The University of Reading

Design and Evaluation of Web Applications and Processes

by

Daniel Rodríguez García

Submitted for the Degree of Doctor of Philosophy
Department of Computer Science
The University of Reading

July 2002

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

To my family and Eva

Acknowledgements

I would like to thank Prof. Rachel Harrison and Dr. Manoranjan Satpathy for their supervision and constant encouragement throughout the duration of my research. I truly believe, without their help and guidance this thesis would never have been completed. If I haven’t learnt enough to become a good researcher, it’s my fault.

Thanks to the University of Reading for financial support and to Southampton University where I spent the first 6 months before moving to Reading. Special thanks to Dr. Javier Dolado (University of the Basque Country) for allowing me to join the SEGESOFT research group, supported by CICYT TIC 99-0351 (Spain), a work that will continue even after the completion of this thesis under the ARGO project CICYT TIC2001-1143-C3.

I would like to thank everyone in the Applied Software Engineering research group for their support and friendship; special thanks to Steve, Rachel (McCrindle), Gerard, Sue, Shirley, James, Pat and others in the Department office. Thanks also to all the participants in the interviews.

Thanks to those who replied to my e-mails with all sort of problems, doubts, comments, papers etc. I should mention the names of E. Mendes, L. Olsina, G. Rossi, D. Pfahl, I. Laconsole and J. Aguilar-Ruiz. My special thanks to B. Kitchenham for giving me a demonstration of the MiniSQUID tool.

I also would like to thank those friends, I made during the way of this thesis, for making my life a bit more enjoyable; To name a few pairs: Sissy and Simeon, Eugenia and Roberto, Anthy and George. I have spent many interesting hours with Joffry, Fuji and Darren. Nils has changed a ‘bit’ our way of living by making it healthy and organic, and made our transition to GNU/Linux much smoother. Very special thanks to Hitesh and Conchi who have never allowed us to miss our families.

Abstract

The popularity and widespread use of Web applications require that the process of Web application development be based on sound software engineering principles, and that the Web applications must satisfy the expected quality needs. This thesis concerns the design and assessment of Web processes and applications.

To begin with, we conducted a series of interviews with Web practitioners with the aim of finding out and characterizing the problems faced by practitioners (Web designers and developers) in industry. The analysis of the results of our interviews suggested areas for further research. Our analysis revealed that it was necessary to formalise Web development processes. We have formalised the Web development process though the definition of a new methodology, called the Parallel Framework. All of the design steps of this methodology can be carried out using the UML (Unified Modelling Language). Our proposed design method is modular and therefore its use can alleviate problems.

Next, for the assessment of Web development processes, we designed a Quality Framework. Our Quality Framework is based on a process model called the Typed Generic Process Model (TGPM), and it, unlike most of the existing quality frameworks, makes the relationship between processes and products explicit. We have also developed tool support for our Quality Framework based on a project management tool which supports, including the usual project management features, the use of Bayesian Belief Networks (BBNs) and System Dynamics.

In the latter part of the thesis, we discuss how to generate BBNs from project data, their validation, and how BBNs can be used to predict the outcome of Product-Process-Dependency Models under the PROFES methodology.

Main Contributions

The major contributions of this thesis are as follows:

1. Design of a new methodology, called the Parallel Framework, for Web Development Process (WDP).
    - The methodology modularises the WDP through independent and parallel development of the authoring part and the infrastructure part. This could be seen as an attempt towards modularisation of the WDP.
    - The methodology is based on UML; so, current tools can be used.
    - This methodology addresses the three major issues as regards to WDP, which are: (i) a clear design architecture that reflects the structure of the Web application at conceptual level, (ii) modular approach to development, and (iii) maintainability.
2. This thesis presents a new perspective to quality. Here, by quality we mean both process quality and product quality.
    - We use the Typed Generic Process Model (TGPM) as the basis of our quality framework. We have extended the TGPM for Web applications. Our quality framework makes the relationship between process attributes and product attributes transparent.
    - Our quality framework addresses important aspects of individual processes of Web development through the use of instantiated process models.
    - We also provide a tool support for our quality framework.
3. Development of a proof-of-concept Project Management Tool (PMT).
    - Our PMT uses TGPM as the quality framework.
    - Internal structure of a process is represented in a graphical notation.
    - The PMT uses modern estimation/prediction/analysis techniques such as Bayesian networks and data mining.

To summarise, we have applied some novel software engineering techniques to Web arena; in this sense, this thesis presents some new contributions to Web Engineering.

Publications

- A Generic Model and Tool Support for Assessing and Improving Web Processes", *IEEE Metrics Symposium* 2002. (D. Rodriguez, R. Harrison, M. Satpathy)
- Tool Support for the Typed Generic Quality Model, In *11th Intl. Workshop on Software Measurement (IWSM)*, 2001, Montreal (Rodriguez, D., Harrison, R., Satpathy M. and Dolado J.)
- An Investigation of Prediction Models for Project Management, *IEEE Computer Software and Applications Conference (COMPSAC)*, 2002, Oxford. (Rodriguez, D., Harrison, R., Satpathy M. and Dolado J.)
- Investigation of Product Process Dependency Models through Probabilistic Modelling, *Empirical Assessment in Software Engineering (EASE)*, 2002 (Satpathy M., Harrison R. and Rodriguez D.)
- Integration of Information in a Training Environment for Software Project Management, *Software Quality Management (SQM)* 2001 (Tuya, J., Fernández, P., Prieto, M. A., Aguilar, J., Ramos, I., Riquelme, J., Ferrer, F., Toro, M., Ruiz-Carreira, M., Rodriguez, D., Satpathy, M., Harrison, R., Ruiz de Infante, A., Dolado, J. J., Matilla, R. and Álvarez, M.)
- Generation of Management Rules through System Dynamics and Evolutionary Computation, *Product Focused Software Process Improvement (PROFES)* 2002. (Aguilar-Ruiz, J. S., Riquelme, J. C., Rodríguez, D. and Ramos, I.)
- Maintenance of Object Oriented Systems through Re-engineering: A Case Study, In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM 2002)*, Montréal, 2002, (Satpathy M., Siebel N. T. and Rodriguez D.)
- Latitudinal and Longitudinal Process Diversity, In *Journal of Software Maintenance and Evolution (JSME)*, 15(1), 9-25, Jan/Feb 2003, Siebel, N., Cook S., Satpathy, M., Rodríguez, D.)
- An Investigation on the use of Bayesian Networks in Project management, In *International Conference on Information Technology (CIT2002)*, Bhubaneswar, India, 2002 (D. Rodriguez, R. Harrison and M. Satpathy)
- Practitioners Views on Web Development: An Industrial Survey by Semi-Structured Interviews, In *13th International Conference Software and Systems Engineering and their Applications (ICSSEA)*, 2000, Paris (Rodriguez, D. and Harrison, R.)
- Les points de vue des praticiens sur le developpement des applications reposant sur le Web: Une enquete aupres d'industriels, *Genie Logiciel*, Vol 56 pp. 2-11, 2001 (D. Rodriguez, R. Harrison)
- Un modele generique pour l'evaluation et l'amelioration des processus du developpement des applications Web', *Genie Logiciel*, Vol. 62, pp. 49-56, 2002 (Rodriguez, D., Harrison, R. and Satpathy, M.)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The advent of the *World Wide Web* during the late 80's and the beginning of the 90's resulted in the emergence of a novel type of application, the Web application. Since then, Web applications have had a profound impact on business, education, entertainment and personal life. They can be broadly defined as software applications which include within their framework the plethora of services provided by the Internet; i.e. they are conventional applications augmented with the capability of the Internet. Examples of Web applications include Web Information Systems (WIS), electronic commerce, digital libraries, distance learning systems etc. Furthermore, applications like Geographical Information Systems (GIS), kiosks etc. attain their fullest capabilities when they are elevated to the status of Web applications.

In the following, we will discuss the evolution of Web applications, their similarities and differences with traditional applications, and the issues and challenges of building Web applications. The various Web specific terms and acronyms which we have used in our thesis have been defined in Appendix C.

1.2 Evolution of Web Applications from a Historical Perspective

Web applications started began as static Web sites; i.e. simple pages containing texts, images and sound arranged as hypertext, and navigation from one page to another was achieved by following links. User requests such as following a link were handled by a program called the Web server. Soon, users were given the capability to interact with databases and other programs by means of forms and CGIs (Common Gateway Interface). Gradually, newer technologies like agents, distributed architectures with loosely coupled databases, object servers and such were incorporated into Web applications.

This evolution also reflects how Web applications are classified in relation to the domain and technology point of views. From the functional point of view, Web applications can be broadly classified into the following categories (Isakowitz *et al*, 1998; Powell *et al*, 1998):

(i) *Static Web Presence*. Web site presence is focused on retrieving static information. Examples are corporate information pages or personal home pages etc. Web development consists of an authoring process only.

(ii) *Dynamic Information Sites*. Data intensive Web sites or database driven applications fall within this group, where the Web has a user interface (UI) for retrieving information from back-end databases. There are tools which allow us to deploy these applications very fast (e.g. Macromedia Drumbeat, MS Access, etc).

(iii) *Web Information Systems (WIS)*. WIS are Web applications in which a workflow process has to be followed; e.g., order entry automation, customer care support systems, etc. Sometimes these Web applications act as front-ends of other legacy systems, where the browser provides the user interface. Takahashi and Liang (1997) enumerates the following essential characteristics of WIS:

- Navigation structure designed to support specific workflow.

- Structured data model(s) representing relationships among pieces of information.
- Features enabling users to process business data interactively.
- Support for distributed collaboration work style.
- Referential integrity for mission critical tasks.

(iv) *Electronic Commerce*. These are also transaction oriented, but they are external to an organisation and require a greater level of security. E-commerce applications are usually classified into three classes: (i) business-to-business; (ii) business-to-customer and (iii) customer-to-customer.

In our thesis, whenever we refer to Web applications, they will mostly fall into the categories from (i) to (iii) of Isakowitz's classification scheme. We are not directly addressing the category (iv) because they involve business standards and some specific communication protocols, which we do not deal with in our thesis.

From the technology point of view, Orfali *et al* (1999) classify Web applications as a variant of client-server computing. In Figure 1.1 we show how different types of Web applications and related technologies have evolved since the inception of the Web.

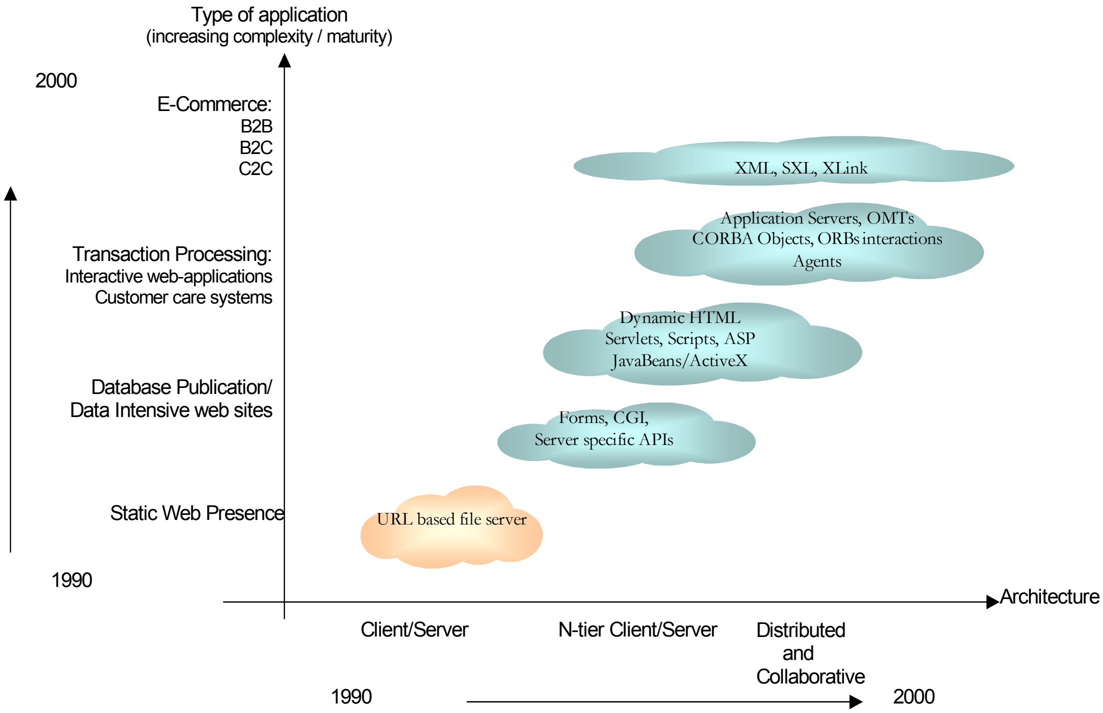


Figure 1.1 Types of Web Based Applications and their Technology

1.3 Web Applications Vs. Traditional Applications

Web applications differ from traditional applications from both the product and the process point of view. As products, they differ from traditional systems in the following ways (Pressman, 2000; Deshpande *et al*, 1999; Lowe and Hall, 1999) :

- Web based applications are distributed and component based, and are part of the client/server paradigm in the sense that they are composed of a number of components such as servers, databases, middleware etc.
- High reliability: Web applications in general and, e-commerce applications in particular, must have high reliability; i.e. servers must be available all the time.
- High scalability: Web applications have the potential of attracting and reaching a very wide audience.
- High usability: The users of Web applications are usually members of the general public, not technical experts. A Web application must have the potential to attract such users. Hence, usability of Web products must be high. Also, there are no

geographical boundaries and so cultural and language issues need to be borne in mind.

- Security: In many Web applications like e-banking, e-commerce etc., security is of prime concern.
- Saleability (advertising, Web-site popularity etc.). Marketing is a stimulus behind many Web applications, so marketing ideas may need to be incorporated.
- Customisation: Web applications should be customisable with respect to various factors like user preferences (e.g. languages), device independence, mobility scenarios etc.

Web applications also differ from traditional applications from the process point of view, such as (Pressman, 2000; Lowe and Hall, 1999):

- Technologies from a wide spectrum (HTML, XML, network protocols, multimedia, Java, and script languages) are involved in Web applications, and thus, many roles (authors, developers, graphic designers, legal issues etc.) have to be managed.
- Shorter time to market and shorter product life cycles.
- Hypermedia characteristics have to be taken into account. These include navigational design, aesthetics, cognitive aspects and user centred design.
- Testing Web applications is more complicated because of their heterogeneous, distributed, and concurrent nature (Powell *et al*, 1998).
- Web applications have an inherently dynamic nature in which maintenance is a continuous process. Practitioners have reported that *"most Web sites change their look and feel once a year to stay attractive. Minor changes in the look and feel of a site several times a year are common, and major modifications occur frequently"* (Kirda *et al*, 2001). Lowe uses the *Web gardening* analogy to describe Web evolution (Lowe, 1999).

1.4 Web Technologies

We can define Web technologies as a set of languages, protocols and tools that allow us to create Web applications. Web technologies include Web authoring tools, content management systems, publishing, database integrators etc.

From the technology point of view, we can say that Web applications are a kind of client/server application. However, they are attracting interest because they have a number of advantages over traditional client/server applications in the sense of the following:

- Ubiquity: The Web enables software to run over any user's desktop irrespective of its location; it makes information and services easily available to the widest possible audience.
- Simplicity: HTML, XML and related standards are mark-up languages that are much easier to learn than other programming languages. In addition, HTTP is a simple text-based protocol.
- User Interface: Browsers are available for all platforms in the sense that provide a universal user interface and cross platform independence.
- Central Administration: All processing activities reside on the server, which transports data content and results to the client as and when needed. The platform-independence of a Web client eliminates the ongoing need to install and upgrade specific versions of applications.
- Low Cost: Web technology offers an inexpensive and easy way to share information across an entire organisation.
- Scalability: Standard-oriented infrastructure offers availability of low-cost tools such as Web server, indexing and search tools, middleware and third party applications etc.

1.5 Web Engineering

Web engineering concerns the development, deployment and maintenance of Web-based systems and applications through the use of scientific, engineering and management principles (Murugesan and Deshpande, 2001). Web engineering has its own issues and challenges. As discussed earlier, Web applications possess many unique aspects from the technology and economic viewpoints. Technology, in relation to Web applications, is evolving at great pace and it may also involve integration of a number of third-party products. Web applications are rich in quality requirements. Offutt (2002) surveyed managers and practitioners of Web applications and found that the three most important factors are reliability, usability and security. Other important criteria include availability, scalability, maintainability and time to market. Orthogonally, the process of Web engineering must also be rich in terms of process quality requirements. It is also necessary to evaluate Web applications and processes as to whether they possess the required quality needs and to what degrees.

1.5.1 Evaluation of Web Processes and Applications

Web applications need evaluation to verify how good they are as products and how good they are in terms of quality requirements. The process of Web engineering needs evaluation to find out how good the process is in terms of cost and efficiency. Evaluations produce indicators which may offer guidance for addressing the deficiencies that the Web application or the process may have. Moreover, evaluation is necessary to identify tools and methods that are ideal for a given organisation or for a given task.

Evaluation of a product or a process is usually based on a measurement framework and quality models. The measurement framework identifies metrics, which provide indicators to evaluate a product or a process. In short, evaluation helps us in (i) understanding what is happening during development and maintenance; (ii) controlling what is happening on projects; and (iii) improving processes and products after evaluation.

1.5.2 What is Quality? What are Quality Models?

Quality of a product or a process is defined in terms of a set of characteristics that the concerned object should have; for instance, a product should be defect free and it should also satisfy user needs. Similarly a process should produce a product in scheduled time and assigned budget. Such quality characteristics are formally defined in terms of quality models.

Garvin (Garvin, 1988) suggests five different perspectives of quality:

(i) *the transcendent view*, which reflects a feeling that quality changes with time;

(ii) *The product-based view*, which related quality with the inherent characteristics of the product.

(iii) *The value-based view*, which relates the quality and value for money

(iv) *The manufacturing view*, which is related to the conformance to specification definition.

(v) *The user-based view*, which relates to the fitness for purpose definition.

It is important to take into account all this views when developing quality models and frameworks. Kitchenham *et al* (1987) and Gillies (1997) view of quality is based upon the five views of quality set out by Garvin.

A *quality model* can be considered to be a framework for unifying different viewpoints of quality requirements. Quality models are usually represented by a quality tree. Higher level quality requirements are important attributes or factors. At the lower level, each attribute (or factor) is composed of one or more subfactors, often known as internal attributes. The internal attributes of a quality model are usually measured using standard metrics. The most commonly cited quality model is the ISO 9126 standard (1991) which is based on early works by McCall (1977) and Boehm (1978). These models concern the quality of products.

There are also quality models for process assessment and improvement. The most influential ones include the Capability Maturity Model (CMM) (Humphrey, 1992), ISO/IEC 12207 (ISO, 1995), ISO/IEC 9000 (ISO, 1994), the BOOTSTRAP model (Kuvaja, 1994), and the ISO/IEC 15504 – SPICE (ISO, 1998).

1.6 The Problem and our Approach

We started with the belief that current approaches towards the development and maintenance of Web applications have deficiencies and further, Web applications are still deficient in terms of performance and quality. For example, a survey by Cutter Consortium (2000) found the following problems with Web projects: (i) schedule delays 79% of the time; (ii) the delivered systems met business needs only 16% of the time; (iii) project exceeded budget in 63% of the cases; (iv) system delivered did not have required functionality in 53% of the cases, and (v) deliverables were of poor quality in 52% of the cases. Therefore, the central motivation for the research presented in this thesis centres around the improvement of the quality of Web processes and applications. We started our research work with the following aims:

(i) To understand the current approaches to Web Engineering in industry and to investigate deficiencies, if any, in them.

(ii) To develop a new methodology to address the existing deficiencies.

(iii) To understand the current approaches to evaluating Web processes and applications.

(iv) To develop a measurement framework to cater to the specific issues of Web processes.

(v) To support the measurement framework and evaluation methods with tools.

1.7 Main Results

The main results of this thesis can be summarised through the following points:

- In order to understand the current methodologies in relation to Web processes and applications, we conducted semi-structured interviews with practitioners in industry. From our interviews, we observed that (i) Web processes are becoming more complex as more technologies are involved; (ii) Web applications require new approaches to design and development as current methods do not have adequate mechanisms for specifying hypermedia concepts, and (iii) although measurement of software metrics is essential to ensure product and process qualities, Web practitioners very rarely use them.

- We developed a new methodology using the Unified Modelling Language (UML) (Booch *et al*, 1999) notation for the development process of Web applications. Our UML based approach ensures that currently available tools for UML can provide tool support for our methodology. Our methodology is better than existing methodologies in terms of (i) cleaner design architecture that reflect the structure at conceptual level, (ii) modular approach to development, and (iii) maintainability.
- We have developed a customised quality model for Web development process which is based on the Typed Generic Process Model (TGPM) (Satpathy *et al*, 2000). Based on this quality model, we have defined a measurement framework for Web processes. The central idea behind our customized model is that it makes explicit the quality relationship between processes and products as regards to Web applications.
- We have developed a project management tool which supports the TGPM, and hence also our customised model. Moreover, the tool supports the integration of different techniques such as Bayesian Belief Networks (Jensen, 1996) and System Dynamics (Abdel-Hamid and Madnick, 1991) for exploring new ways of solving problems in software engineering in general, and Web applications in particular. For instance, we focused on Bayesian Belief Networks (BBN) to estimate quality characteristics of both software processes and products.
- Creation of BBNs is an intellectual task. We have discussed how they can be constructed from historical data using data mining techniques including their advantages and limitations.

1.8 Related Work

In this section, we will discuss briefly how our research contribution stands in relation to the work available in literature. However, work related to each chapter is discussed in the respective chapters.

Barry and Lang (2001) have carried out an industrial survey and their results suggest that no uniform approach exists to multimedia systems development and that practitioners are not using the multimedia models cited in the literature. They also conclude that developers need new techniques that capture requirements and integrate them within a systems development framework.

During the last decade, researchers have proposed hypermedia and Web methodologies based on the *Dexter Reference Model* (Halasz and Schwartz, 1990, 1994) which provides important abstractions (e.g., node, link, anchor etc.) found in a wide range of hypertext systems. Relevant methodologies related to our work include: (i) the Object Oriented Hypermedia Design Method (Schwabe and Rossi, 1998; Schwabe *et al*, 2001; Schwabe and Rossi, 1995); (ii) the Relationship Management Methodology (Isakowitz *et al*, 1995); (iii) Conallen's model (1999; 2000); (iv) and the methodology by Baumeister *et al* (1999). These models and methodologies will be considered in more detail in Chapter 3.

In relation to Web process improvement, Bazzana and Fagnoni (1999) describe PI3 (Process Improvement In Internet service providing), a process improvement plan for an IT company. They used CMM (Humphrey, 1992) to improve key process areas (KPA) in several issues: technical, business, organisational and cultural. However, we use the TGPM (Satpathy *et al*, 2000) since it makes explicit the relationship between process and product attributes as will be explained in Chapters 4 and 5.

If we consider that all the software engineering techniques are needed when developing Web applications, we must also consider project management techniques. Project management is defined as the application of knowledge, skills, tools and techniques to project activity to meet or exceed stakeholder needs and expectations from a project (PMI, 1996). From medium to large size projects, tool support for project management becomes imperative. However, such tools usually deal with tasks, time and resources but do not take into account quality issues. Our project management tools provide quality as an orthogonal axis based on the TGPM.

Finally, the PROFES (PROduct Focused process improvement or Embedded Systems) methodology (2002a) is related to our work in the sense that we demonstrate how to make use of BBNs to predict the outcome of the improvement actions which are identified from a PPD (Product Process Dependency) repository (PROFES, 2002b). This methodology identifies important quality goals of the final product which need to be improved through actions that are identified in the PPD repository.

1.9 Outline of the Thesis

Chapter 2 is concerned with the semi-structured interviews that we conducted with Web practitioners. It describes a qualitative survey aimed at analysing the processes involved in the development of Web applications in industry. The findings of this interviews identified areas of subsequent research which are covered in the following chapters.

Chapter 3 discusses a new development framework for Web applications. This chapter also describes how this development framework addresses many deficiencies of the existing methodologies.

Chapter 4 describes the design and development of a project management tool which provides mechanisms for constructing processes and dealing with different quality metrics generated during the development of a project. In addition, it provides an integrated framework in which different techniques such as Bayesian Belief Networks, System Dynamics etc. can be used. We have also discussed how the toolset fits into the CMM (Capability Maturity Model) framework.

Chapter 5 uses the TGPM for evaluating Web processes and products which makes the relationship that exists between products and processes explicit. The aim is to perform process assessment by using techniques from empirical software engineering such as the GQM (Goal/Question/Metric) method.

Chapter 6 describes how to generate BBNs from project data and BBNs advantages and limitations. We also discuss how BBNs can be used to predict the outcome of PPD (Product-Process-Dependency) models under the PROFES methodology (2002a) and hence the impact of the associated improvement actions. Finally, it also discusses their advantages and limitations and how BBNs can be validated using a framework proposed by Kitchenham *et al* (2002).

Chapter 7 summarises the results of this thesis and discusses future research.

Chapter 2
An Investigation of Views on Web Development by Practitioners

2.1 Introduction

This chapter reports on a series of semi-structured interviews which were conducted with Web practitioners in several organisations. A semi-structured questionnaire was used to investigate their development processes, quality assurance (including metrics used), tool support etc. The main objectives behind this study were:

(i) To study the current development processes and to determine their deficiencies, if any.

(ii) To determine whether the current practices meet necessary quality requirements from both product and process point of views.

(iii) To suggest ways for improving any deficiencies.

2.2 Empirical Assessment Frameworks

Investigation of a technique, tool or method involves an evaluation exercise. Fenton and Pfleeger (1997) consider three ways of organising an evaluation exercise:

- *Survey* is a retrospective study of a situation to try to document relationships and outcomes.
- *Case study* is a research technique in which the investigator identifies key factors that may affect the outcome of an activity and then documents the activity.
- *Formal experiment* is a rigorous, controlled investigation of an activity, where key factors are identified and manipulated to document their effects on the outcome.

Formal experiments, case studies and surveys can be done either quantitatively or qualitatively. Qualitative research is concerned with studying objects in their natural surrounding. Qualitative research aims at interpreting a phenomenon based on explanations by people (Denzin, 1994). Quantitative research is mainly concerned with quantifying a relationship or to compare two or more groups. The aim is to identify a cause-effect relationship (Creswell *et al*, 1996). Daly (1996) points out the value of using all three forms of empirical assessment to support each other in generating hypotheses and establishing the results. The qualitative survey contributes to the formulation of a relevant hypothesis, formal experiments confirm if a relationship exists, and finally, case studies determine whether the results can be generalised. In addition, Perry *et al* (2000) suggest empirical studies can be used not only retrospectively to validate ideas, but also proactively to direct research, identifying and justifying its potential value.

2.2.1 Qualitative Evaluations via Interviews

Interviews are a qualitative method used to collect historical data from the memories of the interviewees (Seaman, 1999). They can be of several types: (i) a structured interview consists of an interview where “the questions are in the hands of the interviewer”; (ii) in unstructured interviews, “the interviewee is the source of both questions and answers” and the objective is to get as much information as possible from a topic; (iii) semi-structure interviews consist of open-ended questions to elicit both the information foreseen and unexpected questions.

The analysis of the qualitative data can be twofold: generation of a theory or confirmation of a theory. Generation of a theory is performed by extracting from a set of field notes a statement or preposition that is supported by the data. Basili *et al* (1999) also state that surveys help in the formulation of hypotheses and in finding out areas for further investigation. In addition, empirical assessment techniques need to be used to check the results of the interviews. In order to generate a theory, the constant comparison method can be used (Glaser and Strauss, 1967; Seaman, 1999). This consists of adding codes to field memos, grouping the information according to the codes and writing a field memo synthesising the findings.

2.3 Design of the Survey

2.3.1 Aim

The aim of our survey was to explore the experiences of practitioners directly in order to discover the main issues involved in the development of Web applications. In particular, we wanted to identify issues surrounding the activities, artefacts and metrics that are needed when developing these kinds of applications. This survey can be seen as the first stage of a multi-method approach to assessment and improvement of Web applications (Daly, 1996).

2.3.2 Design of the Questionnaire

Our survey was conducted by semi-structured interviews based on a questionnaire presented in Appendix A. Our interview was semi-structured because we wanted to determine to what extent best practices in literature were followed in industry and further what were the customised variations. The questionnaire was structured following the software process life cycle in order to facilitate the analysis and coding of the results as per the constant comparison method (Seaman, 1999). Members of the Applied Software Engineering group at the University of Reading reviewed the questionnaire and their feedback prompted some changes to it.

2.3.3 Selection of Subjects

There is always a trade-off between an extensive but superficial survey and an in-depth but possibly unrepresentative survey. With a small set, a wider exploration of the topic and a higher level of confidence in the answers are possible. We selected a few successful companies involved in Web applications in order to get subjects with experience in Web development. Companies of different sizes and types were selected so that the set of subjects interviewed would be a representative one.

Participating organisations were initially contacted by email with a brief outline of our aims and the questions that would be asked. The interviews were structured around an open-ended questionnaire but the interviewees were encouraged to digress and elaborate on topics as much as they felt necessary. Table 2.1 shows a list of the practitioners who were interviewed. In organisation A, we were able to carry out three different interviews at two different departments.

ID of the organisation	*Type of Web Apps*	*Type of interviewee*	*Org. Size*	*Size Typical Project (people)*
A	Web IS (3interviews/2dep't.)	Software engineers	1,500	2-3
B	Software Systems	Project manager	450	12
C	Consultancy and Systems Manufacturer	Sales consultant	>100,000	3 to hundreds
D	ISP and Web design	Software engineer	15	2
E	Insurance Web solutions	Software engineer	18	6
F	Financial and Pharmaceutical Web solutions	Project manager	25	4-5
G	Web Design	Web designer	6	2
H	Government IS	Project manager	20,000	50 (Size of the project carried out at the time of the interview)

Table 2.1 Participating Organisations

2.4 Highlights of the Survey

Our results were obtained from a limited sample. Therefore, the results were viewed as a starting point for further research.

2.4.1 Project Management, Metrics and Quality Assurance

It is usually believed that Web projects are managed in a way similar to other software projects. However, we observed that the pace of change, the reduced time to market, fast turnaround and fast prototyping of Web applications add additional challenges to management. For example, an interviewee in organisation A commented: *a problem that has frequently occurred is that a prototype has been rushed into production either at the customer's insistence, or at the wish of our own management, with lots of resultant problems*.

Our view of these problems is that they are not specific to Web development but they are more severe in this type of application than other application domains. One interviewee from organisation A and another from B noted that more research in project management would be beneficial.

2.4.1.1 Standard Certification

Organisations B, C, D and some departments in A had ISO certification. Organisation B was expected to reach CMM Level 2 within the next two months. Regarding standard certification, one of the interviewees suggested that ISO certification does not necessarily mean a good and well defined development process, and having defined key process areas, as in CMM could be more useful. This is consistent with the observation made by Bazzana that in a European-wide awareness survey 65% of the respondents agreed that certification against ISO 9000 is not enough to guarantee the quality of software products (Bazzana *et al*, 1993). From the collected data, we observed that the bigger the company, the more standards and procedures were defined. For example, configuration management and reuse are more ad-hoc in small organisations than in the large ones.

In general, quality is only seen in two ways: the absence of errors and customer satisfaction.

2.4.1.2 Estimation

The surveyed companies estimate schedules, cost and resources by analogy using judgement and experience from previous projects, adding some risk factors depending on the use of novel technologies (e.g., using XML for the first time). The interviewees commented that estimation involving new technologies may be very difficult and the estimates only become more accurate after experience with similar projects. The common technique is to create a WBS (Work Breakdown Structure) as accurately as possible, which in turn would help in planning estimation, milestones setting, and resources assignment.

From our point of view, this poor use of process estimation methods in contrast to modern methods like COSMIC Function points (Abran, 1999), Bayesian Belief Networks (Jensen, 1996) etc. means that there is a lack of rigor in process activities. Furthermore, we noticed that small companies did not use project management tools.

2.4.1.3 Process Metrics

All organisations but E and G surveyed collect metrics related to time and effort by means of timesheets, which are used for billing clients and/or for better estimations in future projects. Other common measures in large organisations were related to maintenance, such as the number of phone calls received by help desks and bug-fixes when the system was operational, as this contributed to users' perception of the usefulness of the system. A well-defined pattern observed from the surveyed companies is that processes were better measured in larger companies.

2.4.1.4 Product Metrics

The use of product metrics is quite sporadic in industry and in general, a measurement program is seen with little respect. The two main reasons raised from the interviews are that metrics are not found to be good value for money. Furthermore, interviewees from organisation A and D were of the opinion that metrics pertaining to product quality are simply common sense. One interviewee at organisation A commented: *"At various times*

I have worked in groups where they were tried, but they were not found to be value for money".

Two of the interviewees commented that quality metrics judgements are really a matter of common sense and this seems to be especially true with Web systems. One of them used the readability example where readability is seen as a very subjective measure. Another commented on the example of user interface performance, where downloads are at the mercy of the intervening networks. Company B measures the basic attributes such as use cases, lines of code etc. When the systems are in production, the metrics collected by the Web server are extensively used to analyse the users. Most of the organisations used this facility.

In general, few metrics are collected in relation to products under development and these metrics are the ones that are easy to collect and automate, i.e., broken links, page size etc.

2.4.1.5 Resources

A pattern that arises from our interviews is that, in general, there is a clear distinction between the front-end and back-end of the development process. According to an interviewee in organisation A, this is very beneficial in separating presentation from functionality. In organisation H, a very large organisation, developers in the front-end only have the high-level view of the application. They need only to know about the high-level functions to calls; for example, ASP pages that called functions in the backend processing. On the other hand, back-end developers work at the object level. The interviewee of the organisation commented that this was quite different from traditional systems where developers are more homogeneous. In the smaller companies where their projects consist of static Web sites or small database driven applications, the distinction is between the authoring/design role and the application development role. An interviewee in organisation B commented that more technologies (Java, JavaScript, HTML, CORBA etc.) are involved in Web-based applications than in traditional development. Thus, more roles are involved and staff with more skills are required.

Table 2.2 summarises the findings related to project management, metrics and quality assurance for all the organisations.

Org^n	*Estimation*	*Standards*	*Process Metrics*	*Product Metrics*
A	experience (without historic data)	None	Only 'ad-hoc' for particular projects	Logs server checked No quality models defined
B	Not defined (new projects at that time)	ISO	Cost and programmer productivity	LOC, # Use Cases (UC). No quality models defined
C	experience (with historic data)	ISO	Basic cost metrics without detail	*(Unknown by the interviewee)*
D	Experience (without historic data)	ISO	Timesheets	No.
E	experience (without historic data)	No	No	Web visits related to policies sold and quotes given. No quality models defined
F	Experience (without historic data)	No	Timesheets	Logs server checked No quality model defined
G	Experience (without historic data)	No	No	Not collected but check readability, page size, …
H	Experience (with historic data)	ISO	Timesheets, data from conf. management sys. and call centre	No

Table 2.2 Summary of Standards and Metrics

2.4.2 Requirements

The most common way of eliciting requirements is by means of use cases, which are used by the companies B, C and F. Company C uses standard questionnaires based on their customised methodology as well. Requirements are generally recorded just with documents in paper and no special tools are used. However, company B uses Rational Suite (Rational, 2002) to support requirements elicitation. In addition, prototypes and developmental releases of Web sites are used in companies D and F (and sometimes in A) to match client's required functionality. Table 2.3 summarises the findings.

Org^n	*Requirement capture approaches*
A	No tools.
B	Rational Requisite Pro from Rational Inc.
C	Standard questionnaires
D	Nothing specific. No tools
E	No tools. Role of the IT Manager
F	No tools.
G	No tools
H	MS Project for the PM

Table 2.3 Summary Requirements

2.4.3 Analysis and Design

There are new activities and roles to manage the content, design and structure of Web applications; however, approaches found in the academic literature such as the Object-Oriented Hypermedia Design Methodology (OOHDM) (Schwabe *et al*, 2001; Schwabe and Rossi, 1995) and Relationship Management Methodology (RMM) (Isakowitz *et al*, 1995) were not strictly followed in any of the organisations; however, some of the companies use some concepts of both of these methodologies, one example being the navigational modelling.

The most common technique was to create flowcharts and storyboards of navigation using a top-down approach. Company C used patterns as a starting point. In comparison, companies B and H defined prescriptive tasks, which constituted procedures and guidelines to be followed. The whole Rational Unified Process was followed by company B, and UML Activity Diagrams were used in order to model the hypermedia structure. One interviewee from organisation A commented that new artefacts would be beneficial in representing the hypermedia domain. He commented that current methods work well when the number of relationships is small, but they are poor notations for modelling large Web sites. In the literature, the same observation has been made by Noack and Schienmann (1999). Table 2.4 summarises the findings.

Org^n	*Hypermedia Design*
A	Storyboards (Web hierarchy)
B	Rational Unified Process
C	Rational Rose and Holosofx
D	Storyboards and prototypes
E	Ad-hoc
F	UML and Use Cases
G	Storyboards
H	Procedures, guidelines

Table 2.4 Approaches to Hypermedia Design

2.4.4 Reuse

Only company C (the largest one) is using a process for reuse supported by the use of Lotus Notes as a database of work products from previous projects. Company B is now implementing a reuse process and there is emphasis on document artefacts to be reused in future.

One interviewee in B pointed out that reuse within a process is very important; however, sometimes components are developed from scratch because they do not trust the quality of the components. One subject in F commented that they only bought components from trusted sources, i.e., companies that are expected to be in the market for a long time. Another interviewee in organisation A commented that *"it is really important when choosing the servers and software that you know that the people who support them are reliable, experienced with the product and not likely to change job (internally or externally) for the next few years. This is why it is often better from the user's point of view to have older technology used rather than the latest thing"*.

Reuse within the small companies (D and G) is also high but ad-hoc. This is because many of the applications or the structures of Web sites are very similar. In addition, many applications can be open source and a lot of their work consists of customising and integrating them. However, this knowledge is dispersed around the people in the company without following any process. An interviewee from A commented that the constant adoption of new technologies makes it difficult to reuse any pre-existing material. Some of the ideas from Design Patterns (Gamma, 1995) can be incorporated but everything else needs to be done from scratch. The interviewee at company B commented about the different treatment for different parts of the application. With Web applications, frameworks and components that will be reused should be developed more carefully and more attention should be paid to them. Table 2.5 summarises the findings related to reuse.

Org^{n}	*Extent of Reuse*
A	No process. Mainly ideas (using an integrated tool)
B	There are designs, guidelines and procedures, which are reused
C	Reuse is built into processes, databases of artefacts from previous projects
D	Reuse of Open source software
E	Components refactoring code
F	Components and documentation
G	Reuse of Open Source software
H	Components

Table 2.5 Reuse Summary

2.4.5 Maintenance

Some of the organisations interviewed use configuration management procedures and tools; three out of the eight companies do not use any kind of configuration management procedures. Usually sites are database driven and toolsets are used to manage links, for example IBM WebSphere Studio (IBM, 2002). Company C is using their own tools and the Tivoli suite for configuration management. Company B is using the Rational Suite Test Studio and Rational TeamTest.

As discussed in Section 2.4.1.4, metrics related to the Web server’s logs are also used for maintenance in the sense of checking the use of the system, broken links, type of users etc. In organisation A, occasionally, metrics like maximum turnaround time for user queries and bug fixes are measured, as they would contribute to user’s perception of the usefulness of the system. Organisation H also collects metrics from the help desk. Table 2.6 summarises the findings related to maintenance.

Org^n	*Maintenance*
A	Ad-hoc. No configuration management
B	Ad-hoc. Rational Clear case for configuration management
C	Maintenance Team. Database driven toolsets used to manage links
D	No maintenance unless the client ask for it. No configuration management
E	Ad hoc. MS Visual Source safe as configuration management tool
F	Procedures. MS Visual source safe
G	No maintenance unless the client ask for it. No configuration management
H	Procedures. MS Visual source and home-developed tools

Table 2.6 Approaches to Maintenance

2.4.6 Technology

2.4.6.1 From the Developers' Point of View

In one case (organisation A) the project used an integrated package for Web development. This integrated package provided limited functionality that sufficed as long as no extra *'out-of-the box'* functionality was required. When this happens, the interviewee said: *we are continually fighting against some severe limitations and restrictions*. Furthermore, he recommended avoiding such integrated tools in future and opting for more general pluggable systems.

In general, device independence is considered highly important in order to get universal user interfaces and cross platform independence; for instance, features specific to Netscape and Internet explorers should be avoided.

In addition, two of the organisations (E and F) have projects that involve the use of WAP technology and other devices such as kiosks. In these cases, the interviewees commented, user interface tests were hard to perform.

2.4.6.2 From the Users' Point of View

Interviewees were of the opinion that most users look for the most effective way of getting their jobs done. An interviewee from organisation A commented *"always remember that the customers do not get excited about novel technology, just about doing the job they are there to do better, faster and more easily"*. In addition, regarding customer satisfaction, the same interviewee stated: *"The ultimate quality of a system is judged by whether the end user uses it regularly instead of their previous method of working, as it speeds up and facilitates their own job, rather than because they are forced to by lack of alternative or management pressure. It is easy to write a beautiful, theoretically perfect system that the users just do not find attractive enough to use. In this case, though technically successful, the project is a failure"*.

2.4.7 Social Aspects

The last question of the questionnaire was whether there were any important issues that were not covered. In this context, one interviewee raised the following social issues:

- Training users is much easier because users are familiar with the UI of the Web and only need to understand the underlying data or some additional functionality. Much of this can be done by means of on-line help.
- There are social changes in organisations; for instance, more democratisation of the information (more information is more accessible to more people). The drawback is that users may become more anonymous.

2.5 Lessons Learned

From the findings of these interviews and from the literature, we can conclude that there are two main problems when developing Web applications. They were used to identify initial research of this thesis. The first problem is related to authoring, i.e., hypermedia development of Web applications. The second, which is more general, is related to their management and quality characteristics. In relation to hypermedia development, there is a lack of guidance and artefacts concerning the design of hypermedia components in a standard way.

We can highlight the following points from the interviews:

- Web applications have two distinct domains: the hypermedia and the application. In industry, these are often known as the front-end and the back-end of a Web application. Our observation is that both domains need different approaches to development and require different skills. Thus, the natural question is whether we can develop both the domains separately, possibly by different development teams, and then link them together to obtain the full Web application. This will not only make the development process and application modular but also the skills of the developers could be used more effectively. We will address such issues in Chapter 3.

- From an interviewee from organisation A and from the literature (Noack and Schienmann, 1999; Barry and Lang, 2001), we can conclude that there is a lack of methods, activities and artefacts to realise hypermedia characteristics such as structure, content and navigation. For example, Noack and Schienmann (1999) report after introducing object-oriented technology in a banking organisation: *"the combination of OO modelling and Web design presents some challenges. [...] We need both methodological guidance and new work products completing the UML modelling kernel to reflect hypermedia, HTML, and XML pages and navigational links in analysis and design"*. Research methodologies like RMM, HDM and OOHDM address such issues; however they do not follow any standard approach (e.g. UML) to modelling and therefore, as we observed, industries do not use them. It would be better if such methodologies followed a standard such as UML for modelling. We are of the opinion that recent approaches by Conallen (1999; 2000) and Baumeister *et al* (1999), in which they have enriched UML for modelling hypermedia applications, could be beneficial to industry.
- Maintenance of Web applications is more difficult than that of traditional applications because in addition to the maintenance of the Web applications which require more technologies than traditional applications, a large amount of content and different media also need to be under control management (Dart, 1999). Technologies such as content management tools, version control systems and standards such as XML should be used for better maintenance.
- As we observed, reuse is carried out in the industries studied in an ad-hoc manner. Sometimes components are developed from scratch because developers do not trust the quality of the components or the sources.

The second problem faced by Web practitioners (and perhaps the software industry in general) is the lack of guidance and support for project management issues and quality framework for Web practitioners.

2.6 Discussion

We have used 8 companies within this survey, and this can be seen as both a strength and a weakness. The strong points are:

- It is difficult to get companies to participate in a study of this kind, and in this sense, eight is a good number.
- We had the opportunity of talking with the participants at length, which we did through our semi-structured questionnaire.

However, there are some weaknesses of the study in the following sense:

- Our selection of eight companies is not a true representative of the whole spectrum of Web practitioners.
- A shortcoming of our questionnaire is that we mostly asked about the Web process but did not ask much about the success/failure of their products or whether they had any problems with the quality of their work.
- A flaw in the design of the survey was that the participants were not asked directly what problems they faced or what they felt they needed to help them with their tasks.

In the light of the above, further studies of this kind need to be done to validate our observations. May be our experience will enable a future interviewer to focus his/her questions in a better manner.

It is our feeling that new and effective methods could be used in the course of Web development which should not put more burdens on the practitioners. For instance, a Web server can automatically collect some metrics as a part of Web server logs, and such metrics can be used to improve the Web applications. Naturally, further research is required to investigate the effectiveness of such metrics.

Web applications are often expected to satisfy some special quality characteristics like shorter time to market, security, satisfying a wide variety of users etc. (Barry and Lang, 2001). If such quality aspects are of paramount importance, then a measurement framework must be in place to facilitate metric based assessment or improvement. However, it may put additional burden on the practitioners. Therefore, the extent of the measurement framework should be defined in relation to the quality needs. Adequate tool support could be used to minimise the burden on the practitioners.

2.7 Conclusion

In this chapter, we have described how we selected Web practitioners from eight organisations of varying dimension and performed semi-structured interviews. The main objective was to understand their current practices as regards to Web development and to identify deficiencies in them. We also discussed the lessons learned from our semi-structured interview identifying a few important deficiencies with current practices. We will address some of these issues in subsequent chapters.

Chapter 3

A Parallel Framework for Web Development

3.1 Introduction

Web applications deliver a complex array of contents and functionalities to a broad population of end users. Web applications usually incorporate hypermedia capabilities into other traditional systems such as information systems, real-time systems etc. They usually cover a broad range of application areas of varying dimensions – from static site presence for marketing purposes to Web Information Systems (WIS) that allow tasks like commerce, banking, groupware systems etc. This broad range also reflects how the Web has matured since its inception, from static Web sites to complex applications involving databases, CGIs (Common Gateway Interface), agents, distributed architectures with loosely coupled databases, object servers etc. Since Web applications do not totally abandon the traditional development process, the same software engineering techniques are still needed but the process should take into account their differences as stated in Chapter 1 and Chapter 2. Furthermore, Web development should rely on engineering approaches based on well-defined standards employing systematic, disciplined and quantifiable approaches to development, operation and maintenance (Lowe and Hall, 1999). Therefore, they require new approaches to design and development to handle the hypermedia aspects. At the same time, they continue to face the same issues and challenges of traditional systems.

This chapter presents a development framework which is a meta-model for the development of Web application. As discussed in Chapter 2, our interviews revealed that there was a lack of methods, activities and artefacts to reflect hypermedia characteristics (structure, content and navigation), and therefore, Web applications created did not scale up or were not maintainable. In addition, the conception of many Web applications can be seen, in many cases, as a process of integrating hypermedia functionality and some elements of a business domain. Therefore, most projects require some underlying infrastructure to create Web based systems, and so it is preferable to differentiate infrastructure development from hypermedia functionality. Keeping these results in mind, we propose a framework that classifies the overall Web development process into two almost independent sub-processes or activities: the authoring process (AUTH process) and the process of developing the infrastructure (INF process). By an authoring process we mean the process of providing hypermedia contents (hypermedia pages with appropriate links between them). The infrastructure development process provides technological support like interfacing the Web pages with databases, security protocols, applets, agents, enterprise servers etc. This separation of concerns defines a parallel path where the AUTH and INF processes can be developed simultaneously and independently with minimal interaction between them. Any such interaction will be dealt at an abstract level (Rodriguez *et al*, 2001a).

The Parallel Framework identifies the process steps and the artefacts involved in Web projects. It will further help us to define quality models for both products and processes in relation to Web applications. As a result, product improvements as well as process improvement tasks will be more systematic and focussed. We will illustrate our approach through a running example.

This framework has benefits from both the process point of view and the application point of view. From the process point of view, it modularises the development. Such modularisation should lead to better maintenance of the Web process. Furthermore, because of the formalised structure of the process, quality aspects of the individual sub-processes can be addressed effectively (Rodriguez *et al*, 2002). In addition, modularisation can lead to development of individual modules by third-party providers.

From the application point of view, the modular approach leads to a clear separation of concerns. Our modularisation approach can be seen as a method to integrate hypermedia functionality into conventional applications, which leads to enhanced maintainability and portability of Web applications. For instance, there are applications such as billing systems, in which a company want may want to provide facilities like on-line payment, invoice checking etc. Initially, such applications could have been developed without taking the Web into account. Therefore, the hypermedia 'plug-in' capabilities need to be incorporated into the existing application. Furthermore, the division of the development process with a flexible interface layer between provides the ideal basis for customisation.

We also aim at following an object-oriented (OO) approach to cover the whole of the hypermedia development process and the specification of the process itself. The UML is the standard language to model OO applications (Booch *et al*, 1999); so, it would be natural to use UML for specifying Web applications. In this chapter, we will show how UML, with its extension mechanism, could be used to model various activities of the parallel framework. Such extension mechanisms are: stereotypes, tagged values and constrains using the Object Constrain Language (Warmer and Kleppe, 1999).

We view our Parallel Framework as a major contribution to the field of Web engineering; we will elaborate more about this in the concluding section of this chapter.

3.2 Requirements for the Parallel Framework

From the results of the interviews reported in Chapter 2 and the research literature, we can highlight the most important requirements that our framework should provide:

- The framework should be model driven in the sense that it can be based upon models and techniques already used in the hypermedia, information systems, and software engineering fields. The modelling language of the method should be the

UML in accordance with industry standards so that: (i) it can be used with current CASE and modelling tools and (ii) UML profiles can be defined extending the core modelling elements to the hypermedia domain, i.e., high-level specification of the hypermedia domain.

- The framework should be aligned with object oriented life-cycle models such as the Open Process (Henderson-Sellers and Unhelkar, 2000) or the Unified Software Development Process (Jacobson *et al*, 1999) so that: (i) it enables project-specific tailoring using their customisation properties. Web architectures will influence micro-processes (Booch, 1994) defining new activities, artefacts, and roles for those generic processes; and (ii) it covers all the software development life cycle, i.e. requirements capture, analysis, design, implementation, testing and maintenance.
- The framework should allow the reuse of hypermedia or domain components.
- The framework should be able to integrate the different ways of modelling the domain (object oriented, data centric using the entity relations model, XML etc.) as well as being able to reuse (within and externally) hypermedia design and components.
- Well defined artefacts so that evaluation criteria can be devised.
- The framework should permit the separation of application and hypermedia domains in the way that industry addresses front-end and back-end developments.
- Since maintenance is one of the most difficult tasks when dealing with Web application (Brereton *et al*, 1998), the framework should facilitate this aspect through modular design.

3.3 Related Work

3.3.1 Hypermedia Models

The *Dexter Reference Model* (Halasz and Schwartz, 1990, 1994) captures important abstractions found in a wide range of hypertext systems. It was conceived after examining important hypermedia applications such as Augment, Concordia, Hypercard, Notecards etc.; so it can provide a standard terminology and standards for interoperability among hypermedia systems. The Dexter model divides a hypertext system into 3 layers that allow modifications and extensions of one layer without affecting the others. Those layers are: (i) the *Runtime Layer* for the presentation of the hypertext, (ii) *Storage Layer*, a database containing nodes and links; and (iii) *Within Component Layer* which handles the internal content/structure of hypertext nodes.

The *Hypermedia Design Model* (HDM) (Garzotto *et al*, 1994; Garzotto *et al*, 1993) provides a model for high-level hypertext design called *authoring-in-the-large*. HDM is based on the ER model, but extends the concept of entity and introduces new primitives as nodes and links. The authors distinguish the following dimensions in the analysis of hypermedia applications: content, structure, presentation, dynamics and interaction.

The hypermedia and Web methodologies which are in use today are mostly based on the concepts defined by these two models. These models were quite complete and only a few extensions have been proposed to deal with some of the limitations of these models. For example, the Amsterdam Hypermedia Model (AHM) of Hardman *et al.* (1994) is an extension of the Dexter model which adds timing and context aspects to multimedia applications.

3.3.2 Hypermedia Methodologies

The two most well-known hypermedia methodologies available today are the Relationship Management Methodology (Isakowitz *et al*, 1995), and Object Oriented Hypermedia Design Method (Schwabe and Rossi, 1995; Schwabe *et al*, 2001; Schwabe and Rossi, 1998).

The Object Oriented Hypermedia Design Method (OOHDM) divides the development process into four steps: domain design, navigational design, abstract interface design and implementation.

- *Domain Design:* a model domain is described using a object-oriented class diagrams.
- *Navigational Design*: navigational structures are described through predefined types of navigational classes and access structures. Then, navigation objects are grouped into sets called contexts to specify their access structure.
- *The Abstract Interface Design* specifies how users perceive navigational objects through the interface, the way different navigational objects will look and interface transformations.
- *Implementation* of the navigational design and the abstract interface design models into concrete objects available in the chosen implementation environment.

These steps, where the order is only indicative rather than prescriptive, can be used in a mix of iterative, incremental and prototype-based styles of development following an object-oriented approach.

The Relationship Management Methodology (RMM) considers hypermedia applications in three layers: storage, logical and presentation level: storage level describes how information is stored; logical level describes the structure of information that the application manipulates; and presentation level deals with what information is presented to users and how to navigate it. The RMM provides a language based on the Entity-Relationship mechanisms, called RMDM (Relationship Management Data Model) for describing information objects and the navigation mechanisms.

A problem with these hypermedia methods is that they do not clearly identify the separation of authoring domain, the conceptual domain (underlying data and business logic) and their mapping mechanism. They impose their own embedded mapping mechanism (Lopistéguy *et al*, 1997) and use their own notation, where in practice, different approaches could be considered (e.g., object-oriented and entity-relationship approaches). Both OOHDM and RMM use proprietary notations that are not supported by current modelling tools, e.g. Rational Rose (Rational, 2000). There have been attempts to extend UML to specify hypermedia applications to gain the benefits of following a standard (Conallen, 2000; Baumeister *et al*, 1999).

Baumeister *et al* (1999) proposed a UML profile to model hypermedia and Web applications. The authors model the navigation and the user interfaces of hypermedia systems using the UML and the ideas of the OOHDM method. Their process comprises a conceptual, navigational and presentational model. The conceptual model consists of a class diagram of the problem domain. The navigational model describes the navigation structure using a class diagram specifying navigational nodes and an object diagram showing how navigational nodes are visited. Finally, the presentational model describes the abstract user interface.

Conallen (1999) defines architectures and UML stereotypes for modelling Web applications, but this does not constitute a method for hypermedia design as it lacks of navigational models etc. It includes stereotypes for components, classes, methods and associations, such as server and client components, server and client pages, forms, links etc. An extension exists for Rational Rose called WAE (Web Application Extension) that handles these stereotypes (Rational, 2000).

3.3.3 Hypermedia Processes

Lowe and Hall (1999) provide a framework for the hypermedia development process, which includes domain analysis, product modelling, process modelling, project planning, development and documentation. *Product modelling* consists of choosing a model for the final product. The framework supports three different product models: programming language-based, model-based and information-centred approach. *Process modelling* consists of selecting phases, activities and artefacts for the development and establishes how these are integrated into the specific development process. Activities performed during the development process are system analysis, design, production, verification and testing as well as maintenance. Some guidelines are suggested such as incremental development and prototyping.

The Hypermedia Flexible Process Modelling (HFPM) presented by Olsina (1998) is another engineering-based approach, which includes analysis-oriented descriptive and prescriptive process modelling strategies. It describes existing processes giving guidelines for planning and managing of hypermedia projects.

3.4 A Parallel Approach towards Web Development

The Web development process can be broadly classified into two almost independent sub-processes or activities, the authoring process (AUTH) and the process of developing the infrastructure (INF). The authoring process adds the hypermedia metaphor to the information. It includes the presentation and linking structure of information and associated components. It also includes the process of developing the content such as audio, video etc. The INF process provides the technological support and includes activities such as interfacing with databases, security protocols and other components not belonging to the Web application. In a Web application, it is necessary to decide how much functionality is performed on each side of the partition.

Each of the AUTH and the INF processes can be modelled using phases such as requirements capture, domain modelling, interaction modelling, detailed design, implementation and testing. Figure 3.1 summarises the various process phases. The idea behind the parallel development process can be summarised as follows. The requirements capture phase collects all the Web-related user requirements, from which the requirements for the AUTH process and the requirements for the INF process could be derived. We will call them $\mathrm{Req}_{\mathrm{auth}}$ and $\mathrm{Req}_{\mathrm{inf}}$ respectively. After this separation, both the processes can proceed in parallel and almost independently in order to generate two independent implementation modules which are also tested independently. After that both modules are integrated through an interface module and the integration is then tested to obtain the final product.

Figure 3.1 also reflects the usual way for developing applications in industry, in which an application is divided into a front-end and a back-end. While the front-end relates the application to the client side (i.e., the part of the application that the user interacts with), the back-end is related to the core of the application (linkage with legacy systems, databases etc). This is very beneficial in separating the presentation from functionality, where developers of the front-end only have the high-level view of the application. They need only to know about the calls to the high-level functions (for example, ASP pages that call functions in the backend processing). On the other hand, back-end developers work at the object level. This is in contrast to traditional systems development. Thus, our partition of the Web development process reflects the distinction between the front-end and back-end processes. Although we present the parallel framework as a linear sequence of activities, in practice, each path may undergo through several iterations.

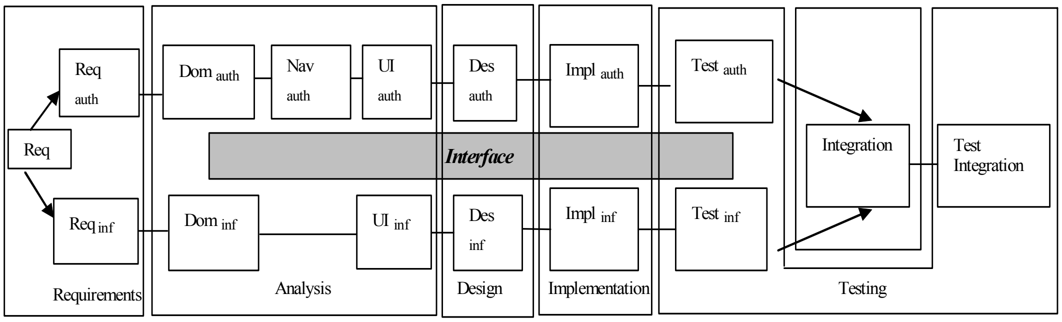


Figure 3.1 Web Development Workflow

We will now discuss the individual phases of the AUTH process and the INF process. Table 3.1 briefly outlines each of the activities of the AUTH process.

Name	***Activity***
Req $_{auth}$	Req $_{auth}$ obtained from the overall Web requirement.
Dom $_{auth}$	From Req $_{auth}$, identify classes, relationships between them and obvious attributes. Produce a preliminary class diagram for authoring.
Nav $_{auth}$	From the authoring class diagram, obtain a navigational model (Baumeister *et al*, 1999). This model describes navigational structure between the Web pages.
UI $_{auth}$	Describe how the navigational structure is presented to the user.
Des $_{auth}$ (Detailed Design)	Complete the preliminary class diagram.
Impl $_{auth}$	Implement contents (pages, sounds, pictures, etc) and links.
Test $_{auth}$	Syntactic testing, image-related problems, document structure etc. (functionality, performance, compatibility, reliability, security etc. need be verified after integration)

Table 3.1 Stages in AUTH Process

Table 3.2 outlines the stages of the INF process. The parallel path for the INF process is similar to the traditional development process.

Name	*Activity*
Req $_{inf}$	Obtain Req $_{inf}$ from the overall Web requirement.
Dom $_{inf}$	Form domain model (preliminary class diagram) to establish the context of the system.
UI $_{inf}$	Produce user interface model for INF (if required).
Des $_{inf}$ (Detailed Design)	Create the design model (interaction diagram, state diagram, complete class diagrams)
Impl $_{inf}$	Write code modules from the Des $_{inf}$.
Test $_{inf}$	Produce test cases and plan for INF components. Remaining testing is to be done after integration.

Table 3.2 Stages in the INF Process

The parallel paths of both the processes (AUTH and INF) join at the integration phase (see Figure 3.1); i.e. the implementation of the AUTH part and the INF part are integrated during this phase through the implementation of the interface. In the Figure 3.1, the module with the label 'interface' implies the logical description of the interface between the AUTH module and the INF module. The integrated product is tested and one round of the process life cycle terminates. The phases may be iterated several times until the Web application stabilises.

Table 3.3 describes the phases of Web development in a formal manner. In the table, '$\cup$' means the independent specification/design/development etc of its constituents; for instance, the expression $Des_{\ Web} = Des_{\ auth} \cup Des_{\ inf}$ means that the design of the Web application consists of the independent design of the authoring and infrastructure path. .Integration (IMPL $_{auth}$, IMPL $_{inf}$) means the integration of the modules of the AUTH part and those of the INF part through the implementation of an interface. Testing $_{Web}$ involves testing of the Impl $_{auth}$, Impl $_{inf}$ and that of their integration. Similarly, maintenance of the application involves maintaining the AUTH, INF part, and the maintenance of the integrated artefacts.

Web Development Process
$\text{Req}_{\text{Web}} = \text{Req}_{\text{auth}} \cup \text{Req}_{\text{inf}}$
$\text{Dom}_{\text{Web}} = \text{Dom}_{\text{auth}} \cup \text{Dom}_{\text{inf}}$
$\text{Nav}_{\text{Web}} = \text{Nav}_{\text{auth}}$
$\text{UI}_{\text{Web}} = \text{UI}_{\text{auth}} \cup \text{UI}_{\text{inf}}$
$\text{Des}_{\text{Web}} = \text{Des}_{\text{auth}} \cup \text{Des}_{\text{inf}}$
$\text{Impl}_{\text{Web}} = \text{Integration}\ (\text{Impl}_{\text{auth}}, \text{Impl}_{\text{inf}})$
$\text{Test}_{\text{Web}} = \text{Test}\ (\text{Impl}_{\text{auth}}) \cup \text{Test}\ (\text{Impl}_{\text{inf}}) \cup \text{Test}\ (\text{Integration}(\text{Impl}_{\text{auth}}, \text{Impl}_{\text{inf}}))$
$\text{Main}_{\text{Web}} = \text{Main}_{\text{auth}} \cup \text{Main}_{\text{inf}} \cup \text{Main}_{\text{auth, inf}}$

Table 3.3 Parallel Web Development Process

Each process or process phase generates or modifies certain artefacts. As mentioned above, not all the process steps are needed for all projects but every project should define what process phases are needed and the corresponding artefacts. Every project should define for each process phase:

(i) pre-conditions or entrance criteria

(ii) objectives and participants

(iii) artefacts created and modified and who is responsible of them

(iv) tools, standards, guidelines, rules etc. to be used, and how they would be used

(v) post-conditions or exit criteria for the process phase. Such post-conditions need to include quality gates. This will prevent that a process steps ends without the necessary exit conditions satisfied. Such quality gates allow us to have a fine control over the quality of the output artefacts.

It is not easy to specify every sub-process or artefact because some artefacts occur in more than one process phase, and all such phases may modify or refine the same artefact. For example, the same class diagram for the INF process may be modified by domain modelling, detailed design etc.

We also need to define a *generic process model* i.e., an abstract representation of a software process, which can be used in many similar projects and organisations, sharing common properties and characteristics (Dermiame *et al*, 1998). Classical life-cycle formalisations are examples of generic process models. Later, every organisation and project will carry out a *process model customisation* refining a generic process model to tailor specific needs. There are several levels of customisation. A *customised process model* is more detailed and is usually derived from a generic model taking into account specific local characteristics, e.g. the application domain. An *enactable process model* defines the process model to be followed by an organisation and an *enacting process model* is the model in place, supporting the actual project (Dermiame *et al*, 1998).

For the definition of the generic process model and its customisation, we have followed the approach by Satpathy *et al* (2002; 2000). Table 3.4 gives an example of further customisation from the generic model, where different criteria of each process step are defined. Further customisation will be necessary to create the enacting process model.

Req $_{auth}$	Preconditions: Global requirements are collected Post-conditions: UC model relevant for authoring created Objectives: Identify UC for authoring from the overall requirements Artefacts: UC Model for authoring Tools: UML toolkit Standards: UML Roles: Requirements engineer Quality gate: Req for authoring is analysed for consistency, completeness, generality etc.
Dom $_{auth}$	Preconditions: UC model for authoring completed Post-conditions: A class model for authoring created Objectives: Identify classes and relationships relevant for authoring Artefacts: Authoring Domain Model Tools: UML toolkit Standards: UML Roles: Author Quality gate: Class model is analysed against the requirements in Req $_{auth}$
Nav $_{auth}$	Preconditions: Authoring Domain Model created Post-conditions: Use model relevant for authoring has been created Objectives: Obtain a navigational model Artefacts: Navigational model Tools: UML toolkit Standards: UML Roles: Author Quality gate: Navigational model is analysed for consistency and completeness
UI $_{auth}$	Preconditions: Navigational Models have been created Post-conditions: User Interface models relevant for authoring has been created Objectives: Describe how the navigational structure is presented to the user Artefacts: UI presentation diagrams. Mock-ups etc. Tools: UML toolkit Standards: UML Roles: Author Quality gate: presentation diagrams, mock-ups, storyboards are analysed against UI requirements
Impl $_{auth}$	Preconditions: UI Models have been created. Post-conditions: Use model relevant for authoring has been created Objectives: Implement contents (pages, sounds, pictures, etc) and links Artefacts: HTML pages, XHTML, XML and XSL documents, JavaScripts etc. Tools: HTML editors, IDE (Integrated Development Environments) etc. Standards: HTML, XML and related standards Roles: Author, designer, media developers Quality gate: Web interface and pages are checked against the requirements.
Test $_{auth}$	Preconditions: The implementation of the authoring is concluded Post-conditions: The authoring components are tested and tests report created Objectives: Perform usability, integrity and syntactic testing Artefacts: Checklists, test reports Tools: HTML Analysers, Client-side scripts analysers etc. Standards: N/A Roles: Testers, authors Quality gate: Web pages and the Navigational model are checked against broken links, edge coverage etc.

Table 3.4 Process Customisation

3.5 Description of the Individual Process Phases

We will discuss each of the process step of the parallel framework though a running example. Our example will be a Web application called SEISN (Software Engineering and Information Systems Network).

3.5.1 The SEISN Website

The SEISN is a forum for academics and industrial practitioners in the area of software engineering. The network provides a forum for colleagues in both the computer science and management disciplines to discover links between academia and real-life problems. The Web site *http://www.seisn.rdg.ac.uk/* is a platform for communication between the SEISN members and any visitor who wants to know about the SEISN activities.

3.5.2 Requirements Capture and Preliminary Analysis

In the proposed Parallel Framework, the starting point of the Web development process is a set of user requirements (Req). In UML, these are expressed in terms of high-level use cases and scenarios (Cockburn, 2000), i.e., the use cases at the *Req* stage are usually at a higher level; such use cases and scenarios are to be separated into two groups called Req_{auth}, the requirements of the authoring path, and Req_{inf}, the requirements of the infrastructure path.

For the SEISN example, Table 3.5 describes some high-level requirements before our splitting approach. We also identify who the actors are in Figure 3.2. An actor's model is a necessary before obtaining the use cases from the requirements.

ID	*Req. description*
#1	Visitors should be able to download meeting minutes in HTML or MS Word format.
#2	Visitors should be able to book on-line in order to attend the next workshop
#3	Users should be able to access a mailing list with information of news and related events.
#4	Visitor should be able to download presentations in several formats (pdf, word)
#5	The Web site should have a search facility
#6	A map site should be available

Table 3.5 Requirements of SEISN Example

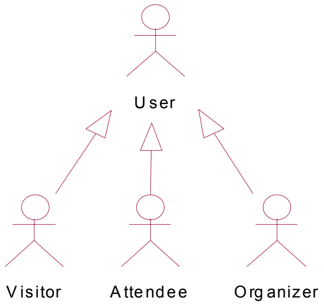


Figure 3.2 Actor's Diagram

Our claim is that any such high-level use case can be split into relatively low-level use cases such that the low-level descriptions can be put either into the Req_{auth} or into Req_{inf}. We illustrate this through an example. Consider the use case (UC): *book for a workshop*. This high-level use case involves (i) the user fills in a registration form and then (ii) commits the registration. A message is sent to acknowledge the registration and a receipt may be generated in the process. Now we are in a position to divide this use case diagram into low-level UCs and scenarios such that some of them fall into the category of Req_{auth} and the others into the category of Req_{inf}. For example, the high-level use case *book for a workshop* can be broken down into the following lower level of use cases:

(i) display the registration form and let the user complete it and submit it

(ii) the system validates the user inputs

(iii) the system generates positive or negative responses

(iv) the system updates necessary databases and generates a receipt

It can be seen that the use cases (i) and (iii) belong to $\mathrm{Req}_{\mathrm{auth}}$; whereas (ii) and (iv) belong to the $\mathrm{Req}_{\mathrm{inf}}$. Figure 3.3 (A) represents the high-level UC for our current example. Its separation into low-level UCs and scenarios are shown in Figure 3.3 (B).

Our claim is that any high level UC can be broken down such that their division into AUTH part and INF part is possible. This is so because the simplification process can go on, possibly repeatedly, until the simpler form of use cases/scenarios fall into one of the categories.

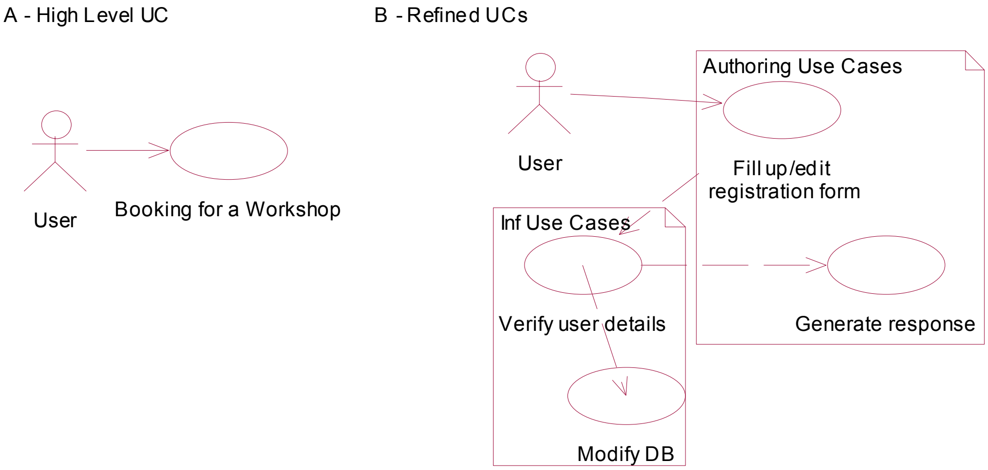


Figure 3.3 Example of the Decomposition of a High Level UC

As a further example, the UC *Register for the next workshop* can generate the following scenario in Table 3.6, which can be divided into INF and AUTH as shown in Figure 3.4.

USE CASE Scenario	***UC – Register for the next Workshop***	
DESCRIPTION	*Step*	*Action*
	1	The visitor opens registration page
	2	The visitor submits the booking form to the organiser.
	3	Organiser receives the booking.
	4	Organiser evaluates the booking
VARIATIONS		Branching Action
	5a	Organiser confirms the booking.
	5b	Organiser refuses the booking

Table 3.6 Example of the Use case/Scenario Register for the next Workshop

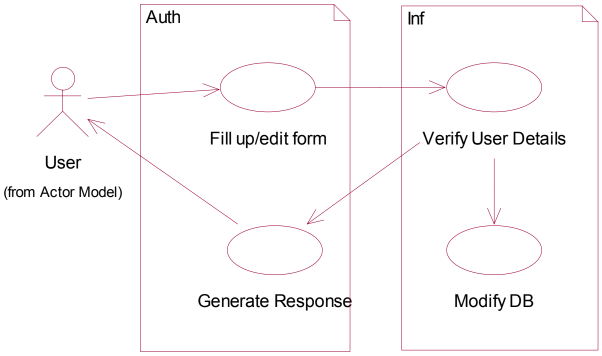


Figure 3.4 UC Register Workshop

Even at the user requirement level, some requirements can be clearly identified as belonging to the authoring domain, and some other belonging to the infrastructure domain. For example, the requirements related to languages (e.g., HTML, XML), the type and number of different media used, the way the users can interact with them, support for people with disabilities etc. fall into the authoring domain. Similarly, functional requirements like financial transactions, data warehousing, CORBA application server, and non-functional requirements like security, performance, scalability, caching etc. mostly fall into the infrastructure domain. If some requirement is not immediately evident as to which category it belongs to, we break this requirement, possibly repeatedly, into sub-requirements until each sub-requirement can be put into one of the above categories. To do this, we follow the use-case splitting approach as discussed in the previous section.

3.5.3 Domain Modelling

At this level, we design the domain architecture in terms of packages. Such an architecture is obtained from the use cases (which have already been separated into the AUTH part and the INF part) and also from domain knowledge. The packages of the authoring domain and those of the INF domain are linked by a special package called the AUTH-INF interface. Figure 3.5 shows the overall architecture. The separation between the authoring domain and the INF domain should be evident at this point.

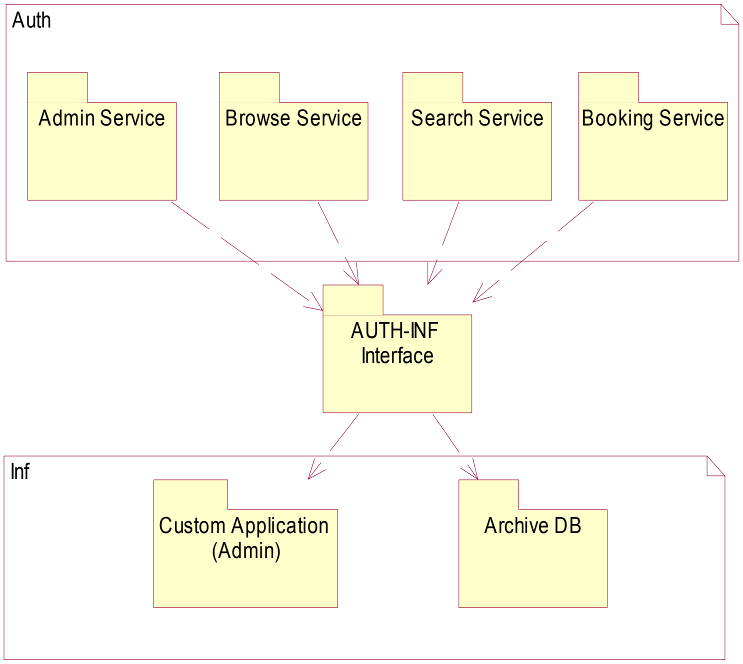


Figure 3.5 Overall Architecture of the Web Application

3.5.3.1 The Authoring Domain

The aim here is to identify entities for modelling the authoring requirements and relationships between them. The authoring domain model must also provide information for user navigation. For our authoring domain modelling, we follow the OOHDM. In order to accommodate the authoring domain modelling in the UML, we follow the approach by Baumeister *et al* (1999) who have extended the UML framework.

Figure 3.6 represents the entities and relationships corresponding to the authoring domain of the SEISN example. The diagram is self-explanatory; for instance, an archive is composed of workshops and Technical papers. The workshops will consist of presentations, and presentations are presented by authors and so on. All the entities from the conceptual domain, which will be used to obtain the corresponding hypermedia entities, are in turn used for navigation. Note that this model is a very simplified class diagram which tries to incorporate the entities which are needed for authoring the Web pages.

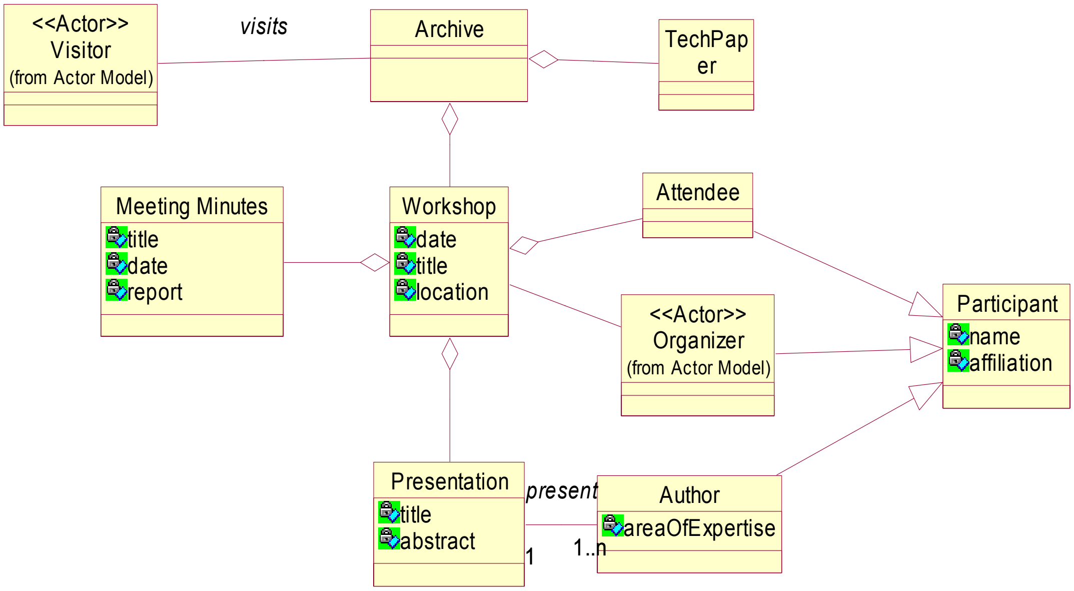


Figure 3.6 Authoring Domain Model

3.5.3.2 The Infrastructure Domain

This domain tends itself to more traditional design methods and conventional approaches towards database modelling etc. Figure 3.7 shows the class diagram for the infrastructure domain. In the diagram, the DB server class has (through aggregation) the database. For designing the database, an entity relationship or ORM (Object Role Modelling) diagram needs to be constructed (Halpin, 2001). Figure 3.8 shows such an ORM diagram.

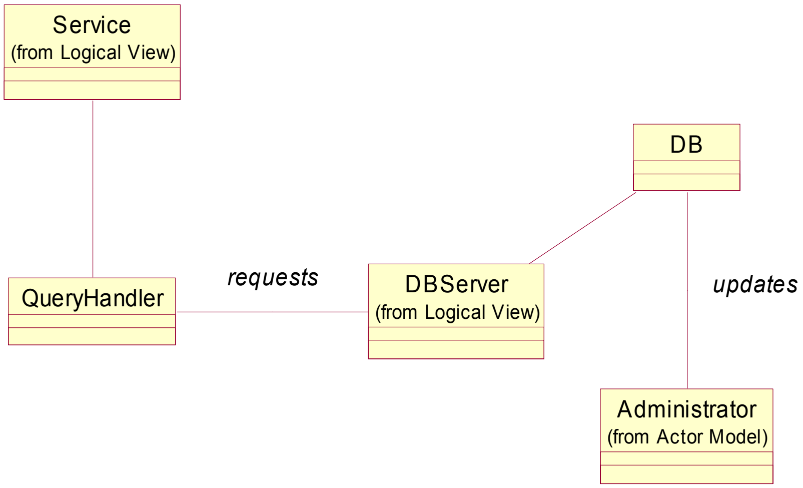


Figure 3.7 Class Diagram for the Infrastructure Domain

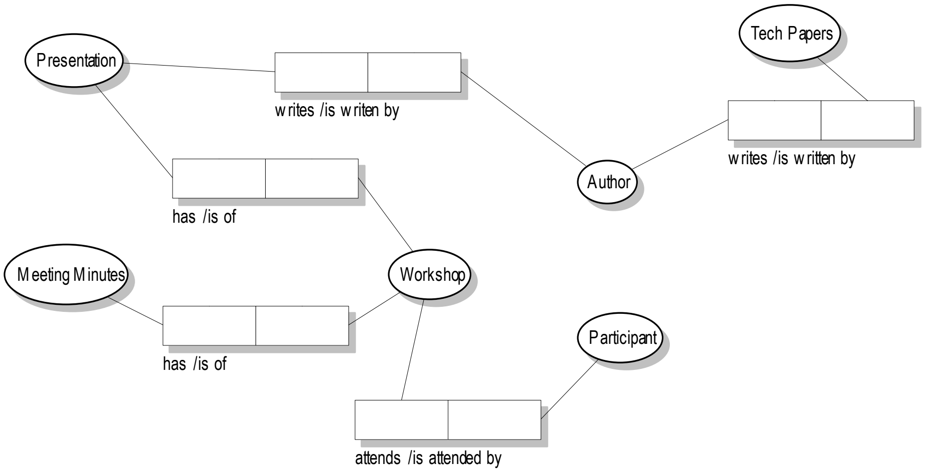


Figure 3.8 ORM Diagram in the Infrastructure

3.5.4 Navigational Modelling

The purpose is to design how hypermedia objects are visited in a way that matches the users' needs. We use the approach by Baumeister *et al* (1999) for navigational design. Starting with the Authoring Domain Model, the Navigational Class Model is obtained by removing classes not relevant to authoring. Relationships between classes are renamed in the process. The navigational class model that we obtain from the authoring domain model in Figure 3.6 is shown in Figure 3.9 where relationships are stereotyped as *«link»* and classes as *«navigation»*.

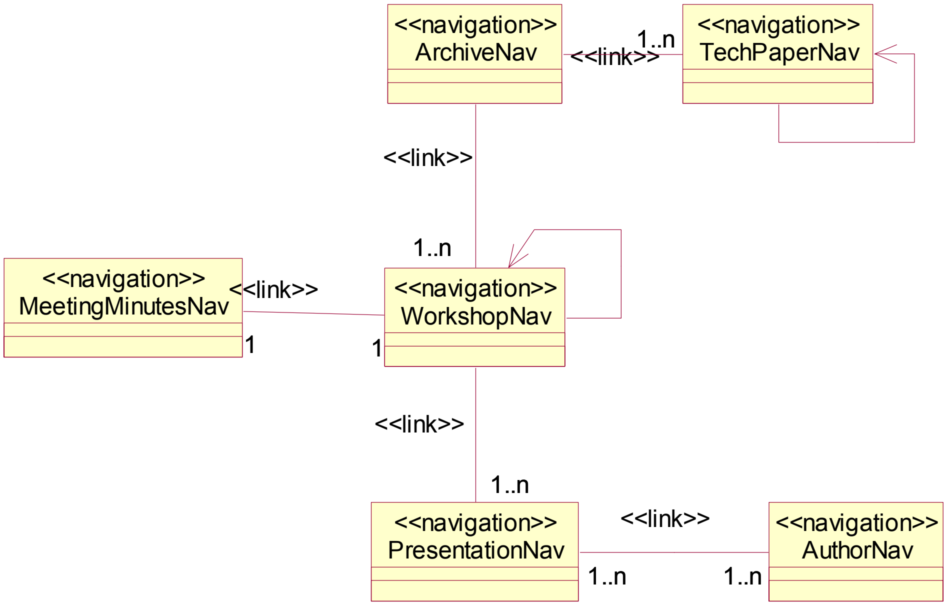


Figure 3.9 Navigational Class Model

From the Navigational Class Model, the Navigational Structure Model is constructed through the addition of navigational classes like indexes, guided tours, index—guided tours. Navigational contexts are also added to incorporate the information.

Figure 3.10 shows the navigational structure model, which represents the navigation from the main menu. This diagram is obtained from the authoring domain to represent the navigational aspects of the application. Note that Figure 3.10 represents an outline of the complete navigational diagram for the SEISN case study where the UML package notation represent contexts, i.e., packages are stereotyped as *«navigational context»*.

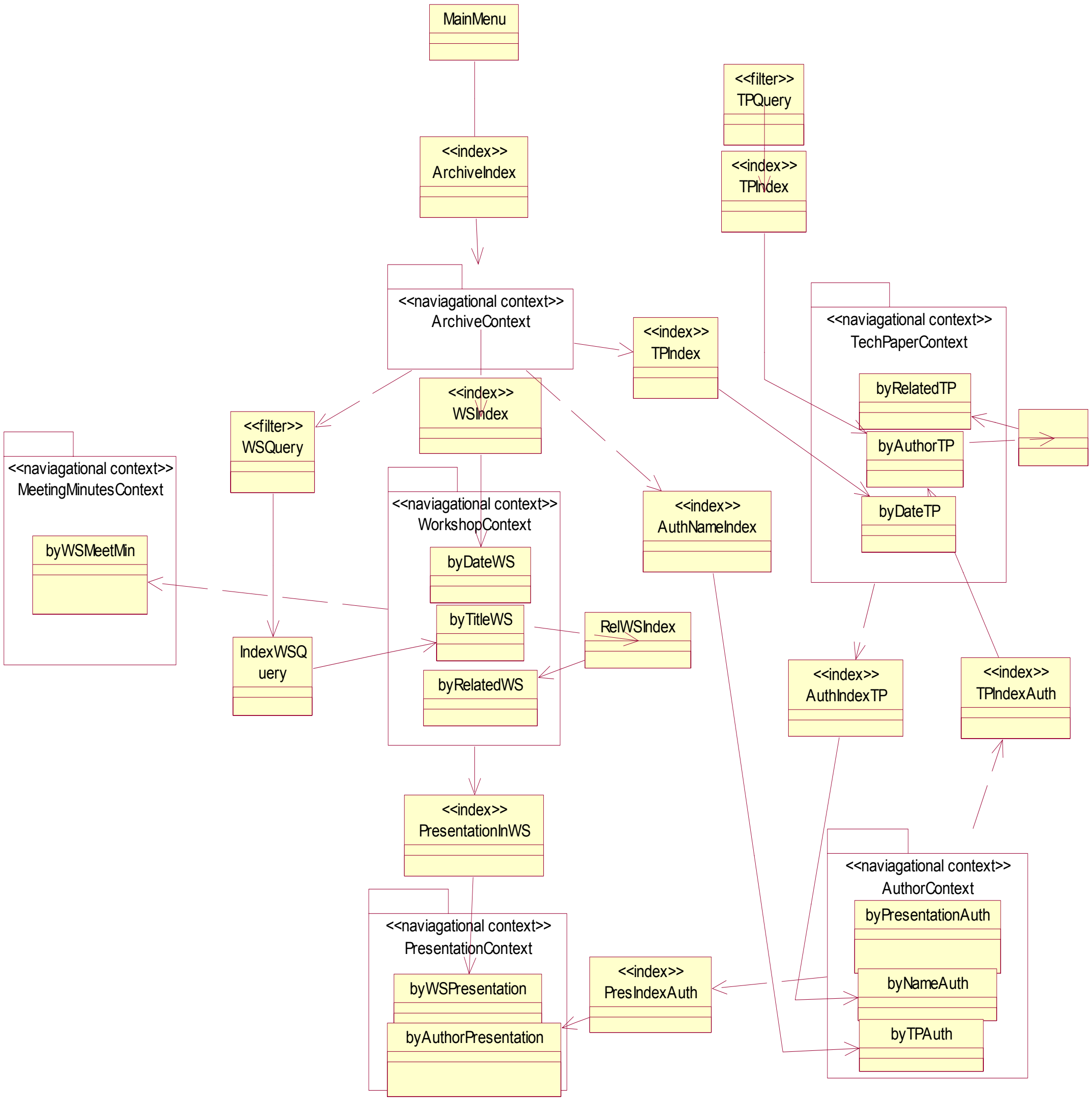


Figure 3.10 Navigational Structure Model

3.5.5 Interface Design

Hypermedia elements such as nodes, links, menus, indexes, guided tours etc. are needed to design the user interaction with the application. This step shows how interface objects (e.g., windows or screens, pictures, sounds etc.) compose the user interface and their behaviour. Such objects map to navigational objects (refer to Figure 3.10). Note that many interface designs are possible with respect to a single navigational model.

3.5.5.1 UI Authoring

UI authoring represents which GUI objects compose the actual user interface and how they interact with each other and with the system. UI authoring is created from the navigational models and navigational contexts. We modify the approach taken by Conallen (2000) for the design of the authoring interface from the navigational structure model. We illustrate our approach through two examples.

Figure 3.11 considers the navigation in which we intend to find all related workshops in relation to a given workshop. Assume we are at a search page from where we fill up a form-page. Then a query is generated from the authoring domain. However, the answer to the query lies in the INF domain. So a query is submitted to the AUTH-INF interface. We assume that the interface will provide us with the answer. Such details we will discuss later. Once the answer is obtained, the JSPServer builds up the page with appropriate format for display. Figure 3.11 shows the whole scenario where *SearchAllRelatedWS* and *WSList* represent Web pages, *FormPage* a Web form and *JSP server* a dynamic page.

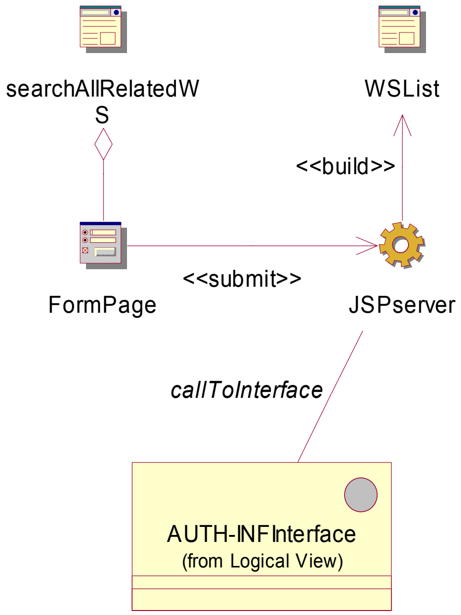


Figure 3.11 Search for Related WS

Figure 3.12 shows the UI authoring design for the navigation 'display all technical papers in the archive by date'.

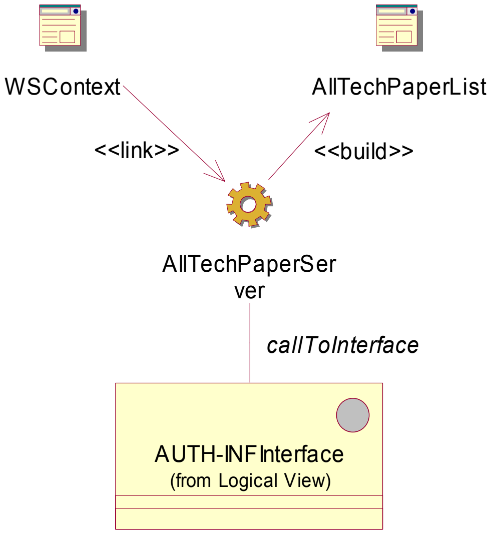


Figure 3.12 Display All Technical Papers

3.5.5.2 UI Infrastructure

Although the UI is always part of the authoring process, there may exist other type of components like applets or ActiveX which can provide GUIs (Graphical User Interface). The Web browser executes such components automatically when a Web page is loaded providing an interface through which a user may interact. Although this approach provides many benefits such as easier deployment, it does not have the generality of hypermedia capability. Their design is not of much interest within the context of the present discussion since such components do not use the Web technologies.

3.5.6 Detailed Design

3.5.6.1 Detailed Design Authoring (AUTH)

By now, we have most of the models required – navigational structure diagram, UI authoring diagrams etc. – which will be used to design the layout of static and dynamic HTML or XML pages. All the queries which will be made to the AUTH-INF interface are obtained, and also the format of the queries.

To implement the authoring, technologies include not only HTML and XML (with their editors) but also numerous technologies for creating dynamic pages; for example, JavaScript or VBScript for creating dynamic pages on the client-side and JSP (JavaServer Pages), ASP (Active Server Pages), PHP etc. for the server-side. In our methodology, these pages must not refer to any low level calls like establishing a connection with a database etc. Such calls must be at done a high level where technical details are hidden. The AUTH-INF interface should take care of the translation of the high level calls to low level calls.

3.5.6.2 Detailed Design Infrastructure (INF)

The INF design involves generation of sequence diagrams, completion of the class diagram, data base design using the entity-relationship diagram. However, since such designs are very much similar to the design of conventional applications, we will not elaborate them here. However, such a design approach must take into account of the AUTH-INF interface meaning that this is the only interface for utilising the services offered by the infrastructure part of the application.

3.5.6.3 AUTH-INF Interface Design

This interface can be seen as a server to the authoring part of the application. It will have operations which will be called by the authoring parts. The implementation of these operations will utilise the services offered by various components in the infrastructure part of the application. The idea is that tags from server pages (for example JavaServer Pages –JSP–) trigger methods in classes in the AUTH-INF interface, which in turn will call methods in the classes in INF part. It can be implemented through the use of Proxy and Façade design patterns (Gamma, 1995). The Proxy provides a surrogate for a remote object through which access to the remote object can be made. The Façade pattern helps in encapsulating low-levels calls and data from several classes into one single class.

The Figure 3.13 shows a sequence diagram which makes the importance of the AUTH-INF interface clear. Figure 3.13 shows the sequence diagram for the UC *'Searching for an entity in the database'*. In this step will be necessary to define the actual interface between both paths of the UC defined previously in accordance with the architecture defined.

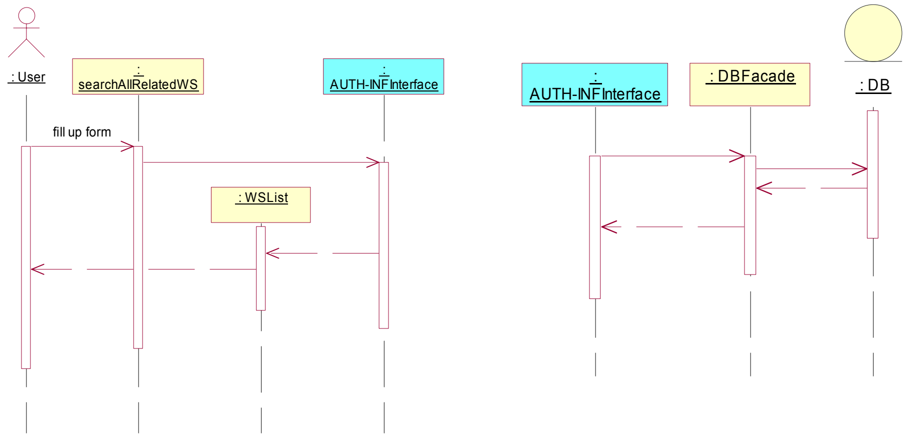


Figure 3.13 Sequence Diagram to Represent Interaction between both Paths

Figure 3.14 shows a generic architecture of a Web application, usually divided into layers. The *data layer* stores the physical data required by the application. The *data access layer* provides a logical view of the physical data and the *application layer* implements the business logic. Client applications interact with the Web application *interface*. The interface is responsible for receiving incoming messages, parsing such messages and dispatching the request to the appropriate infrastructure component. The interface is also responsible for sending the responses back to the client.

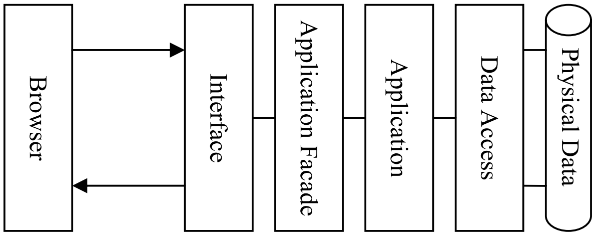


Figure 3.14 Generic Web Application Architecture

This generic architecture can easily incorporate our AUTH-INF interface approach. A request from the AUTH part can be transferred to the AUTH-INF interface by making a local procedure call, and then the interface can interact with the appropriate Web service listener.

3.5.7 Implementation

The detailed design is used for the implementation of contents, graphics, scripts, databases etc. using appropriate technologies. Our Parallel Framework follows the MVC (Model-View-Controller) pattern (Gamma, 1995), which can be implemented using Web services and other current Web technologies. Web services (Curbera *et al*, 2002) are building blocks for constructing Web applications based on Internet standards. The most important feature is that Web services allow us to create interfaces and establish communication between components using open standards such as HTTP and XML, i.e., the interface is specified in WSDL (Web Services Description Language), and SOAP (Simple Object Access Protocol) is a protocol for exchange of information between components. Current technologies for implementing Web services include Microsoft's .NET and Sun's Java 2 Enterprise Edition (J2EE) (Singh and Johnson, 2001). Such applications extend the standard two-tiered client and server model providing fully scalable solutions. As an example, J2EE allow us to implement Web applications and Web services using a range of technologies such as Servlets, JavaServer Pages (JSP) pages, custom tags, JavaBeans etc. The choice of such technologies needs to consider many factors which include in addition to the technical properties, many human factors such as organisation's experience and knowledge, partnerships, cost etc.

3.5.8 Integration and Testing

Testing of the Web application involves independent testing of $\text{Impl}_{\text{auth}}$ and Impl_{inf} and finally the testing of their integration.

$\text{Testing}_{\text{auth}}$ consists of testing related to the authoring such as:

- **Compatibility Testing:** Different browser versions and multiple operating systems associated with the Web application are tested for compatibility. Syntax checking is done over the HTML, CSS, and XML codes.
- **Navigation Testing:** Navigational paths are checked for correctness. It also involves checking of the various accessibility methods to the information.
- **Testing Client-Side Functionality:** Checklists for testing forms, client-side scripts (JavaScript and VBScript) or client-side objects such as Java applets and ActiveX controls.
- **Usability:** Various usability criteria are checked depending on the requirements. Checking that the contents (information) are syntactically and semantically correct and their granularity.

Testing $_{\text{inf}}$ mostly involves the traditional testing approaches; however, more emphasis is given to security checks especially when it is an e-commerce application.

Some important tests need to be done after integration of $\text{Impl}_{\text{auth}}$ and Impl_{inf} through the implementation of AUTH-INF interface. The following are the prominent ones:

- **Load Testing:** It involves simulating how users will use the Web application in the real world.
- **Acceptance Testing:** We need to make sure that the Web application fits the use for which it was intended. One way to do this is by setting up a beta test for the Web application.

3.6 Feature Analysis

In this section we analyse and compare the most important hypermedia models and where our methodology deviates from them or borrows their concepts. Most of the existing hypermedia and Web methodologies consider the following activities in different ways (Lopistéguy *et al*, 1997):

(i) definition of the application domain model (e.g. UML models, E/R models etc.);

(ii) definition of the hypermedia domain model (e.g., nodes, links, landmarks, guided tours etc.)

(iii) definition of the relationships between application and hypermedia domains.

In industry, they are often referred to as the front-end (hypermedia domain) and the back-end (application domain). Therefore, we will take these three activities when analysing hypermedia methods as well as our parallel framework.

The Dexter model is the most referenced model because it provides features that many hypermedia systems do not have even at present, e.g., it is capable of modelling multi-way links and composites that are not supported by current Web applications. Therefore, we should take it into account when updating current UML profiles for modelling Web applications. For example, features such as bidirectional links will be introduced by the XLink and XPath standards and new stereotypes can use the Dexter terminology.

Although HDM is defined as a design model, it implicitly defines activities to create the model, so it could be considered as a design method (Lowe and Hall, 1999). Its importance lies in being the base for the most important hypermedia methods (RMM and OOHDM). HDM model extends a domain defined in a similar way to E/R model with hypermedia features (entity, component, perspective, unit, and relationship). The HDM hypermedia model is fixed and not enriched. The mapping between the domain and the hypermedia navigation is also fixed, e.g., units relate to nodes etc.

The OOHDM method is based on HDM, but extends it by applying object oriented concepts. OOHDM divides the system in a way that can address separately domain aspects, hypermedia aspects and navigational semantics. This allows different linking topologies (according to the user profile) over the same domain and different interfaces over the same navigational schema. In fact it is usually extended with some primitives such as attribute perspectives. The hypermedia model is defined by means of concepts based on classes: node, class of node, link, link class, anchor, navigational context, navigational classes and access structures. However, these extensions are based on a proprietary notation. The mapping between the application domain and the hypermedia domain are suggested but they are not mandatory. OOHDM only focuses on design. As Baumeister *et al*, we have used their process steps to define the authoring path in our parallel framework.

Although RMM defines a series of steps in the life cycle of the application, it focuses on the design keeping open aspects as requirements etc. The domain is specified by means of the E/R model, which is enriched with hypermedia features to create the hypermedia model. Therefore, it merges the domain and hypermedia models so the correspondences are fixed and it is not possible to extend hypermedia elements with specific capabilities. Moreover, this method do not support different perspectives over the same domain. As a consequence, as stated by the authors, RMM is only suited for applications with a regular structure of the domain and where there is a frequent content modification of the database. This method has an associated modelling tool, called RM-Case (Díaz and Isakowitz, 1995).

The interface modelling of Baumeister *et al* uses the UML notation for representing physically the placement of the interface objects. For example, they use nested classes to represent where the visual object should be located, i.e. a class stereotyped with *«presentation class»* contains other classes (e.g. *«picture»*, *«link»* etc.) to represent the relative position within the node of such objects. We believe that this approach does not comply with the purpose of UML. First, in our opinion the authors are using UML to represent mock-ups of the interface windows; moreover, the relationships of those classes are not shown using their notation. Conallen approach is more convenient for this task; hence we have used this approach for the interface design.

Conallen models Web Applications without taking into account any traditional hypermedia methodologies and focuses on the implementation. Therefore, his proposal loses some of the hypermedia characteristics such as navigational modelling or support for customisation. As a consequence, these stereotypes are appropriate to model layout and implementation aspects, but not for defining the navigation structure of the Web application. An interesting Web architecture is presented in this work. Conallen treats

HTML pages (and client and server scripts) as pieces of code that are modelled using UML that can be used to replace the step in the user interface design by Baumeister.

Our framework intents to provide a way to clearly differentiate both the hypermedia and application domain in a more flexible way than previous methods. It uses many of the ideas provided by previous methods, extending and adapting them to current standards. For example, the Parallel Framework covers the whole software development life-cycle from requirements to testing in line with current generic process models. It also uses the UML to specify all hypermedia artefacts. With the exception of Conallen and Baumeister, hypermedia methods propose their own notations for all or for part of their models, so they do not facilitate the use of standard modelling tools.

Figure 3.15 summarises how and where the Parallel Framework utilises the existing OOHDM design method and the notations of Conallen (1999). We follow the OOHDM process steps after the requirements are separated into the requirements of authoring and the infrastructure. However, we have modified the OOHDM notation so as to describe the artefacts in UML. We have used the notation of Baumeister *et al* for modelling the navigational steps and that of Conallen for implementation.

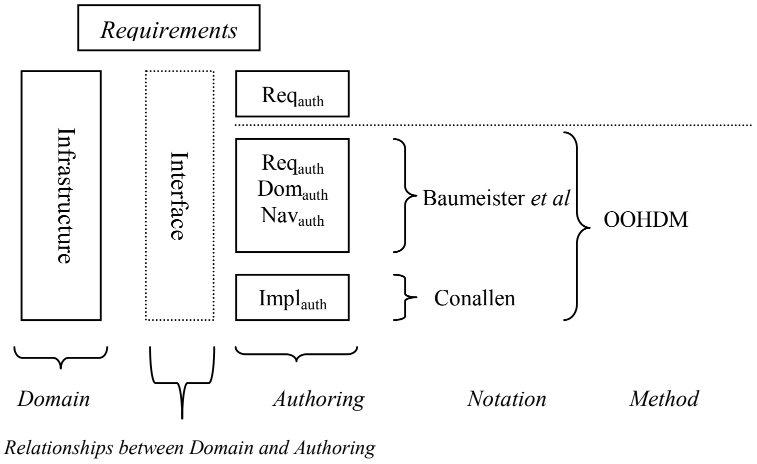


Figure 3.15 The Use of Existing Models in the Parallel Framework

3.7 Discussion

The main benefits of our methodology are:

- **Modular design.** The designs of the authoring and that of infrastructure are like two independent components of the Web application, both interacting with each other through the AUTH-INF interface. The benefits are that both components can be developed independently, possibly by different developers. It is worth to note that our methodology provides a higher level approach towards modular design of Web applications; and further modularisation can be done within authoring and infrastructure components.
- **Better Maintainability.** Any fault in a Web application can be immediately identified whether it belongs to the authoring part or to the infrastructure part. This helps in corrective maintenance (Fenton and Pfleeger, 1997; Satpathy *et al*, 2002b). If we want to add a new use case, by following our methodology, the same use case can be partitioned into AUTH and INF parts. Therefore we get a better picture of what to add and where. Furthermore, our methodology encourages a cleaner design structure, and better documentation improving perfective maintenance. Since the AUTH part does not directly interact with the INF part, it is possible to upgrade the INF component whilst the authoring component remains unchanged and vice versa. Therefore, adaptive and preventive maintenance are improved.
- **Model Driven Methodology.** Our methodology is model driven in the sense that we encode all the artefacts during our development process in UML notation. Therefore, standard commercial tools like Rational Rose (Rational, 2000) can be used to develop and maintain the artefacts of the application. In this sense the artefacts of the application can be easily understood and modified by new developers. Furthermore, documents can also be better maintained.
- **Ease of Integration/Testing.** In the integration phase, the authoring and the inf components are integrated to build an Web application. However, in our methodology, integration is nothing but the correct implementation of the AUTH-INF interface. Therefore integration effort is very much localised. Furthermore,

testing also becomes easier. Since the individual components and the interface can be tested independently, testing is to some extent compositional.

- **Better Distribution of Development Skills.** INF and the AUTH components of a Web application have their own idiosyncrasies, and so, they require different skills. Therefore, it is better to isolate and allocate them to developers with appropriate skills. That is what our methodology advocates.

In view of the above benefits of the Parallel Framework, we are of the view that, this is a significant contribution to the field of Web engineering.

Our study has also the following limitations.

- The SEISN website as it is now is a static one; therefore, it cannot be a considered as a typical example. However, the design and implementation of this website has given us reasonable amount of experience of the issues involving Web development. We intend to apply our methodology over larger Web applications (i.e. with complex infrastructure component) in future.
- In our research, we have not dealt directly with issues such as security and performance. These issues, although of primary importance, were not within the scope of our short study. The degrees of interactivity, maintenance and traffic in the SEISN Web site were also low. We are of the belief that our approach is scalable as we have already analysed the uses cases of large applications and separated them into UC_{inf} and UC_{auth}. Splitting is a major task and our experience is that the rest of the development will not create much problem with the exception of testing when both are integrated. This needs further research.

3.8 Conclusions

In this chapter, we have discussed a parallel framework for the development of Web application which separated, starting from the requirement analysis phase, the development of the authoring and the infrastructure component of the Web applications. The individual components are developed independently and then they are integrated to build the whole of the Web application. The approach towards the design is very much compositional. We have also demonstrated the effectiveness of our methodology through the running example of a small Web application. However, the efficacy of our methodology needs to be validated by using it to build large scale Web applications. The benefits of our methodology are many. The prominent among them are:

(i) Modular design facilitates better maintainability.

(ii) The methodology promotes a cleaner architecture of the design.

(iii) The whole development process is carried out in the UML framework.

So far, we have discussed the problems with current approaches to Web development and presented a meta-model to address the issues concerned. Constructing complex large scale Web applications is a difficult task, and efficient tool support can only help developers to meet the challenge. In the remaining chapters, we will discuss such a tool for managing Web projects. In addition, we will discuss the assessment and improvement of Web processes. Such assessment and improvement approaches are essential for enhancing the quality of Web processes and applications.

Chapter 4

Development of a Project Management Tool

4.1 Introduction

Development of Web applications in particular and of any software in general, requires proper management of the various processes and the associated artefacts such that the final product is developed ‘on time and within budget’ and that it must satisfy the desired quality needs. Project management involves the selection and systematic application of a number of software engineering principles and techniques. Improved techniques concerning estimation, tracking, data interpretation etc. will lead a project closer to its goal. In actual management settings, the manager has to take decisions in relation to an approximate and often abstract picture of the project environment. The fact that much of the information about past projects is usually unknown or uncertain makes it more difficult to ascertain the final project parameters. Moreover, the available data comes from multiple sources and the integration of all such data may not be easily performed with the available software tools.

In this chapter we describe the design and development of a proof-of-concept project management tool which provides an environment where project management and quality aspects are given due importance. Current project management tools mostly deal with time, tasks and resources. Note that, currently, our tool does not have its full capability. At this stage, the tool can be seen as a proof of research ideas.

4.2 Rationale for the USEGESOFT Tool

The motivation behind USEGESOFT, our new project management tool, can be summarised as follows:

- Project management tools usually deal with tasks, time and resources but do not adequately address quality issues. Our tool stores and manipulates quality information for making important managerial decisions. By quality we mean both product quality as well as process quality. Current tools mostly address product quality, not process quality. Our tool links process quality with product quality by taking as reference some recent quality models like the TGPM (see Section 4.2.2). Information about this relationship is crucial for making process improvement decisions. Addressing quality aspects is of paramount importance to Web applications (Chapter 1: Section 1.3). Therefore, our tool will be useful for handling Web applications and processes.
- For performing process assessment and process improvement, it is necessary to have a clear understanding of the process details (Fugetta *et al*, 1998). Without a clear knowledge of the structure of the process, process improvement will look like black-box actions. Our PMT defines process structure details in a graphical notation (Rodriguez *et al*, 2001b; Satpathy and Harrison, 2002).
- The subprocesses of a development process usually have their specific characteristics. Our PMT can address the quality needs of individual processes by using instantiated process models which are obtained from the generic model TGPM.
- USEGESOFT incorporates new modelling techniques like Bayesian networks and data mining.

4.3 Requirements for our Project Management Tool

The idea of this tool is to provide support for basic software project management activities and to create a platform for research in these issues. From the literature, we can summarise software project management activities into project inception, execution and wrap-up (Jalote, 2002).

Therefore, the USEGESOFT should provide support to handle different project entities and activities during project inception including:

- Definition of processes, project structure, resources, milestones, temporal sequences, quality models and quality objectives.
- Calculation of cost and effort estimates.
- Risk management.

During the execution of the project, the USEGESOFT tool should support activities for controlling, measuring and evaluating activity performance towards planned objectives including:

- Project tracking and monitoring.
- Quality metrics collection and monitoring.
- Project adjustments.

At the end of the project, the USEGESOFT tool should give support for post-mortem analysis and reporting activities. Examples of such analysis are:

- GQM based process assessment and improvement
- Prior analysis of improvement actions

Among the non-functional requirements, it is important to provide an intuitive and friendly graphical user interface. However, other non-functional requirements such as performance, security and internationalisation have not been taken into account at this stage. These issues will be important in subsequent releases of our project management tool.

4.4 Related Work

4.4.1 Project and Quality Management Environments

4.4.1.1 MILOS

The MILOS (Minimally Invasive Long-term Organisational Support) project (Maurer *et al*, 2000) is an environment for modelling, planning and execution of software development processes. This system follows a dynamic model for coping with changes by providing a notification mechanism which tracks product changes and informs the concerned people of the change. There are four major components in the MILOS system: (i) the resource pool; (ii) the process model; (iii) the project plan and (iv) the workflow engine. Although this system copes with processes and products related to the different processes, it does not cover quality aspects.

4.4.1.2 SQUID

SQUID (Bøegh *et al*, 1999) is a metric based analysis environment which can be used during project management. It handles some important aspects of project management. The tool identifies what are the metrics to be measured, their target/expected values and at what point they should be measured. After performing measurements, it outputs an analysis report like whether the deviations from expected values were acceptable or not. Consistency of the analysis is ensured by projecting the current attributes/values against past but similar projects. The Squid Data Model (SDM) and the Squid Quality Process (SQP) together provide a rigorous environment for identifying, defining, collecting and analysing quality measures.

The SDM identifies essential characteristics of a development process which includes the development model, and identification of model objects like activities, deliverables and review points; quality measures of the deliverables are taken at review points. Different parts of a product, SQUID calls them product portions, have different quality characteristics. The measures associated with the quality attributes can have actual values, estimated values and target values. In addition, SQUID prescribes measurement protocols which specify the conditions under which measurements should be taken.

The SPM is composed of the following four steps:

(i) *quality specification*: It involves selection of a quality model like the ISO 9126, identification of product portions, their quality requirements (external attributes) based on the selected quality model, targets values of the external quality attributes and feasibility analysis of the targets by comparing them against past projects.

(ii) *quality planning*: Different development processes may be adopted for different product portions. Quantitative target values are assigned to internal quality attributes of the deliverables and they are associated with the concerned activities.

(iii) *quality control* i.e.: Progress is monitored throughout the development. Measurements are taken at review points and deviations from target values are identified. Software components with unusual values are identified in the process.

(iv) *quality evaluation* : Based on the measures taken, the actual values of the external attributes are compared against their target values and any shortfall in the quality requirements of the product portions are reported.

4.4.1.3 MiniSQUID

Kitchenham *et al* (2001) have proposed a standard for storing project data and their subsequent analysis. Software datasets which may come from multiple sources need to be trustworthy, comparable and repeatable. The proposed standard centres around the notion of storing adequate amount of metadata along with measured values (or datasets) so that the datasets when analysed in conjunction with the stored metadata, it will minimise the risk of misanalysis. They have proposed an Entity-Relationship data model which allows complex software data sets to be modelled and their data values stored. They have also developed a prototype tool called MiniSQUID which supports most of the features of their standard.

4.4.1.4 SEGESOFT

We developed SEGESOFT, a project management tool (Tuya *et al*, 2001), which integrates simulation and dynamic modelling, knowledge discovery, quality models etc so that a manager can obtain and manipulate a broad spectrum of information associated with various projects. We have also improved this tool to produce the Upgraded SEGESOFT (or USEGESOFT). The detailed discussion on USEGESOFT is the subject matter of this chapter.

4.4.2 Products and Processes Quality Models

By a software process we will mean any software activity associated with the development and maintenance of software: from requirement analysis through to maintenance. A product is an entity, which a process (e.g. any software activity) produces as output. Products may also be fed to processes as inputs.

One of the important motivations behind USEGESOFT is that it must provide a framework for metric based software engineering practices. Therefore, quality models, of process as well as of products, need to be a part of the tool design. Since TGPM is the reference process model of USEGESOFT, we will discuss it in detail and we will briefly outline a few important quality models here.

4.4.2.1 The Typed Generic Process Model (TGPM)

4.4.2.1.1 Products and Processes

Satpathy *et al* (2000) in their description of TGPM (Typed Generic Process Model) give types to products and processes. For example, Table 4.1 enumerates some of the products and their types. Product ***rd*** belongs to the set of requirement documents, ***spc*** belongs to the set of specifications and so on. TGPM models the set of requirements by the set PRE_ELICIT_REQ. TGPM defines a process as a relation from a set of products to another set of products; the set of relations from *m* input products to *n* output products is denoted by:

$$\mathrm{IP}_1 \times \mathrm{IP}_2 \times \ldots \times \mathrm{IP}_m \leftrightarrow \mathrm{OP}_1 \times \mathrm{OP}_2 \times \ldots \times \mathrm{OP}_{n,}$$

where IP_i and OP_j are the types of the *i-th* input product and the *j-th* output product respectively. Table 4.2 shows the types of a few processes. A process may have many type definitions. For example, if an organisation develops formal specifications from requirements only, then the type of the formal specification process (FS process) is: *RD* ↔ *FS*. On the other hand, if an organisation (in relation to a project) also decides to interview the users for developing a FS, then the type of FS process would be: *PRE-ELICIT-REQ* × *RD* ↔ *FS*.

req: *PRE-ELICIT-REQ*	// Set of pre elicitation requirements
rd: *RD*	// Set of requirement documents
spc: *SPC*	// Set of specifications
des: *DESIGN*	// Set of designs
doc: *DOC*	// Set of documentations
impl: *IMPL*	// Set of implementations
Des: *DESIGN*	// Set of designs

Table 4.1 Types of Some Products

Requirement analysis: *PRE-ELICIT-REQ* ↔ *RD*
Specification: *RD* ↔ *SPEC*
Formal design: *RD* × *FS* ↔ *FD*
Maintenance: *RD* × *SPEC* × *DES* × *IMPL* × *DOC*↔ *RD*× *SPEC* × *DES* × *IMPL* × *DOC*

Table 4.2 Types of Some Processes

4.4.2.1.2 The Internal Process Model

TGPM assumes that the internal structure of a process is well defined. A process may be composed of subprocesses. A subprocess is also a process in the sense that it takes a product set as input and gives out a product set as output. Whether a process should be decomposed into subprocesses or not is decided by the process designer. When considered from a process point of view, the inputs and outputs of subprocesses are called intermediate products. A subprocess which is not decomposed further is called an atomic process. An atomic process is defined as a set of process steps, like the steps of an algorithm. The steps usually do not have rigorous definitions: a process executer usually has degrees of flexibility while executing them. Figure 4.1 shows the internal details of the testing process. An organisation is expected to have its own definition of the internal structure of a process.

In Figure 4.1, the testing process takes an implementation (a set of program modules) and a set of test cases as input and produces a tested implementation and Test results as output. As shown in the Figure 4.1, the testing process consists of two subprocesses: unit testing and integration testing. Unit testing consists of two atomic processes: black box testing and white box testing. Integration testing is an atomic process. Black box testing and white box testing can proceed in parallel, both producing their respective test results. For simplicity, we will consider the only scenario where the implementation passes the test cases. After the unit testing, the program modules are integrated by the integration process. Since the process of integration is not a part of the testing process, following the notations of TGPM, it has been shown outside the scope of the testing process. The integrated modules and the test cases are then fed to the subprocess integration testing which produces a tested implementation along with test results.

There are three aspects to the steps of an atomic process: *what are its process steps, who will execute those process steps* and *how they will be done*? An atomic process is defined in terms of process steps and each process step is in turn defined by a set of flexible guidelines. Further, some resources like supporting tools may be needed to execute the process steps. All of these contribute to the 'how' part of the questions. Each atomic process, in addition to its input and output product sets, will have its starting and ending conditions. They correspond to the 'what' part of the questions. The 'who' part will be taken care of by a person or some automatic tool. ISO/IEC 12207 (ISO, 1995) follows a similar (though less formal) approach to process modeling.

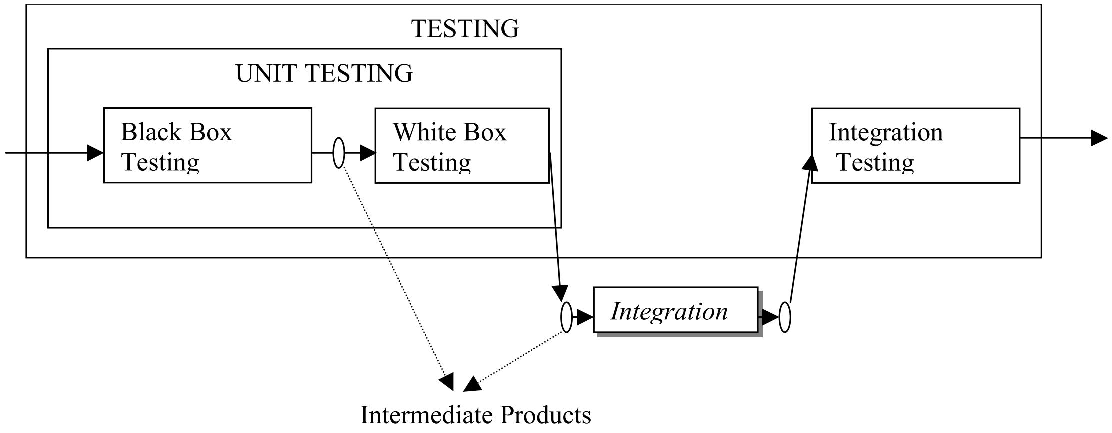


Figure 4.1 Intermediate Products in the Testing Process

The TGPM that we are going to describe is always viewed in relation to well defined internal structure of processes.

4.4.2.1.3 The TGPM and its Instantiation

The TGPM is a generic process model, which could be instantiated to obtain the process quality model for an individual process. The TGPM is defined in terms of eight factors: Functionality, Usability, Efficiency and Estimation, Visibility and Control, Reliability, Safety, Scalability, and Maintainability; each subfactor is in turn defined by a set of subfactors. The definitions of the subfactors of TGPM can be found in (Satpathy *et al*, 2000). The most important aspect of TGPM is that it defines attributes from a dual perspective. Consider, for instance, the understandability attribute. The two perspectives are (i) the logical concepts of a process should itself be understandable to the process executer and further (ii) the process should make its output product sets understandable. When we consider the formal specification process, the method of creating the formal specification and the formal specification language itself must be understandable to the specifier, and further the specification process must make the formal specification understandable to its users. In conclusion, any process assessment must consider such a dual perspective. Of course since we are looking at product attributes from the process quality point of view, we may miss out some important product factors which are not directly addressed by process quality factors. In order to alleviate this problem, the TGPM uses major product quality models (like ISO 9126) as references while defining our generic process quality model.

TGPM could be seen as a template with three parameters:

TGPM (input-product-set-type, output-product-set-type, application-domain)

The input product and the output product sets have been discussed earlier. By application domain we mean whether it is a safety-critical application, a real-time application, a business application etc.

The customisation of TGPM proceeds in two steps: (i) the substitution step and (ii) the refinement step. In the substitution step, we substitute the parameters with their actual bindings. Let us take the example of the FS process and let its type be: ***RD*** ↔ ***FS***. The substitution is illustrated by the expression:

TGPM [<u>RD</u> / input-prod-set-type] [<u>FS</u> / output-prod-set-type]

What the above expression signifies is that, all occurrences of the input product set in the definitions of the factors and the subfactors in the definition of TGPM are substituted by RD (requirement document). Similarly, all occurrences of output product set are substituted by FS (formal specification). Thus, at the end of the substitution step, we have a crude definition of each of the subfactors of the process concerned.

For the *application-domain parameter*, suppose we have a safety critical application. Now, with the knowledge that we are dealing with a FS process in a safety critical application, the refinement step refines the crude definitions that we have obtained after the substitution step. The result then will be the customised quality model for the FS process. As an illustration, consider the first part of the subfactor ‘completeness’. After the substitution step, we obtain the following definition: *the degree to which the process* *transforms all of the functionalities of the RD into the FS*. In the refinement step we know that it is the FS process and the application domain is the ‘safety critical application’. Further FS process achieves transformation through specification; and at the RD level, a functionality is understood by a ‘feature’. So the refined definition is: *the degree to which the FS process specifies all of the features (including the safety critical features) of the RD in the FS.*

The distinctive feature of TGPM is the dual perspective of a process attribute. In plain terms, it says how good a process attribute is in relation to the process itself and then how it is related to the output product(s) of the process and also to the end-product. This duality point of view identifies certain process features such as (a) process faults, (b) process understandability, (c) process scalability, (d) process reliability, (e) process stability etc. These features were not properly addressed by any of the previous process models.

4.4.2.2 ISO/IEC 9126

ISO/IEC 9126 (ISO, 1991) is a product quality model describing a generic model for specifying and evaluating the quality of software products. The model isolates six factors, called Functionality, Usability, Reliability, Efficiency, Maintainability and Portability; and the quality of a product is defined in terms of the quality of the above factors. Each factor may in turn be defined by a set of subfactors. The FURPS+ model (Grady and Caswell, 1987) used by HP is quite similar to ISO 9126. Kichenham and Pfleeger (1996) point out that the models like ISO 9126 have problems as regards to consistency and completeness; i.e. the selection of quality characteristics and subscharacteristics are to some extent arbitrary, and further, it is not clear as to how the low-level metrics associated with the subcharacteristics are composed to obtain an assessment of high level characteristics.

Focusing on product quality alone may not guarantee that an organisation will deliver products of good quality as a result of software processes. So, based on an orthogonal view that improving the quality of a process will deliver products of good quality, many models have been developed. Prominent among them are the CMM (Humphrey, 1992) and ISO/IEC 9001 (ISO, 1994). Models like BOOTSTRAP (Kuvaja, 1994) and ISO/IEC 15504 – SPICE (ISO, 1998) are variants of the CMM. ISO/IEC 12207 (ISO, 1995) does a classification of all processes associated with the software development and offers general guidelines which can be used by software practitioners to manage and engineer software. The scope of most of these standards covers an entire organisation.

4.4.2.3 Dromey's Quality Framework

The ISO 9126 standard takes a top-down approach towards formulating a quality model for software products; a set of high-level quality attributes are identified, and each such high-level attribute is then decomposed into a set of low-level attributes. Dromey points out that the relationship between the low-level internal attributes and the high-level quality attributes is never well defined under ISO 9126; i.e. there exists a semantic gap between the high-level and low-level quality attributes (Dromey, 1995, 1996). In order to bridge this semantic gap, Dromey's quality framework takes a top-down view and a bottom-up view of software quality. A product has a structural form in the sense that it can be decomposed into components and then components can be decomposed further into sub components. This process may have to be repeated to reach at atomic components. The entities at the atomic level have some quality-carrying properties, which influence the high-level quality attributes of the product. For instance, giving meaningful names to variables and subroutines influence the maintainability of the product. If any of the quality carrying properties of a structure is violated, then this will lead to a quality deficiency at the next higher level structure and therefore to a deficiency in the quality of the product. Therefore, quality can be built into the software by ensuring that all the quality-carrying properties associated with each structural form are satisfied. And hence, detecting quality defects in software reduces to systematically checking whether, for each structural form, any of the quality carrying properties is violated. Thus, Dromey's model makes the context of a high-level quality attribute transparent.

Dromey takes the view that, we cannot ignore the process when addressing product quality. Therefore, to ensure that quality-carrying properties of a product are adequately addressed, the process must have a level of maturity.

One problem that we have observed with Dromey's model is that he assumes that the composition of components satisfies the *compositional property* which may not be the case. If the composition is not compositional, then correctness of components may not establish the correctness of the composition and therefore it must be addressed explicitly.

4.4.2.4 The Capability Maturity Model (CMM)

The SEI (Software Engineering Institute) has suggested the Capability Maturity Model (Paulk *et al*, 1993; Humphrey, 1992) to classify software organisations in relation to the processes adopted by the organisation. The CMM deals with the capability of a software organisation to consistently and predictably produce high quality products. The model defines five maturity levels: from Level 1 to Level 5. A maturity questionnaire assesses the following three areas of the organisation: (i) organisation and resource management, (ii) software engineering process and its management and (iii) the tools and technology; and based on the assessment the model certifies that the organisation is at certain maturity level. The model also identifies the weak areas (called *Key Process Areas* or KPAs) and prescribes a set of guidelines; following those guidelines, an organisation can address those areas and attain the next higher maturity level. Measurement, quantitative feedback from the processes and continuous process improvement are some of the highlights of the model. Figure 4.2 shows the five levels of maturity.

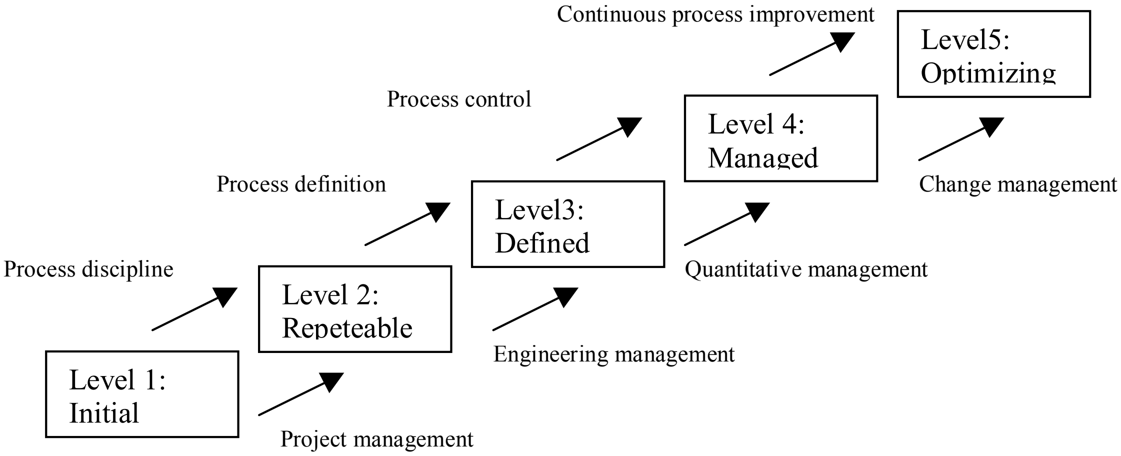


Figure 4.2 The CMM Levels

Level 1 – The Initial Level. At this level, an organisation typically has an unstable and disorganised process. Success depends entirely on having a very good manager and an experienced and effective software team.

Level 2 – The Repeatable Level. Organisations establish policies for managing a software project and procedures to implement those policies. New projects are planned and managed based on experience with similar projects but the internal processes are unknown; i.e. they are like black-box processes. The KPAs at Level 2 include (Paulk *et al*, 1993):

- Requirements Management
- Software Project Planning
- Software Project Tracking
- Software Subcontract Management
- Software Quality Assurance
- Software Configuration Management

Level 3 – The Defined Level. At this level, the standard process for developing and maintaining software across the organisation is implemented. The KPAs at Level 3 include (Paulk *et al*, 1993):

- Organisation Process Focus.
- Organisation Process Definition
- Training Program
- Integrated Software Management
- Software Product Engineering
- Inter-group Coordination
- Peer Reviews

Level 4 – The Managed Level. Software processes are instrumented with measurements, which allow the evaluation of the projects' software processes and products. The KPAs at Level 4 focus on (Paulk *et al*, 1993):

- Quantitative Management
- Software Quality Management

Level 5 – The Optimising Level. At this level, the entire organisation is focused on continuous process improvement. Software project teams analyse defects to determine their causes. The KPAs at Level 5 cover (Paulk *et al*, 1993):

- Defect Prevention
- Technology Change Management
- Process Change Management

4.4.3 The Goal-Question-Metric Method

The Goal-Question-Metric (GQM) method (Basili *et al*, 1994; Solingen and Berghout, 1999) proposes a measurement method for assessing or improving the quality of entities like products, processes or people. It starts with a set of business goals and the goals are progressively refined through questions until we obtain some metrics for measurement. The measured values are then interpreted in order to answer the goals. Existing approaches choose a quality model from those that exist so as to generate the business (or the primary) goals of the GQM formulation for any individual product or process.

4.4.4 Application of Metrics in Industry

Application of Metrics in Industry (**ami**) (Kuntzmann-Combelles *et al*, 1996) combines CMM and the GQM method, and the result is that it provides a complete framework for process improvement. The **ami** approach is iterative, goal-oriented, quantitative, involves everyone in the organisation and integrates necessary management commitment. It covers the whole of the process improvement cycle. CMM or other standards like Bootstrap, SPICE, ISO 9001 etc. are used to identify weak areas in the development process. This information along with the business and environment specific objectives is used to define some software process goals. The goals are validated and then refined into sub-goals. The sub-goals are further refined into metrics and a measurement plan is made to collect data. The data are then analysed and related to the original goal. Based on the data collected an action plan may be made to improve the development process. New goals are then defined and the cycle is repeated.

4.4.5 Techniques for Software Project Management

In our tool we use the following techniques and methodologies which are available in literature.

4.4.5.1 Estimation Techniques

The basic COCOMO (Boehm, 1981) model calculates relationships between product size, project effort and project duration. It predicts the total effort (*eff*) that would be necessary to generate a source program of certain complexity. It is represented by the equation: $eff = c \cdot L^b$, where *eff* is the effort in man months, L is the code size in *KLOC*, *c* and *b* are constants dependent on the complexity of the problem. COCOMO defines that the complexity of a problem can be *organic*, *semi-detached* or *embedded*. And the respective values of *c* and *b* are given in Table 4.3.

Complexity	*C*	*B*
Organic	2.4	1.05
Semi-detached	3.0	1.125
Embedded	3.6	1.2

Table 4.3 COCOMO Parameters

The intermediate and detailed COCOMO models make plans (effort and schedule estimates) for each development phase of the standard software project using a set of so-called cost driver attributes (with associated effort multipliers). The detailed COCOMO model differs from the intermediate COCOMO model in that it uses different effort multipliers for each phase of a project. The phase dependent effort multipliers yield better estimates than the intermediate model.

4.4.5.2 Knowledge Discovery in Databases

The use of computational techniques and tools for extracting useful knowledge from databases is known as machine learning or Knowledge Discovery in Databases (KDD). In general, KDD techniques try to extract in an automatic way the information useful for decision support or exploration of the data source (Fayyad *et al*, 1996). This will be discussed in detail in Chapter 6.

4.4.5.3 System Dynamics

System Dynamics (Forrester, 1961) is a method for studying how complex systems change over time where internal feedback loops within the structure of the system influence the entire system behaviour. The application of system dynamics to software projects provides the perspective of considering them as complex socio-technological dynamic systems, whose evolution will be determined by their internal structure as well as the relations established inside the working team. This allows the development of dynamic models to describe the feedback structure of the system being modelled as well as the mental process followed by project managers in making decisions. It is to be noted that decision making has been traditionally based on a manager's experience. The building of a dynamic model for a software project is based on the evolution of the project. Therefore, the attainment of the project goals such as meeting the deadlines, a project phase being within budget etc. depends on: (i) the initial estimations of the necessary resources; (ii) the management policies to be applied; (iii) the characteristics of project; and (iv) the characteristics of the organisation.

4.4.5.4 Bayesian Belief Networks

A Bayesian Belief Network (BBN) (Jensen, 1996; Fenton and Neil, 2000) is a directed graph in which the nodes represent uncertain variables and the arcs represent the causal relationship between the variables. Each node has a probability table, which stores the conditional probabilities for each possible state of the node variable in relation to each combination of its parent state values. For a node without any parent, such a table stores the marginal probabilities for each possible state of that node. If the state of certain node is known then its probability table is altered to reflect this knowledge. Such knowledge is then propagated to determine the changed probabilities of (of all possible values associated with) other nodes. Note that the initial probabilities of the nodes in a BBN are obtained from expert judgment and/or past project data. In fact, tools are available to help in the generation of BBNs from historical project data (Tiga, 2002; Cheng *et al*, 1997). We will discuss BBNs in greater detail in Chapter 6.

4.4.5.5 GANTT, PERT/CPM, Earned Value Management System

Program Evaluation and Review Technique (PERT) or (Critical Path Method) CPM charts depict tasks, duration and their dependencies using a network diagram consisting of numbered nodes representing events or milestones linked by vectors representing tasks in the project.

A GANTT chart is a matrix where the horizontal axis represents time and the vertical axis lists all tasks to be performed (see Figure 4.3). Horizontal bars of varying lengths represent the sequences, timing, and time span for each task. There are various ways of representing the bars to include resources, skills or progress when the project in under way. PERT and GANTT techniques must be augmented with a system in which effort, cost and schedule are summarised in a way that can be easily understood by the project manager.

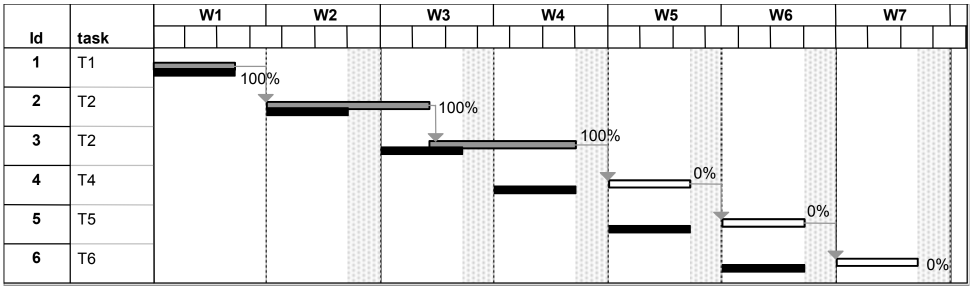


Figure 4.3 Example of a GANNT Chart

The Earned Value System or Earned Value Performance Measurement (EVPM) (Flemming and Koppelman, 1996) allows us to monitor the actual status of a project and extrapolate both cost and schedule for estimating the project completion in a graphical way. Christensen enumerates several benefits of EVPM (Christensen, 1998) which include: (i) single management system; (ii) integration of cost and schedule; (iii) early warning of trouble; (iv) accurate predictor of final cost and (v) to reduce information overload.

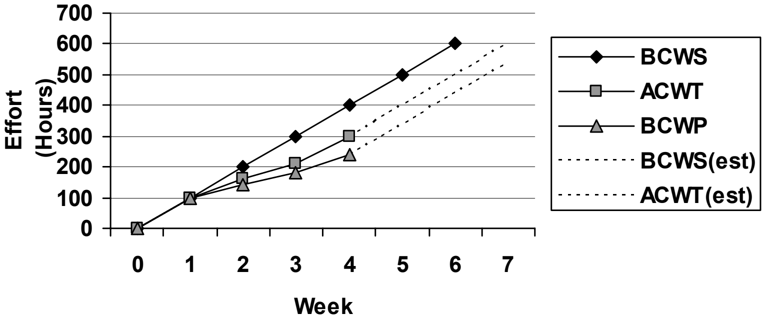


Figure 4.4 Earned Value Performance Measurement (EVPM)

Figure 4.4 is an example of an EVPM graph. The horizontal axis represents time (weeks) and the vertical axis representing accumulated effort. The BCWS (Budgeted Cost of Work Scheduled) in relation to a project is obtained by adding the planning effort of all the constituent tasks. The ACWP (Actual Cost of Work Performed) represents the accumulated effort until the current time. Finally, the BCWP (Budgeted Cost of Work Performed) measures the percentage of the job completed. For instance, the project in Figure 4.4 is behind schedule but used less resources than planned.

4.5 Structure of USEGESOFT

The goal of this tool is to have a flexible structure so that analytical techniques can be incorporated into the structure in an incremental manner. The system collects and records both actual and simulated project data and implements different techniques such as machine learning, project tracking, dynamic modelling etc. The tool also accommodates the distinctive features of the TGPM for quality assessment and improvement, and supports a database where all information associated with processes, products and their relationships can be stored. It also provides a graphical editor for process representation. The tool also helps in building and maintaining the internal structure of a process. Finally, the USEGESOFT tool intends to provide an environment for training project managers.

The idea is that the project manager makes the initial plan, creating a process model, artefacts needed for each activity and assigning resources to activities. The project members would carry out their tasks and report their status to the project manager. They would also collect necessary metrics as desired by the project manager (see Figure 4.5). The USEGESOFT tool is intended to provide an environment to facilitate the above tasks.

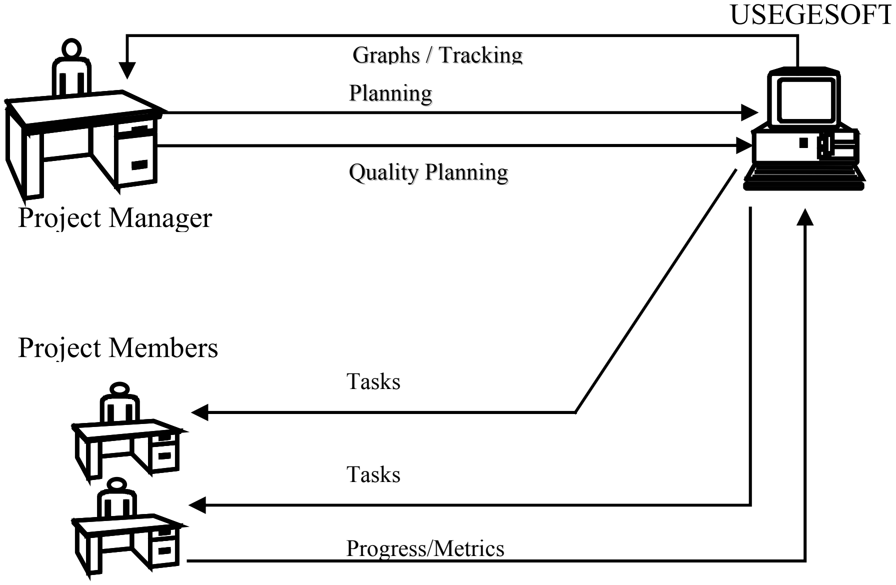


Figure 4.5 The USEGESOFT Environment

From the point of view of the implementation, the development of this tool is centred on open source development. Currently, the whole development is based on Java 2 environment. This approach allows us to reuse and extend software already developed under open source licenses. This decision had many benefits in the subsequent development, the most important being the incorporation of many a freely available software utility.

4.6 Functionalities Offered by the System

The high level Use Case Diagram of Figure 4.6 shows the functionalities covered by the system. The system provides functionality for representing processes and sub-processes as defined by TPMG, the artefacts involved for each process and resources. In addition, every entity (process step, artefact, and resource) may have relevant quality attributes. For each project, effort and cost data are recorded. Data for tasks and resources are also recorded in each version of the software project. The visualisation module can plot the trends of the parameters recorded. Other capabilities of the system include visualisation techniques such as EVPM, GANTT charts etc. The system also provides simulation capabilities using Bayesian Belief Networks and Systems Dynamics etc.

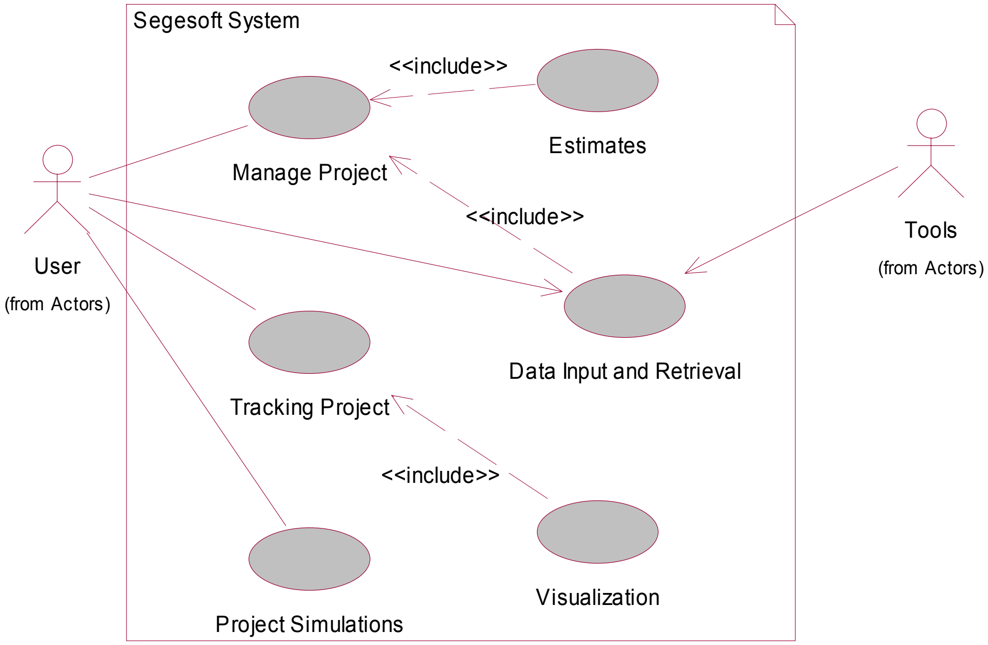


Figure 4.6 General View of the System

4.6.1 General System Architecture

The overall structure of the system is depicted in the package diagram of Figure 4.7. In the figure, packages represent the individual components of the tool, and the arrows represent relationships between the packages.

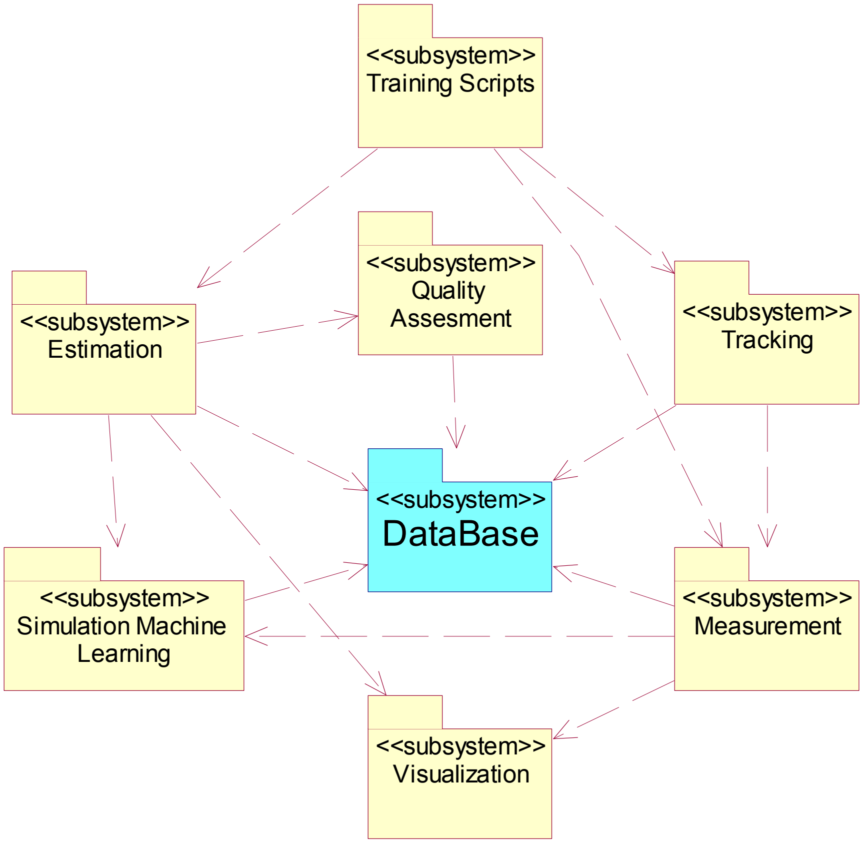


Figure 4.7 General Structure and Main Components of the System

- The *Measurement* module collects information about the metrics of the projects.
- The *Estimation* module provides the estimation methods like COCOMO, Function Points etc.
- The *Tracking* module monitors the evolution of the project.
- The *Modelling and Simulation* module allows the user to model the organisation by using approaches like the System Dynamics and the Bayesian Networks.
- *Machine Learning* module implements some common algorithms for summarising high volumes of data. Those algorithms are based on data warehouse algorithms.
- *Metrics Database* is the core of the system. It maintains all relevant information related to actual and past projects.

- *Visualisation* module provides ways of presenting and representing meaningful information in the forms of graphics and diagrams.
- *Quality Assessment* module implements some criteria for continuous assessment of the quality of the project. Processes and products are assessed based on the metrics collected.
- The *Training Scripts* module is used to train project managers.

4.6.2 System Database

For a given organisation, the project database contains all relevant information in relation to a set of projects as shown in Figure 4.8. For each project, the following information is stored:

- General data about the project, versions, initial estimations.
- Work-Breakdown-structure (WBS) which is composed of the hierarchy of tasks in relation to a project.
- Resource-Breakdown-structure (RBS) is composed of project members and tools which are related to process activities.
- Simulation-Breakdown-structure (SBS) stores the necessary parameters for the simulation.
- Machine-Learning-Break-down-structure (MLBS) where the necessary information for the Machine learning module is stored including management rules.
- Metrics-Breakdown-structure (MBS) which contains the tree of factors, subfactors and metrics for each entity.

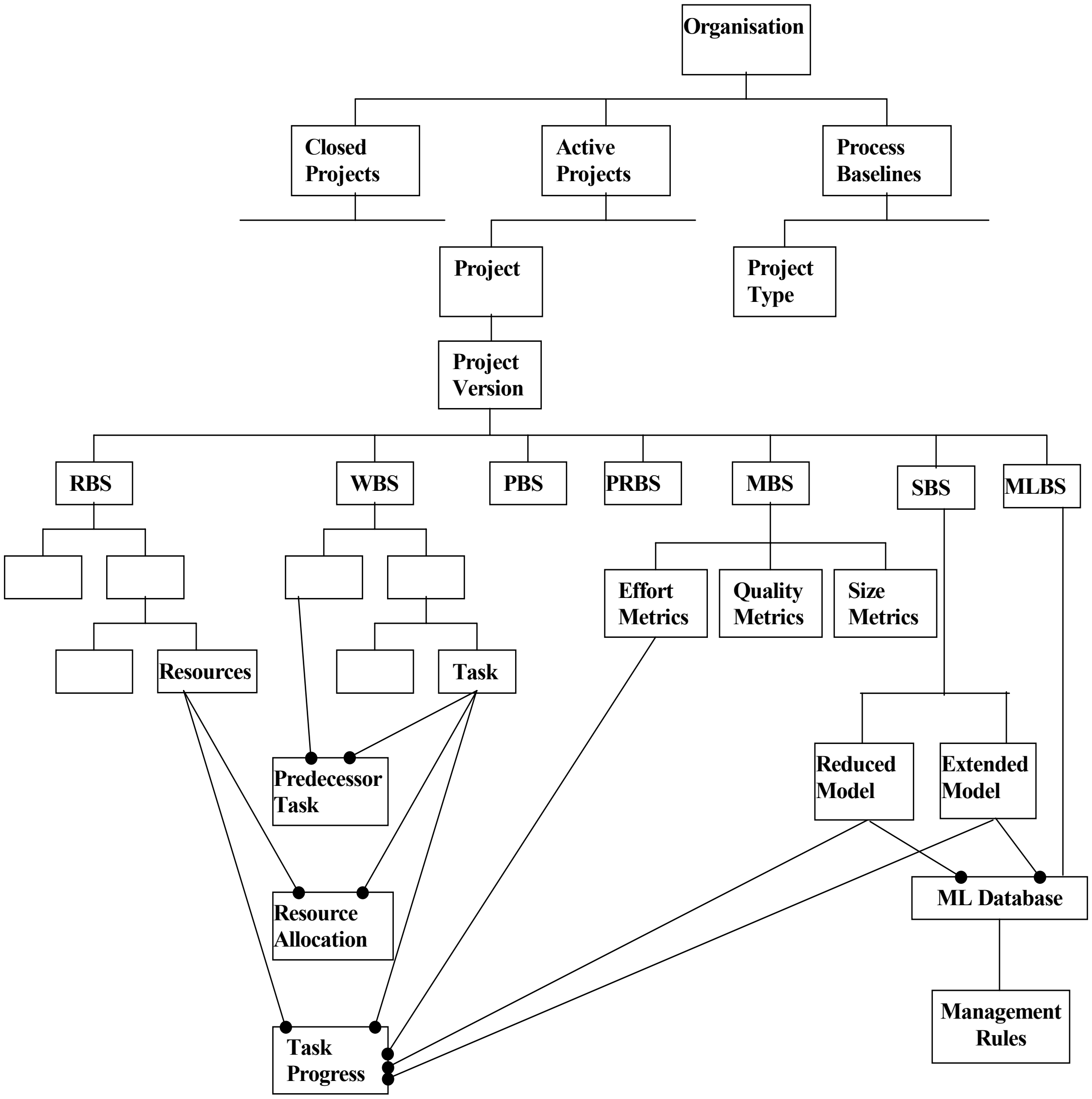


Figure 4.8 Outline of the Database Structure

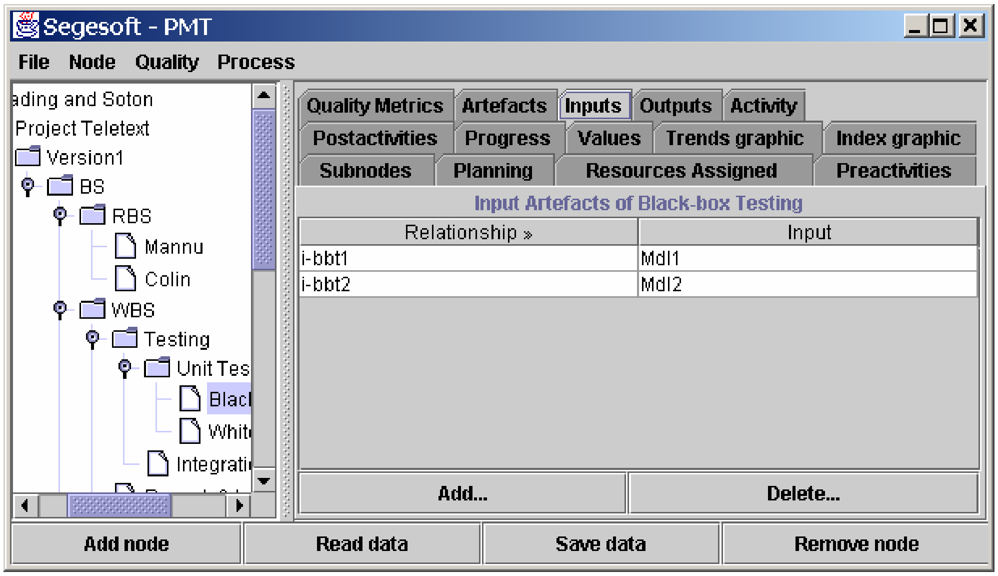
At the moment, our system is a standalone tool which stores information using data serialisation of Java objects and XML files. As we will discuss later, this approach is not scalable. Or future work involves moving all these information into a database and to make use of current standards such as CORBA and Web services.

4.7 Implementation of the Functionalities

Our project management tool integrates several modules using different technologies. In the following subsections, we will present succinctly some of the implemented techniques and the overall system structure.

4.7.1 Storing Data

At present all information is introduced to the system through a GUI and XML files. Figure 4.9 shows the main GUI of the system and provides an idea about the structure of the information which are recorded in the database. On the left-hand side of the main window there is a uniform search facility for all entities of the projects in the organisation (for modelling the process and quality metrics another window is opened). On the right-hand side, related information, properties, subcomponents of each element are easily accessed. It also shows some of the results generated with the techniques mentioned in previous sections.

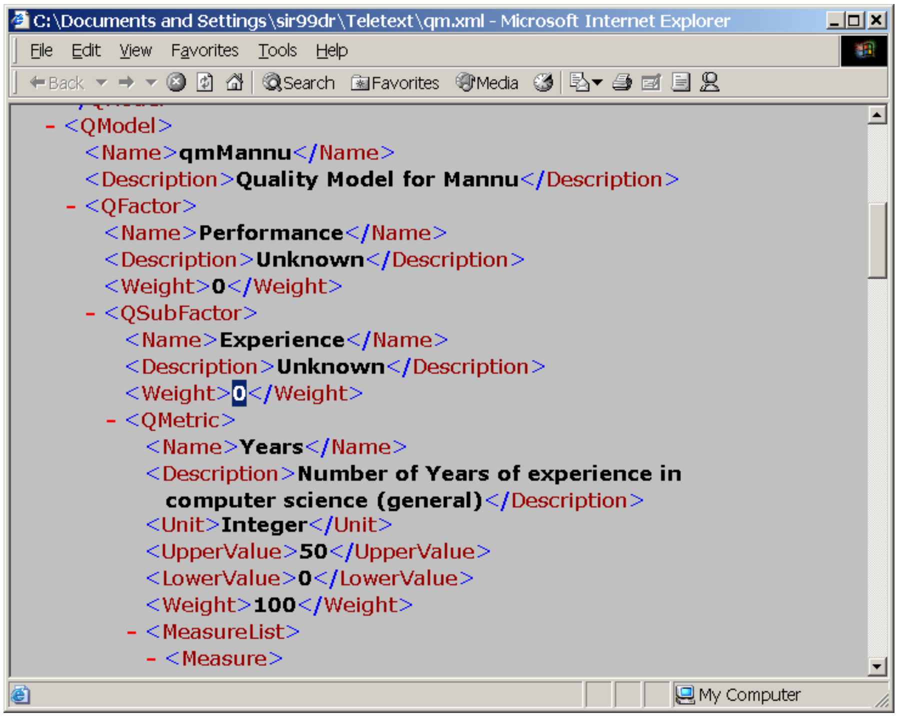


Input Artefacts of Black-box Testing

Relationship »	Input
i-bbt1	Mdl1
i-bbt2	Mdl2

Figure 4.9 Accessing the Data from the Main Window

Figure 4.10 shows a partial XML file that represents a quality model in XML.


C:\Documents and Settings\sir99dr\Teletext\qm.xml - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
My Computer

```
- <QModel>
    <Name>qmMannu</Name>
    <Description>Quality Model for Mannu</Description>
  - <QFactor>
      <Name>Performance</Name>
      <Description>Unknown</Description>
      <Weight>0</Weight>
    - <QSubFactor>
        <Name>Experience</Name>
        <Description>Unknown</Description>
        <Weight>0</Weight>
      - <QMetric>
          <Name>Years</Name>
          <Description>Number of Years of experience in
            computer science (general)</Description>
          <Unit>Integer</Unit>
          <UpperValue>50</UpperValue>
          <LowerValue>0</LowerValue>
          <Weight>100</Weight>
        - <MeasureList>
          - <Measure>
```

Figure 4.10 XML File Corresponding to a Quality Model

4.7.2 Quality Control

Measurement is essential in order to manage, predict and improve the quality of software products and processes. Current project management tools mostly deal with time and resources and usually do not consider quality aspects. Our tool provides quality as an additional axis in this respect. As discussed in Section 4.4, there are tools like SQUID and MiniSQUID which provide environments for managing product quality metrics; however, current project management tools do not usually provide facilities for dealing with such metrics. As we would like to control the quality of products and processes, we need to define quality models for both. For products, ISO 9126 is the standard quality model; however, different products (intermediate or final) emphasise different quality aspects. Therefore, for each product, the quality factors which are specific to the product are considered. In Figure 4.11, the top row of boxes represent various processes, while the boxes in the middle row represent the artefacts associated with the processes of the first row. For each artefact, our tool collects metrics which are specifically important to the artefact. Note that for dealing with product metrics we borrow some of the ideas of SQUID and MiniSQUID; however, at this stage our data collection, storage and analysis mechanisms are not as rigorous as they are in those tools.

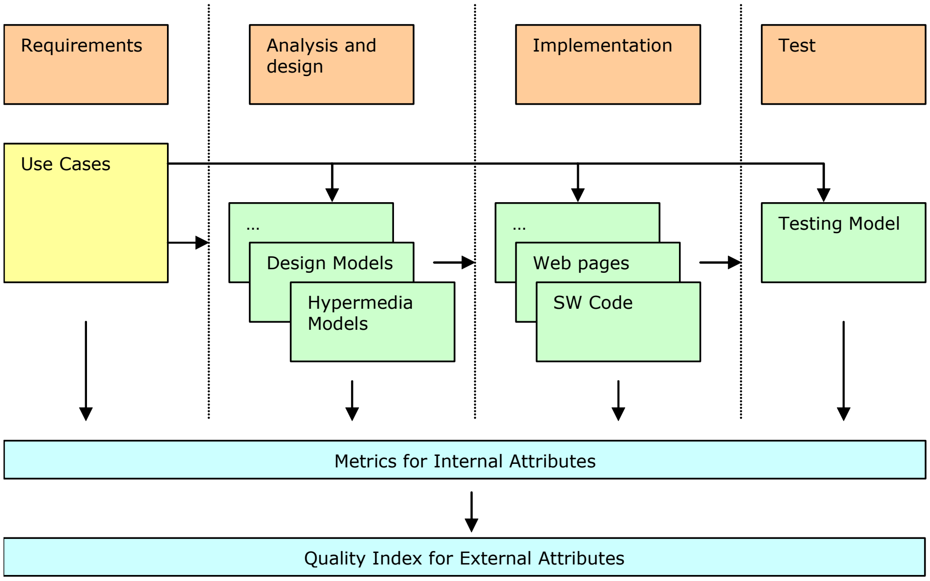


Figure 4.11 Individual Process as an Instantiation of the Measurement Framework

Figure 4.12 shows a UML diagram in which a quality object is related to an entity. A quality object stores the important quality factors for the entity, in which each quality factor is decomposed into quality sub-factors. Finally, metrics are direct measures that are used to evaluate sub-factors.

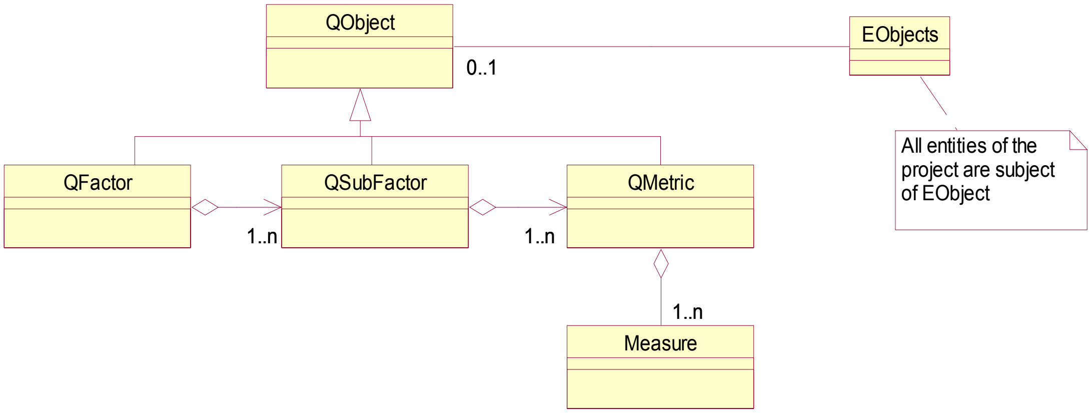


Figure 4.12 UML Class Diagram of Quality Attributes

Metrics relating to various entities are stored in the project database and are used by different software management techniques. USEGESOFT stores direct metrics, average values, upper and lower thresholds etc. Figure 4.13 shows the testing process (refer Figure 4.1) as created by the Process Editor of USEGESOFT. This editor is based on JHotDraw (Gamma, 1995) which was extended and linked to USEGESOFT to provide a tool support to the TPGM model (Rodriguez *et al*, 2001b).

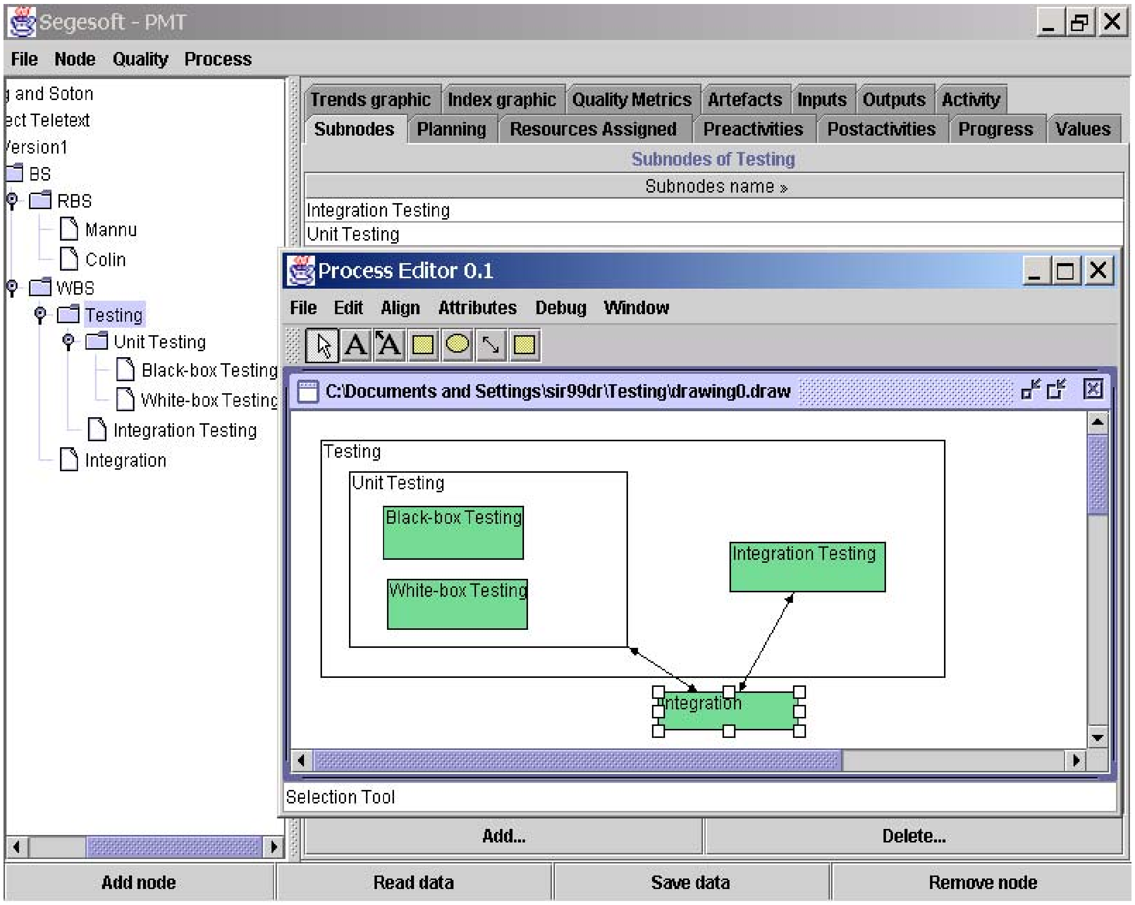


Figure 4.13 Drag and Drop Mechanism of the Process Editor

Processes usually have hierarchical structures (Satpathy *et al*, 2000). We need to store the following four types of information in relation to a process following the conventions of TGPM:

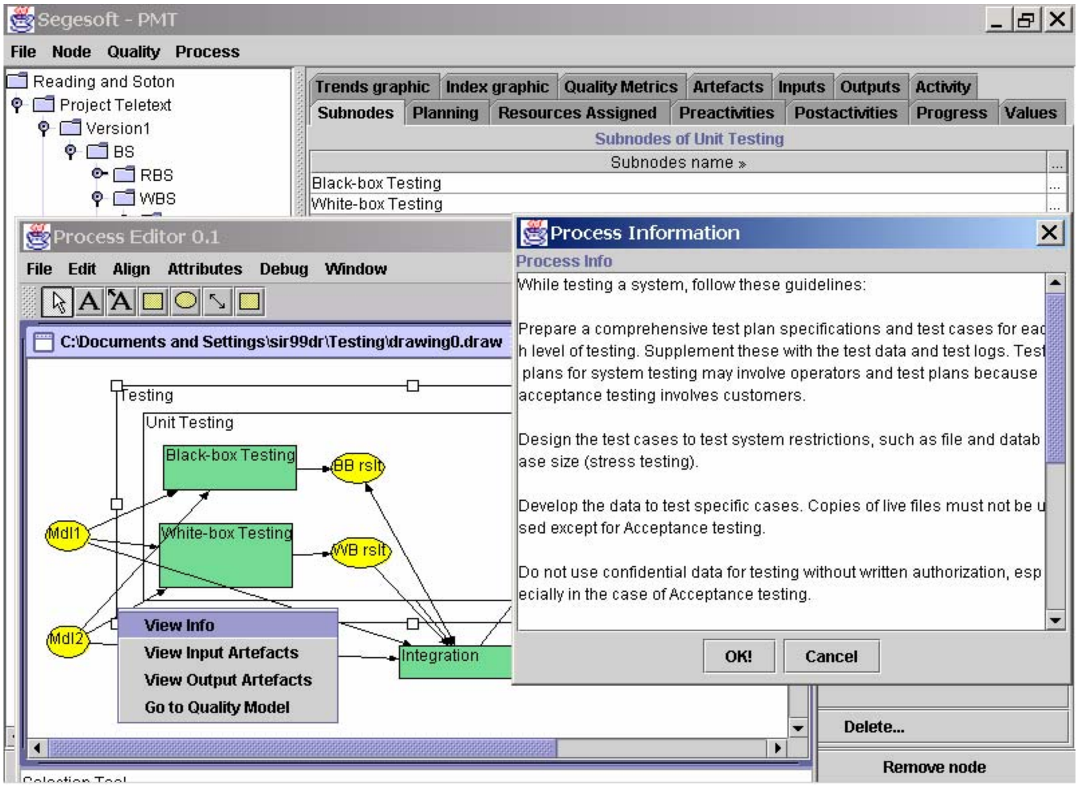
(i) A process is either an atomic process or it is defined in terms of sub-processes. So, documentation is necessary to specify the process structure.

(ii) A process is defined as a relation between a set of input products and a set of output products. Some documentation is necessary to say what such products are.

(iii) A process can have a hierarchical structure (see Figure 4.13). Each process or subprocess takes a set of input products and a set of output products. If we follow the ISO 9126 product quality model, then each such product can have quality factors and subfactors. Further each subfactor may be associated with metrics. So we need to store all such information which may be associated with the products. Since a process can be hierarchical, the number of such products and hence the volume of information associated with them could be significant.

(iv) Any process can have an instantiated model (instantiated TGPM). In relation to the instantiated model, each process will have quality factors and subfactors. Further, corresponding to each subfactor there may be metrics. So, each process should store all such information.

The USEGESOFT tool provides a good documentation mechanism to specify, store and display all the four types of information which can be associated with a process or subprocess. This information can remain hidden, or it can be displayed through a pop up menu. For example, when a process component is selected in the Process Editor, a context menu associated with it allows us to store or visualise all information associated with that component. The snapshot of Figure 4.14 shows the high-level information of the testing process through a pop-up menu. It is also possible to follow from the pop-up menu in a nested fashion to extract further information, if any. Alternatively, information can also be obtained by following the trail of entities from the main window.



While testing a system, follow these guidelines:

Prepare a comprehensive test plan specifications and test cases for eac
h level of testing. Supplement these with the test data and test logs. Test
plans for system testing may involve operators and test plans because
acceptance testing involves customers.

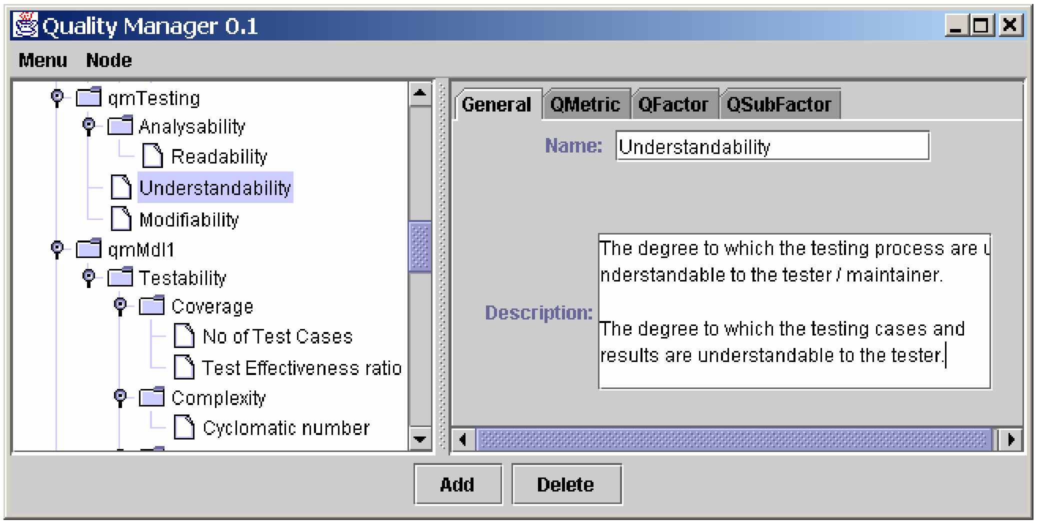
Design the test cases to test system restrictions, such as file and datab
ase size (stress testing).

Develop the data to test specific cases. Copies of live files must not be u
sed except for Acceptance testing.

Do not use confidential data for testing without written authorization, esp
ecially in the case of Acceptance testing.

Figure 4.14 Accessing the Process Information from the Process Editor

Figure 4.15 shows how definitions of various process quality attributes according to the TGPM, along with their dual definitions, can be obtained.



Name: Understandability

Description: The degree to which the testing process are u
nderstandable to the tester / maintainer.

The degree to which the testing cases and
results are understandable to the tester.

Figure 4.15 Process Quality Attributes According to TGPM

USEGESOFT stores a lot of information in relation to a process and the associated products. Depending on a requirement, we need to obtain relevant information from the database, which stores all the documentation. Furthermore, we need to use this information to obtain a graphical display or make inferences by using some statistical modelling approach. Currently, as quality information is stored using the open standard XML (Extensible Markup Language) (W3C, 1998), we are able to extract this information manually or transform it using XSL (Extensible Stylesheet Language) (W3C, 1999). In this way it is possible to manipulate or display data using a spreadsheet or other data processing applications.

4.7.3 Simulation and Dynamic Modelling

This module tries to mimic the main components and behaviour of already well-known software applications such as Vensim (Vensim, 2003), Stella (Stella, 2004), Powersim (PowerSim, 2003) etc. It is composed of a whiteboard for connecting the basic elements of the system dynamics method: levels, flows, constants, auxiliaries and clouds. This module implements a simplified dynamic model that can be used to simulate the temporal behaviour of the project. It contains the controls for setting three groups of required parameters: the parameters of the project (size, number of initial technicians etc.), the parameters of the organisation (average delays related to hiring, dismissals etc.) and, finally, the parameters controlling the simulation process.

Note that the System Dynamics module of USEGESOFT is not a part of this thesis; it has been developed by project collaborators (Aguilar-Ruiz *et al*, 2001).

4.7.4 Bayesian Belief Networks

USEGESOFT supports modelling using Bayesian Networks. To do so, we have upgraded the JavaBayes (Cozman, 2001) tool and linked it to USEGESOFT. BBNs will be discussed in detail in Chapter 6.

4.8 Feature Analysis

We list here the general features provided by currently available tools and compare those with the ones covered by the USEGESOFT tool. There are specific tools available in the market to cover specific areas of project management. They can be classified into tools for project planning, software cost estimation, quality metrics, project risk analysis, project reporting and process support (ESA, 1995).

Project planning tools should provide support for the project related activities like (ESA, 1995):

- Defining work packages and resources.
- Allocating resources to work packages.
- Defining the duration of work packages.
- Constructing activity networks using the PERT method.
- Defining the critical path.
- Defining the schedule in a Gantt chart.

The USEGESOFT provides most of the functionalities offered by commercial project planning tools but in a less flexible way when compared with tools like MS Project (2002) or MrProject (CodeFactory, 2002). This is because the current version of the USEGESOFT do not provide different views of the information, printing facilities, export and import facilities etc. From the functionality point of view, we have also found some deficiencies when compared with commercial tools such as fixed rate scheduling; so it is necessary to provide support to split tasks into different time frames.

Software cost estimation tools work in many different ways depending on the approach used (COCOMO, analogy, case base reasoning, COSMIC FPP, etc) but in general, they should allow managers to (ESA, 1995):

- Produce reliable estimates for entities like cost, effort, time etc.
- Provide explanations of how estimates were arrived at.

- Provide reliable estimates even when part of the information is not readily available.

The USEGESOFT tool has implemented the most general approaches, i.e., COCOMO, Mark II estimation methods. The approach is to use a measure of software size such us lines of code or function points to produce the initial estimate of effort. In addition, our tool supports the use of Bayesian Networks which enable estimates to be calculated even when part of the required information is not available. Furthermore, our tool, through the use Bayesian Networks, can refine an estimate as and when more and more information are available. Currently, only some specific estimation prototypes supports the use of Bayesian Networks.

Project risk analysis tools are usually based on simulation using Monte Carlo, system dynamics, or Bayesian Networks. Many available tools are based on heuristics and they are supposed to handle common risks. Since they use static modelling, their estimates are unlikely to be effective in a general scenario. The USEGESOFT tool uses dynamic modelling approaches like System Dynamics and Bayesian Networks which can simulate the temporal behaviour of a project; so, it is possible to make effective decisions as regards to possible risks.

Project reporting tools should allow project managers to understand data recorded in a user friendly way. The USEGESOFT tool only includes the Earned Value technique (Christensen, 1998) as reporting tool but its integration with spreadsheets and statistical packages is feasible.

Process Support Systems (PSSs), also called WFMSs (WorkFlow Management Systems), help to synchronise and coordinate a project. PSS should provide the following capabilities (ESA, 1995):

- Instantiation of the process model.
- Enactment of each process model instance.
- Viewing of the process model instance status.

- Interfaces between the process model instance and the actors.
- Interfaces to project planning tools.

A process modelling tools should (ESA, 1995):

- Support the definition of procedures and processes.
- Contain a library of process templates.
- Make the process model available to a process support tools.

The USEGESOFT tool is able to model software processes using a graphical environment and provides mechanisms to specify, store and display all types of information which can be associated with a process or subprocess. However, at this stage, it does not provide any integration with process support tools; i.e., a process engine capable of driving development activities.

There are tools like SQUID and MiniSQUID (refer to Section 4.2) which provide environments for quality planning, control and evaluation. MiniSQUID provides a robust data model framework to store datasets with a lot of semantic information so that data obtained from them can be reliable and trustworthy. At this stage, while storing metric data in databases, we do not store much semantic information, and therefore our datasets are not as rigorous as we expect them to be. One of our future works is to use the standard proposed by Kitchenham *et al* (2001) for storing and analysing datasets. Tools like SQUID and MiniSQUID mostly concentrate on product metrics; however, the USEGESOFT gives equal importance to process metrics.

Product Metrics tools are used to collect information in an automatic manner from the different artefacts generated during the software development. The general approach is to parse code files to generate metrics such as lines of code, cyclomatic complexity etc. or other object-oriented metrics. The USEGESOFT, at this stage, does not provide facilities to collect such product metrics but it supports their storage and analysis.

Table 4.4 summarises the specifications of the USEGESOFT tool, the types of tools that are able to handle such requirements and the functionalities provided by the USEGESOFT to do so.

	Activities	*Type of tool that handle the activities*	*USEGESOFT functionalities to handle the activities*
Project Planning	Define project entities: work packages, artefacts, resources and their availability	Project Planning Tools	Core USEGESOFT
	Define quality models	Quality Tools	TGPM based tool support
	Conduct a risk assessment for the project	Risk Analysis Tools	BBNs
	Define duration and relationships among work packages (predecessors and successors tasks) and allocating resources to work packages	Project Planning Tools	Core USEGESOFT PERT method Gantt charts
	Calculate initial estimates	Estimating Tools	COCOMO and Mark II BBNs System Dynamics
	Plan report	Project Reporting Tools	Visualisation techniques ; Earned Value technique, Gantt PERT
Project Execution	Project data collection	Project Planning Tools	Core USEGESOFT
	Quality Data collection	Metrics tools	TGPM based tool support
	Project Tracking	Project Planning Tools	Core USEGESOFT
	Planning adjustment	Project Planning Tools	Core USEGESOFT
Project Wrap-up	Metrics updates	Project Metrics tools	Core USEGESOFT
	Final report	Project Reporting Tools	Visualisation techniques: Earn Value technique, Gantt PERT
Organisational	Process Assessment Process Improvement	Quality Tools Process databases Process Support Tools	Project Database Information can be used in GQM analysis

Table 4.4 Summary per Activities

In the following, we will outline the relevance of the USEGESOFT tool in relation to the requirements of a project manager.

1. A project manager needs to address to and interfere with a multitude of managerial activities during software development. A project manager requires tool support to deal with each individual activity (such as cost estimation, quality estimation etc). The SEGESOFT tool integrates many of these individual tools in a single environment; therefore, a project manager through a common interface can analyse or obtain information about an individual task concerning projects,

present or past. Further, this integrated environment makes it easy to produce collated reports which range over multiple managerial tasks; for example, we can assess the quality of a product, and at the same time we can find out possible improvement actions which can enhance the product quality if there is a deficiency (discussed in Chapter 6).

2. Each project management activity may have a lot of information associated with it. The USEGESOFT tool intends to handle information of all such activities in a common database.
3. An integrated database facilitates the use of data mining techniques; however, excepting the construction of BBNs from past project data (discussed in Chapter 6), we have not addressed such techniques in this thesis.

4.9 USEGESOFT in relation to the CMM

In this section we discuss CMM levels and where the project management techniques of USEGESOFT stand in relation to them.

Organisations at *Level 1* do not have any measurement program; therefore, USEGESOFT is only relevant for higher CMM levels.

At *Level 2*, a project is executed according to a project plan. An office package using spreadsheets can be sufficient as this level. However, the core USEGESOFT tool can help with software estimates and tracking of the software project.

At *Level 3*, the KPAs are mostly related to managerial issues and organisation-wide standards. The USEGESOFT is very relevant here since it assumes that quality models are based on TGPM and data are collected in a standard way throughout the organisation. Moreover, data collection is integrated in the development process and at this level.

At *Level 4*, an organisational measurement program is established for measuring the quality of the products and processes. The USEGESOFT can be used to store such information in a consistent manner since data collection is done in relation to the TGPM; i.e. it is easy to know which product data is related to which process data. It is at this level, where all the project management techniques like BBNs, System Dynamics etc. attain their fullest capabilities.

At *Level 5*, measurement practices can be improved by automating the collection of software metrics. Analytical tools must be used to carry out continuous process improvement actions. The data stored in USEGESOFT in conjunction with the dynamic techniques discussed in this chapter can be used for continuous process improvement following methodologies such as GQM and **ami**. Since TGPM is the guiding principle behind USEGESOFT, the relation between process quality and product quality can be easily related from the collected data. Therefore product focused process improvement can be implemented easily (Satpathy and Harrison, 2002).

4.10 Conclusions

The use of quantitative information tends to promote a more scientific approach to project management, and use of project data can have considerable impact on the outcome of the project. We have developed a proof of concept environment that allows us to manage a project data and associated metrics and show how recent techniques can generate valuable information which can guide a project effectively.

Quality aspects of the tool are based on the TGPM, which is instantiated to obtain process models for individual processes. The amount of information that needs to be stored with processes could be very high. USEGESOFT is a tool that provides a framework for storing various artefacts associated with the instantiated process and quality models. We have also discussed how different types of information can be stored and how, in response to a specific requirement, relevant information can be retrieved to make inferences. In addition, we have compared the features of USEGESOFT with those of various specific tools (not necessarily project management tools), and further, we have analysed how the functionalities of the USEGESOFT are related to the various levels of the CMM.

Chapter 5

Assessing and Improving Web Processes using the Typed Generic Process Model

5.1 Introduction

In Chapters 2 and 3, we discussed about the problems surrounding the development of Web applications, and the design of a Parallel Framework for modular development of Web applications. The management and evaluation of Web processes and applications requires tool support. In Chapter 4, we discussed the development of a project management tool which can support many of the managerial activities staring from estimation up to quality assessment. Although this tool can be used during the development of any kind of applications, in particular, it could also be used for Web processes. In the present Chapter, we will focus on the assessment and improvement of Web processes.

There are numerous reasons for improving Web applications. From the product point of view, usability, reliability etc. have room for improvement. From the process point of view, time to market, maintainability etc. need also to be improved. An improvement action is usually preceded by an assessment phase. Process assessment and process improvement are the means to develop quality applications within time and budget. Our aim is to perform process assessment and process improvement of Web processes using techniques from empirical software engineering such as quality models (Rodriguez *et al*, 2002).

Among the models that are available for process assessment and improvement, the most influential ones include the CMM (Humphrey, 1992), ISO 12207 (ISO, 1995), ISO 9000 (ISO, 1994), the BOOTSTRAP model (Kuvaja, 1994), and the ISO 15504 (SPICE) (ISO, 1998). These process models are high level and are not specific enough to cater to the needs of each of the individual processes since: (i) the nature of the processes vary widely; (ii) most of the models are more oriented towards enhancing the maturity of an organisation and take a monolithic view of the overall development process; and finally (iii) the process models, while emphasising process activities, often put too little importance on the products which are the results of the process activities. Product quality aspects are usually addressed by models such as ISO 9126 (ISO, 1991), FURPS+ model (Grady and Caswell, 1987) etc. However, how such product models are related to the above process models, has not been properly addressed. A process model should aim to increase the quality of the products they produce; however, the relationship between the above models and product quality is far from clear. In this chapter, we will use the Typed Generic Process Model (TGPM) (Satpathy *et al*, 2000) as our quality framework; TGPM is used since it makes the relationship between process and product attributes explicit.

5.2 Related Work

The most common approach to evaluation of Web applications is to focus on usability (Nielsen, 2000). For example, Garzotto *et al* (1999) decompose usability into learnability, comprehensibility, memorability, handling ability and niceness. In turn, they decomposed these into other lower level factors. Olsina *et al* (1999) have defined a more general approach defining a tree of quality attributes based on the ISO 9126 standard (ISO, 1991).

Fewer attempts have been made to evaluate Web processes; most notable is the work of Lowe *et al* (1999) and Christodoulou *et al* (1998). Christodoulou *et al* compare various hypermedia methodologies and systems, and suggest which methodology is more suitable for certain type of applications. Lowe *et al* define a series of abstract tasks which state what process entities are to be assessed, what aspects of the development process are involved and what activities need to be performed. These abstract tasks are defined in a template. Lowe (1999) also proposes a reference model for identifying elements (resource, activities and artefacts) in various development process models in order to provide a common terminology. Mendes *et al* (1999) have carried out some empirical studies to estimate Web authoring effort.

Bazzana and Fagnoni (1999) describe PI3 (Process Improvement In Internet service providing), a process improvement plan for an IT company. They used CMM to improve key process areas (KPA) in several issues: technical, business, organisational and cultural. They describe how a company at CMM level 1 can address improvement plans in accordance with business objectives. They defined the required status at the end of the improvement program as follows:

(i) Definition of processes and adoption of tools for the KPAs of testing and configuration management

(ii) Adoption of defined practices and of the tools in the daily routine work of projects

(iii) Increased maturity of the measurement systems, tracking the most relevant indicators for the business of the company

(iv) Initial formalisation of the experiences gained by means of standard operating procedures constituting an initial kernel of the company QMS

At the end of the process, they achieved a maturity level of 2, where the Key Process Areas (KPA) are requirements management, software configuration management, software project planning, software project tracking and software quality assurance.

The GQM method of Basili *et al* (1994) is often used to make a measurement plan for assessing the quality of products and processes, details of which has already been discussed in Chapter 4.

We will also use the TGPM as our quality model for process assessing and improving Web processes. Details of this model have been discussed in Chapter 4. It is to be noted that a process is always performed within the context of a process environment (Bazzana and Pioti, 1999). The TGPM assumes that a process has a well defined structural representation within the environment; this we have called the 'internal process model' in Chapter 4. The TGPM (i.e. the instantiated models obtained from TGPM) is always viewed in conjunction with such an internal process model. This combination of the TGPM and the internal process structure takes into account of the following process attributes which have significance in the CMM:

1. Standard software Process: the internal process model has a clear documentation of the internal details of a process. The specifications of the process steps of an atomic process and the intermediate products including the guidelines as to how to perform the process steps conforms to some of the requirements of CMM levels 2 and 3.
2. Process assessment and improvement are performed on the basis of quantifiable metrics. The internal process model also documents what metrics are to be measured, at what point, by whom and how. The metrics that need to be measured are obtained from the instantiated TGPM. This addresses some of the issues of CMM level 4.

Note that at this stage, our approach to assessment and improvement of Web processes is more in line of a proof of concept rather than a robust assessment or improvement framework. Building a robust framework is a part of our future work.

5.3 Evaluation of Web Processes

Assessment and improvement of a process in general and a Web process in particular is based on a measurement framework which usually uses a quality model. For our purpose, we will use TGPM (Satpathy *et al*, 2000). First, TGPM needs to be instantiated to generate a customised process model which can be used in the measurement framework. The customised model helps us to identify relevant metrics for assessment or improvement purposes. The whole process consists of the following four steps:

(i) *Internal Process Model:* The process under consideration must have a well defined internal process model (i.e. the process steps, inputs, outputs and intermediate products) in place. The internal process model elucidates the internal structure of a process.

(ii) *Upgrade the TGPM:* To perform this task, we may need to add or remove new subfactors which are of importance to Web applications to make a generic model for Web processes.

(iii) *Instantiation:* Instantiate the upgraded generic model to the process concerned.

(iv) Identify the metrics associated with the subfactors of the instantiated model. Not all of them will be necessary. The measurement framework will select a subset of these metrics for analysis.

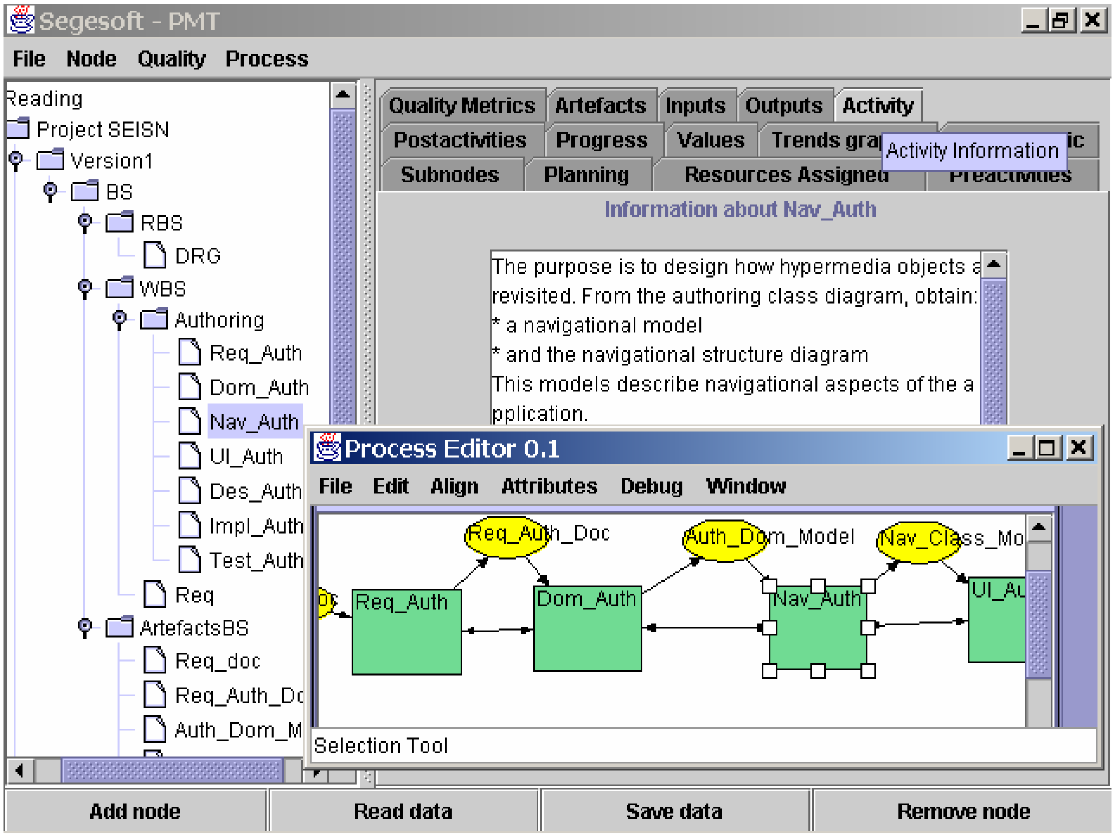
In Chapter 3, we classified the Web development process into two almost independent sub-processes: the authoring process (AUTH process) and the process of developing the infrastructure (INF process). Figure 3.1 in Chapter 3 summarises the main phases of the AUTH as well as the INF processes. Assessment of Web process means the assessments of the AUTH and the INF processes. Since, the assessment of INF process amounts to the assessment of conventional development processes, in this chapter we will only concentrate on the AUTH process.

The internal structure of each of these phases can be defined as per the guidelines of the internal process model of Chapter 3. For instance, Table 5.1 shows the internal structure of the subprocess Nav_{auth} in which we treat Nav_{auth} as an atomic process, and some of the guidelines of this atomic process have also been shown in the table.

Nav$_{auth}$	
Goal	This process takes authoring class diagram and produces the navigational structure model. It involves two steps.
Step 1	Create the Navigational class model from the authoring class diagram following the approach by Baumeister *et al* (1999) (combining or dividing authoring domain classes and creating relationships with *«link»* as stereotype).
Step 2	Enrich the navigational class with menus, indexes, guided tours and contexts using the approach by Baumeister *et al* (1999) to obtain the navigational structure model.

Table 5.1 Process Steps of the Atomic Process Nav $_{auth}$

Figure 5.1 shows the snapshot of the USEGESOFT tool showing the Nav$_{auth}$ subprocess.



The purpose is to design how hypermedia objects a
revisited. From the authoring class diagram, obtain:
* a navigational model
* and the navigational structure diagram
This models describe navigational aspects of the a
pplication.

Figure 5.1 Internal Process Structure of the Navigational

We will now instantiate the TGPM to generate the customised model for the AUTH process, i.e. we need to customise the definitions of the subfactors of the TGPM.

Appendix B shows such customised definitions. By now, we have performed the steps (ii) and (iii) of the measurement framework.

FACTOR	***SUB-FACTORS***	***METRICS (Examples)***
Functionality	Compliance	Strict use of the HTML/XML standards (Yes/No) No. of deviations from prescribed standard
	Completeness	Trend of missing features in the content No. of Use Cases implemented vs. planed over a no. of projects
	Generality	Additional features in content that the AUTH process supports No. of different types of authoring languages (HTML, XML) the process can use No. of different types of links the process can use
	Consistency	Average number of structural inconsistencies (over *n* projects)
	Correctness	Average no. of mismatches between the design and the content
	Interoperability	No. of different types of systems to which the contents can be linked
Usability	Understandability	Effort to understand the AUTH process Effort to understand the products (presence of On-line help, manuals etc) No. of search related problems
	Learnability	Effort to do authoring (No. of hours/ page) Level of training required to do authoring
Efficiency and Estimation	Cycle time	Estimated cycle time vs. actual cycle Integration frequency (daily, weekly etc.)
	Cost/Effort	Average of the planned development time vs. actual time HTML pages/person month
	Complexity estimation	Estimated complexity vs. actual complexity (Compactness, Stratum (Botafogo *et al*, 1992), Graph complexity etc.) Link complexity Psychological complexity (Subjective assessment)
	Schedule/Priority	No. of times schedules are not fulfilled
	Resource usage	Estimation resources vs. actual resources (people, tools)
Visibility and Control	Automatic checks and feedback	Interactions (e.g. no. of link failures, dangling links, inland pages, etc)
	Traceability	Subjective assessment of difficulty in tracing between analysis, design, and implementation
	Progress monitoring	No. of milestones No. of times milestones are not met No. and kind of approaches to monitor progress
	Improvement measures	No. and type of improvement actions
Reliability	Failure frequency	MTBF (Mean Time Between Failures) MTTF (mean time to failure) Average no. of times failures encountered during authoring (tool crashing etc.)
Scalability	Scalability	Time, effort, cost against projects of varying sizes
Maintainability	Reusability	Average no. of modules reused vs. total no. of modules
	Testability	Average no. of faults discovered during testing/Totals no. faults over a number of projects Average no. of faults/lines of HTML/XML)
	Portability	No. of browsers and versions considered (e.g. palm appliances)
	Analysability	Effort to analyse causes of an authoring defect (e.g. broken links etc).
	Modifiability	Effort to correct an authoring defect
	Defect trend	Average no. of defects in AUTH process Average no. of incident reports Average no. of change requests
Customer satisfaction	Communicativeness	Average no. of customer complaints

Table 5.2 Quality Tree of the AUTH Process

Table 5.2 represents the quality of the AUTH process; the first column shows the various factors of the AUTH process, the second column shows the sub-factors associated with the factors, and the third column shows the major metrics which could be associated with the corresponding sub-factors. In the third column, average means the average value taken over *n* projects. TGPM does not have some sub-factors which could be very important in relation to Web applications. One such factor is customer satisfaction which we have added to Table 5.2. Similarly, we have removed the Safety sub-factor because it is not relevant to the AUTH process (which could be of importance to the INF process). With these minor modifications, the step (iv) of the measurement framework is complete in relation to the AUTH process.

The attributes of Table 5.2 needs further explanation. Dromey's generic quality framework (discussed in Section 4.4.2.3, Chapter 4) decomposes a software product into a hierarchy of structures, and the entities of a structure have some quality-carrying properties, and by addressing these properties, we can ensure that the product satisfies the desired high-level quality characteristics. The responsibility of satisfying the quality-carrying properties of structure entities rests on a matured process. The TGPM, in a similar manner, emphasises that process quality attributes cannot be considered in isolation; processes can only enhance the quality of a product. This is the idea behind considering process quality attributes from a dual perspective (discussed in Section 4.4.2.1, Chapter 4). Let us consider the attribute 'Process defects' (*Defect Trend*). The TGPM defines this attribute as follows:

(i) *The average number of defects that are observed in the AUTH process itself (defects linked to the process − during or after process execution)*

(ii) *The degree to which the AUTH process detects defects or deficiencies in the* Req_{auth} *so that defects in the hypermedia content are minimal.*

The first definition tells that the process itself should be defect free, and the second tells that the process should make its output product defect free. Let us assume that the

Authoring process has produced *n* products. Let the number of defects in each product be $def_{i.}$ Then the average number of defects of the past products ($Def_{average}$) of the Authoring process can be given by the expression:

$$Def_{average} = \frac{1}{n}\sum_{i=1}^{n} Def_i$$

If the quanity $Def_{average}$ is low, it implies that the Authoring process is good in relation to the product aspect of the attribute 'Process Defect'. In this sense, $Def_{average}$ is a process attribute. Furthermore, we should be careful while choosing the number *n* in the above expression. Let us assume that initially the process was not good in terms of ensuring product quality and then it was improved. This improvement can only be noticed in the products developed after the improvement. Therefore, what should be the value of *n* depends on the context of the process.

5.4 Interpretation of Metric Measures

In order to interpret the metrics collected, we may need to consider several metrics at a time and the dependency relationships between them. One way is to assign weights to each of the individual metrics and to come up with a numeric value. Another way is to use a dynamic model like Bayesian networks (Jensen, 1996) which takes into account the metric measures as well as many context factors to arrive at a conclusion. For instance, although it may be useful to know the number of defects in the authoring of a Web application, it is more useful if we consider its complexity (e.g., compactness, stratum) and effort metrics (e.g., HTML pages/person-month). Bayesian networks also can handle uncertainties and inaccuracy in measures. Bayesian Networks will be the subject matter of our next chapter.

Validation of the metric measures is also an important issue. We are not addressing this issue in this thesis; however, the approach taken by the SQUID and MiniSQUID projects (Kitchenham *et al*, 2001) is a good choice. In them, each metric has a target value which is obtained by referring to a historical database and comparing the metrics of the current project with the metrics of the similar projects in the past. The measures values when seen within the context of past similar projects, gives us a confidence in them.

5.5 A GQM based Measurement Framework

Our measurement framework uses the Goal-Question Metric (GQM) (Basili *et al*, 1994) method. The GQM method helps us to identify, focus, document and analyse a small but relevant number of measurements that need to be collected on a regular basis. The method is composed of 3 levels. First, the conceptual level identifies a quality and/or productivity goal, i.e., the purpose of measurement in relation to an entity (product, process or resources) from a specific point of view (manager, developer, maintainer etc.). A goal can be an assessment goal or it can be an improvement goal. The second level, called the operational level, breaks down the goal by means of questions that characterise the entity. Finally, the quantitative level specifies metrics that need to be collected in order to answer those questions. All three steps are summarised in a tree structure, called the goal-tree.

This measurement framework offers a systematic way of creating a goal tree in three steps. First, we need to identify what factor(s) of our instantiated model concerns the goal. Then from the instantiation of the quality model (Appendix B) or from Table 5.2, we identify the relevant subfactors. The definitions of the subfactors will help in generating the questions of the goal-tree. Finally, from Table 5.2, we can obtain the appropriate metrics which are essentially the metrics associated with the subfactors in the table. In relation to a given goal (i.e. some factor(s)), we need to identify a subset of the associated metrics (i.e. the metrics associated with the subfactors of the given factor(s)). It is worth noting that because of the duality of the sub-factors, the definitions cover both the process as well as the product aspects of the goal. The quality tree as represented by Table 5.2 helps us in building such goal trees in a systematic manner (Fugetta *et al*, 1998).

As an example, we will create two assessment goals: (i) assess the estimation of the AUTH process from the viewpoint of the manager; and (ii) assess the maintainability of the AUTH process from the viewpoint of the manager/maintainer. Table 5.3 and 5.4 summarise the corresponding goal-trees.

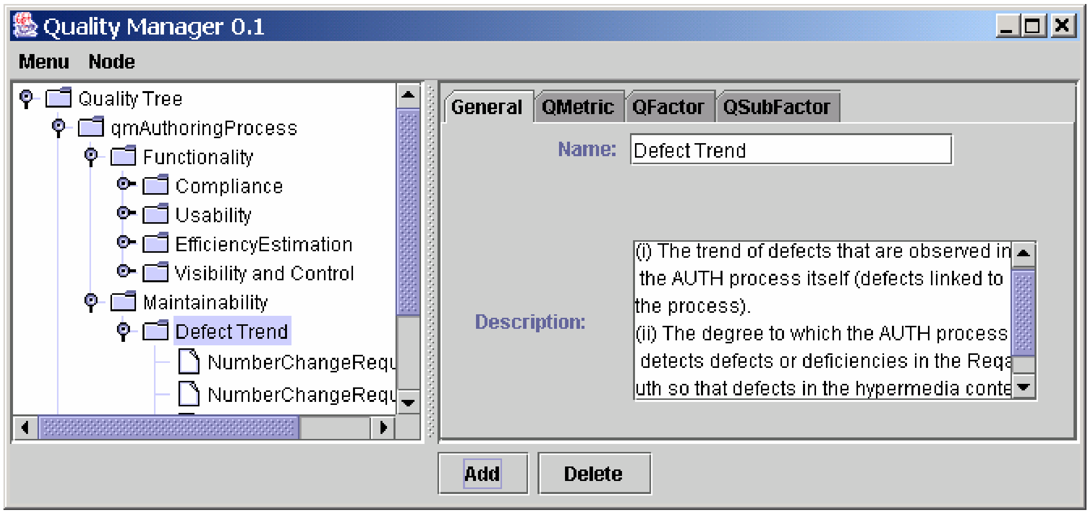
Goal	*Object of Study:* ***AUTH process*** *Purpose:* ***To assess*** *Focus: Efficiency and Estimation* *Point of view:* ***Manager***
Q1	What is the underlying authoring language?
M1.1	Note the languages used (HTML, JavaScript, XML, Xlink etc.).
Q2	Does the estimated complexity match the actual complexity? What are the techniques and tools used for complexity estimation?
M2.1	Note the difference between estimated/ actual complexity
M2.2	Note the technique. For instance, Stratum, Compactness, Depth, Imbalance (Botafogo *et al*, 1992)
Q3	Does the estimated authoring cycle time match the actual cycle time? What is the technique used to estimate the cycle time?
M3.1	Note the difference (actual cycle time/ estimated cycle time)
M3.2	Note the technique/ tool support
M3.3	Integration frequency (how often the modules of the application are integrated)
M3.2	Average cycle time
Q4	Does the estimated effort match the actual effort? What is the technique to estimate the effort?
M4.1	Note the difference (actual effort/ estimated effort)
M4.2	Note the technique/ tool support
Q5	Does the estimated schedule match the actual schedules? What is the technique to estimate the schedule?
M5.1	No. of times schedules are not fulfilled
M5.2	Note the technique/tool for estimating the schedules
Q6	Do the estimated resources match the actual resources? What is the technique to estimate the resources?
M6.1	Note the differences (estimated no. of people/ actual no. of people)
M6.2	Note the technique/ tool for estimating the resources

Table 5.3 GQM to Assess the Estimates in the Authoring Process

Goal	*Object of Study:* ***AUTH process*** *Purpose:* ***To assess*** *Focus: Maintenance* *Point of view: Manager/Maintainer*
Q1	Is the current analysability performance acceptable? Is the tool-support adequate?
M1.1	Effort to analyse the cause of an authoring error (link, misspelling etc.)
M1.2	Efficiency of the tool support (subjective assessment)
Q2	Is the current modifiability performance acceptable?
M2.1	Average time for addressing a change request
M2.2	Average time for addressing an incident request
Q3	Are the contents free of defects? Is the process defect-free itself?
M3.1	Trend of defects in the contents after release
M3.2	Trend of process defects and their cause (e.g. a tool crashing)
M3.3	No. of change requests after release for each project
Q4	Does authoring addresses portability issue adequately?
M4.1	No. of browsers that the process addresses
M4.2	No. of versions that the process addresses
Q5	Is authoring testing adequate?
5.1	No. of defects discovered during testing/ Total number of estimated faults
Q6	Is the AUTH process re-usable?
M6.1	Are the same AUTH process steps used for all projects (Yes/No)

Table 5.4 GQM for Maintainability of the Authoring Process

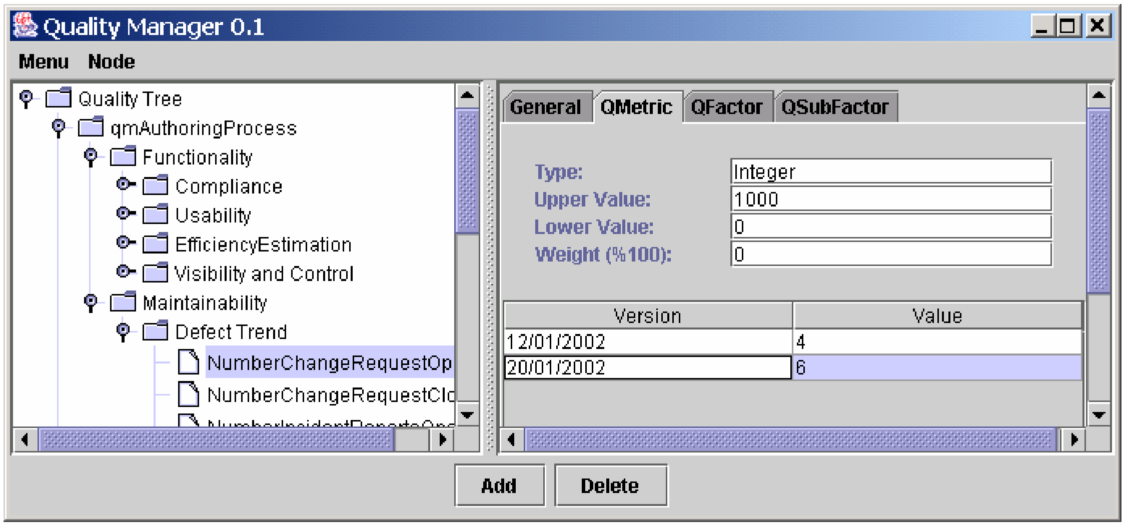
Figure 5.2 and Figure 5.3 are screenshots of the project management tool supporting the customised measurement framework. For example, Figure 5.2 shows the dual definition of the Defect Trend subfactor and Figure 5.3 shows quality metrics associated with the same subfactor.



Name: Defect Trend

Description: (i) The trend of defects that are observed in the AUTH process itself (defects linked to the process).
(ii) The degree to which the AUTH process detects defects or deficiencies in the Reqa uth so that defects in the hypermedia conte

Figure 5.2 Quality Tree Showing the Defect Trend Definition



Type: Integer
Upper Value: 1000
Lower Value: 0
Weight (%100): 0

Version	Value
12/01/2002	4
20/01/2002	6

Figure 5.3 Metrics related with the Defect Trend Subfactor

Usually, assessment is done to characterise the current status of a process. If the assessment reveals that a certain aspect of the process is not satisfactory, then this needs to be improved. Following the **ami** terminology (Kuntzmann-Combelles *et al*, 1996), this is done by defining an improvement or *change* goal. A goal-tree can be constructed in relation to this change goal by following the GQM method. From this goal-tree, improvement actions can be inferred, which may correct the unsatisfactory aspect(s) of the process. For example, suppose the assessment finds that the actual cycle time of the AUTH process is too high as compared to the estimated cycle time. Then the improvement goal can be: *improve the estimated cycle time of the AUTH process from the viewpoint of the manager*. Then we need to find out what improvement actions need to be taken so that cycle time could be improved. The goal-tree for this improvement goal is shown in Figure 5.3 (Debou, 1999). From this goal-tree, we obtain the improvement action which is then integrated to the AUTH process, and then another cycle of assessment follows. This new assessment verifies whether the improved AUTH process produced the desired result.

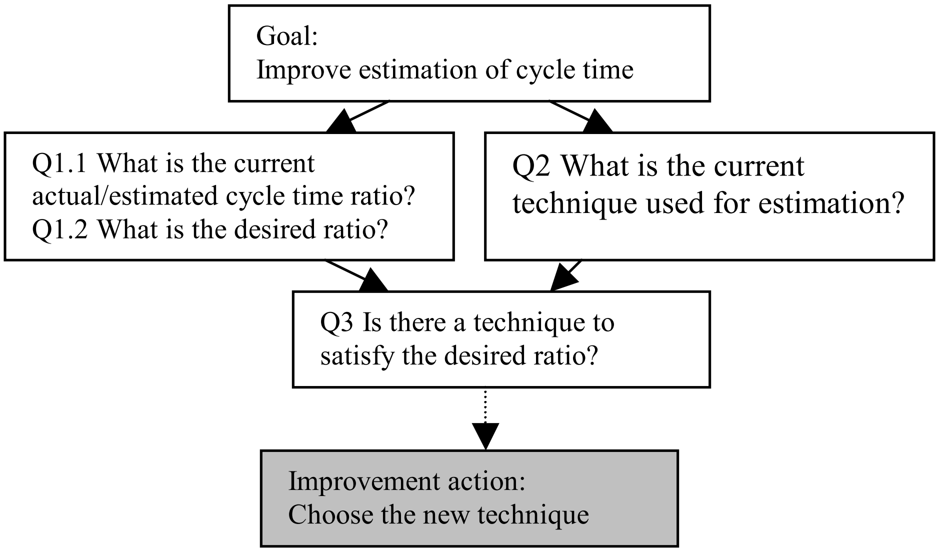

Goal:
Improve estimation of cycle time
Q1.1 What is the current actual/estimated cycle time ratio?
Q1.2 What is the desired ratio?
Q2 What is the current technique used for estimation?
Q3 Is there a technique to satisfy the desired ratio?
Improvement action:
Choose the new technique

Figure 5.4 GQM Tree for an Improvement Goal

5.6 Conclusions

TGPM provided a systematic way for creating a customised measurement framework for any software process. In this chapter we have demonstrated the customisation of the TGPM to obtain a measurement framework for the authoring process. We have presented the step by step approach of the customisation process. This customised measurement framework forms the basis of assessment and improvement of the authoring process. In this chapter, we have not considered the customisation of TGPM to the INF process, since such an activity resembles the customisation in relation to any conventional development process.

We have also used the GQM method to make measurement plans. The GQM method involves building goal trees in a systematic manner by using the customised framework of the authoring process. For assessing the authoring process or any of its subprocesses, a goal tree is created to obtain the necessary metrics to be collected. Such metrics data need to be analysed to find the deficiencies in the process. Then improvement goals are obtained which could remove the deficiencies. A new GQM tree is then built to obtain improvement actions. We have demonstrated such approaches through examples.

In the next Chapter, we will introduce Bayesian networks as our selected tool for predicting, monitoring, controlling and assessing software development. Bayesian networks, in addition to possessing many interesting properties, allow us to analyse a group of metric measures within a problem context.

Chapter 6
An Investigation of the Use of BBNs for Project Management

6.1 Introduction

Dynamic models such as system dynamics and Bayesian Belief Networks (BBNs) are becoming increasingly popular among the software engineering research community as they are capable of providing better solutions to some of the problems in the area of estimation and prediction when compared with the solutions provided by the traditional static models (Fenton and Neil, 1999b). Traditional models such as COCOMO or Function Points are usually static in nature in the sense that they are based on some mathematical equations. However, software applications even in the same domain do not have uniform characteristics, and further capabilities of organisations are also not uniform. Dynamic models consider many such context related factors and therefore estimations or predictions obtained from them are likely to be more accurate. BBNs are one such type of dynamic model which are used for making estimations and predictions. In this chapter, we will focus on the use of BBNs in project management.

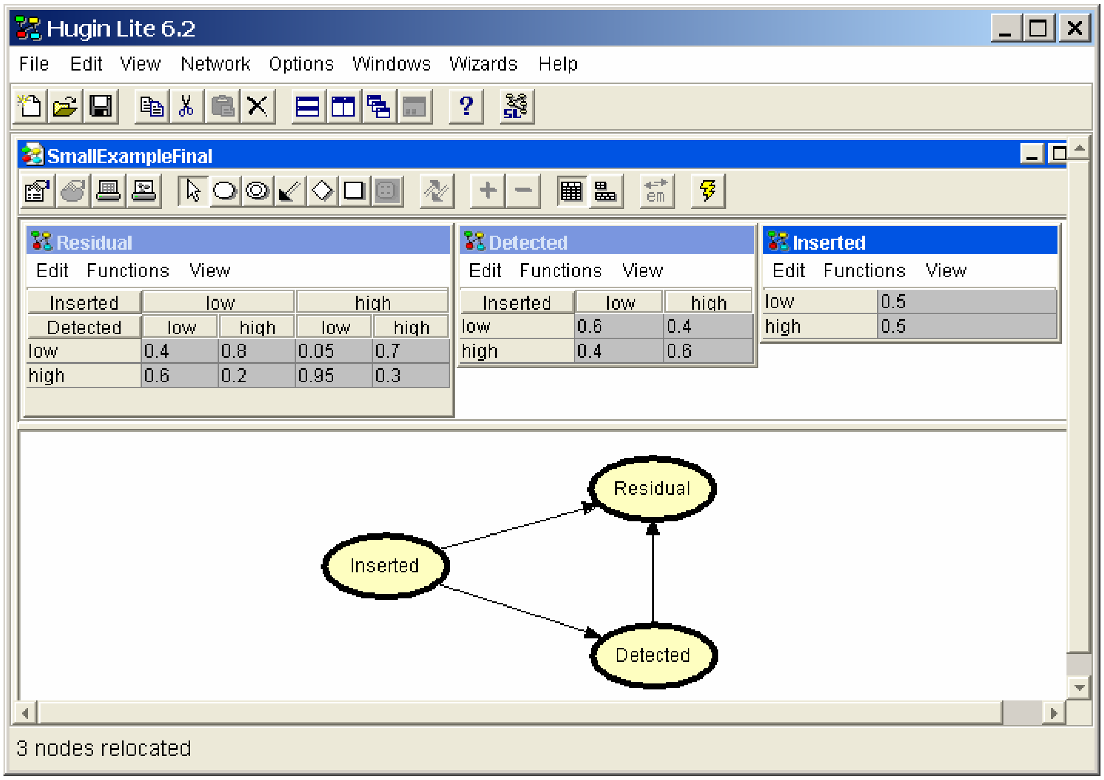
As we have seen in Chapters 4 and 5, we need to collect a lot of data relating to product and process quality metrics. However, a metric may be related to many other metrics, and such relationships need to be taken into account when we analyse the data that we measure. Sometimes relationships are direct and sometimes they are transitive. BBNs model such complex relationships in a graphical framework, and thereby offer a degree of visibility to such relationships.

Creation of a BBN to reflect the domain of a software engineering problem is a challenging task. They are usually obtained from expert judgement or from past project data using data mining techniques, or by using both the approaches. There are many semi-automatic approaches available in literature for generating a BBN from project data. In this chapter, we will discuss the step by step construction of a BBN using the data set from a Web-related project involving 37 Web sites. In addition, we will discuss a use of BBNs in process improvement.

6.1.1 Bayesian Belief Networks

In Chapter 4, we have given a very brief introduction to BBNs. We will now make the definition of a BBN more formal. A BBN (Krause, 1998; Jensen, 1996) is a Directed Acyclic Graph (DAG) which is composed of nodes and directed arcs. The nodes represent domain variables $X_1, X_2,..., X_n$ and the directed arcs represent the causal relationships between the variables. BBNs assume variable independence, i.e., a variable depends on its parents only and on no other variable in the network. A node has a Node Probability Table (NPT) which stores the conditional probability values of the associated variable. Let us assume that the node representing variable X which can have states $x_1, x_2, ..., x_k$ and $P(x_j)$ is the probability of X satisfying the state xj. Then the NPT stores the value: $P(x_j \mid Parents\ (X))$ for each possible state x_j of the variable X. If X is a root, i.e. a node without parents, then NPT stores the marginal probability distributions i.e. the value of $P(x_j)$. for each $j = 1...k$. It is worth noting that nodes are usually discrete represented by a set of mutually exclusive and exhaustive states (although they can also be continuous and use parametric probability models such as the normal distribution).

Figure 6.1 represents a simple BBN for defect estimation. The variable *residual defects* (defects that remain after delivery) depends on the variable *defects introduced* in the source code and the variable *defects detected* during testing. Each such variable can be in either of the states “low” and “high”. The NPTs for the nodes are as given in the figure. For instance, when defects *inserted* is high and defects *detected* is low, then the probability that the *residual* defect is high is 0.95. Other entries in the NPTs can be similarly explained. Note that the node defects inserted has no parents; so, its NPT stores only the marginal probabilities.



Residual

Inserted	low		high	
Detected	low	high	low	high
low	0.4	0.8	0.05	0.7
high	0.6	0.2	0.95	0.3

Detected

Inserted	low	high
low	0.6	0.4
high	0.4	0.6

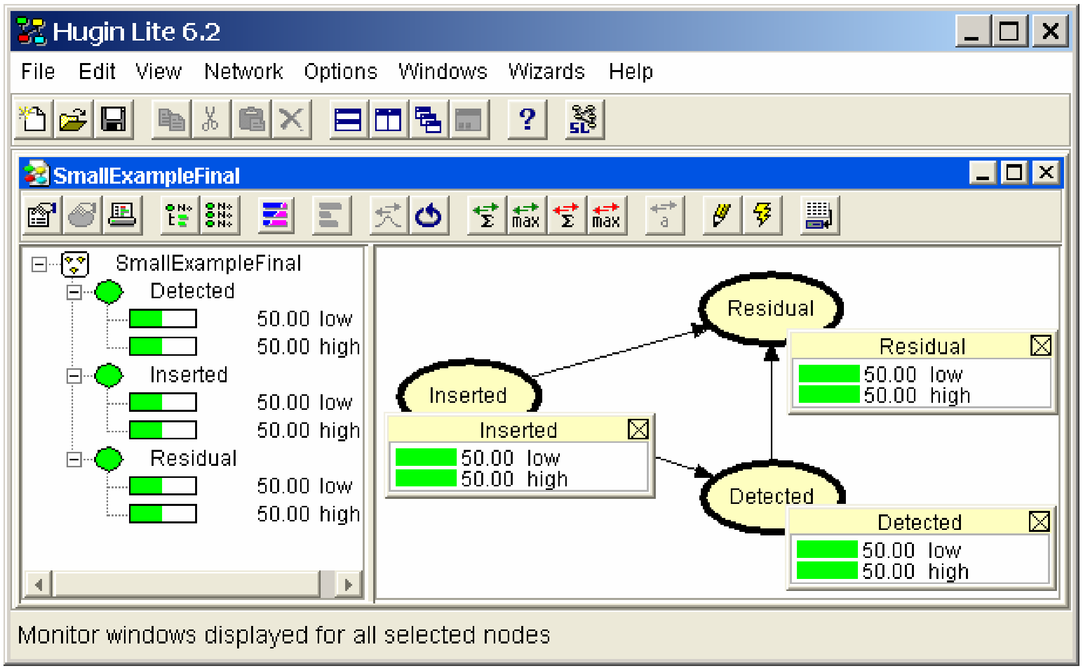
Inserted

low	0.5
high	0.5

3 nodes relocated

Figure 6.1 Example of BBN for Estimates of Residual Defects

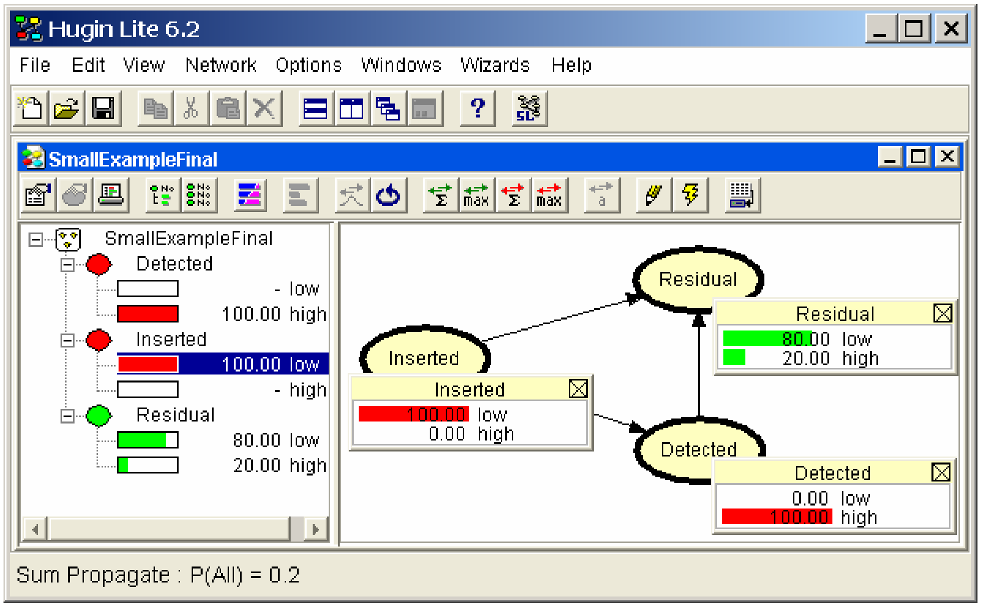
Tools supporting BBNs usually provide two modes of operation: (i) *edit mode* allowing creation of networks and probability tables, and (ii) *query mode* in which evidences are provided and the resulting probabilities are visualised (the evidence results in probabilities getting propagated in the network). For instance, if we have no additional information about a project, Figure 6.2 shows the default probability values of each variable. Therefore, we cannot make any meaningful inferences.



Monitor windows displayed for all selected nodes

Figure 6.2 Query mode without Observations

However, if we have the additional information that defects *inserted* is low and a high number of defects were *detected* during testing (variable inserted and detected get the values low and high respectively), the BBN predicts that the probability of *residual* defects being low is 0.8 and the probability of being high is 0.2. Figure 6.3 demonstrates such a scenario. Dark bars in the probability windows show where evidences have been entered; and the light grey bars show the impact of the evidence.



Sum Propagate : P(All) = 0.2

Figure 6.3 Run-mode with Observations

Construction of BBNs involve a lot of intellectual activities in the sense of identifying the variables, the relationships between then and obtaining the entries in the NPTs. Therefore, expert guidance along with information from past project data, if available, are used in the generation process. In Section 6.3, we will describe how to automate some of the activities of the process of creating BBNs when data about the concerned variables are available.

6.1.1.1 Inference Using Bayesian Belief Networks

A BBN can be seen as an inference system such that if the state of a certain node is known (evidence or observed fact) its probability table is altered to reflect this knowledge. Such knowledge is then propagated in the network to determine the changed probabilities of other nodes, i.e., the inference mechanism calculates the posterior probability distribution $P^{*}(X \mid E)$ given evidence E (X is the variable under consideration).

The complexity of making inferences using BBNs is known to be a NP-hard problem (Cooper, 1987). However, exact and approximate algorithms have been proposed in the literature to propagate evidences in BBNs very efficiently for some restricted types of BBNs which are sparse in nature. It is believed that for domains in which humans can reason effectively are usually represented by sparse graphs (D'Ambrosio, 1999). Therefore, software engineering problems, especially the ones concerned with project management, can be represented by sparse graphs, and hence tools can do computation over them with reduced complexity.

6.1.1.2 BBN Modelling

There are two issues related to the construction of BBNs. The first one concerns with the structure of a network and the second concerns with determining the probabilities associated with each node in the network. We can obtain such information from domain experts, from past project data or by using a combination of both. The construction procedure involves the following steps:

- First, it is necessary to find out the variables of interest from the problem domain. Domain knowledge should be used to select only the meaningful variables.
- For each such variable, we determine a set of mutually exclusive and exhaustive values (or ranges of values). Each variable becomes a node in the network.
- We then obtain the topology of the network, i.e., causal relationships between the selected variables is determined; the relationships are then modelled as directed edges in the network.
- The last step concerns with the elicitation of probability values for the NPT of each node in the network.

It is usually not easy to build a model in a single sequence; several cycles may be needed to obtain a more accurate network.

6.2 Related Work

6.2.1 BBNs in Software Engineering

Neil and Fenton have used BBNs (1999a) to estimate defect density. They have considered high-level factors like developer skill, and traditional factors such as effort, complexity of the design etc. They have shown how it could be possible to include variables in a software reliability model that correspond to processes as well as product attributes. The authors have also developed, under the SERENE project in the area of safety and risk evaluation, a tool to create a larger network from smaller networks (Neil *et al*, 2000). Fenton *et al.* also carried out the DATUM (Dependability Assessment of safety critical systems through the Unification of Measurable) project in which they have used information from diverse sources which were often uncertain and subjective (Fenton, 1995). They have used BBNs to estimate a system's reliability and safety.

Wooff *et al* (2002) discuss how the Bayesian approach could be applied to the problem of software testing. The authors state that BBNs can be used to drive test design, answer what-if questions, and provide decision support to managers and testers.

Delic et al have used BBNs as an aid for the assessment of software dependability and for the feasibility of their use (Delic *et al*, 1995). Their goal is to validate the probability of failure per demand of the target system. They used historical data for the initial construction of the BBN.

According to Ziv (Ziv and Richardson, 1997), the Maxim of Uncertainty in Software Engineering (MUSE) states that uncertainty is inherent and inevitable in software development process and products. As a consequence software uncertainties should be modelled and managed explicitly and BBNs are the chosen mechanism. MUSE shows a detailed presentation of uncertainty in software testing and lists three possible categories that uncertainty modelling could be used. (i) confirmation of certain characteristics and behaviours of software systems; (ii) evaluate the software qualities, properties, or the design/test processes; and (iii) prediction of the system properties by exploring the uncertainties in the new system requirements and corresponding new designs.

6.2.2 Related Approaches to Estimation and Prediction

Shepperd *et al* (1996) classify estimation and prediction techniques into three main categories: (i) expert judgement; (ii) algorithmic models; and (iii) machine learning.

6.2.2.1 Expert Judgement

Expert judgement is based on experts' previous experience on similar projects. Although it has been formalised using the Delphi's approach (Boehm, 1981) to combine several expert opinions, its major drawback is that the means of deriving estimates are not explicitly defined (black-box) and therefore, not repeatable. Although it is difficult to quantify, expert judgement can be an effective estimating tool on its own or as an adjusting factor for other.

6.2.2.2 Algorithmic models

Algorithmic models represent the relationship between variables, usually effort (e.g. person-month) and one or more of a project's characteristics such as software size and complexity. Such models include COCOMO (Boehm, 1981), SLIM (Putman, 1978) and function points in many flavours such as COSMIC FFP (Abran, 1999), Albretch (Albretch, 1979), Mark II (Symons, 1988) etc. Algorithmic models provide repeatable estimations and are well understood by all stakeholders of software development.

Regression techniques (Briand *et al*, 1999) are a kind of algorithmic techniques which looks for an equational model to fit a set of observed data values. The linear least square method finds the line that minimises the sum of squared errors. A problem with this method is that it assumes a normal distribution and cannot cope well with outliers. Regression analysis can be made more robust for outliers using the least median squares method. Least median square regression analysis is suitable for small data-sets, and it is sensitive to abnormal observations and errors.

6.2.2.3 Machine learning

We briefly summarise here some machine learning techniques and expert systems paradigms used in software engineering.

6.2.2.3.1 Rule-based Systems

A rule-based system (Rich and Knight, 1991) consists of a library of rules of the form:

if (assertion) *then* action.

Such rules are used to elicit information or to take appropriate actions when specific knowledge becomes available. These rules reflect a way to reason about the relationships within the domain. Their main advantage is their simplicity. However, they are mainly appropriate for deterministic problems, which is not usually the case in software engineering. To overcome this problem, rules can also contain a certainty measure in the premises and/or in the conclusions. Fuzzy logic is the most popular scheme for incorporating such uncertainty in rule based systems. An advantage of fuzzy logic is that fuzzy values such as ‘low’, ‘medium’ etc are intuitive and easy to understand.

6.2.2.3.2 Case-based Reasoning

This model intends to mimic the process of an expert making a decision using previous experience (Gray and MacDonell, 1997). Earlier observations are stored, and when faced with a new observation, the closest matching stored values are retrieved. A case-based reasoning system includes a data pre-processor, a similarity function to fetch similar cases, a predictor to estimate output value and a memory updater to store new cases.

6.2.2.3.3 Neural Networks

Neural Networks (NN) are used for pattern recognition and classification (Picton, 2000). The architecture of a NN consists of an input layer, an output layer and possibly several hidden layers in between them; except for the output layer, nodes in a layer are connected to nodes in the succeeding layer. Such connections have weights and the nodes have threshold functions to determine the network behaviour. In software engineering, the input layer may be comprised of attributes such as lines of code, development time etc. and the output nodes could represent attributes such as effort and cost. NNs are trained with past project data, so that when a new project arrives, NNs can estimate the new project attributes according to previous patterns.

6.2.2.3.4 Decision Trees

Decision trees (Breiman *et al*, 1984) are used for predicting or explaining outputs from observations. In such a tree, each node is a leaf indicating a class or an internal decision node that specifies some test to be carried out. If the output values conform to intervals, then the decision trees are called regression trees, whereas if they do correspond to a nominal or ordinal scale they are called classification trees. There are many tree-building algorithms such as C4.5 (Quinlan, 1993) which determine which attributes best classifies the remaining data, and then the tree is constructed iteratively. The main advantage of decision trees is their immediate conversion to rules that can be easily interpreted by decision-makers (Aguilar-Ruiz *et al*, 2002).

6.3 Mining Software Engineering Data using BBNs

In this section, we will show how data mining techniques and BBNs can help to acquire knowledge about the software engineering process of an organisation. We first describe some of the terminology that we will use. *Domain knowledge* refers to the whole information specific to a field. Users may have a limited subset of the whole knowledge, called *background knowledge*. New or discovered knowledge can be obtained in a *deductive* or *inductive* way. Deduction infers information as a logical consequence of the data (e.g. average cost of a project). Induction infers information by generalising the information. The computational techniques and tools designed to support the automation of inductive learning, i.e., extraction of useful knowledge from data, are called machine learning techniques. More recently, terms like *data mining* or *Knowledge Discovery in Databases* (KDD) have come to be used in place of machine learning. In general, KDD techniques try to extract in an automatic way the information useful for decision support or exploration of the data source (Fayyad *et al*, 1996).

Since data may not be organised in a way that facilitates the extraction of useful information, typical KDD processes are composed of the following steps:

(i) Data preparation. The data is formatted in a way that tools can manipulate it.

(ii) Data selection and cleaning. There may be missing, noisy and uncertain data in the raw dataset.

(iii) Data mining. It is in this step when the automated extraction of knowledge from the data is carried out.

(iv) Proper interpretation of the results, including the use of visualisation techniques.

(v) Assimilation of the results.

6.3.1 Learning BBNs From Data

For generating BBNs from past project data some semi-automatic procedures are available in the literature (Castillo *et al*, 1997). The process is composed of two main tasks: (i) induction of the best matching qualitative dependency model from data and prior knowledge; and (ii) estimation of the local probabilities. There are two approaches to learning Bayesian Networks from data (Castillo *et al*, 1997): (i) search and scoring methods and (ii) dependency analysis methods.

In search and scoring methods, the problem is viewed as a search for a structure that fits the data best. Starting with a graph without edges, the algorithm uses a search method to add an edge to the graph. Then the scoring method checks whether the new structure is better than the old one. If so, the edge is kept and a new edge is tried. The algorithm stops when there is no better structure. This problem is known to be an NP-hard problem (Castillo *et al*, 1997), therefore, many of the methods use heuristics to reduce the search space with node ordering as an input. Well-known search and score algorithms include: Chow and Liu (1968) Tree Construction Algorithm, K2 (Cooper and Herskovits, 1992) etc.

Dependency analysis methods follow a different approach. These methods try to find dependencies from the data which in turn are used to infer the structure. Dependency relationships are measured using conditional independency tests (Castillo *et al*, 1997).

6.3.2 The Process of Creating BBNs from Data

The typical steps of a BBN based data mining process are presented in Figure 6.4. We will illustrate this process following an example in the context of estimation in Web Engineering using the dataset provided by Mendes *et al* (2002).

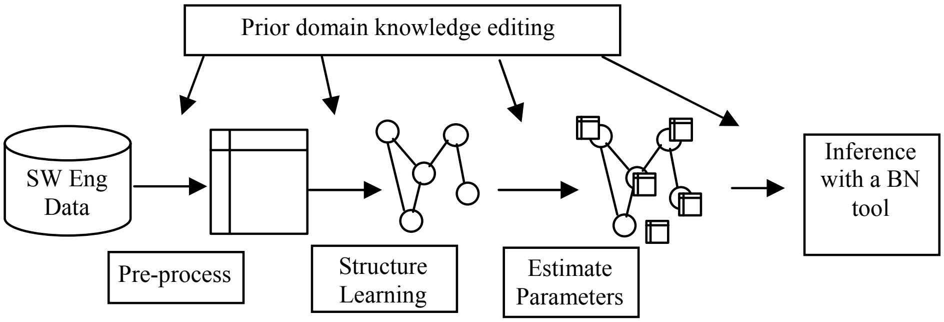


Figure 6.4 Process Steps to Obtain BBNs from Data

The Dataset:

The dataset is composed of 37 Web applications which were created by final year undergraduate students from the University of Auckland attending the Hypermedia and Multimedia Systems course (Mendes *et al*, 2001). The experiment consisted of designing and authoring Web applications using a minimum of 50 documents. A questionnaire was used to measure characteristics of the Web applications developed and the effort involved in designing and authoring those applications. The dataset provided by Mendes *et al* (2001) discarded six questionnaires because they were considered outliers. Subjects were given a training prior to designing and authoring the Web applications, and therefore, subjects' authoring and design experiences were considered similar. All Web applications used a hierarchical structure without any server side component (i.e., infrastructure) and were developed using only simple text or Web editors.

This dataset is shown in Table 6.2. From this dataset, the authors identified 8 variables that characterise a Web hypermedia application and its development process (refer to Table 6.1).

Metric	***Description***
Page Count (PaC)	Number of HTML or SHTML files used in the application
Media Count (MeC)	Number of media files used in the application (not reused)
Program Count (PRC)	Number of JavaScript files and Java applets used in the application
Reused Media Count (RMC)	Number of reused/modified media files (black- or white-box reuse)
Reused Program Count (RPC)	Number of reused/modified programs
Connectivity Density (CoD)	Total number of internal links divided by *Page Count* (students did not used external links, i.e. links to other Web sites)
Total Page Complexity (TPC)	Average number of internal links divided by *Page Count*
Total Effort (TE)	Estimated amount of effort in person hours to design and author the application

Table 6.1 Size and Complexity Measures (Mendes *et al*, 2002)

ID	***TEFFORT***	***Page Count***	***Media Count***	***Program Count***	***Connectivity Density***	***Total Page Complexity***	***Reused Media Count***	***Reused Program Count***
1	79.13	43	.00	.00	8.72	1.18	42	0
2	133.1	53	53.00	1.00	17.73	2.21	53	0
3	145.5	75	21.00	.00	16.85	1.00	64	0
4	135.4	100	2.00	.00	9.02	1.00	0	0
5	128.4	50	82.00	.00	13.90	1.00	27	0
6	106.6	53	11.00	.00	7.58	1.00	57	0
7	100	54	.00	.00	2.57	1.26	54	0
8	112.6	52	36.00	.00	6.36	1.07	43	0
9	101.3	54	26.00	.00	13.20	1.07	27	0
10	87.05	50	13.00	2.00	10.62	1.00	8	0
11	81.54	60	.00	.00	21.43	.28	2	0
12	113.8	51	.00	.00	7.19	1.00	89	0
13	58.36	41	4.00	.00	3.19	1.00	2	0
14	153.8	51	74.00	1.00	21.62	1.94	75	0
15	112	61	8.00	.00	9.00	2.07	50	0
16	122.2	66	.00	.00	2.57	.77	66	0
17	125.1	59	66.00	.00	16.54	1.88	15	0
18	139.8	62	21.00	.00	12.27	1.99	87	0
19	128.5	59	13.00	.00	15.52	2.51	82	0
20	115.5	50	5.00	1.00	12.24	1.00	81	0
21	119.7	53	63.00	.00	23.30	1.11	7	8
22	106.1	53	30.00	3.00	1.69	.17	10	0
23	73.81	55	.00	.00	6.83	.00	0	0
24	147.4	44	126.00	.00	13.95	1.00	30	0
25	152.8	66	28.00	.00	7.21	.98	94	0
26	120	66	27.00	.00	13.56	1.00	31	0
27	73.01	43	.00	.00	8.72	1.19	30	0
28	101.8	53	1.00	.00	1.69	1.05	59	0
29	97.3	56	25.00	.00	2.76	1.75	15	0
30	76.23	53	.00	.00	4.86	.19	10	0
31	137.2	51	20.00	.00	16.67	1.00	112	0
32	117.4	55	25.00	.00	4.32	1.00	57	0
33	60.79	33	16.00	1.00	5.00	1.00	6	0
34	141.4	52	48.00	5.00	16.30	1.85	45	0

Table 6.2 Raw Dataset

Data pre-processing:

In order to create a BBN from data, it needs to be pre-processed. Typically, this process consist of discretising data and dealing with missing values. In our case, we did not have any missing values but it was necessary to discretise most of the variables. We need to decide whether a variable is continuous, the number of intervals and the discretisation method, usually by frequency or range. Naturally, we lose some of the available information in the process.

Table 6.3 partially shows the pre-processed dataset where new columns with discretised data have been added. Columns that were already discrete and new columns with discretised data will be used in the learning process.

ID	***TE***	***PaC***	***MeC***	***PRC***	**…**	***TE_d***	***PaC_d***	***MeC_d***
1	79.13	43	0	0	…	<92.175	<50.5	<0.5
2	133.1	53	53	1	…	>128.45<150.1	>50.5<53.5	>27.5<78
…	…	…	…	…	…	…	…	…
34	141.4	52	48	5	…	>128.45<150.1	>50.5<53.5	>27.5<78

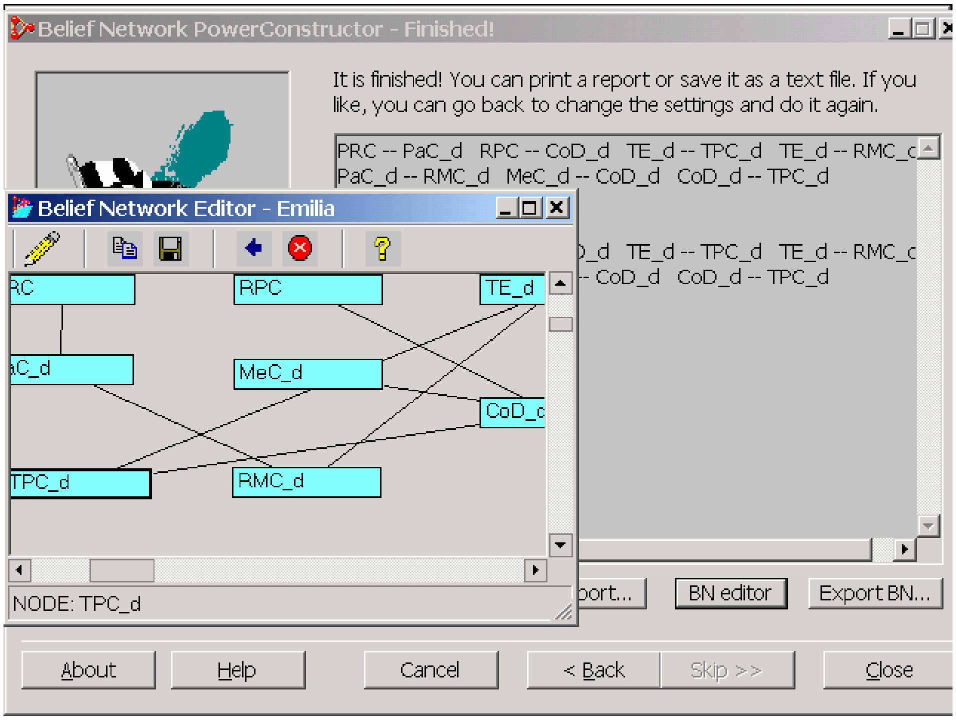
Table 6.3 Preprocessed Data

Learning process:

This process consists of learning the network structure and generating the node probability tables.

In order to learn the structure, we need to apply domain knowledge both before and after the execution of the learning process. Before executing the learning process, it may be necessary to totally or partially define the structure of the network (node ordering for known causal relationships, forbidden links and root/leaf nodes) and some parameters such as thresholds for the accepted dependencies. After executing the algorithm, we may need to edit the network to add, remove or label arcs if it was impossible to do this from the data.

There are several projects under the GNU license that implement well-known *search and scoring* and *dependency analysis* algorithms in Java. Those algorithms can be integrated into the USEGESOFT tool easily. However, we found that using wizard-like GUI interfaces to define the structure facilitates the above task. Among these software packages we used BNPC tool (Cheng, 2001) and Bayesware Discoverer (Ramoni and Sebastiani, 2002). Figure 6.5 shows a possible BBN generated after running the BBN Power Constructor tool.



It is finished! You can print a report or save it as a text file. If you like, you can go back to change the settings and do it again.

```
PRC -- PaC_d  RPC -- CoD_d  TE_d -- TPC_d  TE_d -- RMC_c
PaC_d -- RMC_d  MeC_d -- CoD_d  CoD_d -- TPC_d

)_d  TE_d -- TPC_d  TE_d -- RMC_c
-- CoD_d  CoD_d -- TPC_d
```

NODE: TPC_d

Figure 6.5 Learning Process using the BNPC Tool (Cheng, 2001)

Figure 6.5 shows a screenshot of the BNPC tool and the generated network. In this case, the tool was not able to assign directions to the links between the nodes with the data available, hence the need to modify the network by hand. It is to be noted that expert domain knowledge may be required for such editing. Once the structure is defined, the probabilities of the links are estimated from the data by the BNPC tool.

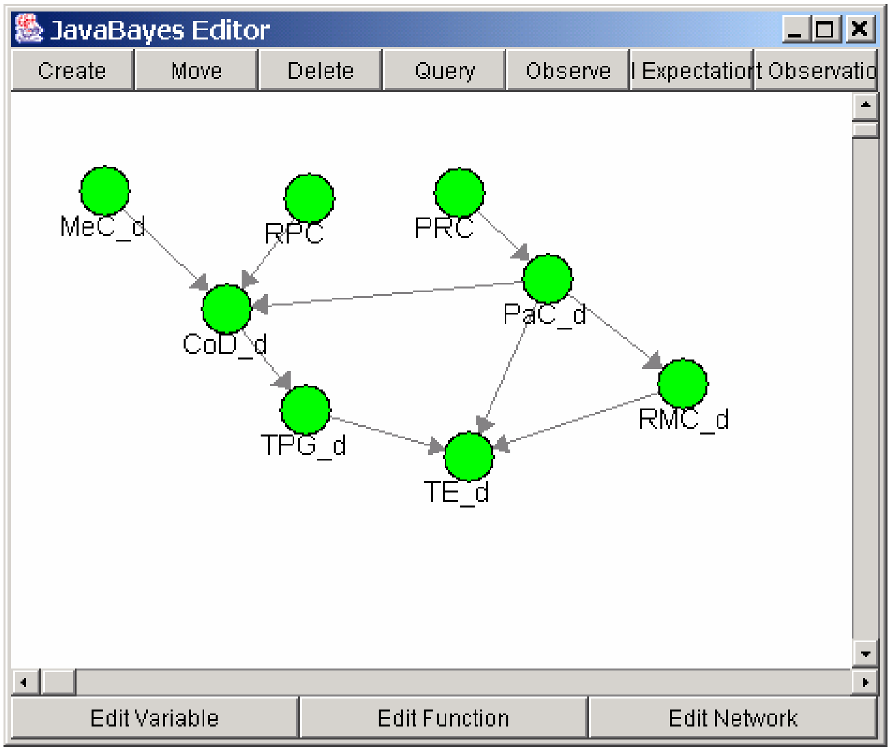


Figure 6.6 BBN generated from Mendes' dataset

At this stage the resulting network can be used by Bayesian network inference tools; in our case, we used USEGESOFT tool which includes a modified version of the JavaBayes tool (Cozman, 2001). Figure 6.6 shows the final network from which we should be able to show how predictions might be made and historical results can be explained more clearly (Figure 6.7 and Figure 6.8). For example, it would be possible to estimate the *total effort* of the authoring of a Web application once we have approximate knowledge about other parameters of the Web application such as number of pages, amount of media used, number of JavaScripts etc. It is worth noting that in this case the output will be a distribution of probabilities instead of a single value. Naturally, the value or the range with the highest probability is taken as the expected answer.

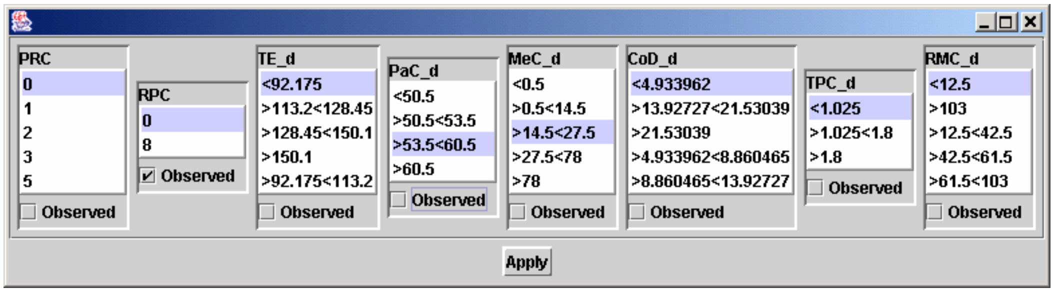


Figure 6.7 Evidences Window of a BBN for Web Authoring Estimates

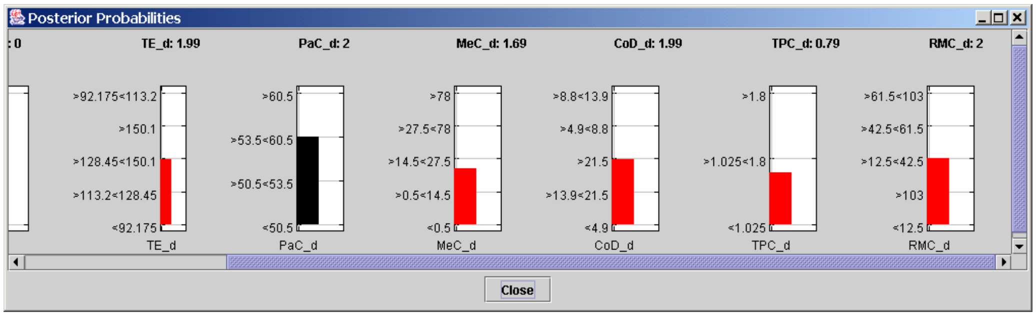


Figure 6.8 Probabilities Window of a BBN for Web Authoring Estimates

Validation of the BBN:

It is important that we need to validate a BBN once it is created from historical data. Presently, we will compare the results obtained from this network with the actual dataset from which we obtained the network. Note that our approach is not a proper validation technique; however, this will give us some confidence about the effectiveness of the network.

Table 6.4 shows the maximum, minimum and the mean values of the 8 variables in the original dataset. We use the network to obtain the value of TE (total effort) by feeding to the network the values of the remaining variables. For the minimum values of the variables (minimum values of each variable in the dataset), Table 6.4 says that TE has a value of 58.36. The network shows (refer to Figure 6.9) that the probability that the value of TE remains less than 92 is 55%, and the other possible values of TE are very low. When the maximum values are considered, the table shows that value of TE is 153.78, whereas the Figure shows that the probability of TE being 150 is maximum (33%). So far as the mean values are considered, the table shows that TE has a value of 111.89, whereas the network shows that the probability of the TE value remaining between 92 and 128 is high. The results that we obtain from the BBN are shown in Figure 6.9. Figure 6.10 show the probability distribution of the variable TE (Total Effort) when minimum values are given as evidence. This is in consistent with the actual results in the given dataset. In a BBN, the expected value and the variance of a node can be obtained by using the standard equations, i.e.:

$$\mu = E[X] = \sum_i x_i P(X = x_i)$$

$$Var\ (X) = E(X^2) - \mu$$

where x_i is the outcome of X at sample point i.

Metric	***Mean***	***Min***	***Max***
PaC	55.21	33	100
MeC	24.82	0	126
PRC	0.41	0	5
RMC	42.06	0	112
RPC	0.24	0	8
COD	10.44	1.69	23.3
TPC	1.16	0	2.51
TE	111.89	58.36	153.78

Table 6.4 Summary Statistics

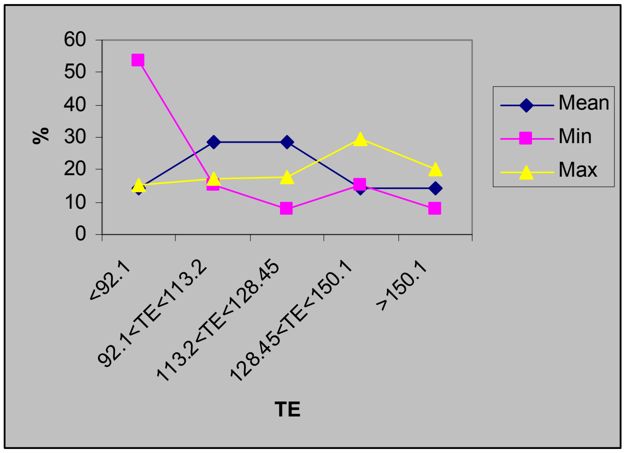


Figure 6.9 Output Probabilities for the Variable *Total Effort* (TE)

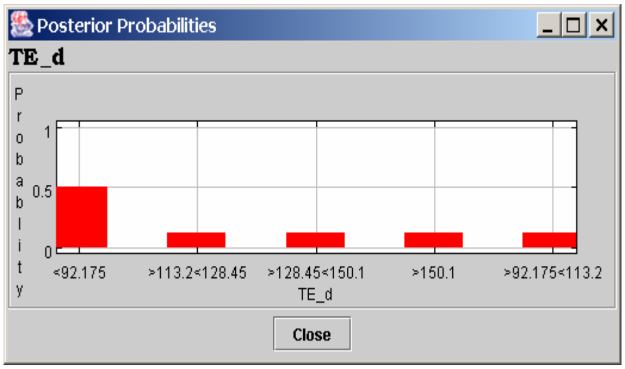


Figure 6.10 Probability Distribution with Minimum Values

What we can say from these results is that the values obtained by the network approximate the actual values to a large extent, although they are not that accurate. The reason is that the number of cases considered in the dataset is not that high (34).

The most common way of validating the predictive accuracy of this type of models is based on cross validation, which divides the database in parts and then, for each part, it predicts the values of a set of variables by estimating the conditional probabilities of the network from the remaining parts. However, Kirsopp and Shepperd (2002) have investigated problems with this hold-out approach concluding that using small dataset can lead to almost random results. Furthermore, the network construction process uses expert knowledge at intermediate stages, which might have not been perfect in our case.

6.4 Use of BBNs in Process Improvement

In this section, we will use BBNs as an aid during process improvement. We will first present an introduction to the type of process improvement that we are dealing with.

6.4.1 The PROFES Methodology

The number of candidates for improvement in a process is large, but also each process has many aspects (or factors) that may need improvement; in other words, many improvement actions are possible for any individual process. However, the cost of implementing improvement actions can be high. Furthermore, an end-product has many quality factors that may be improved. So, the optimal choice is to choose a few of the end-product quality factors, find out which improvement actions can lead to the improvement of such product quality factors, and implement those specific actions. This is the philosophy behind Product Focused Process Improvement (PFPI) (PROFES, 2002a).

Implementation of improvement actions is a costly activity; therefore, it would be better if we could predict the impact of an improvement action before it is implemented. In this chapter we will use Bayesian networks to predict the impact of an improvement action in the context of the PROFES (PROduct Focused process improvement or Embedded Systems) methodology (PROFES, 2002a).

The PROFES improvement methodology identifies a few important quality goals of the final product which need to be improved. Typical such goals are: (i) to decrease the defect density of an artefact, (ii) to increase the reliability of the end-product, (iii) to improve the fitness for use of the final product, and (iv) to improve the predictability of the quality, time and cost of the development of the product etc. Next, all improvement actions are identified from a PPD (Product Process Dependency) (PROFES, 2002b) repository which, possibly, will lead to the desired product quality improvements. As the number of such improvement actions could be large, a few important ones need to be identified and implemented.

PPD repositories are the core element of the PROFES improvement methodology. They contain an organised collection of PPD models. A PPD model describes the impact of a particular improvement action using a template on a certain software quality characteristic when applied in a certain development process in a specific project context. Table 6.5 and Table 6.6 respectively show the structure of two PPD models that we have used from (Pfahl and Birk, 2000). PPD model A indicates that to get a better defect density in the final product, formal inspection should be applied to code development process under the following context characteristics. The project type should be either semi-detached or embedded; it can be of any project size, and the manpower skill must be either average or high. The meaning of PPD model B can be interpreted similarly.

PPD Model A		
Product Quality		Defect Density
Process		Code Development
Improvement action		Formal Inspection
Context Section		
CF-1	Project Type	Organic,semi-detached,embedded
CF-2	Project Size	Small,average,large
CF-3	Manpower Skill	Small, average, high

Table 6.5 PPD Model A (Pfahl and Birk, 2000)

PPD Model B		
Product Quality		Defect Density
Process		Design Development
Improvement action		Formal Inspection
Context Section		
CF-1	Project Type	Organic,semi-detached,embedded
CF-2	Project Sise	Small,average,large
CF-3	Manpower Skill	Small, average, high

Table 6.6 PPD Model B (Pfahl and Birk, 2000)

The six phases of PROFES improvement cycle are (van Latum and van Uijtregt, 2000):

- **Characterise:** Current product quality is evaluated by analysing available product quality data. Customer feedback, customer surveys, market research results, internal interviews etc. provide information for product quality needs.
- **Set Goals:** Final product improvement goals are set. Necessary process changes are determined by referring to the PPD repository. Those PPDs which match the current project context and the product quality needs are the candidates for selection. Corresponding process improvement actions are then obtained. If there are many improvement actions, then the most important ones are selected by expert judgment or from previous simulation results (Pfahl and Birk, 2000).
- **Plan:** Plans for implementing the process changes are made which may include training needs, progress training etc. A measurement program, possibly based on the GQM method (Basili *et al*, 1994), is initiated.

- **Execute:** The improvement actions are implemented according to the plan. If necessary, corrective actions based on feedback analysis are taken.
- **Analyse:** Evaluate the measured data to find out if the improvement actions indeed resulted in improved product quality. The lessons learned are captured and evaluated.
- **Package:** Experience gained from the project is stored for further use. The PPD models in the repository may be enriched based on the acquired knowledge.

6.4.1.1 Evolution of a PPD Repository

An organisation is expected to have its own PPD repository for its PFPI programme. But questions may arise as to how to do this. The organisation should start with a tentative list of PPD models based on experience or obtained from the PROFES PPD repository. The PROFES project has identified a set of such models from an extensive investigation of the process-impact on product quality. It is obvious that such PPD models are generic in nature and they must be tailored for a specific organisation. Such refinement of a PPD model can be performed by using past project data and/or by conducting process assessment(s). PFPI can then be initiated on the basis of the refined PPD models. Every improvement cycle will come up with new knowledge, which, in turn, can be used to refine the corresponding PPD models. Thus, over a period of time, the customised PPD repository of an organisation will attain a level of stability.

6.4.2 Prediction of PPD Models through BBN

Under the PROFES improvement methodology, it is crucial that we should have some idea about the outcome of a PPD model before it is implemented. One way to have such knowledge is to generate empirical evidence from pilot applications. However, such an approach is not only expensive but it also can be risky. Therefore, it would be appropriate to use simulation models to predict the outcome of PPD models before they are selected for implementation. Pfahl and Birk (2000) have used system dynamics to perform such a simulation of PPD models. We intend to use BBNs, already discussed in Chapter 6.

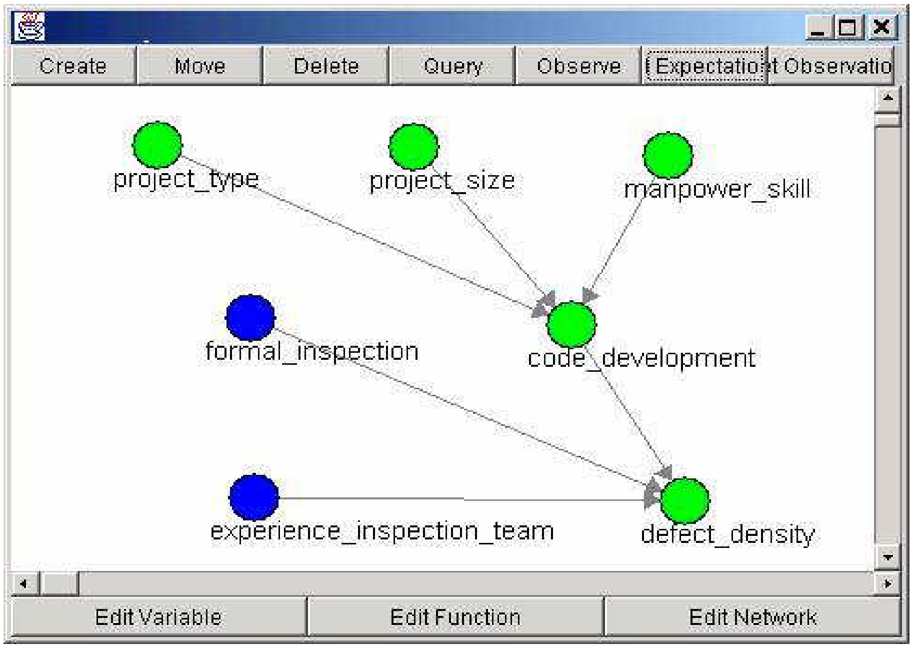


Figure 6.11 BBN for PPD Model A

A PPD model can be represented as a BBN (Satpathy *et al*, 2002a); for example, the BBN of Figure 6.11 represents the PPD model of Table 6.5. The BBN shows that the quality of code development depends on project-type, project-size and manpower-skill. The defect density in the final product depends on formal inspection and the experience of the inspection team. Conditional probabilities are assigned to each node through expert judgment. Figure 6.12 shows that when there is no formal inspection and the quality of developed code is medium, the defect density of the final product is likely to be high. However, if we use an experienced inspection team, then the defect density can be brought down. Such a scenario is shown in Figure 6.13.

The probabilities associated a BBN representing a PPD model are supposed to be tuned to the capability of an organisation; in other words, such values represent to what extent each of the context factors influence the outcome of the application of the PPD model. Such values are refined as new experience is gathered from implementation of improvement actions. So when we use a network to predict the outcome of the corresponding PPD model in advance, in all likelihood, we would succeed in our prediction. Coming back to the example discussed above, if a number of PPDs are available for decreasing defect density then their corresponding BBNs can be analysed, and such a analysis will reveal which PPD model is most effective in bringing defect density down. Furthermore, the impact of a combination of PPD models can also be obtained by constructing a BBN which can predict the combined impact of the corresponding improvement actions. The appropriate combination can then be selected for implementation. Currently, we implement this by hand; though clearly it could be done with tool support.

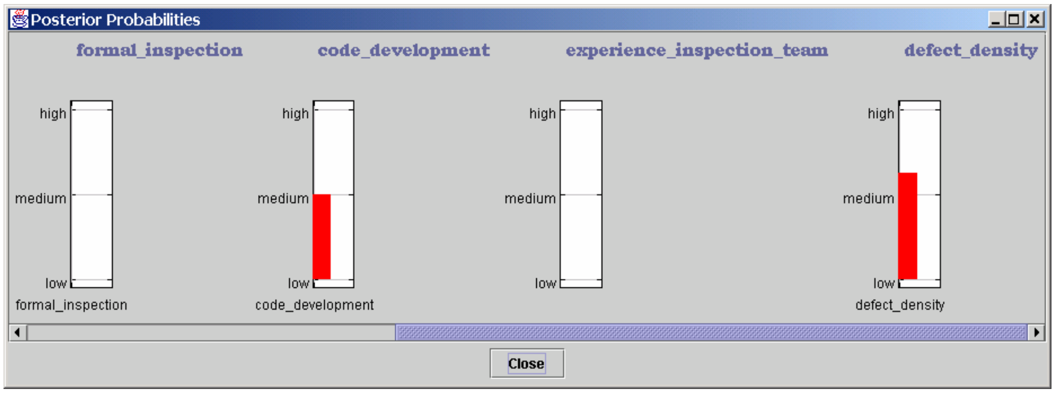


Figure 6.12 Analysis of BBN without Formal Inspection

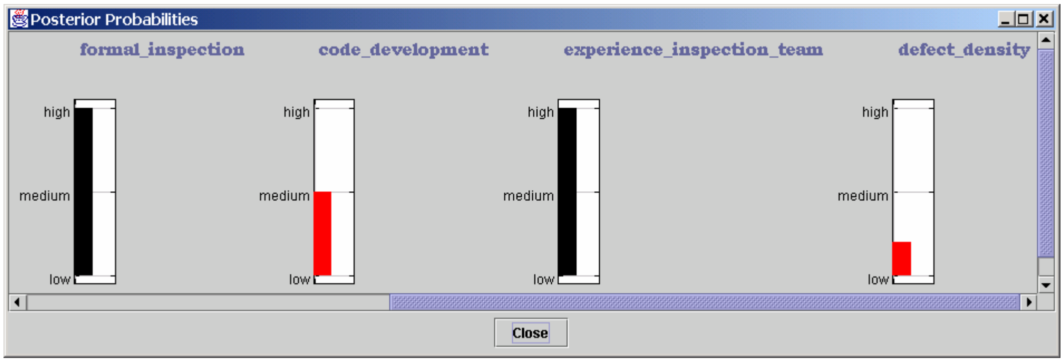


Figure 6.13 BBN Analysis with an Experienced Formal Inspection Team

6.5 Discussion

6.5.1 Properties and Limitations of BBNs

We first look at the general properties of BBNs and then focus on how BBNs can overcome some the problems with traditional metrics and methods in software engineering. As Fenton (1999a) states: *"The key to more powerful use of software metrics is not in more powerful metrics. Rather it is in a more intelligent approach to combining different metrics …. Once you are comfortable with such an approach you may need to look at more advanced statistical techniques for achieving a real prediction system. BBNs is such an approach that we feel provides the most promising way forward"*.

BBNs have a number of features that make them suitable for dealing with problems in the software engineering arena:

- Graphical representation: In a BBN, a graphical notation represents in an explicit manner the dependence relationships between the entities of a problem domain.
- Qualitative and quantitative modelling. BBNs are composed of both a qualitative part in the form of a directed acyclic graph and a quantitative part in the form of a set of conditional probability distributions. Therefore, BBNs are able to utilise both subjective judgements elicited from domain experts and objective data (e.g. past project data).
- Bi-directional inference. Bayesian analysis can be used for both forward and backward inference, i.e. inputs can be used to predict outputs and outputs can be used to estimate input requirements. For example, we can predict the number of *residual* defects of the final product based on the information about *project size*, *project complexity*, *design effort* etc. Furthermore, given an approximate value of *resisual defects* the BBN will provide us with a combination of allowable values for the *design effort*, *testing effort* etc. which could satisfy the desired number of *residual defects*.
- *Explaining away*: If we have more accurate information about some variable in the network, its impact on all other variables can be observed in the network. Because of the dependency

- *Sensitivity analysis:* It is possible to perform sensitivity analysis using Bayesian networks quite easily, i.e., how sensitive is a conclusion given set of evidences just changing some of the evidences.
- *Uncertainty*. There are many sources of uncertainty, such as distortion, incompleteness and irrelevancy (Krause and Clark, 1993). In common with all Bayesian systems, Bayesian networks model "degrees of belief" (equivalent to probabilities) rather than exact value or range of values. This means that uncertainty can be handled effectively and represented explicitly.
- *Confidence Values*. The output of a BBN is a probability distribution for each variable instead of a single value. This sort of information can be used as a measure of confidence in the result, which is essential if the model is going to be used for decision support.

In spite of previously stated advantages, Bayesian networks have the following limitations:

- *Structure*. It may be difficul to describe a complex structure which could even be complex for experts especially if the problem area is new. So there is the possibility of mismatch between the problem domain and the constructed model.
- *Hidden variables*. If two variables had a hidden relationship, then there would often be a dependency between them and the model might have not taken this into account.
- Inconsistent probabilities. In a system with a large number of variables, it can be difficult ensure that probabilities are consistent. Inferences of a BBN may only be useful if the prior knowledge is reliable. A poor estimation of prior beliefs will distort the entire network and invalidate the results.
- When constructing the BBNs from data there are also numerous sources of error. For example, as we have commented in previous sections when discretising variables there is always loss of information, missing values from the database, etc.

From the software engineering point of view, BBN are a useful tool as they provide a way of dealing with some of the limitations of traditional approaches:

- BBN deal with uncertainty. As stated by Kitchenham and Linkman (1997) many areas in software engineering are driven by uncertainty. A typical example of reasoning under uncertainty is as follows. If a project manager, for instance, makes the observation that a software module has many post-release defects, then, the project manager has a higher belief that the module was very complex or poorly tested. BBN models can predict events based on partial or uncertain data, i.e., making good decisions with data that is scarce and incomplete. Uncertainty is inherent in Software Engineering. For example, Kitchenham and Linkman (1997) state that estimates are a probabilistic assessment of a future condition and that is the main reason why managers do not obtain good estimates.
- Software Engineering entities are typically influenced by many factors. BBNs allow us to create and manipulate complex models to understand chains of events (causal relationships) in a graphical way that might never be realised using conventional methods. Moreover, it is possible to include variables in a model that correspond to processes as well as product attributes.

6.5.2 BBNs vs. other Machine Learning Techniques

Neural Networks vs. BBNs: NNs have been shown to produce favourable results when used with sufficiently large datasets. A difference between BBNs and NNs is that NNs cannot handle uncertainty. Furthermore, NNs offer a black-box view in the sense that they do not provide information about how the results are reached. Another difference is that hidden layers and weights do not provide information about the domain (they only have a meaning in the context of the functionality); however, all nodes in a BBN and their probability tables provide information about the domain and can be interpreted. Another disadvantage of NN compared to BBNs is that expert knowledge cannot be incorporated into a NN. Probabilities can be introduced into a BBN using expert knowledge, past data or using a combination of both, while in NNs it is only possible through training with past project data. Moreover, it is not possible to change the parameters of the network once the learning process has finished. Finally, it is to be noted that a over-trained NN loses its ability to generalise.

Rule-base systems vs. BBN: The main difference between BBNs and rule based systems is that rule based systems model experts' way of reasoning while BBNs model dependencies in the domain. Another difference is that the propagation of probabilities in BBNs uses a global perspective in the sense that any node in a BBN can receive evidences, which are propagated in both direction of the edges. In addition, simultaneous evidences do not affect the inference algorithm. BBNs as well as fuzzy logic systems also provide less commitment to exact values as predictions can be defuzzied into linguistic labels to handle uncertainty.

6.5.3 Further Extensions to BBNs

The BBNs that we have seen so far are not very flexible, that is, once a network is created, it provides no way to adapting to a changing domain. Moreover, some complex systems cannot be represented by BBNs properly because those systems contain a large number of variables or because we may need to model many similar situations with different configurations. In order to mitigate these problems, researchers are providing mechanisms to create a complex system from several smaller systems.

Pearl (1986) defines *idioms* as fragmented structures of causal organisations which are being assembled on the fly from a stock of building blocks. Idioms act as building blocks that can be joined together to obtain bigger systems. Furthermore, using idioms can help to identity the granularity of the modelling process. Neil *et al.* (Neil *et al*, 2000) extended this notion and proposed an idiom framework to help practitioners build large scale BBNs exploiting ideas from Object Oriented Bayesian Networks. The authors also provide a BBN life-cycle process using idioms.

Object Oriented Bayesian Networks (OOBN) provide a way to represent large scale BBNs through modularisation (Koller and Pfeffer, 1997). The authors exploit the fact that the same components in a large system may appear several times in a model having their own identity. They provide a mechanism to quickly combine the different sub-models to model a particular configuration and reduce the amount of work by reusing sub-models. In OOBNs, both knowledge declaration and probabilistic inference are modular.

Influence Diagrams (ID) are Bayesian networks which have been extended for decision making (Jensen, 1996). In a decision making scenario, it may be beneficial for the decision maker to acquire additional information before a decision is made. Influence diagrams are BBNs extended with new types of nodes called utility and decision nodes to explicitly model various decision alternatives and the utilities associated with these. An example is a decision on whether or not to carry out code inspection. The result of carrying out code inspections can reduce the number of defects but it may not be worth the cost. Althoug it is possible to construct a model for decision making with BBNs, the concepts of utility and decisions are not explicitly covered by general Bayesian networks.

Dynamic Bayesian Networks (Ghahramani, 1997) are able to model time relationships. Many entities in problem domains are modeled with variables whose values change over the time. Dynamic Bayesian networks capture this process by representing multiple copies of the state variables, one for each time step.

6.6 A Validation Framework for BBNs

In this section, we will present the framework proposed by Kitchenham *et al* (2002) which validates influence diagrams and show its relevance in relation to BBNs. Their evaluation framework is composed of five quality aspects: (i) syntactic quality, (ii) semantic quality, (iii) pragmatic quality, (iv) test quality and (v) value quality. For each quality aspect, the authors define goals and means to achieve them. Although their evaluation framework was designed for a bidding problem, it can also be used for evaluating dynamic systems such as BBNs and System Dynamic models. We will now discuss such issues in detail.

(i) The *syntactic quality* goal is syntactic correctness. BBNs have a well defined syntax based on graph and probability theories and therefore they are amenable to automation.

(ii) The *semantic quality* goal comprises of feasible validity and feasible completeness. In the context of BBN, feasible completeness mean that the model include all the relevant causality relationships (nodes and arcs) in the domain. Feasible validity refers to the correctness of such relationships to the domain. Kitchenham *et al* define 'traceability to the domain' as the model property that supports the semantic quality goals. This is valid in the context of BBN because the causality relationships have to be reflected in the domain.

In our opinion, semantic quality goals are the most difficult to achieve in the context of BBNs and since they are related to the problem of knowledge elicitation. Since the problem of building BBNs inevitably involves domain experts, it is necessary to provide them with techniques (i) for constructing the structure of BBNs, and (ii) to elicit their conditional probability values. Fenton *et al* (1999a) have concentrated on the problem of specifying sensible graph structures extending the idea of Object Oriented BBNs (OOBBNs), where fragments of BBNs can be combined together into larger ones. The authors have developed a prototype called SERENE based on these ideas (Neil *et al*, 2000). Other approaches include the similarity graph of Heckerman (1991), which exploits the clinical technique of differential diagnosis to elicit directed graphical structures. The work by Druzdzel and van der Gaag (1995) is an example of how to elicit probabilities for the NPTs.

There is always an element of subjectivity in the specification of BBNs. Use of data mining techniques can help to reduce this because the structure of the BBNs and their parameters can be learnt directly from data. However, when following this approach, there can be other problems related to the quality of data, size of the dataset, algorithms utilised etc. For example, missing data complicates the learning process; or what is worse, if the data is 'false', the learned model will also be ‘false’. Discretisation of variables also results in loss of information.

Kitchenham *et al* proposed inspections aimed at checking that the model includes definitions, detail and scope information of the variables. This is an appropriate technique for evaluating BBNs because experts can notice whether the model behaves differently when compared with reality. Other techniques that can be applied include sensitivity analysis to identify unnecessary features, consistency checking etc. When constructing the Bayesian network using data mining techniques, thresholds that are used to decide node dependencies should be considered carefully.

(iii) *Pragmatic quality* deals with the issue of how to express a model and it comprises of two goals: feasible comprehension and feasible understandability (Kitchenham *et al*, 2002). Methods to facilitate comprehension include visualisation, explanation and filtering which can be assessed through empirical studies. Expressive economy and structuredness are model properties that enable the feasible understandability quality aspect. Documentation standards and guidelines will also help to improve the model (Kitchenham *et al*, 2002). In the context of BBNs, the current problems with tools is that they are mainly research prototypes which in general do not handle the visualisation in an aesthetic and user friendly manner. Layout algorithms, hypermedia features and integration of the models and inference engines with specialised user interfaces could improve this quality aspect.

(iv) The goal for the *test quality* aspect is feasible test coverage. Simulation studies can be used to assess this quality aspect. When data is available, cross-validation methods for estimating generalisation error based on re-sampling can be used (Castillo *et al*, 1997).

(v) Finally, *value quality* refers to practical utility. Kitchenham *et al* include appropriate user manuals, training etc. as means to achieve this. These can be

> evaluated through empirical studies. In the context of BBNs and software engineering, project managers and quality assurance engineers need to understand the underlying models to apply them correctly.

An important point highlighted by Kitchenham *et al* is the difference between generic and specialised models. The same applies to BBNs, where causal relationships of software processes could be defined in a generic way but once these generic processes are modelled it can be difficult to adapt them to other more concrete environments.

6.7 Conclusions

In this chapter we have discussed the use of BBNs in the area of software engineering. BBNs can be useful because: (i) they can explain some cause-effect relationships of software engineering better than static models; (ii) the backwards propagation of the probabilities in the BBN can help in taking managerial decisions that might not be possible using static models.

Knowledge discovery should be an important feature of a project management tool for making important managerial decisions. One such example is the creation of Bayesian networks from past project data. The network so created could make accurate analysis of dynamic behaviour. This is an important motivation behind the design of our tool.

Construction of BBNs is an intellectual task. We have discussed a semi-automatic approach which uses many prototype tools for constructing BBNs. We have also discussed a practical example and constructed a BBN in relation to the given dataset.

We have also discussed an example which shows how BBNs can be used in process improvement.

Chapter 7

Conclusions

This chapter summarises the research carried out in the previous chapters, analyses the research contributions and offers suggestions for future research.

7.1 Summary of the Thesis

Web applications are different from traditional applications to a significant degree, and so they offer new challenges and issues in technological, economic and social areas. The aim of this thesis was to improve Web processes so that they could produce quality Web products by means of software engineering concepts from the point of view of developers and managers.

We first carried out semi-structured interviews with Web practitioners to understand the current practices as regards to Web development in industry, and to identify deficiencies. Some deficiencies that we identified from our interviews were addressed in later Chapters.

In Chapter 3, we discussed a framework for the Web development process based on our interviews and related literature. Our framework separated (starting from the requirement analysis phase) the development of authoring and infrastructure components of Web applications. Such a formalisation of the Web development process opens up a modular approach to Web development.

The design and development of a project management tool as a proof of concept and to provide an environment where project management and quality aspects are related have been described in Chapter 4. Current project management tools mostly deal with time, tasks and resources. They usually do not provide any mechanisms for constructing processes nor for dealing with the different quality metrics generated during the development of a project. Our tool also provides a framework within which different techniques such as Bayesian Belief Networks (Jensen, 1996), System Dynamics (Abdel-Hamid and Madnick, 1991) etc. can be applied. We also described how our tool and some of its functionalities fit into the CMM framework.

In Chapter 5, we demonstrated the customisation of the TGPM to obtain a measurement framework for Web processes. This customised measurement framework can form the basis for assessment and improvement of Web processes; in particular, we have focused on the authoring process, and have shown how to use the GQM method for assessing and improving the authoring process.

The last chapter before the conclusions is related to the use of Bayesian networks in software engineering. We describe how to generate BBNs from project data using an dataset of Web authoring projects. We also discussed advantages, limitations and their validation. Finally, we also demonstrated how Bayesian networks can be applied for process improvement based on the PROFES Methodology (PROFES, 2002a), and how by representing PPD (Product-Process Dependency) models as BBNs, it would be possible to predict the outcome of improvement action(s).

7.2 Main Contributions

In this section, we revisit in detail the aims stated at the beginning of the thesis and discuss the degree to which we have addressed them, including the limitations of our work.

Aim: To understand the current approaches to Web Engineering in industry and to investigate deficiencies in them, if any.

To have an in-depth knowledge of current practices to Web development, we first performed a survey of the existing literature. Based on our study, we formulated a semi-structured questionnaire to carry out a series of interviews with practitioners in industry. From such interviews, we highlight the following findings. Some important deficiencies identified were: (i) Web applications have two distinct application domains – authoring and infrastructure – but current design methods hardly reflected this difference, (ii) lack of proper quality models and measurement frameworks for Web process and applications.

A limitation of our semi-structured interviews is the small number of subjects; consequently, our survey should be considered an preliminary approach. There is always the risk of it being unrepresentative of the general population. However, we selected our subjects with adequate experience, and organisations of different sizes and types. A flaw in the design of the survey was that the participants were not asked directly what problems they faced or what they felt they needed to help them with their tasks.

Aim: To develop a new methodology to address the existing deficiencies

We developed a UML based methodology that could fit with practices used in industry. Our design methodology separates hypermedia and application domains, and both can be developed by distinct teams, possessing different skills. Integration of both the modules is necessary to produce the final Web application.

Benefits of our methodology include: (i) a cleaner design architecture that reflects the structure at conceptual level, (ii) a modular approach to development, so the different skills of different developers can be effectively used; and (iii) the whole development process is carried out in the UML framework so that currently available tools can be used.

In our methodology, we have utilised much of the best practices available in literature; including the processes steps defined by OOHDM, UML notations from Conallen and Baumeister *et al* for specifying the navigation structure and the internal process structure of the TGPM. Our approach could be seen as addressing some of the problems related to 'hypermedia engineering crisis'.

One limitation of our methodology is the definition of the interface between both domains (the hypermedia and the application domain); further research would be necessary to address this issue.

Aim: To understand the current approaches to evaluating Web processes and applications, and to develop a measurement framework to cater to the specific issues of Web processes

We adopted the TGPM as the basis of our quality framework and extended this model to make it suitable for Web applications (Rodriguez *et al*, 2002). The extended model could be instantiated to obtain quality models of individual processes in Web development. In particular, we have instantiated this model to obtain the customised model of the authoring process.

The TGPM in combination with GQM forms the basis of our measurement framework for various Web processes and subprocesses. We have shown how this framework could be used for process assessment and improvement following a top-down approach for the assessment and improvement of Web processes.

Although the TGPM provides a framework for creating GQM quality trees, it does not provide any obvious way of identifying metrics. We have provided tool support for TGPM; however, further research is necessary to make the TGPM understandable to managers.

Aim: To support the measurement framework and evaluation methods with tools

We developed a project management tool which supports the TGPM, and hence our customised models. This tool supports integration of different techniques such as Bayesian Networks and System Dynamics for exploring new ways of solving problems in software engineering in general, and Web applications in particular. For instance, we focused on BBNs to estimate quality characteristics of both software processes and products.

We have discussed how to create BBNs from historical data.We have also shown how BBNs could play a crucial role in the context of the PROFES methodology for predicting the outcome of an improvement action.

Our tool should be considered as a proof of concepts, i.e., it is a prototype developed with purely research objectives in mind. A significant amount of further research would be necessary to make it robust enough for commercial situations.

7.3 Future Work

We have identified the following areas for future research.

(i) Further research on the Parallel Framework

We have shown the feasibility of Web development through our Parallel Framework through the development of a relatively simple example. Though we believe that scalability would not be an issue for applying the parallel methodology, this needs to be validated by applying the methodology to build larger applications.

As mentioned in the previous section, one limitation of our parallel methodology is a systematic definition of the interface between the authoring and infrastructure domains; further research would be necessary to provide an algorithmic approach to solve this problem.

(ii) Quality evaluation and process improvement

We showed how the TGPM can be instantiated to Web Authoring. However, we have not yet performed any industrial case studies. The application of customised models for various Web processes would validate the appropriateness of the TGPM when used in the context of Web applications.

(iii) BBN vs. other similar models

It would be of interest to compare the BBN approach with other dynamic techniques such as System Dynamics in areas such as effort and cost estimates, and then, compare the BNN approach with others such as estimation by analogy (Shepperd and Schofield, 1997). Moreover, further experiments should consider real data from industrial projects to investigate the advantages of BBNs over traditional regression models.

(iv) Validation of BBNs

In Chapter 6, we have presented a semi-automatic procedure for obtaining BBNs from historical data. However, the fact that the BBNs so constructed are valid is an important issue. Our future work involves obtaining a robust validation procedure for validating BBNs in relation to the problem domains.

(v) PPD models as BBNs

We have shown how a PPD model represented as a BBN can predict the outcome of a single improvement action in relation to the given PPD model. In order to predict the outcome of a combination of improvement actions, it would be necessary to combine the BBNs of the respective PPDs. Currently, we create such a network manually. Constructions of such a combined BBN in relation to a set of PPD models through a tool will be a part of our future research.

It would be of interest to develop specific PPD models for Web and hypermedia applications.

References

Abdel-Hamid, T. K. and Madnick, S. E. 1991, *Software Project Dynamics: an Integrated Approach*, Prentice Hall, Englewood Cliffs, N.J.

Abran, A. 1999, 'FFP Release 2,0: An Implementation of COSMIC Functional Size Measurement Concepts', in *FESMA*, Amsterdam.

Aguilar-Ruiz, J. S., Ramos, I., Riquelme, J. C. and Toro, M. 2001, 'An evolutionary approach to estimating software development projects', *Information and Software Technology,* vol. 43, no. 14, pp. 875-882.

Aguilar-Ruiz, J. S., Riquelme, J. C., Rodriguez, D. and Ramos, I. 2002, 'Generation of Management Rules through System Dynamics and Evolutionary Computation', in *Product focused software process improvement*, eds. Oivo, M. and Komi-Sirvio, S., Berlin, Rovaniemi, Finland, pp. 615-628.

Albretch, A. J. 1979, 'Measuring Application Development', in *Proc. of INM Applications Development Joint SHARE/GUIDE Symposium*, Monterey, CA, pp. 83-92.

Avison, D. E. and Fitzgerald, G. 1995, *Information Systems Development: Methodologies, Techniques and Tools*, 2nd edn, McGraw-Hill, Maidenhead.

Barry, C. and Lang, M. 2001, 'A Survey of Multimedia and Web Development Techniques and Methodology Usage', *IEEE Multimedia,* vol. 8, no. 2, pp. 52-61.

Basili, V., Shull, F. and Lanubile, F. 1999, 'Building Knowledge through Families of Experiments', *IEEE Transactions on Software Engineering,* vol. 25, no. 4, pp. 456-473.

Basili, V. R., Caldiera, G. and Rombach, H. D. 1994, 'The Goal Question Metric Paradigm', in *Encyclopedia of Software Engineering*, John Wiley & Sons, Inc., pp. 528-532.

Baumeister, H., Koch, N. and Mandel, L. 1999, 'Towards a UML Extension for Hypermedia Design', in *UML'99 - The Unified Modeling Language. Beyond the Standard. Second International Conference*, eds. France, R. and Rumpe, B., Springer-Verlag, Fort Collins, CO, USA, pp. 614-629.

Bazzana, G., Brigliadori, R., Andersen, O. and Jokela, T. 1993, 'ISO 9126 and ISO 9000: Friends or Foes?' in *Software engineering standards symposium*, IEEE Computer Society Press, Brighton, pp. 79-88.

Bazzana, G. and Fagnoni, E. 1999, 'Process Improvement in Internet Service Providing', in *Better Software Practice for Business Benefit. Principles and Experience*, ed. Richard Messnar, C. T., IEEE CS, pp. 267-279.

Bazzana, G. and Pioti, M. 1999, 'Process and Product Measurement', in *Better Software Practice for Business Benefict*, eds. Messnarz, R. and Tully, C., IEEE Comp. Society, Los Alamitos, pp. 151-176.

Berson, A. 1992, *Client/Server Architecture*, McGraw-Hill.

Bøegh, J., Depanfilis, S., Kitchenham, B. and Pasquini, A. 1999, 'A Method for Software Quality Planning, Control and Evaluation', In *IEEE Software*, vol. 16, pp. 69-77.

Boehm, B. W. 1978, *Characteristics of Software Quality, TRW series of software technology ; v. 1*, North-Holland Pub. Co. : American Elsevier, Amsterdam ; New York.

Boehm, B. W. 1981, *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, N.J.; London.

Booch, G. 1994, *Object-Oriented Analysis and Design with Applications, The Benjamin/Cummings series in object-oriented software engineering*, 2nd edn, Benjamin/Cummings Pub. Co., Redwood City, Calif.

Booch, G., Rumbaugh, J. and Jacobson, I. 1999, *The Unified Modeling Language User Guide, The Addison-Wesley object technology series*, Addison-Wesley, Reading, Mass.

Botafogo, R. A., Rivlin, E. and Shneiderman, B. 1992, 'Structural Analysis of Hypertexts: Identifying Hierarchies and Useful Metrics', *ACM Transactions on Information Systems,* vol. 10, no. 2, pp. 142-180.

Breiman, L., Friedman, J., Olshen, R. and Stone, C. J. 1984, *Classification and Regression Trees*, Chapman and Hall, New York.

Brereton, P., Budgen, D. and Hamilton, G. 1998, 'Hypertext: The Next Maintenance Mountain', *Computer,* vol. 31, no. 12, pp. 49-55.

Briand, L. C., El Emam, K., Surmann, D. and Wieczorek, I. 1999, 'An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques', in *Software engineering*, IEEE Computer Society, Los Angeles; CA, pp. 313-323.

Castillo, E., Gutiérrez, J. M. and Hadi, A. S. 1997, *Expert Systems and Probabilistic Network Models*, Springer-Verlag.

Cheng, J. 2001, *Belief Network (BN) Power Constructor*, Available: [http://www.cs.ualberta.ca/~jcheng/bnpc.htm].

Cheng, J., Bell, D. A. and Liu, W. 1997, 'Learning Belief Networks from Data: an Information Theory Based Approach', in *Sixth ACM International Conference on Information and Knowledge Management*, ACM Press.

Chow, C. K. and Liu, C. N. 1968, 'Approximating discrete probability distributions with dependence trees', *IEEE Transactions on Information Theory,* vol. 14, pp. 462-467.

Christensen, D. S. 1998, 'The Costs and Benefits of the Earned Value Management Process', in *International Society of Parametric Analysts and Society of Cost Estimating and Analysis*, Ispa, Toronto; Canada, pp. 1-16.

Christodoulou, S. P., Styliaras, G. D. and Papatheodrou, T. S. 1998, 'Evaluation of hypermedia application development and management systems', in *HYPERTEXT '98. Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia: Links, Objects, Time and Space - Structure in Hypermedia Systems*, ACM Press, Pittsburgh, PA USA, pp. 1-10.

Cockburn, A. 2000, *Writing effective use cases, The Crystal series for software development*, Addison-Wesley, Boston.

CodeFactory 2002, *MrProject*, Available: [http://mrproject.codefactory.se/].

Conallen, J. 1999, 'Modelling Web Application Architectures with UML', In *Communications of the ACM*, vol. 42, pp. 63-70.

Conallen, J. 2000, *Building Web Applications with UML, Addison-Wesley object technology series*, Addison-Wesley, Reading, Mass. ; Harlow.

Conklin, J. 1987, 'Hypertext: An Introduction and Survey', In *IEEE Computer*, vol. 20, pp. 17-41.

Consortium, C. 2000, 'Research Briefs', In *Research Briefs*.

Cooper, G. F. 1987, *Probabilistic Inference Using Belief Networks is NP-hard.*, Medical Computer Science Group, Stanford University, KSL-87-27.

Cooper, G. F. and Herskovits, E. 1992, 'A Bayesian Method for the Induction of Probabilistic Networks from Data', *Machine Learning,* vol. 9, pp. 309-347.

Cozman, F. G. 2001, *JavaBayes 0.346*, Available: [http://www-2.cs.cmu.edu/~javabayes/].

Creswell, J. W., Goodchild, L. F. and Turner, P. P. 1996, 'Integrated Qualitative and Quantitative Research: Epistemology, History, and Designs', *Higher Education,* vol. 11, pp. 90-136.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N. and Weerawarana, S. 2002, 'Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI', *Ieee Internet Computing,* vol. 6, no. 2, pp. 86-93.

Daly, J. 1996, *Replication and a Multi-Method Approach to Empirical Software Engineering Research*, University of Strathclyde.

D'Ambrosio, B. 1999, 'Inference in Bayesian Networks', *AI Magazine,* vol. 20, no. 2, pp. 21-36.

Dart, S. 1999, 'Containing the Web Crisis Using Configuration Management', in *Web Engineering Workshop at the International Conference on Software Engineering (ICSE'99)*, Los Angeles, USA.

Daskalantonakis, M. K. 1992, 'A Practical View of Software Measurement and

Implementation Experiences Within Motorola', *IEEE Transactions on Software Engineering,* vol. 18, no. 11.

Debou, C. 1999, 'Goal Based Software Process Improvement Planing', in *Better Software Practice for Business Benefit. Principles and Experience*, ed. Richard Messnar, C. T., IEEE CS, pp. 107-150.

Delic, A. K., Mazzanti, F. and Strigini, L. 1995, *Formalizing a Software Safety Case via Belief Networks*, Center for Software Reliability, City University, London, U.K.

Denzin, N. K. 1994, 'Evaluating qualitative research in the poststructural moment: the lessons James Joyce teaches us', *International Journal of Qualitative Studies in Education,* vol. 7, no. 4, p. 295.

Dermiame, J. C., Kaba, B. A. and Wastell, D. 1998, *Software Process: Principles, Methodology and Technology, Lecture Notes in Computer Science 1500*, Springer Verlag, Heidelberg.

Deshpande, Y., Hansen, S. and Murugesan, S. 1999, 'Web Engineering: beyond CS, IS and SE', in *Web Engineering Workshop (ICSE'99)*, Los Angeles, pp. 10-16.

Díaz, A. and Isakowitz, T. 1995, 'RMCase: Computer-Aided Support for Hypermedia Design and Development', in *International Workshop on Hypermedia Design (IWHD'95)*, Springer-Verlag, Montpellier, France, pp. 3-15.

Dromey, R. G. 1995, 'A Model for Software Product Quality', *IEEE Transactions on Software Engineering,* vol. 21, no. 2, pp. 146-162.

Dromey, R. G. 1996, 'Conering the Chimera', In *IEEE Software*, pp. 33-41.

Druzdzel, M. J. and van der Gaag, L. C. 1995, *Elicitation of Probabilities for Belief Networks: Combining Qualitative and Quantitative Information*, Utrecht University, Institute of Information and Computing Sciences, UU-CS-1995-23.

ESA 1995, *Guide to Software Project Management*, ESA (European Space Agency), Paris, France, ESA PSS-05-08.

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. 1996, 'The KDD Process for Extracting Useful Knowledge From Volumes of Data', In *Communications of the ACM*, vol. 39, pp. 27-34.

Fenton, N. E. 1995, *Multi-criteria Decision Aid; with emphasis on its relevance of in dependability assessment*, CSR, City University, London, U.K., DATUM/CSR/02.

Fenton, N. E. and Neil, M. 1999a, 'A Critique of Software Defect Prediction Models', *IEEE Transactions on Software Engineering,* vol. 25, no. 5, pp. 675-689.

Fenton, N. E. and Neil, M. 1999b, 'Software Metrics: Successes, Failures and New Directions', *Journal of Systems and Software,* vol. 47, no. 2-3, pp. 149-157.

Fenton, N. E. and Neil, M. 2000, 'Software Metrics: A Roadmap', in *The Future of Software Engineering*, ed. Finkelstein, A., ACM.

Fenton, N. E. and Pfleeger, S. L. 1997, *Software Metrics: a Rigorous and Practical Approach*, International Thomson Computer Press: PWS Pub, London.

Flemming, Q. W. and Koppelman, J. M. 1996, *Earned Value Project Management*, The Project Management Institute (PMI).

Forrester, J. W. 1961, *Principles of Systems*, Norwalk, CT: Productivity Press.

Fugetta, A., Lavazza, L., Morasca, S., Cinti, S., Oldano, G. and Orazi, E. 1998, 'Applying GQM in an Industrial Software Factory', *ACM Transactions on Software Engineering and Methodology,* vol. 7, no. 4, pp. 411-448.

Gamma, E. 1995, *Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley professional computing series*, Addison-Wesley, Reading, Mass.

Garvin, D. A. 1988, *Managing Quality: the Strategic and Competitive Edge*, Free Press, New York.

Garzotto, F., Mainetti, L. and Paolini, P. 1994, 'HDM2: Extending the ER Approach to Hypermedia Application Design', *Lecture Notes in Computer Science,* vol. 823, pp. 178-188.

Garzotto, F., Matera, M. and Paolini, P. 1999, 'Abstract Tasks: a Tool for the Inspection of Web Sites and off-line Hypermedia', in *Proceedings of the tenth ACM Conference on Hypertext and Hypermedia*, ACM Press, Darmstadt, Germany, pp. 157-163.

Garzotto, F., Paolini, P. and Schwabe, D. 1993, 'HDM A Model Based Approach to Hypertext Application Design', *ACM Transactions on Information Systems,* vol. 11, no. 1, pp. 1-26.

Ghahramani, Z. 1997, 'Learning Dynamic Bayesian Networks', in *Adaptive processing of sequences and data structures*, eds. Giles, C. L. and Gori, M., New York, Vietri sul Mare; Italy, pp. 168-197.

Gillies, A. 1997, *Software Quality: Theory and Management*, 2. edn, International Thomson Computer, London.

Glaser, B. G. and Strauss, A. L. 1967, *The Discovery of Grounded Theory:*
Strategies for Qualitative Research, Aldine Publishing.

Grady, R. B. 1992, *Practical software metrics for project management and process improvement*, Prentice Hall, Englewood Cliffs, NJ.

Grady, R. B. and Caswell, D. L. 1987, *Software Metrics: Establishing a Company-wide Program*, Prentice-Hall, Englewood Cliffs, N.J.

Gray, A. R. and MacDonell, S. G. 1997, 'A Comparison of Techniques for Developing Predictive Models of Software Metrics', *Information and Software Technology,* vol. 39, no. 6, pp. 425-437.

Halasz, F. and Schwartz, M. 1990, 'The Dexter Hypertext Reference Model', in *NIST Hypertext Standardization Workshop*, pp. 95-133.

Halasz, F. and Schwartz, M. 1994, 'The Dexter Hypertext Reference Model', *Communications of the ACM,* vol. 37, no. 2, pp. 30--39.

Halpin, T. A. 2001, *Information Modeling and Relational Databases: from Conceptual Analysis to Logical Design, The Morgan Kaufmann series in data management systems*, Morgan Kaufman Publishers, San Francisco, Calif. ; London.

Hardman, L., Bulterman, D. C. and van Rossum, G. 1994, 'The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model', In *Communications of the ACM*, vol. 37, pp. 50-62.

Heckerman, D. E. 1991, *Probabilistic Similarity Networks*, MIT Press, Cambridge, MA.

Henderson-Sellers, B. and Unhelkar, B. 2000, *Open Modeling with UML*, Addison-Wesley, Harlow, England; NY.

Humphrey, W. S. 1992, *Introduction to Software Process Improvement*, Software Engineering Institute, CMU/ SEI- 92- TR- 7.

IBM 2002, *WebSphere Studio*, Available: [http://www-3.ibm.com/software/ad/adstudio/].

Isakowitz, T., Bieber, M. and Vitali, M. 1998, 'Web Information Systems', *Communications of the ACM,* vol. 41, no. 7, pp. 78-80.

Isakowitz, T., Stohr, E. A. and Balasubramanian, P. 1995, 'RMM: A Methodology for Structured Hypermedia Design', *Communications of the ACM,* vol. 38, no. 8, pp. 34-44.

ISO 1991, *Information technology - Software product evaluation*, International Organization for Standarization (ISO), ISO/IEC 9126.

ISO 1994, *Quality Management and Quality Assurance Standards*, International Organization for Standarization (ISO), ISO 9000-1:1994.

ISO 1995, *Information technology - Software life cycle processes*, International Organization for Standarization (ISO), ISO/IEC 12207.

ISO 1998, *Information technology - Software Process Assessment*, International Organization for Standarization (ISO), ISO/IEC TR 15504.

Jacobson, I., Booch, G. and Rumbaugh, J. 1999, *The Unified Software Development Process*, Addison-Wesley, Reading, Mass.; Harlow.

Jalote, P. 2002, *Software Project Management in Practice*, Addison-Wesley, Boston.

Jensen, F. V. 1996, *An Introduction to Bayesian Networks*, UCL Press, London.

Kirda, E., Jazayeri, M., Kerer, C. and Schranz, M. 2001, 'Experiences in Engineering Flexible Web Services', *IEEE Multimedia,* vol. 8, no. 1, pp. 58-65.

Kirsopp, C. and Shepperd, M. 2002, 'Making Inferences with Small number of Traning Sets', *IEE Procedings of Software Engineering,* vol. 149, no. 5, pp. 123-130.

Kitchenham, B. and Pfleeger, S. L. 1996, 'Software quality: The Elusive Target', *Ieee Software,* vol. 13, no. 1, pp. 12-21.

Kitchenham, B. A. 1987, 'Towards a constructive quality model. Part I Software Quality Modelling, Measurement and Prediction', *Software Engineering Journal*.

Kitchenham, B. A., Hughes, R. T. and Linkman, S. G. 2001, 'Modeling Software Measurement Data', *IEEE Transactions on Software Engineering,* vol. 27, no. 9, pp. 788-804.

Kitchenham, B. A. and Linkman, S. G. 1997, 'Estimates, Uncertainty, and Risk', In *IEEE Software*, vol. 14, pp. 69-74.

Kitchenham, B. A., Pickard, L. M., Linkman, S. G. and Jones, P. 2002, 'A Framework for Evaluating a Software Bidding Model', in *Empirial Assessment in Software Engineering*, Keale.

Koller, D. and Pfeffer, A. 1997, 'Object-oriented Bayesian networks,' in *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, eds. Geiger, D. and Sheno, P. P., Morgan Kaufmann, San Francisco, pp. 302--313.

Krause, P. J. 1998, 'Learning Probabilistic Networks', *Knowledge Engineering Review,* vol. 13, pp. 321-351.

Krause, P. J. and Clark, D. 1993, *Representing Uncertain Knowledge: An Artificial Intelligence Approach*, Intellect Books, Oxford.

Kuntzmann-Combelles, A., Pulford, K. and Shirlaw, S. 1996, *A Quantitative Approach to Software Management: the ami Handbook*, Addison-Wesley, Wokingham.

Kuvaja, P. 1994, *Software process assessment and improvement : the BOOTSTRAP approach*, Blackwell Business, Oxford.

Lopistéguy, P., Pagorret, P. and Losada, B. 1997, 'Hypermedia Design Methodologies Versus Hypermedia Functionality Integration', in *Workshop on Incorporating Hypertext Functionality - ACM Hypertext'97*, ACM, Southampton.

Lowe, D. 1999, 'Engineering the Web. Web Engineering or Web Gardening?' *WebNet Journal. Internet Techonologies, Applications and Issues,* vol. 1, no. 1.

Lowe, D. and Hall, W. 1999, *Hypermedia & the Web: an Engineering Approach, Worldwide series in computer science*, John Wiley, Chichester, Eng.

Lowe, D. and Webby, R. 1999, 'Utilisation of Process Modeling in Improving the Hypermdia Development Process', in *The New Review of Hypermedia and Multimedia*, Taylor Graham, Longon, pp. 133-150.

Lowe, D. B., Bucknell, A. J. and Webby, R. G. 1999, 'Improving Hypermedia Development: a Reference Model-based Process Assessment Method', in *Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots*, Darmstadt, Germany, pp. 139-146.

Maurer, F., Dellen, B., Bendeck, F., Goldmann, S., Holz, H., Kötting, B. and Schaaf, M. 2000, 'Merging Project Planning and Web-Enabled Dynamic Workflow Technologies', In *IEEE Internet Computing*.

McCall, J. A., Richards, P. K. and Walters, G. F. 1977, *Factors in Software Quality*, US Rome Air Development Center Reports, RADC TR-77-369.

Mendes, E., Mosley, N. and Counsell, S. 2001, 'Web Metrics-Estimating Design and Authoring Effort', *IEEE Multimedia,* vol. 8, no. 1, pp. 50-57.

Mendes, E., Watson, I., Triggs, C., Mosley, N. and Counsell, S. 2002, 'A Comparison of Development Effort Estimation Techniques for Web Hypermedia Applications', in *8th IEEE Metrics Symposium*, IEEE Press, Ottawa, Canada, pp. 131-140.

Mendes, M. E. X., Hall, W. and Harrison, R. 1999, 'Applying Measurement Principles to Improve Hypermedia Authoring', *The New Review of Hypermedia and Multimedia,* vol. 5, pp. 105-132.

Microsoft 2002, *Microsoft Project*, Available: [http://www.microsoft.com/].

Murugesan, S. and Deshpande, Y. 2001, *Web Engineering: Managing Diversity and Complexity of Web Application Development, Lecture Notes in Computer Science ; 2016*, Springer, Berlin ; New York.

Neil, M., Fenton, N. E. and Nielsen, L. 2000, 'Building large-scale Bayesian Networks', *The Knowledge Engineering Review,* vol. 15, no. 3, pp. 257-284.

Nielsen, J. 2000, *Designing Web Usability: the Practice of Simplicity*, New Riders, Indianapolis, Ind.

Noack, J. and Schienmann, B. 1999, 'Introducing OO Development in a Large Banking Organization', *IEEE Software,* vol. 16, no. 3, pp. 71-81.

Offutt, J. 2002, 'Quality Attributes of Web Software Applications', *Ieee Software,* vol. 19, no. 2, pp. 25-33.

Olsina, L. 1998, 'Building a Web-based Information System applying the Hypermedia Flexible Process Modeling Strategy', in *1st International Workshop on Hypermedia Development Held in conjunction with Hypertext'98*, Pittsburgh, PA.

Olsina, L., Godoy, D., Lafuente, G. J. and Rossi, G. 1999, 'Specifying Quality Characteristics and Attributes for Websites', in *First ICSE Workshop on Web Engineering (WebE-99)*, eds. Murugesan, S. and Deshpande, Y., Los Angeles, USA.

Orfali, R., Harkey, D. and Edwards, J. 1999, *Client/server Survival Guide*, John Wiley, New York ; Chichester.

Paulk, M. C., Curtis, B., Chrissis, M. B. and Weber, C. V. 1993, 'Capability Maturity Model, Version 1.1', In *IEEE Software*, vol. 10, pp. 18-27.

Pearl, J. 1986, 'Fusion, Propagation and Structuring in Belief Networks', *Artificial Intelligence,* vol. 29, pp. 241-248.

Perry, D., Porter, A. and Votta, L. 2000, 'Empirical Studies of Software Engineering: A Roadmap', in *The Future of Software Engineering*, ed. Finkelstein, A., ACM Press.

Pfahl, D. and Birk, A. 2000, 'Using Simulation to Visualise and Analyse Product-Process Dependencies in Software Development Projects', in *Product focused software process improvement*, eds. Bomarius, F. and Oivo, M., Berlin, Oulu, Finland, pp. 88-102.

Picton, P. 2000, *Neural networks*, Palgrave, Basingstoke, Hampshire ; New York.

PMI 1996, *A Guide to the Project Management Body of Knowledge*, Project Management Institute, Upper Darby, PA.

Powell, T. A., Jones, D. L. and Cutts, D. C. 1998, *Web Site Engineering: Beyond Web Page Design*, Prentice Hall, Upper Saddle River, N.J.

PowerSim 2003, *Powerim: Power Simulator*, Available: [http://www.powersim.com/].

Pressman, R. S. 2000, 'What a Tangled Web We Weave', *IEEE Software*, pp. 18-21.

PROFES 2002a, *The PROFES Methodology*, Available: [http://www.profes.org/].

PROFES 2002b, *The PROFES PPD Repository: Understanding patterns of Product/Process Dependence*, Available: [http://www.iese.fhg.de/projects/profes/PPDRepository].

Putman 1978, 'A general empirical solution to the macro software sizing and estimating problem', *IEEE Transactions on Software Engineering,* vol. 4, no. 4, pp. 345-361.

Quinlan, J. R. 1993, *C4.5: Programs for machine learning*, Morgan Kaufmann, San Mateo, California.

Ramoni, M. and Sebastiani, P. 2002, *Bayesware Discoverer*, Available: [http://www.bayesware.com/].

Rational 2000, *Rational Rose*, Available: [http://www.rational.com/].

Rational 2002, *Rational Suite*, Available: [http://www.rational.com/products/rs/index.jsp].

Rich, E. and Knight, K. 1991, *Artificial intelligence*, 2nd edn, McGraw-Hill, New York.

Rodriguez, D., Harrison, R. and Satpathy, M. 2001a, 'A Generic Model for Assessing and Improving Web Processes', in *14th International Conference Software & Systems Engineering and their Applications (ICSSEA)*, Paris, France.

Rodriguez, D., Harrison, R. and Satpathy, M. 2002, 'A Generic Model and Tool Support for Assessing and Improving Web Processes', in *8th IEEE International Software Metrics Symposium (Metrics 2002)*, IEEE Press, Ottawa, Canada, pp. 141-151.

Rodriguez, D., Harrison, R., Satpathy, M. and Dolado, J. 2001b, 'Tool Support for the Typed Generic Quality Model', in *Current Trend in Software Measurement (Proceedings of 11th International Workshop on Software Measurement)*, eds. Dumke, R. and Abran, A., Shaker Verlag, Montreal, Canada.

Satpathy, M. and Harrison, R. 2002, 'A Typed Generic Process Model for Product Focused Process Improvement', in *Computer Software and Applications Conference (COMPSAC 2002)*, IEEE Press, Oxford.

Satpathy, M., Harrison, R. and Rodriguez, D. 2002a, 'Investigation of Product Process Dependency Models through Probabilistic Modeling', in *Empirical Assessment in Software Engineering (EASE 2002)*, Keele.

Satpathy, M., Harrison, R., Snook, C. and Butler, M. 2000, 'A Generic Model for Assessing Process Quality', in *New Approaches in Software Measurement - 10th International Workshop on Software Measurement (IWSM'00)*, vol. 2006, eds. Dumke, R. and Abran, A., Springer-Verlag, Berlin, Germany, pp. 94-110.

Satpathy, M., Siebel, N. T. and Rodriguez, D. 2002b, 'Maintenance of Object Oriented Systems through Re-engineering: A Case Study', in *IEEE International Conference on Software Maintenance (ICSM 2002)*, IEEE Press, Montreal, Canada.

Schwabe, D., Esmeraldo, L., Rossi, G. and Lyardet, F. 2001, 'Engineering Web Applications for Reuse', *IEEE Multimedia,* vol. 8, no. 1, pp. 20-31.

Schwabe, D. and Rossi, G. 1995, 'The Object-Oriented Hypermedia Design Model', *Communications of the ACM,* vol. 38, no. 8, pp. 45-46.

Schwabe, D. and Rossi, G. 1998, 'An Object Oriented Approach to Web-based Applications Design', *Theory and Practice of Object Systems,* vol. 4, no. 4, pp. 207-225.

Seaman, C. B. 1999, 'Qualitative Methods in Empirical Studies of Software Engineering', *IEEE Transactions On Software Engineering,* vol. 25, no. 4, pp. 557-572.

Shepperd, M. and Schofield, C. 1997, 'Estimating Software Project Effort Using Analogies', *IEEE Transactions on Software Engineering,* vol. 23, no. 11, pp. 736-743.

Shepperd, M., Schofield, C. and Kitchenham, B. 1996, 'Effort Estimation Using Analogy', in *Software engineering*, IEEE Computer Society Press, Berlin, pp. 170-178.

Singh, I. and Johnson, M. 2001, *Designing Enterprise Applications with the J2EE*, 2nd edn, Addison Wesley.

Solingen, R. v. and Berghout, E. 1999, *The Goal/Question/Metric Method: a Practical Guide for Quality Improvement of Software Development*, McGraw-Hill, Maidenhead.

Stella 2004, *Stella*, Available: [http://www.stella.com/].

Symons, C. R. 1988, 'Function Point Analysis: Difficulties and Improvements', *IEEE Transactions on Software Engineering,* vol. 14, no. 1.

Takahashi, K. and Liang, E. 1997, 'Analysis and Design of Web-based Information Systems', in *6th International World Wide Web Conference (WWW6)*, eds. Genesereth, M. R. and Patterson, A., Santa Clara, USA, pp. 377-389.

Tiga 2002, *PRONEL*, Available: [http://www.tiga-technologies.com/].

Tuya, J., Fernández, P., Prieto, M. A., Aguilar, J., Ramos, I., Riquelme, J., Ferrer, F., Toro, M., Ruiz-Carreira, M., Rodriguez, D., Satpathy, M., Harrison, R., Ruiz de Infante, A., Dolado, J. J., Matilla, R. and Álvarez, M. A. 2001, 'Integration of Information in a Training Environment for Software Project Management', in *Software Quality Management 2001 SQM'01*, Loughborough, UK.

van Latum, F. and van Uijtregt, A. 2000, 'Product Driven Process Improvement PROFES Experiences at Drager', in *Product focused software process improvement*, eds. Bomarius, F. and Oivo, M., Springer Verlag, Oulu, Finland, pp. 232-243.

Vensim 2003, *Vensim: Ventana Simulator*, Available: [http://www.ventana.com/].

W3C 1998, *Extensible Markup Language (XML) 1.0. W3C Recommendation*, World Wide Web Consortium.

W3C 1999, *XSL Transformations (XSLT). Version 1.0 W3C Recommendation*, World Wide Web Consortium.

Warmer, J. and Kleppe, A. 1999, *The Object Constraint Language. Precise Modeling with UML*, Addison Wesley.

Wooff, D. A., Goldstein, M. and Coolen, F. P. A. 2002, 'Bayesian Graphical Models for Software Testing', *IEEE Transactions on Software Engineering,* vol. 28, no. 5, pp. 510-525.

Ziv, H. and Richardson, D. J. 1997, 'Constructing Bayesian-network Models of Software Testing and Maintenance Uncertainties', in *Software maintenance*, IEEE Computer Society, Bari; Italy, pp. 100-113.

Appendix A

Questionnaire Survey

General Information

1) With respect to your company:

   a. What is the nature of your organisation?

   b. How many employees does your organisation have? And how large is the development team?

   c. How much of experience (in years) does your company have in Web development for itself? For others?

2) Do you develop Web product for your own organisation or for others?

Requirements

3) Which methods and metrics do you use to estimate the development effort of Web applications (time, effort, cost)?

4) What tools are used when managing requirements?

Design and Development

5) Which development methods are used by your organisation when developing Web applications? (e.g. Structured A/D, OO A/D methods, Data driven methods)

    a. Do you use any hypermedia specific methods such as OOHDM, RMM or HDM?

    b. How do you model the hierarchy and design of the Web applications? (e.g. UI Patterns, storyboards, UI testing)? Please state the work products used in analysis and design to reflect hypermedia and navigation (e.g., storyboards, state chars, etc).

    c. Is device independence (browser, TV, phones etc.) considered?

6) What kinds of tools are used when developing these types of applications?

System test and integration

7) How do you test these types of applications?

8) What tools are used when testing?

Maintenance/Evolution

9) How do you maintain Web applications in relation to the application executables (CGI, ASP, Servlets etc.)?

10) How do you maintain the content and navigational structure (content and links)?

11) Does your organisation have Configuration Management procedures? If so, please state which tools and techniques are used.

Reusability

12) Do you reuse application executables from previous applications? If so,

    a. What do you reuse (components/frameworks, documentation…)?

    b. How does the component acquisition process/development fit into the software life cycle? How do you reuse it (ad-hoc, within a process,)?

13) Do you reuse data (content) from previous systems? If so,

    a. How do you reuse the content (granularity of the information, etc)?

14) What are the tools used for reuse of data, contents, components etc?

Metrics and Quality Assurance

15) Does your organisation have a quality standard certification? (e.g., ISO, CMM etc.) Quality Management team? Standard procedures and project standards?

16) Do you collect metrics at your organisation (system development metrics and process metrics)?

    a. If so, please list these metrics and why are they collected?
       i. as part of the your life-cycle process
       ii. as part of the Web application product (e.g. size metrics)

17) Do you define quality objectives for each project? If so,

    a. How do you define these quality objectives?

    b. Do you have definitions of the ranges for acceptable values for each quality characteristic (e.g. size limits, page layout etc.)?

    c. Do you consider different quality characteristics for different parts/artefacts of the application?

Project Management

18) How is the typical organisation of your Web project? Are there differences with other types of projects? If so, can you state them (activities, models, roles, …)

19) How do you manage the authoring process (content) and the development process (application)?

20) Is an iterative and incremental process followed?

Others

21) Do you have any comments not covered so far?

Appendix B

Definitions of the Sub-factors of an Authoring Quality Model

(a) Sub-factors of **Functionality**

- *Compliance*:
  (i) The degree to which the AUTH process conforms to prescribed (IEEE, ISO or company-specific) standards.
  (ii) The degree to which the AUTH process conforms to its own model (i.e. it consists of a set of sub-processes and the sub-processes in turn consisting of sub-processes or sets of steps).
- *Consistency*:
  (i) The degree to which the AUTH process uncovers contradictions in the requirement document input (i.e. Req_{auth}).
  (ii) The degree to which the AUTH process does not introduce contradictions in the hypermedia content.
- *Completeness*: the degree to which the AUTH process transforms all of the features of the input requirements into the hypermedia content.
- *Generality (robustness)*: The ability of the AUTH process to address by over-specifying/ over-implementing conditions, which are not covered by the input product specification but are relevant to its context.

- *Correctness*: The degree to which the process makes the functionalities of the hypermedia content match accurately to the features requested.
- *Inter-operability*: The degree to which the AUTH process contributes to the ability of the product to interact with specified systems.

(b) Subfactors of **Usability**:

- *Understandability*:
  (i) The effort with which a typical author understands the logical concepts of the AUTH process.
  (ii) The degree to which the AUTH process contributes to the understandability of the hypermedia content.
- *Learnability*:
  (i) The effort required for a typical author to learn to use the process.
  (ii) The degree to which the AUTH process makes the hypermedia content easy to use (through informative help messages, good user interface etc.).
- *Operability*:
  (i) The effort required for a typical author to perform the process steps of the AUTH process.

(c) Sub-factors of **Efficiency & Estimation**:

- *Cost/ Effort estimation*: The degree to which the cost/effort of AUTH process remain within a specified range and the ability of the process to support their estimations.
- *Cycle Time estimation*: The degree to which the AUTH process meets its expected cycle time and the ability of the process to support its estimation.
- *Complexity estimation*: The ability of the process to support the prior estimation of various forms of complexity (Structural Complexity (Botafogo *et al*, 1992), Navigational Complexities, Cognitive Complexity etc), and the degree to which the estimates are accurate.
- *Schedule/ Priority estimation*: The priority of various stages in the process and the ability of the process to support their estimations and scheduling.
- *Resource Estimation*: The degree to which a process keeps its resource usage in a specified range, and its ability to support their estimations.

- *Process Maturity*: The CMM maturity level and/or any ISO certification of the organisation.

(d) Sub-factors of **Visibility & Control**:

- *Progress Monitoring*: The ability of the AUTH process to facilitate monitoring of its progress at any point of time during the process execution to show that progress so far has been correct and effective (e.g. Work product analysis (Grady, 1992), PERT charts etc.).
- *Automatic Feedback*: The ability of the AUTH process to provide feedback data and to support corrective actions if necessary.
- *Improvement Measures*: The ability of the AUTH process to support the analysis of feedback data in combination with the data of previous runs and improve itself, or result in the improvement of a sibling process, continuously.
- (e) Sub-factors of **Reliability**:
- *Failure Frequency*:

  (i) The number of (and the interval between) failures encountered during the AUTH process (e.g. tools used for authoring may crash).

  (ii) The degree to which the AUTH process makes the hypermedia contents failure-free.

(f) **Scalability**:

The degree to which the AUTH process maintains its efficiency level in terms of time, cost and resource usage in handling problems of large sizes.

(g) Sub-factors of **Maintainability**:

- *Analysability*: The effort with which a typical author can analyse the cause of a fault, process failure or unexpected feedback data. A fault could be one of the following categories.

  (i) the fault is discovered during the process execution and the fault may be with the process itself, or it may be from the requirements.

  (ii) the fault is discovered at a later point in time, and the cause of the fault is linked to the AUTH process.
- *Modifiability*: The effort with which a maintainer addresses failures, detection of faults and unexpected feedback data during the AUTH process; or faults

discovered later but linked to the process.

- *Stability*:
  (i)The degree to which addressing a process fault adversely affects the process itself (say, process efficiency or properties like process consistency).
  (ii)The frequency of changes done to the process (less the number of changes, more is the stability).
- *Testability*:
  (i) The degree to which the AUTH process could be validated itself (success history of its hypermedia contents).
  (ii) The degree to which the process contributes to the testability of its artefacts or the final product (say, for generating test cases).
- *Defect Trend*:
  (i) The trend of defects that are observed in the AUTH process itself (defects linked to the process - during or after process execution).
  (ii) The degree to which the AUTH process detects defects or deficiencies in the Req_{auth} so that defects in the hypermedia content are minimal.
- *Reusability*:
  (i) The degree to which the process contributes to the reusability of the hypermedia contents.
  (ii) The degree to which components of the process could be reused in a different context (e.g.: testing process of HTML pages can be applied to WAP pages).
- *Portability*: The degree to which the AUTH process contributes to the portability of the hypermedia contents; i.e., the process should facilitate the migration of the product to a different environment (by taking into account of different versions, releases).

Appendix C

Glossary and Acronyms

AMI: Application of Metrics in Industry. It is a framework for process assessment and improvement.

Anchor represents a linking point, i.e., the start or endpoint of navigation in hypermedia system.

Artefacts are tangible piece of information that are created, changed and used during the software life cycle by workers. Example: models, documentation etc.

Attribute is a feature or property of an entity (Fenton and Pfleeger, 1997).

BBN: Bayesian Belief Network.

Client-Server: The term client/server originally applied to a software architecture that described processing between two programs -- an application and a supporting service. At that time the client program and the server program did not have to be physically separated -- they could be a calling and a called program running on the same machine. Thus, usually the client/server discussions were limited to interactions between one client and one server. As computer science and the theory of programming evolved, however, the concepts of programs capable of providing services or managing resources on behalf of a number of other programs became widely accepted (Berson, 1992).

CMM (Capability Maturity Model) is a process improvement and assessment framework. It defines as five-level for how an organisation matures its software process from ad hoc, chaotic process to mature, disciplined software processes. key process areas (KPAs), which represent the areas on which an organisation should focus if it wants to move to a particular level.

COCOMO (Constructive Cost Model) is a model for estimating cost, effort, and schedule when planning a software development (Boehm, 1981)

Constant Comparison Method: Theory generation method in qualitative research (Glaser and Strauss, 1967).

CPM: Critical Path Method

Entity is an object or an event.

EVPM: Earned Value Performance Measurement

External attributes of a product, process or resource are those that can be measured only with respect to how the product, process, or resource relates to his environment (Fenton and Pfleeger, 1997).

GANTT charts are a matrix where the horizontal axis represents time and the vertical axis lists all tasks to be performed.

GQM: Goal-Question-Metric.

GNU: recursive acronym for "GNU's Not Unix; [http://www.gnu.org/]

GUI: Graphical User Interface.

HTML: HyperText Mark-up Language.

HTTP: HyperText Transfer Protocol.

Hypermedia application can be defined as an application that allows us to navigate through an information space using associative links.

Hypermedia Engineering (Lowe *et al*, 1999) is defined as the employment of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of hypermedia applications.

Hypermedia is the conjunction of hypertext and multimedia. i.e., the elements include graphics, digitalised speech, audio etc. the hyperdocument is called hypermedia. Hypertext can be defined as a Web of chunk of information inter-related through links. The origin or destination of a link is an anchor. Anchors can be words, sentences, areas of an image etc.

Hypertext is defined by Conklin (1987) as follows: *"windows on the screen are associated with objects in a database, and links are provided between these objects, both graphically (as labelled tokens) and in the database (as* *pointers)"*. The collection of objects in the database is defined as a hyperdocument. When the elements that ban be networked together are only text the hyperdocument is called hypertext.

Hypothesis in a experiment is defined as a predicted relationship between variables

Internal attributes of a product, process or resource are those that measured purely in terms of the product, process or resource itself and external attributes (Fenton and Pfleeger, 1997).

KDD (Knowledge Discovery in Databases) computational techniques and tools designed to support the extraction of useful knowledge from databases.

KPA (Key Process Areas) are areas on which an organisation should focus if it wants to move to a particular level.

Link is usually defined as an object which represents a connection between zero or many *anchors* in a hypermedia system.

Maintainability is defined as "the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment". IEEE, "IEEE Software Engineering Standards Collection", Institute of Electrical and Electronics Engineers, New York, NY 1993.

Measurement is defined as "mapping from the empirical world to a formal, relational world. Consequently, a measurement is the number or symbol assigned to an entity by this mapping in order to characterise an attribute." *Direct measure* of an attribute involves no other attribute or entity (length, duration of testing process, number of defects, time a programmer spends,…). *Indirect measures* (programmer productivity = LOC/persons month of effort, etc). (Fenton and Pfleeger, 1997)

Method (Methodology) for system development as a set of phases which guide the developers in their choice of techniques that might be appropriate at each stage of the project (Avison and Fitzgerald, 1995).

Metric (software) "a method of quantitatively determining the extent to which a software process, product, or project possesses a certain attribute. This includes not only the formula used for determining a metric value, but also the chart used for presenting metric values, as well as the guidelines for using and interpreting the charts and metrics in the context of specific projects" (Daskalantonakis, 1992).

Model is an abstraction of reality, allowing us to strip away detail and view an entity or concept from a particular perspective (Fenton and Pfleeger, 1997).

Model is an abstraction of reality, allowing us to strip away detail and view an entity or concept from a particular perspective (Fenton and Pfleeger, 1997).

MTBF: Mean Time Between Failure.

MTTR: Mean Time To Repair.

Node (hypermedia) is each of the navigational objects that composed the hypermedia application.

OCL (Object Constraint Language) is an expression language that enables that enable one to describe constrains (restrictions) on object-oriented models and other object modelling artefacts.

OOHDM: Object Oriented Hypermedia Design Method.

PERT (Program Evaluation and Review Technique) charts depict tasks, duration and their dependencies using a network diagram consisting of numbered nodes representing events or milestones linked by vectors representing tasks in the project.

PERT: Program Evaluation and Review Technique

Process assessment in software engineering is the evaluation of methods, tools, resources and practices.

Process (software) any software activity associated with the development and maintenance of software: from requirement analysis through to maintenance.

Process (engineering). The goal is to develop or enhance a process model; corresponds to a business use case in business engineering" (Jacobson *et al*, 1999).

Product is an entity, which a process (i.e. any software activity) produces as output and may also fed processes as input.

PROFES: PROduct Focused process improvement or Embedded Systems

RMM: Relationship Management Methodology

Software Engineering (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1). (IEEE, 1991)

Software maintenance is defined as "the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment (defined by IEEE standard 729)."

Software metric is defined as "a method of quantitatively determining the extent to which software process, product, or project possesses a certain attribute. This includes not only the formula used for determining a metric value, but also the chart used for presenting metric values, as well as the guidelines for using and interpreting the charts and metrics in the context of specific projects" (Fenton and Pfleeger, 1997).

UML (Unified Modelling Language) is a general-purpose modelling language that is used to specify, visualise, construct and document the artefacts of a software system (Booch *et al*, 1999).

UML profile are extension of UML (using its standard mechanisms - stereotypes, tagged values and constraints to the core UML language-) to create concise and effective models for specific domains. These profiles ensure that the resulting models conform to and are optimised for a specific target domain.

URI: Uniform Resource Identifier

WAP: Wireless Application Protocol

WBS: Work Breakdown Structure

Web applications could be defined as an application that runs on the Web protocols. Web based applications are attracting interest because they have a number of advantages over traditional client-server applications.

Web Engineering Web engineering is concerned with the establishment and use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality Web-based systems and applications. [http://aeims.uws.edu.au/WebEhome/]

XML (eXtensible Mark-up Language) describes a class of data objects called XML documents and partially describes the behaviour of computer programs which process them. XML is an application profile or restricted form of SGML, the

Standard Generalised Mark-up Language. By construction, XML documents are conforming SGML documents. (XML, 2001)

XSL (Extensible Stylesheet Language) is a language for expressing stylesheets. It consists of two parts: (1) a language for transforming XML documents, and (2) an XML vocabulary for specifying formatting semantics (XSL, 2001).