=Medium, failure tolerated but not desired 1=Low, can tolerate failure	P3 Project Handbook Vol II - Internal Project Document
42	ProductPerformance	Int	The performance of the product is	3=Essential for acceptance 2=Preferred for acceptance 1=Not critical to its acceptance	P3 Project Handbook Vol II - Internal Project Document
43	ProductHCIRequirements	Int	Users must interface with the product	3=always 2=sometimes 1=never	P3 Project Handbook Vol II - Internal Project Document

44	ProductMaintenanceCosts	Int	The cost of maintaining this product is	3= High, this product will need continuous maintenance 2=Medium. this product will require a limited amount of maintenance 1= Low, this product will require little or no maintenance	P3 Project Handbook Vol II - Internal Project Document
45	CustomerFinancialCapability	Int	The customers ability to further finance this project is	3=Low, the customer has expended their budget for this project 2=Medium, a possible increase in investment is foreseeable but would require justification 1= High, the customer is closely associated with the project and will provide full financial backing	P3 Project Handbook Vol II - Internal Project Document
46	ProjectConcurrency	Int	The degree to which concurrent development can be applied to this project is	3= Low, few tasks can be parallelised 2=Medium, some tasks are inherently sequential but some may be parallelised 1= High, most tasks may be performed in parallel	P3 Project Handbook Vol II - Internal Project Document
47	ProjectBudget	Int	The Budget for this Project is in thousands	User entered value	P3 Project Handbook Vol II - Internal Project Document
48	ProjectEquipment	Int	The equipment at the disposal of the development team is	3=Less than adequate 2=Adequate 1=More than adequate	P3 Project Handbook Vol II - Internal Project Document
49	ProjectTeamTraining	Int	The level of training required for the development team is	3=Team requires an entirely new set of skills	P3 Project Handbook Vol II - Internal Project

				2=Some training is required to augment current skills 1=None, no training required – current skills are adequate	Document
50	ProjectSubcontracting	Int	The proportion of the project that will be subcontracted is	3= 41% - 100% 2= 11% - 40% 1= 0 – 10%	P3 Project Handbook Vol II - Internal Project Document
51	StaffSize	Int	The number of staff on the project is	User entered value	P3 Project Handbook Vol II - Internal Project Document.
52	TeamSoftwareDevelopmentExperience	Int	Overall the teams experience of professional software development is	3= Low, the team is inexperienced 2= Medium, the team has worked on software projects before 1= High, the team has extensive experience of software development	P3 Project Handbook Vol II - Internal Project Document
53	SafetyCriticalUserEnvironment	Int	How essential is this product to the safety of the user	3=Essential, failure compromises the safety of the user 2=Medium, the product must degrade gracefully to preserve user safety 1=None, no danger to the user pending failure	P3 Project Handbook Vol II - Internal Project Document
54	EconomyCriticalUserEnvironment	Int	How essential is this product to the economical success if the user	3=Essential, failure compromises the economical success of the user 2=Medium, failure must not be frequent and when failure occurs the user must be notified 1=None, no economical risk to the user pending failure	P3 Project Handbook Vol II - Internal Project Document
55	UserCulture	Int	The user has a software usage culture which is	3=Not very mature, the user does not have much previous software	P3 Project Handbook Vol II - Internal Project

				usage experience 2=Medium, the user has limited software usage experience 1=Mature, the user has experience in using various software products.	Document
56	CustomerSchedule	Int	The customer schedule is	3= inflexible, changes will cause schedule problems 2=fairly flexible, changes may require unfavourable alterations to schedule 1= very flexible, changes to schedule are open to negotiation	P3 Project Handbook Vol II - Internal Project Document
57	UserTrainingRequired	Int	The level of training required to allow the user to use the product is	3=High, user has very little familiarity with this type of product 2=Medium, some training will be needed 1=Little or no user training will be required	P3 Project Handbook Vol II - Internal Project Document
58	UserApplicationDomainExperience	Int	The amount of experience the user has in the application domain is	3=Low, the user has little experience of this application domain 2=Medium, the user is familiar with this line of business 1=High, the user has a recognised level of expertise in this area of business	P3 Project Handbook Vol II - Internal Project Document
59	MarketCompetition	Int	The level of market competition facing this product is	3=High, the market is flooded with alternative products 2=Medium, there are a number of alternative products for the user 1=Low, this product has a particular market niche or this	P3 Project Handbook Vol II - Internal Project Document

				product is dedicated to a specific user	
60	StandardISO9001	Boolean	Your organisation meets the requirements set to reach ISO 9001 certification	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
61	StandardCMMLevel2	Boolean	Your organisation meets the requirements set to reach CMM Level 2 certification	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
62	StandardCMMLevel3	Boolean	Your organisation meets the requirements set to reach CMM Level 3 certification	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
63	StandardCMMLevel4	Boolean	Your organisation meets the requirements set to reach CMM Level 4 certification	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
64	StandardCMMLevel5	Boolean	Your organisation meets the requirements set to reach CMM Level 5 certification	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
65	StandardISO15504	Boolean	Your organisation wishes follow the ISO 15504 (SPICE) standard	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
66	OrganisationCodingStandard	Boolean	Your organisation has a coding standard which is in place and followed	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
67	OrganisationDocumentationStandard	Boolean	Your organisation has a documentation standard which is in place and followed	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
68	OrganisationConfigManagementStandard	Boolean	Your organisation has a standard in place and followed for Configuration Management	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
69	CustomerProductStandard	Boolean	The customer has an evaluation procedure for the product	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
70	OrganisationInternalProductStandard	Boolean	A defined internal standard for	True=Yes	P3 Project Handbook Vol

			product evaluation is in place	False=No	II - Internal Project Document
71	SubcontractorStandardRequired	Boolean	A defined standard is in place for evaluation of subcontractors	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
72	SoftwareReuseStandard	Boolean	A defined standard is in place that is expected of software that is reused/off-the-shelf	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
73	HardwareCost	Int	The hardware cost of developing this product is X number of thousands	User entered value	P3 Project Handbook Vol II - Internal Project Document
74	SoftwareCost	Int	The software cost of developing this product is X number of thousands	User entered value	P3 Project Handbook Vol II - Internal Project Document
75	TravelCost	Int	The travel cost of developing this product is X number of thousands	User entered value	P3 Project Handbook Vol II - Internal Project Document
76	EffortCost	Int	The effort cost of developing this product is X number of thousands	User entered value	P3 Project Handbook Vol II - Internal Project Document
77	TrainingCost	Int	The training cost of developing this product is X number of thousands	User entered value	P3 Project Handbook Vol II - Internal Project Document
78	EstimationStandard	Boolean	You have a formal standard of estimation used on all projects	True=Yes False=No	P3 Project Handbook Vol II - Internal Project Document
79	HardwareCostAccuracy	Int	The accuracy of your hardware cost estimation is	3=Not Very accurate 2=Average approximation 1=Very accurate	P3 Project Handbook Vol II - Internal Project Document
80	SoftwareCostAccuracy	Int	The accuracy of your software cost estimation is	3=Not Very accurate 2=Average approximation 1=Very accurate	P3 Project Handbook Vol II - Internal Project Document
81	TravelCostAccuracy	Int	The accuracy of your travel	3=Not Very accurate	P3 Project Handbook Vol

			cost estimation is	2=Average approximation 1=Very accurate	II - Internal Project Document
82	EffortCostAccuracy	Int	The accuracy of your effort cost estimation is	3=Not Very accurate 2=Average approximation 1=Very accurate	P3 Project Handbook Vol II - Internal Project Document
83	TrainingCostAccuracy	Int	The accuracy of your training cost estimation is	3=Not Very accurate 2=Average approximation 1=Very accurate	P3 Project Handbook Vol II - Internal Project Document
84	ProductPortability	Int	The portability of your product is	3= High, the product will operate on multiple platforms 2= Medium, certain modules are platform specific 1= Low, required for a specific platform	P3 Project Handbook Vol II - Internal Project Document
85	ExpectedDuration	Int	The duration of your project is expected to be X weeks	User entered value	P3 Project Handbook Vol II - Internal Project Document
86	ActualDuration	Int	The actual duration of your project is X weeks	User entered value	P3 Project Handbook Vol II - Internal Project Document
87	WorkEnvironmentErgonomics	Int	The environment in which your team operates is	3=Unsuitable, the atmosphere is detrimental to the comfort of the team 2=Mixed, the atmosphere may be compromised by disturbances 1=Ideal, the atmosphere suits software development	P3 Project Handbook Vol II - Internal Project Document
88	*Activity*Duration	Int	The duration of this Activity is	User Entered Value	P3 Project Handbook Vol II - Internal Project Document
89	ProjectLifeCycle	Int	The life cycle chosen for this project is	5=Spiral Model 4=Evolutionary Model 3=Incremental Model	P3 Project Handbook Vol II - Internal Project Document

				2=Waterfall/V 1=None	
90	IndependentVandV	Int	The level of independent Verification and Validation at each stage of the project is	3= Low, there is little or no external verification and validation 2= Medium, there is verification and validation at each stage of development 1= High, the project schedule and continuity depends on external verification and validation	P3 Project Handbook Vol II - Internal Project Document
91	EstimatedProjectRisk	Int	This is a result of risk calculation for this project The user will not set this token – this is an output result token	Output value	P3 Project Handbook Vol II - Internal Project Document
92	CustomerSoftwareCulture	Int	The level of software usage experience at the customer organisation	3=Low, the customer does not have much previous software purchasing experience 2=Medium, the customer has limited experience in general software use but not in this particular application area 1=High, the customer has purchased software applications of this nature previously	P3 Project Handbook Vol II - Internal Project Document
93	MetricDevelopmentBug	Int	Number of bugs observed during a development phase	2 = each week 1 = each month 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
94	MetricValidationBug	Int	Number of anomalies observed during the validation tests	2 = each week 1 = twice a month 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
95	MetricReportedAnomalie	Int	Number of classified anomalies	3 = each week	Metrics suggested by the

			reported	2 = each month 1 = at each milestone 0 = rejected	AMI Handbook (Pulford, 1996)
96	MetricOriginBug	Int	Number of bugs whose origin is requirements/design/coding against time of discovery	3 = each week 2 = each month 1 = at each milestone 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
97	MetricErrorLocation	Int	Error location	3 = each week 2 = each month 1 = at each milestone 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
98	MetricTestingCoverage	Boolean	Testing coverage of each testing phase	1 = adopted 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
99	MetricErrorKLOC	Boolean	Number of errors/KLOC (high level design review errors, code inspection errors, unit test errors, integration test errors etc.)	1 = at each review 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
100	MetricPhaseDelay	Boolean	Delay of each phase and percentage of deviation	1 = at each milestone 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
101	MetricMilestone	Boolean	Percentage of milestones on time	1 = at each milestone 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
102	MetricProductSize	Int	Expansion ratio of product size	2 = twice a month 1 = each month 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
103	MetricProductivity	Int	Productivity: KLOC/person-month	3 = each week 2 = twice a month 1 = each month 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
104	MetricReview	Boolean	Number of hours to prepare a	1 = at each review	Metrics suggested by the

			review vs. number of errors reported and time to fix it	0 = rejected	AMI Handbook (Pulford, 1996)
105	MetricEffort	Int	Effort spent per phase; deviations	3 = each week 2 = twice a month 1 = each month 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
106	MetricTestProductivity	Boolean	Number of test cases passed per unit of time	1 = adopted 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
107	MetricRework	Int	Total effort in rework/phase	3 = each week 2 = twice a month 1 = each month 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
108	MetricProductivityError	Int	Productivity: KLOC with a fixed number of errors/person-month	3 = each week 2 = twice a month 1 = each month 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
109	MetricBugFix	Boolean	Average time to fix a bug (over a month, over a year)	1 = adopted 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
110	MetricComponent	Boolean	Maximum number of components to be corrected in case of error or change request	1 = adopted 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
111	MetricNonRegression	Boolean	Average number of test cases to ensure non-regression	1 = adopted 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
112	MetricNewCode	Int	Percentage of new code in a system	2 = twice a month 1 = each month 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
113	MetricChange	Int	Percentage of changes which introduce faults	2 = twice a month 1 = each month 0 = rejected	Metrics suggested by the AMI Handbook (Pulford, 1996)
114	MetricUnresolvedError	Int	Number of unresolved	2 = twice a month	Metrics suggested by the

			problems/number of solved ones	1 = each month 0 = rejected	AMI Handbook (Pulford, 1996)
115	User Requirements Duration	Int [0..100]	The duration of this Activity is as a percentage of overall duration	User Entered Value	P3 Project Handbook Vol II - Internal Project Document
116	System Requirements Duration	Int [0..100]	The duration of this Activity is as a percentage of overall duration	User Entered Value	P3 Project Handbook Vol II - Internal Project Document
117	Software Requirements Duration	Int [0..100]	The duration of this Activity is as a percentage of overall duration	User Entered Value	P3 Project Handbook Vol II - Internal Project Document
118	Architecture Design Duration	Int [0..100]	The duration of this Activity is as a percentage of overall duration	User Entered Value	P3 Project Handbook Vol II - Internal Project Document
119	Detailed Design Duration	Int [0..100]	The duration of this Activity is as a percentage of overall duration	User Entered Value	P3 Project Handbook Vol II - Internal Project Document
120	Implementation Duration	Int [0..100]	The duration of this Activity is as a percentage of overall duration	User Entered Value	P3 Project Handbook Vol II - Internal Project Document
121	System Integration And Test Duration	Int [0..100]	The duration of this Activity is as a percentage of overall duration	User Entered Value	P3 Project Handbook Vol II - Internal Project Document
122	Acceptance Testing Duration	Int [0..100]	The duration of this Activity is as a percentage of overall duration	User Entered Value	P3 Project Handbook Vol II - Internal Project Document
123	Operational Testing Duration	Int [0..100]	The duration of this Activity is as a percentage of overall duration	User Entered Value	P3 Project Handbook Vol II - Internal Project Document

Appendix B: Prompter Use Cases

1. Use Cases

This section outlines the Use-case analysis for process selection and project instantiation This use-case analysis is a direct extract from the User Requirements document of the P3 Project This document was deliverable to EC as part of the P3 project contract These use-case diagrams are different to an actual use-case with the developed ***Prompter*** tool This is due to the refinement and elimination of various user requirements throughout the project Despite the disparity between the developed product and these use cases, these use cases still provide an insight into how ***Prompter*** was intended to be used

Roles and actors

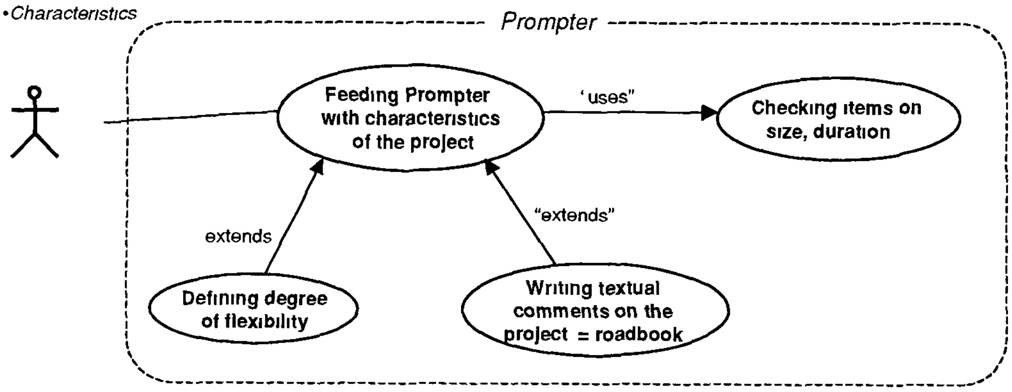
The main actor in ***Prompter*** is the project manager During the process selection phase the project manager performs different roles, as described in the ***Prompter*** Handbook

- Firstly, the project is described *the main characteristics* of the project are specified and *the business drivers* and goals identified
- *Preparation for the process selection,* checking the coherence of what has been defined
- *Selecting the process* with help of these elements
- *Instantiating the process into a project* i e feeding the framework of the process with the values of the project

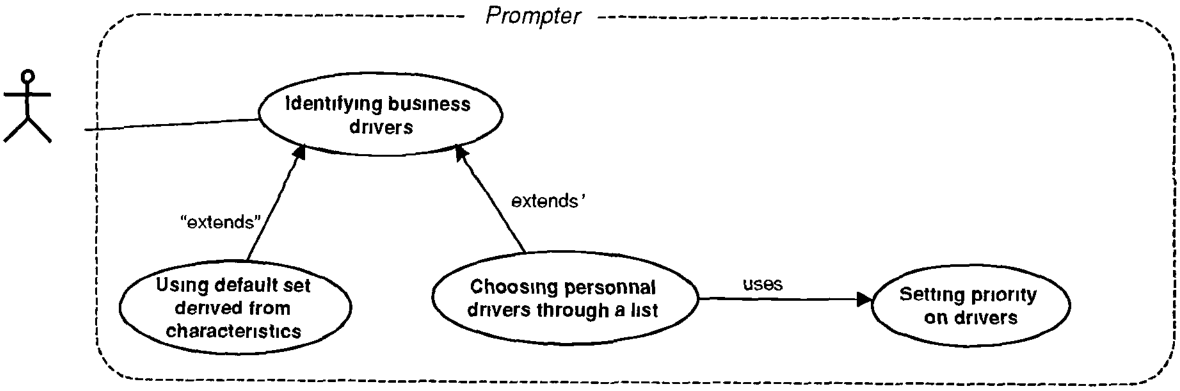
Use cases diagrams

Describing the Project

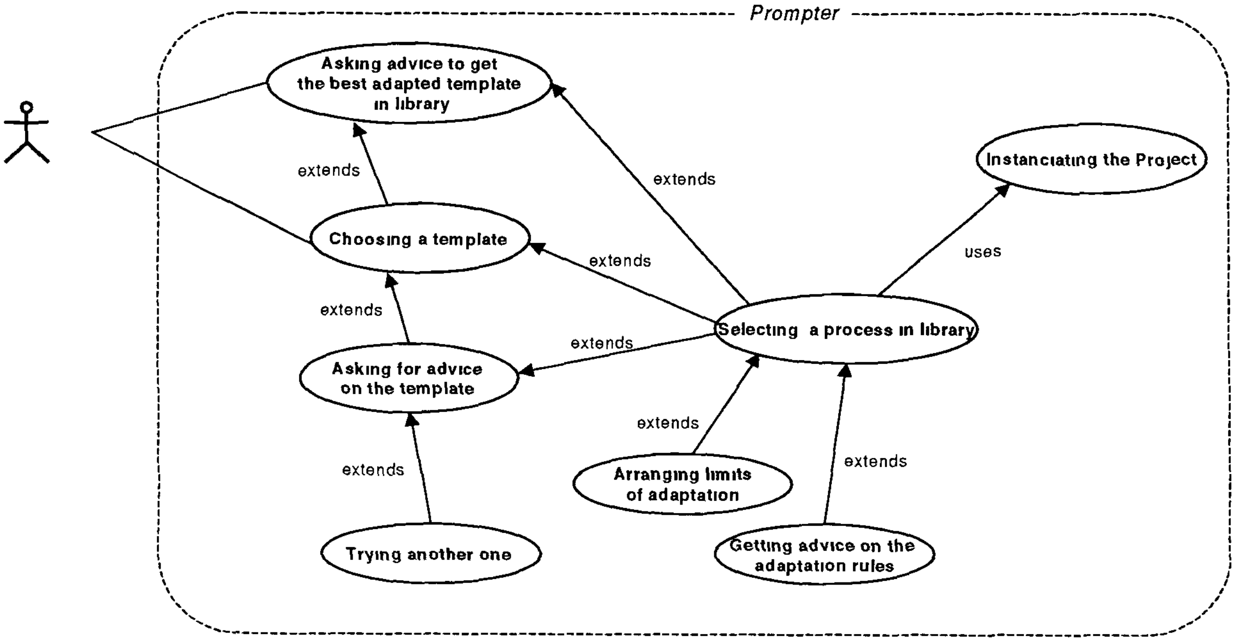
•Characteristics


Prompter
Feeding Prompter with characteristics of the project
' uses"
Checking items on size, duration
extends
"extends"
Defining degree of flexibility
Writing textual comments on the project = roadbook

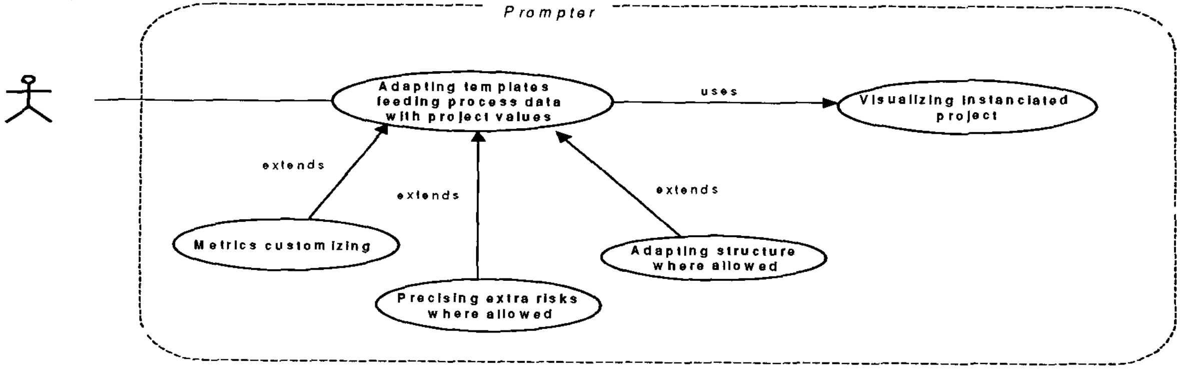
•Business goals


Prompter
Identifying business drivers
"extends"
extends'
Using default set derived from characteristics
Choosing personnal drivers through a list
uses
Setting priority on drivers

Process selection


Prompter
Asking advice to get the best adapted template in library
Instanciating the Project
Choosing a template
Selecting a process in library
Asking for advice on the template
Arranging limits of adaptation
Trying another one
Getting advice on the adaptation rules
extends
extends
extends
extends
extends
extends
extends
extends
uses

Project instantiation


Prompter
Adapting templates feeding process data with project values
uses
Visualizing instanciated project
extends
extends
extends
Metrics customizing
Precising extra risks where allowed
Adapting structure where allowed

Use Cases description

- Describing the project characteristics

description : the user has to feed ***Prompter*** with general data on the project if it is long or short , if it is a sub-project or if there are partners, if new technologies are used or if a similar project as already been undertaken, the degree of flexibility of the project, etc in order to make clearer any constraints on the project Writing textual comments on the project, even if not directly used by ***Prompter*** for assessment, may be a good way to provide the project manager with a kind of "road book" or "diary" of the project Much of this may be achieved by checking or annotating predefined items suggested by the Tool
end : characteristics of the project are known by the Tool

- Describing the project business goals

description : the user has to identify the project business drivers These can be done automatically because the tool could provide a default list derived from project characteristics The users may also want to defined special goals and drivers by setting the priority of general drivers that ***Prompter*** provides
end : business drivers are defined

- Preparing for process selection

description : this step is provided to allow the user to verify if what has been provided to ***Prompter*** is what was intended ***Prompter*** will also use this step to check coherence of the user choices If these are OK, then ***Prompter*** will allow the user to go further and to select the process If not or if the user is not satisfied with his choices, he can re-enter the previous steps and change what is wrong
end : the user is sure that his choices are coherent and that therefore ***Prompter*** should be able to find a process adapted to the characteristics of his project

- Process Selection Choosing the APM

description : this step is where the choice of the relevant process for the project is done Much advice can be provided by ***Prompter*** depending on whether the user knows the life-cycle to be chosen, or needs to consider several possible choices, or lets ***Prompter*** propose the best it can find in process library Rules of adaptation of the process to the current project is also dealt with in this step and the user may ask information on it or may be allowed, for certain processes, to tailor this adaptation
end : the framework of the project is defined Instantiation can be done

- Project Instantiation APM → IPM

description : the user fills the structure of the project with the real data The process can be adapted according to the rules defined before, the metrics customised and risks factors added that are peculiar to its project
end : the project defined and ready to be worked with Summary views of the project are displayed