# Chapter 1

## Introduction

### 1. The problem and its context

Software engineering is the process whereby software systems are being developed to solve problems that are defined by certain user requirements. There are, however, effectiveness and efficiency problems that have become accepted facts in the software engineering industry. What is alarming however, is that software will increasingly absorb a larger percentage of the overall development cost for computer-based systems (Longstaff, 2000, p. 43).

### 1.1. What gave rise to the existence of the problem?

The problems concerning the software engineering process arise because of how software is developed, how a growing volume of existing software is maintained and the growing demand for software which result in:

- Inaccurate schedules and cost estimates;
- Low productivity of people involved in the development of software;
- The poor quality of software.

These problems relate to the inherent characteristics of software and software engineering (Pressman, 1993, p. 19).

**The characteristics are:**

**Complexity**

Software engineering is inherently a complex process because fundamentally software consists of large numbers of variables and

unique components. Apart from the actual software system being developed, software engineers also have to capture, organise, analyse and present huge volumes of interrelated development information (Andersen, 1999). These variables, components and information are interdependent and influence each other (Roth, 1994, p. 164). The fact that software is largely invisible because of a lack of visual representation also adds to complexity (Brooks, 1987).

**Scale**

Large-scale projects are more difficult to develop because the number of variables increases exponentially with the size of the project. The number of people involved also adds an extra dimension to the complexity. It is safe to say that the size of a software project is directly proportional to the difficulty of developing it (Kraut, 1995, p. 69).

**Uncertainty**

Software engineering is not a routine activity. Each system being developed is largely unique (Boehm, 2000, p. 32). Uncertainty arises because:

- Software and the software engineering process is inherently unpredictable;
- Requirements change over time;
- Requirements and specifications are to some extent always incomplete;
- People involved make the whole process even more dynamic and unpredictable because they bring their own ideas and agendas to the process (Kraut, 1995, p. 70).

**Informal communication**

Because people develop systems together, their work has to be coordinated. This is often a huge challenge and can pose serious

problems if it is not done successfully. Formal communication is a necessary part of this process, "but often fails in the face of uncertainty, which typifies much software work" (Kraut, 1995, p. 70). Informal communication accommodates this shortcoming, but is difficult to coordinate.

**Human orientation**

Because people develop software according to other people's requirements, software is bound by and conforms to human constraints. For this reason software is constantly changing because requirements change and requirements are subject to the human mind and nature (Roth, 1994, pp. 163, 164). Software also has a "logical rather than physical character" (Pressman, 1993, p. 19). Software is engineered from beginning to end. It is not manufactured like products in other engineering disciplines (Glass, 1995, p. 15).

## 1.2. Stating the problem

The problems surrounding the software engineering process can largely be attributed to the lack of proper coordination and integration of information used for development.

### 1.2.1. Hypothesis

The characteristics of hypermedia technology seems to provide a solution to the problem of coordinating and integrating information in the software engineering process.

The coordination and integration of information used for development has to do with the transfer of information. Because people are so much involved, it can be defined as a communication

problem. This strongly relates to the characteristics of software and software engineering, which are listed in 1.1.

It is hypothesised that hypermedia technology can help to solve the communication problem mentioned above.

This problem viewed from a communications perspective can be broken down into the following:

- Communication problems between the user and the developers;
- Communication problems between people in the development process;
- Communication problems between developers and development information;
- Communication problems between user and user documentation.

The problem will now be identified in the context of the software engineering process.

### 1.2.2. The problem in the context of the software engineering process

#### 1.2.2.1. Problems during analysis

A classical problem in the development of software is the phenomenon that the system being delivered is in many instances not what the user requested or it is what the user requested, but not what he or she requires. The problem, as far as the user is concerned, is therefore not solved (Winograd, 1995, p. 71). The problem encountered here is mainly a breakdown in communication between the user and developer. A breakdown in communication occurs because:

- Developers do not involve users enough in the development process;
- Developers misinterpret user requirements and/or

- Users do not state requirements properly which in turn occurs because:
  - Developers do not properly analyse user requirements and/or
  - Developers do not properly analyse the domain where the system will function in and/or
  - Users do not participate effectively when requirements and domain analysis are being done.
- Communication between developers is not effective while developers are doing analysis to come to a common understanding of the problem from a development perspective.

#### 1.2.2.2. Problems during design

During design all the information gained during analysis must be structured to model the solution to the user's problem (Winograd, 1995, p. 71). It has to be taken into account, however, that analysis and design are concurrent tasks for the greater part of development.

The problems during design are: First of all most of the information gained from analysis and design gets lost because there is too much information to handle. This includes ideas, arguments, collaboration activities and feedback etc. The information is lost because of a lack of proper communication between the developer(s) and the information that is available (Kraut, 1995, p. 70).

Secondly, the designer must provide a conceptual model that the user can relate to. This should be done so that the user can give feedback to the designer. Usually this part of design is not effective because the communication is too technical and not understandable to the user (Loucopoulos, 1995, p. 66). Design is

therefore not user orientated enough and results in systems being developed that do not entirely meet user requirements.

Thirdly, when a large system is being developed and a large number of developers are involved, the formal and informal communication between developers is not always effective (Kraut, 1995, p. 70). Although formal communication is recorded, it is not always structured well enough. For this reason it is not very accessible. Informal communication is usually not documented at all and mostly involves small groups of people. Informal communication is very important because in-depth, invaluable knowledge about the system under development and technical knowledge is communicated. This needs to be accessed by every member of the development team (Hayne, 1996). Apart from non-effective communication, individuals also have different perspectives on design.

Lastly, the integration between the different levels of design is not effective in most cases. Software design involves several transformations. Each of these may be viewed as an instance of the previous set in a new context. For example a requirement specification becomes a design specification which becomes source code which becomes object code. These transformations must be matched with a high degree of precision.

#### 1.2.2.3. Problems during coding and testing

During coding and testing, the design model is transformed into a real application. Apart from the difficulties in moving from an abstract model to a physical model that consists of computer code, the people that are involved usually change from being analysts or designers to application programmers and/or database programmers and/or technical programmers. Therefore, information has to be communicated to another group of developers.

#### 1.2.2.4. Problems during implementation

During implementation, the tested system is transferred to the user domain where it must solve the requested problem. Apart from the physical implementation, the users also have to be trained in using the system. User-manuals and other relevant systems documentation, like database documentation, must also be supplied.

It is specifically in regard to the user-manuals that another problem arises. Because technical people, who developed the application and know it by heart, usually write the user-manuals, the manuals are almost always too technical for the user's needs and comprehension (Loucopoulos, 1995, p. 66).

#### 1.2.2.5. Problems during maintenance and support

Maintenance and support involves changes that have to be coded. This is often done by someone who was not involved in developing the system. Without proper documentation and references to specific information in the documentation, the person doing the maintenance has only the code as way of determining how the system functions. With large projects it is disastrous, if not impossible. The same is true for the process of reverse engineering where proper structuring and integration of development information is needed.

#### 1.2.2.6. Problems surrounding the documentation created during development

The volume of documentation created during software development is enormous and the information it contains is very important to developers. If this information is not well structured and linked, retrieving needed information can be very difficult. In this case, the information does not serve the purpose it was intended for. Development documentation is generally not linked and referenced well enough. This presents a huge problem because quality information is needed where the development of software changes hands from analysts to designers to development programmers to maintenance programmers. Documentation is also not always updated when changes or enhancements are made to the system. This results in system documentation becoming worthless (Glass, 1995, p. 27).

## 1.3. The importance of solving the problem

The size and complexity of software projects are increasing and fast, effective changes will increasingly be expected. Therefore, better integration and coordination of software development along with better access to development information is vital (Cochran, 2001).

## 1.4. Determining the scope of the study

The scope of the study is to determine the support hypermedia technology can provide to the software engineering process in solving the problem of integration and coordination of development information. By relating the problem to the characteristics of hypermedia technology and the type of problems that are typically solved by hypermedia technology, the value of hypermedia technology in solving the problem can be determined.

### 1.5. The importance of the study in providing a solution for the problem

Hypermedia technology seems to be able to provide a solution for the problem of coordinating and integrating development information as encountered in the software engineering process. In light of the fact that effective and efficient coordination and integration of information in the software engineering process is of vital importance because of the increasing size and complexity of software projects, the study is important because it will provide information that will help solve this daunting problem (Cochran, 2001).

## 2. Overview of the state of research on the problem

### 2.1. Nature of the theory and research on the specified problem area

Although the area of hypermedia support for the software engineering process is not as well researched as the software engineering process or hypermedia technology, a number of researchers have noted the relationships and possibilities. Although most of them just mention the relationships in their work, others like Bottaci (1991, pp. 219 - 235) and Roth (1994, pp. 149 - 169) conducted more in-depth research in this area.

### 2.2. Important findings as reflected in the literature

Hypermedia technology has made significant contributions to software development in three primary areas:

- Coordinating and accessing the massive amounts of information used and generated in developing software;

- Linking heterogeneous information in different documents and media;
- Providing access to everyone involved in order for them to add and manipulate information (Roth, 1994, pp. 149 - 169; Andersen, 1998).

Hypermedia technology has strong relationships to many other kinds of software technology like:

- Databases (particularly relational and object-oriented databases);
- Spreadsheets;
- Text processors;
- Outliners;
- Desktop publishing;
- Electronic mail;
- Programming packages;
- Electronic publishing;
- Client/server systems for example the Internet;
- Expert or knowledge based systems;
- Object-oriented programming (Woodhead, 1991, p. 32).

**Hypermedia as a unifying paradigm**

Hypermedia technology is characterised by enormous flexibility. Apart from integrating information in different formats, it can also integrate different applications into a unified, seamless whole while remaining independent of them all. This helps to increase the quality of heterogeneous information (Woodhead, 1991, p. 10).

**Hypermedia technology introduces a more user-orientated approach to working with development information**

As previously mentioned, hypermedia technology integrates heterogeneous information. It also makes the use and manipulation of this information easier. The user plays a much more active role. Instead of being presented with pre-defined knowledge, users can define the information themselves and can also add their own views to the knowledge base. With the structural flexibility of hypermedia technology and variety of media, multiple different views of the same information can be created. This will accommodate people with different cognitive preferences. The interactive interface also enables more active participation by the user (Roth, 1994, p. 157).

**The relationship between hypermedia technology and software engineering**

Hypermedia technology provides a software engineering environment with the capabilities of linking broad categories of software engineering artifacts including:

- Management reports;
- Specifications and requirements;
- Design and program documentation;
- Implementation notes;
- Source code;
- Test specifications and results;
- Object code;
- Products (Roth, 1994, p. 153).

Hypermedia technology, also "provides low overhead task switching in situations where users are performing concurrent tasks such as analysis and design" (Roth, 1994, p. 156).

## The importance of hypermedia technology in interface development

The relationship between hypermedia systems and interfaces is important because:

- Hypermedia technology successfully provides access to reference information;
- Hypermedia technology provides an easy to use, flexible, interactive medium for prototyping, developing and evaluating interfaces;
- Hypermedia, being an integrating medium, is very useful as an interface for distributed, heterogeneous information;
- "Hypermedia can almost be considered as an interface attribute" (Roth, 1994, p. 156).

## The importance of hypermedia technology in prototyping

Prototypes are used to define requirements and to provide feedback. It is therefore primarily used to give feedback of the developer's and the user's understanding of the problem and of each other. The relationships between the developer's and the user's conceptual models must be accommodated. "Hypermedia bears the same relation to cognitive mapping that automated systems analysis tools bear to structured systems analysis methodologies. That is, they can substantially reduce the tedious 'paper-crunching' of numerous redrafts" (Woodhead, 1991, p. 141).

A prototype must also accommodate the user's level of knowledge. Storyboarding techniques and interactive prototypes result in better conceptual understanding than conventional, pre-defined, static prototypes (Jetly, 1999). Also, when a prototype is based on terminology, functions and images that are familiar to the user, it will result in the user understanding the problem better. This will result in the developer understanding the user

and therefore the requirements better. Hypermedia technology also makes easy, rapid prototyping possible.

**The importance of hypermedia technology in handling large volumes of information**

When internalising or processing information, people are limited to the amount of information that can be handled simultaneously. Hypermedia technology can extend a person's capability by optimizing the structure and content of the information.

Hypermedia technology can provide access to large collections of development reference materials. This fits neatly into what Schneiderman calls the "Golden Rules of hypertext":

- "There is a large body of information organised into numerous fragments;
- The fragments relate to each other;
- The user needs only a small fraction at a time" (Roth, 1994, p. 157).

**The importance of hypermedia technology in development documentation**

Hypermedia technology provides the means to link documents quickly and easily for access to a body of integrated information. When developing, hypermedia technology is useful in integrating documents that range from initial requirements to code and maintenance documents (Woodhead, 1991, p. 46).

**The importance of hypermedia technology in structuring information**

Because hypermedia technology is structurally so flexible, information can be organised to be more intuitive and closer to how humans process information than other media. This feature accommodates people in forming and structuring ideas and patterns from information on different abstraction levels. An overview of a system's layout can be organised in a hierarchical structure and links can be made to more detailed information. Related information can also be linked, thereby forming a web-like structure. Information content can also include different types of media (Bottaci, 1991, p. 223).

Not much research has been done on the support hypermedia technology can provide to the software engineering process itself. However, the qualities of hypermedia technology in accommodating and manipulating information relates very well to the type of technology needed to manipulate information created during development. In literature related to the topic of software engineering, it was suggested that development information and documentation play a much more active role in development. With the hypothesis stated in 1.2., it is suggested that hypermedia technology will be useful in achieving this goal.

## 2.3. Motivation for continuing the research as reflected in the literature

The following aspects will provide the motivation for continuing research:

- All aspects of the software engineering process needs better integration and coordination (Kraut, 1995, p. 69);
- Hypermedia technology is important for:
  - Structuring documentation (Woodhead, 1991, p. 46);
  - Structuring information (Bottaci, 1991, p. 223);

- Enriching information content;
- Coping with massive amounts of information (Roth, 1994, p. 164);
- Providing efficient access to large collections of development reference materials (Roth, 1994, p. 157).

## 3. Method that is to be used

The research will be done in the form of a literature study, including a critical analysis and synthesis of the research results. From literature, the characteristics of hypermedia technology will be identified independently of software engineering aspects. Also from literature, the characteristics of software engineering, the criteria for good software engineering as well as the problems involved will be identified. Further, human information processing, communication, documentation and the technical aspects that are involved in software engineering, as well as related problems, will be identified. What is needed to solve these problems will be determined by interpreting what others say in literature and also by the researcher's experience as a computer programmer in developing software systems.

To conclude, all the issues raised above will be discussed in relation to hypermedia technology as a proposed solution in terms of what was identified as what is needed to solve these software engineering problems. This conclusion will then be consolidated into a table to give an integrated perspective of the problem and the proposed solution.

## 4. Chapter layout

The chapter layout represents the methodology that will be followed.

### 4.1. Characteristics of hypermedia technology

The characteristics of hypermedia technology are researched independently of software development issues to render an unbiased view of hypermedia technology.

### 4.2. Characteristics of the software engineering process

The characteristics of the software engineering process are researched. The emphasis is placed on the problems that exist because of these characteristics and what is needed to solve them.

### 4.3. The role of information processing and documentation in the software engineering process

The characteristics of information processing and conventional software documentation are researched. The emphasis is placed on the problems that exist because of these characteristics and what is needed to solve them.

### 4.4. Methods, tools and applications in the software engineering process

The research entails the methods and tools used, the applications developed and what is needed to assist these operations.

### 4.5. Hypermedia technology as a proposed solution in supporting the software engineering process and solving the problems associated with it

Hypermedia technology is proposed as a solution to the problem of coordinating and integrating the software engineering process. The focus is on the problems identified in the previous chapters.

# Chapter 2

## Characteristics of hypermedia technology

### 1. Introduction

Hypermedia technology has a broad and accommodating set of characteristics. In this chapter, these characteristics will be loosely grouped into structural and human-orientated characteristics.

Structural characteristics address the primary aspects of the technology itself and deal with:

- The architecture;
- Structure of nodes and links;
- The associative structure that arises as a result of the nodes and links;
- How the structure improves functionality;
- The content in terms of the different types of media involved.

Human-orientated characteristics are secondary characteristics and result from the nature of the primary characteristics mentioned above. Human-orientated characteristics pertain to:

- Structure in terms of storing information;
- Integrating capabilities;
- Favourable orientation towards the human mind;
- Communication capabilities;
- Ease of use and effectiveness.

### 2. Structural characteristics

The structure of a hypermedia system can range from physical to abstract, which involves associations with certain functionality and value attached. The structure will be described in terms of:

- Architecture;
- Structure of nodes and links;
- The associative structure that results from the nodes and links;
- The content in terms of the different types of media involved;
- The functionality that results from this structure.

## 2.1. Architecture

Like all computer systems, hypermedia systems have a specific architecture. This architectural model, however, is only a generalised model and may be customised to benefit specific system needs.

Generally, hypermedia systems consist of three architectural levels or layers:

- A presentation layer which is also the client-interface and which is usually a Graphical User Interface (GUI) (Gronbaek, n.d.);
- The second layer is called a Hypertext Abstract Machine (HAM). It manages the nodes and links and thus the structure of the information. The heart of a hypermedia system is the Hypertext Abstract Machine (HAM) (Andersen, 1999; Gronbaek);
- The last layer or database layer is where the information is physically stored. The database layer can range from simple, flat text files to sophisticated structures like relational databases, for example SQL server or Oracle (Nielsen, 1995, p. 131; Gronbaek, n.d.).

The operations of each layer are largely transparent to the other

layers which makes layers modular and weakly coupled. The operations of the system as a whole are also transparent to the user. The result is a set of seamless interfaces between layers and between the system and the user.

A hypermedia system can therefore be defined as an information system, which provide the possibility of non-sequential access to information through a network of nodes connected by links (McRae, n.d.). The nodes and links are the essential features of the hypertext structure.

## 2.2 Structure of nodes and links

A node is a self-contained, modular unit of encapsulated information. Each node should present a single, unique idea or piece of information (Nielsen, 1995, pp. 50, 309). The information inside a node can consist of various different media or functionality. A node contains information or content that can be related to the information or content of other nodes. This can be done by linking the node to other nodes. These links provide continuity between nodes. Links are very important because in connecting nodes, structure emerges. Linking or referencing provides structure to an otherwise fragmented group of nodes. Links give the content in a node a particular context within a larger hypermedia system (Marshall, 1995, p. 88). Links are used to form associations between different nodes of information in a hypertext structure, thereby creating an integrated frame of reference information. These different nodes can reside in the same document or can be different documents altogether. In addition to facilitating the making of associations between documents or nodes, nodes and links can also be made active by adding behaviour to them. This is achieved by using programming and scripting languages (Jetly, 1999). Links can also point to applications or embedded components.

Links can basically be grouped as:

- Referential - which is a reference from source to destination with no relation to structure between them;
- Parental - which involves a hierarchical organization of information and where links provide references between parent and child nodes (Jonassen, 1989, p. 8).

Within these two broad categories, there are also different types of links with different functions. For instance, an annotation provides only a small portion of additional information whereafter the user returns to the primary material. Unidirectional links traverse from one node to another and do not return to the node that initiated the link as in the case of annotations. The new node becomes the active node. Bi-directional links work technically the same as unidirectional links, except that there is a link back to the source document (Ashman, 1994). Links can be defined explicitly. Explicit links are predefined by the author and do not change unless changed explicitly. An implicit link is defined automatically at run-time by adding some programmatic conditions or behavior to a link (Definitions of concepts, 1997). A link of this type can be made to point to references for different conditions. These links are known as super-links.

Nodes can further be linked together and referenced by a single name, thus forming a composite node that represents a composite concept, which in turn can be related to other composite nodes (Balasubramanian, 1994). Another advantage of being able to link nodes is that the same information can be molded into different structures without making changes to the content. Therefore, when the content of a node is designed to represent a single idea, nodes will be modular, avoiding duplication (Nielsen, 1995, pp. 50, 309). Characteristics and functionality of nodes or objects

can be inherited instead of being recreated. Hypermedia systems can be developed to use object-oriented methods like operation overloading and inheritance (Nielsen, 1995, p. 264). An important issue concerning this however is to maintain balance between the size of nodes and fragmenting information.

## 2.3 Associative structure

The combination of nodes and links results in making the information in a hypertext system an integrated whole. It has the effect that in a hypermedia system information is organised as an associative network of nodes and hyperlinks that link these nodes. As a result the emerging structure is non-sequential and non-linear (Aedo, 1994, p. 111). This structure is free from the linear structure that is dominant in other media. Because of the associative structure of hypermedia technology, systems can be developed that are structurally very flexible. Any structure can be accommodated, ranging from rigid structures like strict hierarchies to structures with crossover links between hierarchies and even to systems with no structure at all. Information in nodes can have one-to-one, one-to-many or many-to-many relationships. It is this inherent flexibility that makes it possible to develop systems that "externalize the structure of subject matter and represent the information as it is stored in human memory" (Jonassen, 1989, p. 13). In terms of structural variety, hypermedia can be categorised as having a multi-dimensional structure.

## 2.4 Functionality

Hypermedia technology has all the conventional functionality that other media have in terms of organising information including:

- Information structuring
  - Table of contents;
  - Overviews;
  - Guided tours.
- Search functions
  - Indexes;
  - Keyword searches;
  - Boolean operators;
  - Filters;
  - Path history;
  - Term-weighting (Ashman, 1994).

Apart from this, hypermedia technology also makes it possible to structure the same information in a variety of ways for a variety of functionality that results in much flexibility (Eklund, 1996).

The structural flexibility of hypertext makes it possible to navigate through the information in the system. Navigation through hypertext is as flexible as the flexibility in structure (Eklund, 1996). Navigation ranges from unstructured exploration or browsing to guided navigation where the user is constantly kept up to date of where in the structure of the system he or she is. This gives the user the option to explore new nodes in context of the whole body of information as well as to backtrack to nodes already visited. This functional flexibility contributes to enhance applications, increase comprehension and enrich context (Bieber, 1995, p. 28). Hypermedia technology is also not limited to its inherent functionality. The functional framework, for example, can be extended or customised using programming languages like C++, Visual Basic, Pascal and Java.

## 2.5. Media

The multi-dimensional structure of hypermedia technology that was mentioned earlier, can further be extended in terms of the different types of media (called multi-media) that can be used to represent information. These different levels of structuring information and the variety of media available can be integrated seamlessly into a system (Narayanan, 1997).

Although acquiring information or communicating using text as a medium is efficient, the material is always subject to a certain interpretation by the reader. This interpretation varies because of a lack of knowledge of the domain, the situation, the background associated with the subject material and the interpreter's background or frame of reference. Using other media in conjunction with text may give the reader a more complete perspective because the subject matter is viewed from different perspectives. Information can also be re-ordered and re-read.

As far as media is concerned, hypermedia technology combines text with audio, animation, graphics and video that results in a higher bandwidth of information being recorded (Roth, 1994, p. 165). For example, when a specific domain has to be described, photographs, drawings, diagrams, comments, text descriptions, video recordings, audio recordings and animated simulations can all be used. Saffo (1997, p. 97) has the following to say about hypermedia technology: "Seeing that hypermedia systems use information in different types of media, dimensions and structures, this will help connect the symbolic universes of our creation with the physical world".

Hypermedia technology can be extended even further with virtual reality technology. Virtual reality is in essence an extension of hypermedia technology (Weiss, 1998). This can be seen on the World Wide Web where VRML (Virtual Reality Modeling Language) is already being merged with HTML (Hypertext Markup Language) and

JAVA to create an environment that is more dynamic, interactive and user-friendly.

Virtual reality increases the sensory breadth of a hypermedia system in that it promises to integrate touching with hearing and seeing in a hypermedia system. In full immersion the user's whole body is the interface to the computer, which responds to human behaviour, thereby making the interface very intuitive and easy to use. With virtual reality, the user is also not restricted to the fixed size of a two-dimensional screen, but is central to a three dimensional environment that has virtually no limit in size. It is also inclusive, interactive and happens in real-time. The user becomes in effect part of the virtual world and can affect what happens in this world by manipulating virtual objects and by moving. Virtual reality is a very intense, immersive, active and believable experience (more so than any other medium). "Virtual reality is the first conceptual, almost intuitive computer system" (Sherman, 1992, p. 71). Currently, virtual reality does not offer full realism, but with future advances in graphics, screen resolution and CPU power, virtual reality environments will become increasingly realistic. "VR, with its augmented reality, allows a smoother, more controlled transition from virtual to real and back" (Heim, 1993, p. 128).

## 3.Human orientated characteristics

### 3.1. Information structure

Hypermedia technology can be said to present a multi-dimensional functionality, which is defined, in terms of the different levels of abstraction into which information can be structured. The structuring ranges from high levels of abstraction, that accommodate abstract conceptualisation, to low levels of abstraction which amount to concrete experiencing of events

(Vatcharaporn, 1994, p. 102; Nielsen, 1995, p. 131). Information can also either be structured, or have no structure at all. In other words, a system can be constructed so that it either supports unconstrained searching that offers free association between different items of information, or is tied to problem solving for deep understanding (Thuring, 1995, p. 57).

Hypermedia structures can show divergent points of view in context (Spiro, n.d.). These structures are subject to a variety of interpretations. It is important that this variety is allowed because it accommodates users with different individual preferences as far as information content and structure is concerned. Information can be added to such a system so that a specific structure is not forced upon the reader, but rather left implicit to be formed when and how it is needed. A hypermedia system can be designed specifically to accommodate the semantic network of a particular user or to resemble a specific subject matter. A system can also be developed to be open-ended. Information can be internalised in a mutually constructive and not just a receptive way. Parallel or lateral structures of information can be provided to suit different users, for example:

- Logical structures to show the semantic relationship between different items;
- Pragmatic structures to show certain views of relationships that may either emphasise or minimise aspects of the logical structure;
- Dynamic structures to show the interaction of relationships over time (Woodhead, 1991, p. 85).

When using an application or searching for information, users normally follow references, thereby building up the necessary context. In terms of a hypermedia system, the information structure within the system can be hierarchical which is valuable for a high level overview or meta-information and have web or

network structures for micro-hierarchies in the system. This structuring is useful because it provides manageable structures in large applications and documents. Users can thus immerse themselves in the local meaning of a sub-system without losing a global perspective of the whole system. At progressively higher levels in the system hierarchy, increasing abstraction occurs, thus decreasing the complexity of the system overall. Such systems accommodate users to expand to lower levels of abstraction or move to higher levels of abstraction. Users of hypermedia applications can also browse through unstructured information and add to or create their own abstractions. This feature of hypermedia technology enables the user to expect and formulate relationships in information.

These features make it possible for users to be lead or to search by themselves. It is up to the user to decide which paths and at what depths to explore, thereby giving a user direct access to the content and interconnections of an information domain (Bieber, 1995, p. 28). It also lets the making of strategic decisions remain with the user. This enables users, rather than direct them, thereby facilitating the customization of individual needs. Many hypermedia implementations go even further by allowing readers to become authors by adding comments (annotations) and additional links to what they read (Ashman, 1994). Users can view and manipulate structure as well as content. It allows users to actively engage in constructing the meaning of text. In all these ways the hypermedia technology accommodates choice and change to a much greater extent than any other form of media.

### 3.2. Integration

Hypermedia is an integrating technology with its potential to unify diverse media, tasks, information structures, applications,

software, hardware, users, technologies and geographic barriers (Jetly, 1999). This unifying quality provides for a seamless hypermedia environment with functionality that prevails between the above- mentioned components rather than being unique to each of them. This model provides the ability to both increase the quality of heterogeneous information and to increase the ease with which it can be used (Woodhead, 1991, p. 10). Hypermedia technology supports computing and communication as well as social scientific areas like teaching (computer-based training) and cognitive science. It is both an information and a communication technology.

Information can also be re-used in hypermedia systems (Spiro, n.d.). With re-use, information is placed in different contexts so that the same material is used in different structures, thereby building multiple relationships around a single piece of information (Nelson, 1995, p. 32; Spiro, n.d.). Re-use promotes efficiency because it reduces the making of mistakes, maintains consistency and less physical space is required (Garzotto, 1995, p. 74). Re-use also facilitates integration because different parts of a system are connected via a re-usable component.

Information can also be shared across multiple locations or machines. Hypermedia systems need not be geographically bound. A good example of this is the World Wide Web. The World Wide Web is platform independent. HTTP (Hypertext Transfer Protocol) is used on the Internet and links different hardware and operating system platforms to provide multi-platform nomadic computing environments (Schase, 1995, p. 72). Information units can therefore be re-used across technological and geographical boundaries.

### 3.3.Mind

Hypermedia technology is very much orientated towards the human mind in that it has striking similarities to the functional level models of neurology and higher level cognitive models of human associative memory. It also takes advantage of the human perceptual system, spatial and geographic memory and spatial intelligence. It thereby accommodates and facilitates concept formation. It also complements visual memory and supports the interpretive process, forming of abstractions and visualising complex structures. It also encourages critical thinking (Spiro, n.d.). People structure information by forming links of associations, sets and composites in order to handle large amounts of information. In the process, emerging patterns can be detected. These functions and characteristics are principles of human learning, writing, collaborating and thinking. It comes together to promote better understanding when humans collect, read and analyse information in light of a problem to be solved or a specific task that must be done (Marshall, 1995, p. 93, Chun, 1995, p. 97). This whole process or set of processes resembles fuzzy logic, which closely relates to how people do problem solving. To further illustrate the cohesion between the human mind and hypermedia technology:

- The human mind is perceived as a system's phenomenon (Capra, 1997, p. 55);
- The brain operates by massive connectivity (Capra, 1997, p. 70).

In the process of problem solving, a person acquires knowledge about the problem to be solved that is vital to the quality of the solution. Knowledge acquisition or epistemology seems to evolve into more concrete or empirical ways of knowing. From there the change to evolutionary epistemology (Spiro, n.d.). This new development integrates well with the theory of constructionism. The constructionist views gaining knowledge in a reflective way, thereby creating a feedback loop in the process.

The learner constructs knowledge rather than being instructed. Initial instruction is merely the frame around which one has to construct knowledge through interpreting experiences. "Rather, because knowledge will have to be used in too many different ways for them all to be anticipated in advance, emphasis must be shifted from the retrieval of intact knowledge structures to support the construction of new understandings" (Spiro, n.d.). Every person's understanding of reality is therefore unique. Therefor, knowledge acquisition is a process of design.

In light of what is mentioned above, hypermedia information can have major advantages over information in other media that do not have this kind of support. Hypermedia technology makes it possible to handle large volumes of information quite effectively, because "coherence (constructing a mental picture that correlates with facts and relations) can be increased while the cognitive overhead can be reduced" (Thuring, 1995, p. 65). This is possible because the connection density of individual information items can be increased up to the mental capacity of the developer or reader.

Hypermedia technology can thus be said to stimulate rational thinking as well as creativity.

### 3.4.Communication

When different people collaborate in solving a problem or developing a system, it is very important to be mutually constructive in order to be successful. To illustrate, "people differ in how they approach learning of new ideas and concepts while solving problems" (Vatcharaporn, 1994, p. 101; Young, 2000). In accommodating these individual differences, a person can work with information in a way that suits him/her the best.

However, the results of these different ways of working have to be reconciled for the collaborative process to be effective.

Knowledge cannot be communicated precisely in an instructive way because each individual's interpretations, experiences and beliefs differ. However, we need to share common knowledge about the objective world. This common knowledge is constructed through our interactions with one another. "Research has established the ability of hypertext for argumentation" (Jetly, 1999). People must accommodate and integrate other peoples' interpretations and ideas into their own frames of reference about reality. Knowledge sharing follows a social path of back and forth negotiation, rather than just a one way transmission of information. This process has many similarities to the constructionist process. This way the reader can communicate with his or her as well as other peoples' ideas, thereby deepening his or her understanding. This way of acquiring knowledge is effective because the reader is not just passively interpreting information, but is an active part of the process.

Hypermedia technology accommodates and assists the seamless integration of different kinds of knowledge in the same system, thereby providing a colourful picture and different angles. Presenting the same material from different points of view not only accommodates different perspectives, but also enhances an application's richness. The valuable implication of this is that a person can have different views of the same material and therefore get a more holistic perspective of the material and other people's understanding of the material (Jetly, 1999).

Therefore, hypermedia technology not only offers cognitive flexibility in accommodating different perspectives, but also offers the communication of information between people. This quality facilitates social interaction during problem solving

(Spiro, n.d.). As Jones states: "hypermedia technology has, from a philosophical point of view, enormous potential for putting texts into contexts and for generating imagined conversations" (Jones, 1992, p. 143; Star, 1995, p. 152). Hypermedia technology provides a rich contextual bridge for communicating information effectively because it presents information that connects simulated environments (including well known cultural symbols) with abstract word symbols. According to Bottino (1994, p. 310), "it is possible to create representations that are not completely arbitrary, but preserve a strong analogical link with the related objects. These representations could act as mediators between the problem situation and its meanings, ideas, relationships and processes".

### 3.5.Usability

Hypermedia technology shields the user to a great extent from the underlying implementation and technological detail that is involved, by making the user-interface largely transparent to the underlying implementation (Nielsen, 1995, p. 311). This promotes ease of use because it does not require a high level of technical skill to develop a hypermedia system. As far as using a hypermedia application, a user only needs minimal familiarity with the technology. In this sense hypermedia technology bridges the gap between experienced and inexperienced users.

Applications also have very intuitive interfaces that users can relate to and offers close man-machine coupling (Bell, 1997, p. 32). For example, through hypermedia technology on the Internet, a user is not just able to access any document, but can also start an application automatically by just choosing a specific file, for example a text, spreadsheet or database file etc. Although the Internet is not a hypermedia application but rather enables hypermedia technology to function in the environment, the

arrival of the World Wide Web caused the use of the Internet to increase exponentially. The World Wide Web is a hypermedia-based system (Andrews, 1996). Software market leaders like Microsoft are seriously incorporating hypermedia technology into their products. It permits users to concentrate on the information content of objects and not the mechanics of acquiring it.

There are also certain advantages to using hypermedia technology as opposed to other information processing technologies or techniques. In using hypermedia technology, users acquire:

- More cognitive, meta-cognitive advantages (Spiro, n.d.);
- Richer and better connected knowledge (Spiro, n.d.);
- More collaboration and planning abilities (Andersen, 1999);
- Greater creativity;
- Better organising abilities (Jetly, 1999).

## 4. Problem characteristics

### 4.1. Uncertainty

The support that hypermedia technology provides for change and non-linearity can and often does cause uncertainty. Because of the structure or lack of a definite structure of a hypermedia system, a user can feel lost. This, combined with the fact that there are no physical attributes, like the size of a book, to guide a user, can hamper the effective use of a system (Nunes, n.d.). This shortcoming, however, can be overcome to a great extent by good design and new advances in the technology.

### 4.2. Interpretation problems

By arranging information into nodes and by linking these nodes, information is subject to unique interpretation, as is the case with other media. However, if more people collaborate to the structuring of such an information base, interpretation can be generalised and therefore become more objective.

## 5. Conclusion

Hypermedia technology accommodates and facilitates indirect communication. This level of communication is communication through information (structures, representations and other people's perspectives). Indirect communication in hypermedia technology accommodates the human mind because firstly, it integrates the views of people with different perspectives. Secondly, coherence (constructing a mental picture that correlates with facts and relationships), can be increased while the cognitive overhead can be reduced. Lastly, representations preserve a strong analogical link between related objects in a problem space. The user and the technology accommodate each other which benefits the integrated system of both the user and the technology. The capabilities of the technology as well as the user's thinking and learning abilities is enhanced. The whole then becomes greater than the sum of its parts.

Hypermedia technology essentially facilitates better knowledge transfer because of more effective communication between people and information than other media. The result of this is that it also facilitates communication between people where the mode of communication is other than direct person-to-person communication.

The next chapter will concentrate on the characteristics of software and software engineering.

# Chapter 3

## Characteristics of software engineering

### 1. Introduction

Whereas the previous chapter focused on the characteristics of hypermedia technology, this chapter will focus on the characteristics of software and software engineering. Software engineering is the discipline associated with the development of large-scale software products or systems (Andersen, 1999). It is a coherent activity and has technical, social, cognitive, organisational and cultural aspects to it (Winograd, 1995, p. 67). The quality of a software product is measured in terms of correctness, reliability, robustness, performance, user friendliness, maintainability, evolvability and re-usability (Ghezzi, 1991, pp. 19-35, Jetly, 1999). "There are, however, problems associated with the development of quality software. These problems are caused by the character of software itself and by the failings of the people developing it" (Pressman, 1993, p. 19).

Through the years, much has been done to build software engineering foundations to improve the development of complex software. However, the problems to be solved have also grown rapidly in complexity (Leveson, 1997, p. 130). What is problematic, is the trend that with distributed and scalable systems, more people are using the same system. Software is also increasingly embedded into other systems, and systems, containing software, are increasingly linked to form an interconnected whole (Boehm, 2000, p. 28). This situation adds considerable responsibility to the developers of a system. Outsourcing is another major trend. Although outsourcing makes sense from a

resource management point of view, it also means that developers of systems do not have the domain knowledge that in-house developers usually have.

There are five key dimensions to software engineering:

- The people that are involved in the process. They are business people, developers and end-users;
- The software engineering process;
- The software system or application;
- The technology used to develop and implement the system;
- The environment where the system will function as part of a bigger process.

"Of these dimensions, people have more impact on productivity than any other factor" (McConnell, 1996, p. 12).

Almost all software engineering problems relate to people. The most critical ones involve domain knowledge, requirements, communication and coordination. Major problems largely disappear when these aspects are engineered effectively (Hoc, 1990, p. 262; Pressman, 1997, p. 65).

The software engineering process has certain characteristics, which define the limits, the essence and the problems associated with the software engineering process. In this chapter, every characteristic will be defined and described individually. However, these characteristics are not independent of each other. They are related and influence each other. These relations will therefore be mentioned in each definition and description by referring to other characteristics when necessary. Firstly, each characteristic and the problems surrounding it will be described. Secondly, solutions to these problems will be suggested. Some of the problem characteristics that were referred to in chapter one

are put into more general categories for more effective organisational purposes.

Six general characteristics can be identified and the structure of this chapter revolves around them. These characteristics are - Software engineering:

- Is complex;
- Has an element of uncertainty;
- Is a non-linear process;
- Is a multi-disciplinary process;
- Is a human-orientated process;
- Is a communication process.

## 2. Software engineering is complex

Software is based on intellectual content. This gives software a unique and highly flexible nature (Boehm, 2000, p. 33). Software can not be measured in the same way as the products of other engineering disciplines. The so-called hardware engineering disciplines use components that obey physical laws and limits, and can be measured using these limits. The limits software has, has more to do with human abilities and the accuracy of the information available (Pressman, 1997, p. 65). For these reasons a software system has a fluidity that is difficult to conceptualise and understand.

In addition, "Computing is the only profession in which a single mind is obligated to span the distance from a bit to a few hundred megabytes, a ratio of 1 to 10 to the power of 9, or nine orders of magnitude. This gigantic ratio is staggering. Compared to that number of semantic levels, the average mathematical theory is almost flat" (McConnell, 1993, p. 774; Jetly, 1999).

There are however tools and techniques to help conquer these complexities. Designing and modeling system components and the relationships between them make the system visible. Decomposing and analysing individual components aids understanding because a person can then concentrate on smaller portions at a time (Pressman, 1997, p. 281). Designing and modeling on the one hand, and decomposition and analysis on the other, are complementing sets of activities of the same interactive process that can be used on any level. "We do this by imposing on the software engineering process the discipline that nature imposes on the hardware engineering process" (Leveson, 1997, p. 129). If, however, it can be accepted that the problems with building complex systems are rooted in people's limited ability to handle complexity, then our techniques and tools need to address this limitation more than any other. Complexity as a characteristic of software and the software engineering process can also be defined in terms of all the other characteristics, because they all influence complexity directly (Roth, 1994, p. 164).

## 2.1. The scale factor

No aspect of software engineering has a more profound influence on complexity than the size of a project (Glass, 1995, p. 27). Size amplifies complexity in all other characteristics and all characteristics influence complexity. Large projects that are beyond the handling of an individual or a small group are difficult to develop because the complexity involved in the development of large projects increases exponentially (Kraut, 1995, p. 69). Except for a minority of systems that are very complex regardless of size, it can generally be accepted that the difficulty of developing a software project is directly proportional to its size. The reason for this is very much a human orientated problem. The larger the number of variables people have to take into account simultaneously, the less

efficient they tend to become. The levels of abstraction also increase proportionally with increase in project size. Because of these reasons, people find it difficult to visualise a large system as a whole. Specifications are also further removed in terms of abstraction for large systems than for small ones.

One way to conquer this complexity is specialisation. Developers with a lot of experience in a specialised field can almost develop good systems intuitively (Ghezzi, 1991, p. 518). Although this way of working results in some success in software engineering, it also presents another problem. Many people are involved in the development of a large system. To make a group of highly specialised developers in different fields work together effectively, requires good coordination and communication (Glass, 1995, p. 40). Therefore, apart from the inherent difficulties discussed in the previous paragraph, members of a development team have to communicate effectively in order for efficient development to take place. This does not present much of a problem if there are only a few team members. However, in large development teams the total number of communication paths amounts to a level of complexity that can be termed as chaotic. This is because the number of communication paths is exponentially proportional to the number of team members (McConnell, 1996, p. 28). To add to this, the quality of communication in development teams is generally poor.

The difficulties inherent to developing large systems result in different perspectives of the problem and solution (Vatcharaporn, 1994, p. 101). The problem of different perspectives combined with the fact that developers generally lack good formal communication skills creates the possibility that communication can be a major factor in impairing development efficiency. Developers also tend to spend more time on reading about, rather than on formal communication of a problem. In other words, communication via documentation is very important.

The communication problems that result because of size can be categorised as:

- Communication of developers involved in different phases of development;
- Communication of developers across different levels of abstraction;
- Communication of developers with different perspectives of what should be done and how it should be done. If team members refuse to acknowledge one another's perspectives, a breakdown in communication occurs (Brooks, 1987).

### 2.2. What is needed

What is needed is some form of representation that is flexible enough to allow developers to understand the system as a whole while working in detail on parts of the system. Developers need to be able to cope with huge amounts of information and variables and the relationships between them (Andersen, 1999). What is also needed is information about the system that different developers can understand. A medium is also needed to allow developers to communicate information regarding the problem efficiently. Documentation is the medium that is most frequently used.

## 3. The software engineering process has an element of uncertainty

The software engineering process has an element of uncertainty because some aspects of it are unpredictable and also because the software engineering process has a non-linear structure. Because of these inherent uncertainties, there are also great risks involved. To control these risks, risk assessment and management

is needed. However, there are also uncertainties in regard to these aspects which can be understood from Heisenberg's Uncertainty Principle which states that it is impossible to measure the exact position and momentum of a particle at the same time. "When we apply this principle to the software life-cycle and to information assurance, it means this: We cannot simultaneously measure the risks associated with software and information assurance when no protective actions are taken and measure the efficacy of deploying risk assessment and management on the system because the system has fundamentally changed. Moreover, software development is a dominantly intellectual enterprise, and the very attributes that make software so attractive (flexibility, low tooling cost, ease of reproduction) also make it hard to measure and quantify" (Longstaff, 2000, pp. 44, 45). This situation results in increased complexity to problem solving because "Uncertainties complicate the problem" (Longstaff, 2000, p. 46).

### 3.1. Unpredictability

Software engineering is not a routine activity. Although there are portions of every system that are standard or common to most systems, every system is mostly unique with aspects that are not clearly understood at first. This gives the development of software an unpredictable nature that causes a lot of uncertainty during the process (Boehm, 2000, p. 27). To solve a problem using software, a developer's or developers' interpretation of a customer's problem must be translated through several layers of abstraction into a machine readable format that is very difficult for people to understand. This becomes apparent in large and complex systems. A software system can therefore be seen as a communication process between the developer and the customer about the problem, using the computer as communication link.

These factors further contribute to the inherent unpredictability of software and its development.

Apart from the inherent unpredictability, unpredictability also results from inefficient communication, especially in large development teams and large, complex systems. This results in making unnecessary changes to the system that might and often does result in compromising system integrity. Because changes in software cannot be prevented entirely, specifications must be flexible enough to anticipate and accommodate change (Ghezzi, 1991, p. 65; Pressman, 1997, p. 288). There will always be information that is not known beforehand that will only become known during the process of development. However, incompleteness can be reduced with more effective communication. Much of the information that needs to be known before development starts, is available (Pressman, 1997, pp. 104 - 106). Developers just have to find it using good communication practices. This will lead to more complete and better quality specifications and better systems. Changes due to bad communications can be greatly reduced, which will have an impact on system stability. These changes are not just limited to user requirement specifications, but also involve specifications generated during other phases of development.

### 3.2. What is needed

What is needed is a documentation and representation medium that makes problem and development information more accessible, comprehensible and stimulates good communication.

## 4. The software engineering process has a non-linear structure

Non-linear systems are systems that are characterised by chaotic processes, involving feedback. In feedback systems, small changes in initial conditions lead to significant differences in ultimate outcomes. Non-linear systems are also characterised by a high degree of component or sub-system interdependence. Change in any individual factor has an effect on the system as a whole (Olson, 1993, p.35).

Chaotic systems generally share the following features:

- Areas of order - areas of well-defined structure, order and pattern;
- Areas of disorder - the boundary between one area of order and another is usually disorderly;
- Self similarity - the whole and the parts of the whole look the same; for instance in design, the layering of abstract data types, designing and coding are structurally the same;
- Self dissimilarity - areas of order can also differ, for example analysis (taking apart) and design or synthesis (putting together);
- Response to changes - complex systems are very sensitive to change and because this process is not linear, changes have unpredictable results (Olson, 1993, p. 55).

Software engineering is such a feedback, chaotic problem solving process from beginning to end. There is constant feedback during all the phases of the software engineering process. This makes software engineering components interdependent. This interdependence is not only true for components of systems being developed, but also for the components of the software engineering process, like the different software engineering phases, people and other systems involved. Interdependence in software engineering is evident from observing people's dependence on communication.

The correlation between well-defined requirements and project success, has led many people to believe that requirements gathering is the most significant factor in the success of software engineering (McConnell, 1996, p. 236). Gathering requirements is a difficult activity because interpretations vary. The process is dependent on feedback from sources (customer, developer) that are constantly changing. Both the developer and customer adjust their understanding according to each other's efforts (Pressman, 1997, pp. 272 - 278). To strive for ordered development, feedback-loops must be kept as short as possible during development. If customers can only give feedback on the completed system, the feedback-loops in development will be very long and the process will be too chaotic to be successful. The result will therefore not be what the customer wants or needs. However, when customers are closely involved in the development process - in other words when constant interaction between developer and customer takes place - feedback-loops will be shorter and development will be more orderly and successful. If the user is kept up to date, development remains synchronised with the user's knowledge and requirements (Olson, 1993, p. 27). This emphasises the utmost importance of effective communication during the software engineering process. Communication of this nature also occurs between developers throughout the development process and the same rules, as was mentioned above between developers and customers, apply.

### 4.1. What is needed

Because of the non-linear structure of the software engineering process, it has to be accepted that change is an inherent part of the process and should be accommodated. However, as mentioned before, effective communication and working flexibility can solve

many issues relating to problems associated with non-linear systems.

## 5. Software engineering is a multi-disciplinary process

Software engineering consists of a variety of components. These components can be loosely grouped as: software, management, business and people. These components are all fields of study in well-established scientific disciplines (Boehm, 2000, p. 31). Software engineering is a relatively new science compared to more established scientific disciplines like physics and engineering. Software engineering is very much influenced by the scientific fields it involves. All these influences give the software engineering process a multi-disciplinary nature. This multi-disciplinary influence gives software engineering a more complete perspective to problem solving (Pfleeger, 1999, p. 32). This is necessary, because large-scale systems must be perceived from a multi-disciplinary perspective. However, the disadvantage of this multi-disciplinary influence is that people in different disciplines do not communicate effectively, because different disciplines, and therefore the people involved, are isolated (Longstaff, 2000, p. 44).

The influences on software engineering that took place over the past four decades mainly came from the so-called hard sciences like physics, chemistry, engineering and mathematics. This resulted in a situation where software was, and to a great extent still is, regarded as an engineered product rather than a human product derived from intellectual content in a social setting (Leveson, 1997, p. 130). As for the future, the so-called soft sciences, including such diverse fields as psychology, neuro-physiology, sociology, philosophy and economics, will have a major impact on information technology. In fact, this impact promises to be much greater than that of the hard sciences.

However, information technology will not just be influenced by all the mentioned disciplines, but in turn it will influence them as well. The influence of information technology on the human sciences, in turn, also promises to be greater than on the hard sciences.

Natural existing systems, like genetic systems and the evolutionary process, also have an impact on computing. In the area of evolving systems, of which neural networks and fuzzy systems are part, genetic programming and genetic algorithms have come onto the scene. These systems employ the principles of reproduction and mutation where many individual programs or potential solutions reproduce and mutate or die in striving for survival of the fittest solution (Boehm, 2000, p. 31).

To illustrate the extreme diversity of the software engineering field even further, software engineering is at the same time a science and an art.

Software engineering is first of all a scientific discipline. It is a problem solving discipline that has a definite structured process and produces standardised, quantifiable, objective results (Shaw, 1996, pp. 10 - 12).

On the other hand, software engineering is also an art (Glass, 1995, p. 92). The reason for this is that the complexity of the systems to be designed often transcends any detailed analysis and specification. Even with a completely specified system, it may not operate the way a person thought it would, especially if it involves user interaction. Intuition plays an important role because often, what "feels right" really is the best solution, even though it might not follow the rules of convention. Experimental studies support this in so far as any creative activity involves an opportunistic mental process (Ghezzi, 1991,

p. 518). The solution to a problem is almost never a simple linear progression from the original requirements. The problem solving process follows a recursive pattern between analysis and design, wherein analysis and design are completely dependent on each other. Feedback is used to integrate these two activities into a functional whole and thereby used to evaluate and modify (Capra, 1997, p. 127). Because this is a creative process and therefore very subjective to each individual, good communication is essential.

These scientific and artistic aspects are interdependent. In developing a system as a solution to a problem or requirements, the software engineer works simultaneously as a scientist and an artist. Software engineering also has another uniqueness in its diversity. Most of the time the problem to solve is in another domain with which the developer is not necessarily familiar. Very often this other domain is another field of study or even another discipline as diverse as accounting, physics or psychology etc.

## 5.1. Phases in the software engineering process

The application of the software engineering process involves "achieving a fit between the people, the discipline, the problem and the organization" (Olson, 1993, p. 54). Apart from this, the software engineer must be able to adapt to the fastest evolving industry in the world. In addition to developing and maintaining applications, the software engineer must also keep abreast of new and changing technology. This means that a software engineer is obligated to do ongoing research. This in effect gives software engineering a development, as well as research characteristic, rather than just a development one.

The software engineering process consists of different phases with a different focus on problem solving in each phase. Although

the different dimensions and phases are essential in making the software engineering process manageable, what happens between the different phases and dimensions are just as important as the phases and dimensions themselves. The bridging of these phases and dimensions can only be accomplished through effective communication.

### 5.1.1. Strategic phase

Software engineering is generally part of a larger process and needs to be treated as such. The larger process is a discipline called systems engineering. Systems engineering focuses on a larger system, rather than just the software system. In fact, the software system is a component of this larger system. Systems engineering emphasises the system as a whole and the relationships between subsystems as the components of this larger system, rather than focusing on isolating the components of the system or subsystems (Pressman, 1997, pp. 234, 235).

Though the strategic phase is not directly involved in development, it is very important because some critical errors can be eliminated during the strategic phase. "Many of the critical insights in software engineering is not code-focused, but strategic and philosophical" (McConnell, 1996 p. xvi).

The strategic phase starts with identifying and prioritising all variables that are relevant to the problem situation. Information regarding these variables is collected. Typical variables involve the existing system to be replaced or changed, its integration with other systems and its users; economical and management issues; organisational philosophies, scope, risk and resources (Boehm, 2000, pp. 114 - 116). Brainstorming plays an important role in this phase. The strategic phase is mainly business driven and done by project managers, business analysts and systems

analysts. System analysts also take part in the development phase. Initial and global specifications are defined on the grounds of the strategic analysis of the problem domain and user requirements undertaken. An initial requirements specification is produced. Using the initial requirements, a number of solutions are proposed. Scenario planning and feasibility studies are conducted, tested and refined for each solution and in accordance with the user, the best solution is chosen (Pfleeger, 1999, pp. 36, 37). An estimation of the cost of development is done and presented to the customer. If the customer accepts, planning for the software engineering process is done by determining what the functions of the system to be developed is in broad terms. A high level or strategic design is done for the solution including hardware, software and people. Using this information, goals and strategies to meet these goals are set. On these grounds, development scheduling, resource planning and allocating, as well as risk assessment is done. The important decisions of what methodology and technology to use, are also made during this phase. The development phase can be defined as an implementation of the strategic phase.

### 5.1.2. Development phase

In the development phase of the software engineering process, development is done on the software component of the high level design in the strategic phase. Although development is done on the software component, it is not isolated from other components of the total solution, namely people, hardware, documentation and the problem domain (Pressman, 1997, p. 232). Using analysis and design activities, step-wise refinement and modeling of the system is done until the system is implemented or when maintenance is done successfully.

The first step in the development phase is to flesh out the initial requirement specification with detail (Pressman, 1997, pp. 272 - 278). Where the initial requirement specification were set up by management personnel in the customer organization, the detailed requirements must be gathered from operational personnel who are working in the problem-domain and who will be working with the new system. Prototyping, animation, natural language paraphrasing and CASE tools can be used to refine the requirements specification (Loucopoulos, 1995, pp. 131 - 136).

The development phase has logical and physical designs. Problem specific logical designs are still abstract, but less than strategic designs. The focus is on the logic of the problem and on the solution of the software system. This is an abstraction level that states the solution in problem-domain specific terms. Implementation specific logical designs are still abstract, but less than problem specific logical designs with an abstraction level that states the solution in implementation-specific logical terms. This lower level logical design models the system in terms of what should technically be done, but not how it should be done. Physical designs focus on how the system should be constructed physically and technically through coding and using testing as feedback mechanism for coding. Apart from feedback received from users, the compiler, that generates executable portions of code, also plays an important feedback role in the physical design.

### 5.1.3. Implementation phase

The implementation phase involves implementing the system in the problem-domain, doing testing with problem-domain specific data, training users in the use of the system and offering system support where small alterations are made to fine-tune the system.

### 5.1.4. Maintenance phase

Maintenance involves the software development life-cycle for existing systems. Maintenance involves making changes to the developed system because of changes in requirements specifications and program errors (Pressman, 1997, p. 32). Maintenance usually takes place on isolated portions of a system. Because system components are interdependent, making changes to isolated parts can have negative effects on the rest of the system. Developers, other than the original developers, also, more often than not, do maintenance on the system, which means that they do not have an in-depth knowledge of the system. It must be accepted, however, that specifications change and that changes must be made accordingly. However, it happens too often that maintenance is done for making changes as a result of bad or misinterpreted requirement specifications. This whole maintenance problem has the effect that generally up to eighty percent of development time is spent on maintenance (Ford, 1994, pp. 7, 135). This situation can largely be avoided by the effective communication of user and development requirement specifications.

In the maintenance phase, user support and configuration management is also done (Mazza, 1996, pp. 256, 257, 365).

## 5.2. Problem solving activities in the software engineering process

Analysis and design activities, in one form or another, are present in all the phases of the software engineering process. Analysis and design are integrated and dependent on each other and forms an iterative process. Apart from building a solution through design, the result of the design process is also used as a feedback mechanism to verify or disprove the analyst's

understanding of the problem or domain. This interactive feedback mechanism allows a developer to handle complexity that is not possible to conceptualise through analysis alone. This feedback process can be seen as a form of communication. The analysis and design process has both a top-down and bottom-up structure and looks at the system from both a high-level and low-level perspective. It is also a heuristic, non-linear process that involves some trial and error and experimentation (Capra, 1997, p. 127). This experimentation also holds true for methodology. Using multiple approaches enables the developer to choose and use the most suitable approach for a specific situation (McConnell, 1993, p. 163).

The objective of analysis is to investigate a problem in a domain. The analyst strives firstly to discover the essence of the problem by collecting data. Data is collected by interviewing users, reading documentation and observing users at work with the current system in the problem-domain. The data is then processed and analysed by the analyst to form an understanding of the problem that is to be used in design (Pressman, 1997, pp. 278 - 284).

With the understanding that was reached during analysis about the problem, a design of a possible solution to the problem is made. This design depends on what it is needed for. For example, designs in the strategic, development or implementation phases range from very abstract designs with a focus on system integration, to less abstract designs with a focus on the problem, to little or no abstraction with a focus on coding. A design, apart from leading to a solution, also reflects the developer's understanding of the problem and might stimulate further analysis.

## 5.3. What is needed

As described, there are many different disciplines, phases, tasks, activities and processes involved in the software engineering process. This multi-perspective influence stimulates creativity but also involves a need for good communication. What is needed is a medium to accommodate and integrate this variety.

## 6. Software engineering is a human-orientated process

Software engineering involves intellectual content in order to develop a system to be utilised by humans.

### 6.1. Software and the human factor

Software is based on thinking. In some ways, the computer and the software are strongly related to the human mind and thought processes. Software engineering has in a way similar goals and limitations to that of psychology. Just as psychology is a study of the soul, software engineering is ultimately about understanding other people's thoughts. "Software comes directly from the thoughts of the human soul. The best software often takes advantage of the creativity of the soul. Furthermore, it will forever be doomed to the psychological makeup of its creator" (Wood, 1998).

Because software is based on intellectual content, software systems also have a logical and social, rather than a physical character (Pfleeger, 1999, pp. 33, 34). A set of instructions is used to construct a software system. This set of instructions can be seen as a vocabulary, also called a programming language. The software engineer is in fact communicating a solution to a problem to a computer via a programming language. A computer can

respond to any vocabulary. The computer's vocabulary is, however, subject to human limitations.

## 6.2. Software engineering in general

Software engineering is a problem solving activity and software engineering techniques and tools are used to assist humans in this activity. Problem solving is chaotic, involving feedback and very soon, while busy solving the problem, "people loose track of the distinction between problem and solution. The two together become a new situation which is more complex than the original situation" (Olson, 1993, p. 55). To accommodate this evolving, complex process, multiple problem solving approaches have to be incorporated for flexibility with the emphasis on user-involvement. This aspect of problem solving is very important. In his book *Am I clever or am I stupid*, Neethling (1996), the creativity expert, starts by acknowledging the fact that people have different mind-orientations, which refers to their different approaches to problem solving. Neethling (1996), places the emphasis on the importance of accommodating, understanding and integrating these different mind-orientations in creative, collaborative problem solving.

Software engineering, being a problem solving activity, involves learning. The whole learning process, however, is an example of a non-linear system. An individual's understanding of the system evolves as learning progresses. "Because people are learning creatures and because learning involves feedback, it can safely be said that every human endeavor is non-linear in nature" (Olson, 1993, p. 171). In light of this, software engineering must be seen as a human orientated activity - "Our art is abstract, but has a profound emotional and social effect on our audience" (Lanier, 1997, p. 56).

Therefore, it makes sense that software engineering is packed with contrasts. Not only does it involve a variety of disciplines, as previously mentioned, but it also involves both high and low level cognitive processes. The high level cognitive processes include high-level problem solving and linguistic skills, which in turn involve concurrent processing of lower level cognitive tasks, like analysis and design with feedback. Another set of contrasts involves outwardly focused activities, like communication and the study of people and their requirements on the one hand, and inward, private activities like programming on the other hand (McConnell, 1993, p. 756). These contrasting situations cause many communication problems.

## 6.3. Purpose of software engineering

Everything in the software engineering process revolves around the user's problem and determining what the user's requirements are (Winograd, 1995, p. 71). Doing this, however, is not that easy because of human nature. Where humans are involved, there are always changes of mind taking place. Therefore, change is an inevitability that has to be catered for and the software engineering process has to be flexible enough to accommodate it. "This, however, is not a substitute for good communication, but an anticipation of change" (Roth, 1994, p. 163).

Taking change into account, the most challenging part of software engineering is conceptualising the problem. If this is done well, success is virtually guaranteed. This is why user involvement in the software engineering process is so important. Apart from satisfying user requirements, the resulting system must be easy to use, efficient, maintainable, portable and flexible (Olson, 1993, p. 48)

## 6.4. People as factors in the software engineering process

The software engineering process takes place within certain environments. Every one of these environments has a culture that is specific and unique to it (Star, 1995, p. 113). Apart from an individual's psychological perspective, the relationships between individuals in an environment create a culture that impacts greatly on development. This is reflected in the software being developed.

The general type of personality profiles of the members of a development team, must also be considered for the sake of better coordination and communication Neethling (1996). Whereas managers have an outward focus towards relationships, people and things, developers are inwardly focused towards their own ideas and the need for stimulation. Developers are also less formal in their methods of working than managers.

In terms of problem solving, managers tend to have a holistic view of matters. Developers on the other hand, focus more narrowly on smaller portions and do detailed analysis (McConnell, 1996, p. 240). If this difference in perspectives is left uncoordinated, it can lead to serious communication problems that might have an impact on the quality of the system being developed.

## 6.5. What is needed

When considering the influence that the human factor has in software engineering, combined with the knowledge of how developers' personalities and perspectives generally differ from those of managers and customers, what is needed is a medium that accommodates these factors for the purpose of better coordination and communication.

# 7. Software engineering is a communication process

## 7.1. Communication defined

A critical success factor in the software engineering process is the efficient communication so that everyone can reach a common understanding of what is involved. Communication is effective when the receiver of a message understands it as it is intended.

Communication is defined in literature as follows:
"Communication relies upon the capacity of members to project themselves imaginatively into the standpoint of others in order to comprehend the dimensions of the situation as a whole in terms of possibilities and actualities" (Langsdorf, 1995, p. 144). "Communication is the art in which social imagination allows one to take different perspectives of the same situation" (Langsdorf, 1995, p. 148). "Communication is an ongoing process that leads to the making of a linguistic product on the one hand and creative doing on the other" (Langsdorf, 1995, p. 204). Taking all these definitions into account, the essence of communication is that it must lead to understanding. Communication is also a non-linear process and part of the ongoing process of action and reflection. In software engineering terms communication also involves analysis and design.

## 7.2. Software engineering and communication

Software engineering is in essence a communication process. According to acclaimed computer scientist and virtual reality "guru" Jaron Lanier: "Information science [= software engineering - HC] will continue to reveal the unsuspected potential in relationships between human beings" (Lanier, 1997, p. 56). People

are in effect communicating with a non-human intelligence, which is a machine, and we are using programming languages to do so (Pressman, 1993, p. 19). "However, the concepts need to be examined in more than a machine context. What is needed is to look past the machines to the communication between people" (Summit, 1995, p. 114). Therefore, software engineering must primarily be seen as communicating with people and only on a secondary level with the computer. "When we treat computers as no more than conduits between human imaginations, grand vistas open" (Lanier, 1997, p. 56). What must be taken into account, however, is that although all forms of communication in software engineering are in essence between people, it also involves interaction between humans and machines; people (be they developers or customers) and the problem-domain; people and the problem at hand and lastly people and information.

## 7.3. Background communication problems

Not only is communication critical to the software engineering process and very often the source of problems, but it also has a powerful effect on the world surrounding software and the software engineering process.

There are harmful perceptions of the world of software that influence software engineering considerably. These perceptions that are created and communicated are:

- That computers, rather than people, should be emphasised (Hayne, 1996). This perception exists because of people's lack of knowledge about computers and the erroneous portrayal of a computer's abilities by the media and art world.

- The business perception that something must be produced as fast as possible. Everyone agrees that good software engineering practices and communication are essential (Boehm, 2000, p. 28), However, it is generally more important to business people to have a working product ready as soon as possible, so that it can be sold. Rapid development tools are therefore popular and they are very powerful in order to produce applications quickly. However, the quality of these products is often not so good (Boehm, 2000, pp. 29, 32).

- People are made to believe that technology solves problems (Boehm, 2000, p. 31). In reality, what is needed is sound software engineering principles geared towards solving the customer's problem. There should be less emphasis on technical aspects. The perspective of software engineering should be geared more towards the involvement of people and the relationships between them, since this is the most critical element in software engineering (Winograd, 1995, p. 68).

The bottom line is that people, especially customers who pay to have their requirements met, should be given honest, realistic expectations (McConnell, 1996, p. 243).

The communication structure and attitude in an organisation form another obstacle to the proper flow and use of information and knowledge. Good communication internally, as well as externally to users, provides a solution to most of these problems (Hoc, 1990, p. 263). Communicating well with users is especially important because it is the starting point of and pivotal to the development of any system. If any misunderstandings can be avoided or resolved early in the process, success is a near certainty. Involving customers in the development process enhances vital communication that results in better understanding and cooperation. In fact, in 1994 the Standish Group concluded

that user involvement is the most important factor in project success (McConnell, 1996, p. 236). "Software products are used or monitored by humans and the way that software is designed to interact with humans is a critical factor in whether the software is useful or not usable by them" (Leveson, 1997, p. 130).

## 7.4. People involved in the software engineering process

It is important to remember that people with different professional backgrounds, knowledge, perceptions and personalities are grouped together in the software engineering process and must therefore communicate with each other (Burgoon, 1994, p. 101).

When a single developer works on a project, communication is limited to interaction with the user. In this case communication is relatively simple and the emphasis is placed on development activities. With a team of developers, interaction becomes a weighty factor. When different teams are working together, good communication becomes a critical success factor. Not only are there many people involved, but also, inter-team interaction takes place through intermediaries, which results in indirect lines of communication. Communication is now no longer a one-dimensional process with one line of interaction between two people, but a multi-dimensional process that grows in complexity relative to the number of people involved, as well as to the number of indirect communication lines (McConnell, 1996, p. 28).

### 7.4.1. Project managers

Project managers manage a project throughout the software engineering process. Software, however, cannot be managed

entirely in a conventional way because of its nature and the complexities involved.

A project manager is involved in project planning and monitoring as well as managing the development team efficiently, which has much to do with assigning tasks to people that suit their skills set best (Ghezzi, 1991, p. 13). To further coordinate the cooperation of people in their assigned tasks as a team, the project manager needs to coordinate good clear communication between team members. This is important because members will engage in more informal communication that is not only very important in software development, but is characteristic of developers. If possible, a team must consist of the same people because "cohesion results in better communication" (McConnell, 1996, p. 288). Business analysts are only involved in the strategic phase and together with project managers, their activities are business-driven.

### 7.4.2. Developers

Development teams consist first of all of system analysts and programmers and often there is no clear distinction between the two roles. Therefore, a relatively new position called analyst programmer was created. Apart from these members, graphic artists, language specialists, interface designers, usability engineers and database administrators might also be involved. Team members often have different beliefs and perspectives on what should be done and how it should be done (Vatcharaporn, 1994, p. 101). However, having people with different viewpoints is not problematic. It is actually quite useful because this way the problem and its solution are pursued from various angles resulting in flexible, more creative solutions. By coordinating, and thereby integrating these views, everyone can have access to this information. This is important because members of a

development team are dependent on one another and therefore need to work towards the same goal (McConnell, 1996, p. 286). Developers also specialise in specific areas like research, technical programming and applications programming. Specialist areas can further be divided into specific technologies like back-end, middle-tier and front-end technologies.

Conventional methods of communicating are essential for high-level routine activities like high-level coordinating and planning. It is however less effective when there exists much uncertainty. This characterises software engineering at large, because very often there is much uncertainty and confusion with regards to user requirements and perceptions. A real danger lies in the tendency of developers to "treat requirements as being explicit and complete rather than as examples of a more general need" (Olson, 1993, p. 49). Developers communicate valuable information in much the same way as they do development - in an informal, unstructured and non-linear way. "Informal communication is needed here for coordination, but problematic because of the size of teams and projects" (Kraut, 1995, p. 70).

### 7.4.3. Customers or users

The customer or user requires a solution to a problem and the software engineering team has to provide the correct solution. The customer measures progress by being kept up to date with development proceedings. Users are also often used to work with finished parts of a system during development. Doing this, the user can give feedback on usability and on whether the system meets requirements (Olson, 1993, p. 27).

## 7.5. What is needed

All the people involved in the software engineering process have to come to a common understanding of what a system solution must be like. This can only be achieved through effective communication with all parties' interests and perceptions taken into account. What is needed is for development information to be in a format that promotes effective communication.

## 8. Conclusion

This chapter aimed to emphasise the critical importance of people and the interaction between them in the software engineering process. Communication envelops the software engineering process and also filters through every aspect of it. Most of the problems associated with the characteristics of the software engineering process can be solved by improved communication. Apart from direct person-to-person communication, communication also takes place through documentation. A medium for documentation is necessary that accommodates effective communication of structured, unstructured and dynamic information. It must also accommodate broader communication bandwidth as far as media presentation is concerned.

This chapter focussed on the general characteristics of software and software engineering. The next two chapters will focus on the functional aspects of software engineering in more depth.

# Chapter 4

## Documentation in the software engineering process and the processes it involves

### 1. Introduction

Software engineering, like all other forms of problem solving, involves information processing to a great extent. Analysing, defining, internalising information to become knowledge, communicating, documenting and coding are all information processing activities that take place during the software engineering process. Whereas the previous chapter focused on the characteristics in general, this chapter and the next will focus on the functional aspects of software engineering in more depth. This chapter will emphasise documentation and the information processing behind it as a vital part of the software engineering process. The following aspects will be covered:

- Fundamentals of human information processing and communication;
- What must be done in the software engineering process;
- Why documentation is needed;
- Problems in the software engineering process;
- What is needed in the software engineering process.

### 2. Fundamentals of human information processing and communication

Documentation is a product of human information processing. To understand the impact and effectiveness of documentation in the software engineering process, one has to realise that software engineering involves information processing by humans and