An Overview of Systems Design and Development Methodologies with Regard to the Involvement of Users and Other Stakeholders

SHAWREN SINGH AND PAULA KOTZÉ
University of South Africa

Systems development methodologies have been developed with the purpose of designing and developing effective information systems. This paper looks at different types of systems development methodologies with particular emphasis being placed on how the human component and other stakeholders are involved or catered for. One general critical flaw we have identified in these processes is that the interface design is not effectively conducted or implemented, but neither are the many other stakeholder issues addressed. This paper investigates this lack of attention and recommend some improvements.

Categories and Subject Descriptors: D.2 [**Software Engineering**]: Requirements Specification - Methodology; H5.2 [**Information Interfaces and Presentation**]: User Interfaces - *Theory and methods*
General Terms: Design, Standardization
Additional Key Words and Phrases: Human-computer interaction, Object-Orientation, Systems Development Life Cycle

1. INTRODUCTION

Almost half of software in systems being developed today and thirty-seven to fifty percent of efforts throughout the software life cycle are related to the system's user interface [Vanderdonckt and Harning 2003]. For this reason issues and methods from the field of human-computer interaction (HCI) affect the overall process of software engineering (SE) tremendously. Yet despite strong motivation amongst organizations to practice and apply both effective SE and HCI methods, major gaps still exist in understanding suggested practice, and how software is actually developed in industry, and between the best practices of each of the fields. There are also major gaps of communication between the HCI and SE fields – the methods and vocabulary being used in each community are often foreign to the other community. As a result, product quality is not as high as it could be, and (avoidable) re-work is often necessary [Vanderdonckt and Harning 2003].

The goal of the development of any information system (IS) is to make human endeavour easier and safer. Whitten et al. [2001](structured systems development life cycle authors) define an IS as an arrangement of people, data, process, information presentation, and information technology that interact to support and improve day-to-day operations in a business as well as support the problem-solving and decision-making needs of management and users. Satzinger et al. [2002] (object-oriented development authors) have a much narrower definition of an IS: an IS is a collection of interrelated components that collect, process, store, and provide as output the information needed to complete a business task. Both definitions are essentially correct. There is, however, an inherent defect in both approaches in that both the structured systems development life cycle (SDLC) and object-oriented (OO) approaches tend to ignore the human aspects during system development. A ‘user friendly’ interface is almost always slapped on to the developed system at the end or late in the development process. In fact the critical flaw, of the Satzinger et al. definition is that it leaves out the human (stakeholder) aspect of the entire system altogether.

This paper addresses this issue and compares various systems development methodologies. Section 2 of this paper looks at an alternative way in which evaluating systems development methodologies, keeping the human element and other stakeholders in mind, can be approached. In Section 3 we discuss information systems development. Section 4 give an overview of the traditional structured systems development methodologies, the object-oriented approaches, as well as development methodologies that focuses on the user or human element. Section 5 assesses the methodologies given the issues identified in Section 2. Section 6 gives an example of where the end-user failed to successfully complete his task, most probably due to an inappropriate development process, and Section 7 concludes with out proposals or suggested improvements.

Author Addresses:
S. Singh, School of Computing, University of South Africa, P O Box 392, UNISA, 0003, South Africa; singhs@unisa.ac.za.
P. Kotzé, School of Computing, University of South Africa, P O Box 392, UNISA, 0003, South Africa; kotzep@unisa.ac.za.

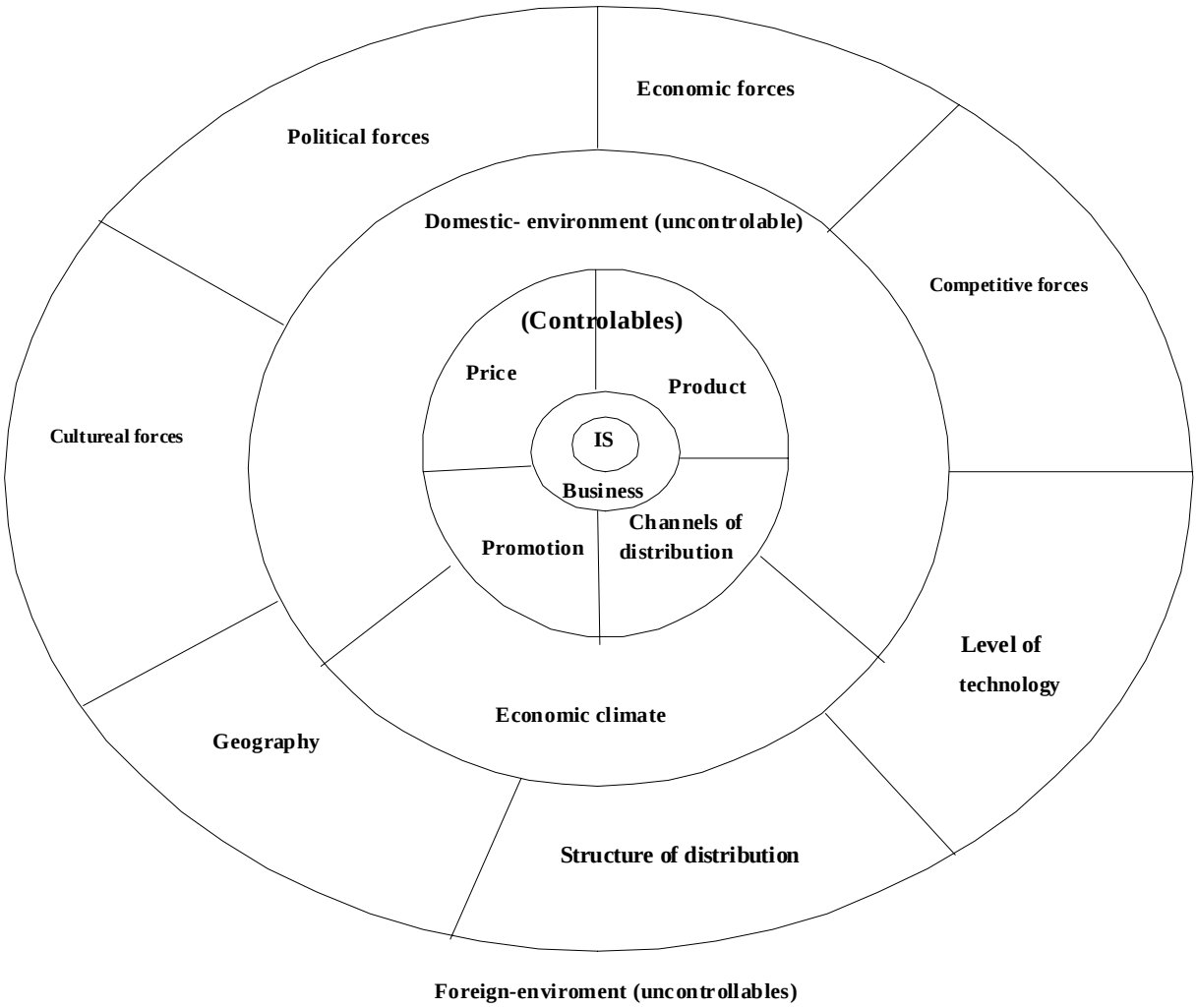

Figure 1. *The Business Environment (Adapted from Ricks [1993])*

2. THE BUSINESS ENVIRONMENT

The typical environment in which contemporary businesses operate, as illustrated in Figure 1, is complex and fragile. The inner sphere represents the heart of the business organisation (IS), the second and third spheres represents the organisation and its controllable variables. Outside those spheres are uncontrollable factors that affect the business organisation. These factors affect the competitiveness of the business organisation. For example, in the outer most sphere, the following aspects are beyond the direct control of the business, but severely impacts on its operation:

— Cultural and language forces: This can be illustrated with an example. In South Africa the car manufacture Volkswagen launched a series of cars under the brand name Polo, aimed at the young to yuppie market. The name Polo [Paroz 1988] sounds inoffensive in English, but in Sotho (a local African language) the word Polo means penis. What compounds the use of the word as well is the descriptions add to the word by the car manufacture like Polo Playa and Polo Classic. If a young African female owned a Polo Playa, it could add a rather negative connotation to her image. The same trap could exist in the development of any IS.
— Geography: The world has become smaller with the development of the Internet. The whole world can be seen as competitors to a business organisation.
— Levels of technology: The average South African does not have access to the Internet. It is still the domain of a privileged few, and so is access to the latest and greatest computer technology.

All these aspects have some human component or connotation. Business is driven by human customers, human users of systems, humans making decisions on investing in a business venture, et cetera. With the development of the correct IS that keep the human element in mind, an organisation will have a competitive advantage over other business.

The traditional approach to eliciting information in an organisation was that a user would request the information from the IT department and the IT department would deliver the information as per request (see Figure 2). This system is rather inflexible and largely closed to the wider company or world. The more contemporary approach is that the

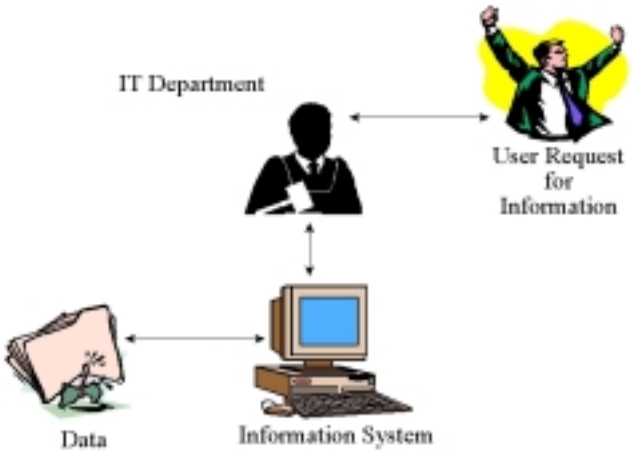

Figure 2. *Traditional Approach [Shelly 2003]*

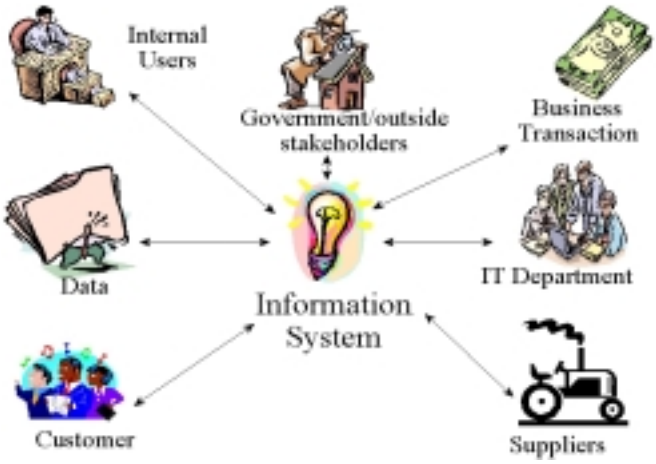

Figure 3. *Contemporary Approach [Adapted from Shelly 2003*]

system is opened to the world and all stakeholders can interact with the system (see Figure 3). We can also see from the Figure 2 that the user interface was the domain of the IT department, be it command driven or graphical user interface, the IT staff would somehow get your information to you. This has changed in the second model. In Figure 3 the system and all stakeholders interface with each other. This means that the interface to the system has to be versatile and flexible to be used by all stakeholders. The domain for the development of the IS and ultimately the user interface is, however, still that of the IT department.

3. COMPARING INFORMATION SYSTEMS DEVELOPMENT TECHNIQUES

There are various development methodologies that are used in developing ISs, some more conventional than others. On the conventional side there are two major approaches to systems development methodologies that are used to develop IS applications: the traditional systems development methodology and the object-oriented development approach. The proponents of HCI and interaction design propose lifecycle models with a stronger user focus than the conventional approaches. Section 4 briefly looks at these three systems development approaches.

Before we look at these approaches we need to argue about the method of comparing and assessing the various methodologies. There are always inherent problems in comparing various development methodologies. Therefore before we compare the two approaches, we would refer to these possible problems and how we tried to overcome them in our approach. The Object Agency [1993] identifies the following common flaws:

— Comparing methodologies is often like comparing apples and oranges (or even fire trucks). The same term often has different meanings in different approaches. While this difference in terminology may seem academic at first glance, the appropriateness and application of these methods are significantly impacted by this distinction. We did not compare terminology, but rather how these approaches addresses the elements identified in Figure 3.

— Any methodology comparison must evaluate a ‘snapshot’ of methodologies. Any comparison would necessarily be restrictive in the information it reviews. We did not look at a specific version of a methodology, but rather at the general steps involved in the process.

— Using an inappropriate framework with which to conduct the evaluation. An evaluator may attempt to evaluate methods using the context (i.e. framework) from their prior development background, often disproportionately biasing their review. We did not quantify issues, but merely identified if a particular issue was addressed or not.

— Utilizing reviewers who are simply ‘looking for words’ and ‘placing check marks in a checklist’ without doing the essential research. Although we looked at certain terms, we did not limit our assessment to that terms, but rather analysed the entire methodology in how it addressed the human issues.

— Using an overly constrained, or unconstrained, definition of methodology. The definition of ‘methodology’ should be clearly stated within the introduction to any methodology comparison. Each methodology comparison must be careful in its selection of definitions in order not to exclude viable methods, or include non-viable methods. We did not look at definitions per se, but rather to whether the human elements were addressed in generic terms.

— Many methodology comparisons seek to identify and document support for particular concepts, but do not rank the degree of support. This can lead to studies showing a number of methods as being relatively equal in support from a numerical standpoint (the number indicating the method ‘addresses’ certain topics, when in reality this is not the case). As stated above, we did not quantify issues, but just identified if a particular issue was addressed or not. We did not add a summative value to these issues.

— It is, in many instances, difficult to repeat the results of a methodology comparison with any accuracy. Since few (if any) of the comparisons cite page references for where a particular methodology comparison item (e.g. a term, concept, or example) is found in the methodology under review, it is difficult, if not impossible, to verify the

accuracy of these methodology comparisons. We did not compare the methodologies step-by-step, but rather as to whether and when they address the human element. We have to acknowledge that methodologies are always in a state of flux. In theory one thing happens and in practice the methodologies are modified to suite individual business needs.

4. DEVELOPMENT METHODOLOGIES

If we consider the definition of an IS (see Section 1), Whitten et al. [2001] mentions people as part of their definition. In Section 2 we established that an IS is the key to business effectiveness. Thus if the key components such as humans and the IS do not communicate effectively with each other, this business organisation is bound to fail. This section gives an overview of the three major groups of development methodologies and the major phases/processes involved. The aim of all these methodologies is to design effective and efficient ISs. But how effective are these when the wider environment is considered?

4.1 Traditional systems development approaches

Traditional (structured) development approaches include methodologies such as Structured Analysis and Design Techniques (SADT) [Ross 1985], The Yourdon Systems Method (YSM) [Yourdan 1993], Specification and Description Language (SDL) [Belina and Hogrefe 1989], Information Engineering and Jackson System Development (JSD) [Jackson 1983], the Dennis and Wixom Approach [Dennis and Wixom, 2000], et cetera. These approaches all have the following general phases in common: planning, analysis, design, and implementation.

Most of these development approaches follow the waterfall approach or is an iterative variation to the waterfall approach. As an example, we analysed one of these approaches, namely the Dennis and Wixom approach, in more detail with regard to how it meets the dynamic environment illustrated in Figure 3.

4.1.1 The Dennis and Wixom Approach

The most contemporary of the structured development approaches, namely the Dennis and Wixom Approach [2000], consists of the following phases:

— Planning (why build the system): Identifying business value, analyse feasibility, develop work plan, staffing the project, and control and direct project.
— Analysis (who, what, when, where will the system be): Analysis, information gathering, process modelling and data modelling.
— Design (how will the system work): Physical design, architecture design, interface design, database and file design and program design.
— Implementation (system delivery): Construction and installation of system.
— Dennis and Wixom [2000] describe their user interface design aspect as consisting of:
— Develop use scenarios (generic user).
— Design interface structure.
— Design interface standards.
— Design user interface template.
— Design user interface.
— Evaluate user interface.

Although included in the Dennis and Wixom approach, these steps are conducted much too late in the design phase. Most of the other structured development methodologies do not even address the issue though. The structured development approach therefore relegates the design of the user interface (and the human related issues) to the design phase of the development life cycle – to quite late in the development process. The user interface will only be tested in the implementation stage.

We assessed the Dennis and Wixom approach against each of the components depicted in Figure 3. Table 1 (section 4.1.1) summarises our findings and will be placed in context in Section 5.

4.2 The OO methodologies

There is a great deal of diversity within the OO community. Many object-oriented designers and developers, for example, seem to focus almost entirely on programming language issues. They tend to cast all discussions in terms of the syntax and semantics of their chosen object-oriented programming language. For example, to fully leverage IBM's design approach, IBM assumes the target language to be Smalltalk. Often an evaluation of methods requires the evaluator to understand the target platforms for which the methodology is intended [The Object Agency 1993].

Another sector of the object-oriented community is interested in formality and rigour. To this group software engineering is largely very systematic, repeatable, and transferable. They view object-oriented software engineering as primarily an engineering process with well-defined deliverables. The quality of the resulting products (and the process itself) can be evaluated in a quantitative, as well as qualitative, manner.

There are various OO methodologies such as Objectory [Jacobson 1994], Unified Modelling Language [Bahrami 1999], Coad and Yourdon Method [1991], Booch Method [1987], Object Modelling Technique [Rumbaugh et al. 1991], IBM approach [1990a, 1990b, 1990c, 1999] and Object-oriented Business Engineering [Jacobson 1994], et cetera.

Although diverse in approach most object-oriented development methodologies follow a defined system development life cycle, and the various phases are intrinsically equivalent for all the approaches, typically proceeding as [Schach 2002]: requirements phase; OO analysis phase (determining what the product is to do) and extracting the objects; OO (detailed) design phase; OO programming phase (implementing in appropriate OO programming language); integration phase; maintenance phase; and finally retirement.

The OO development approach in general lends itself do the development of more effective user interfaces because of the iterative design process, although this process does not seem to be effectively managed and guidelines for doing so are often absent.

We analysed three OO methodologies: The Rumbaugh, Coad and Yourdan, and IBM approaches and their relationship to the aspects illustrated in Figure 3. In our attempt to link it with the aspects identified in Figure 3, we considered issues related to how these approaches cater for the dynamic environment.

4.2.1 The Object Modelling Technique

The Rumbaugh et al. [1991] OO model has three distinct phases, which are analysis, system design and object design:

— The goal of the analysis phase is to develop a model of what the proposed system will do. The model is expressed in terms of objects and relationships, dynamic control flow, and functional transformations. The process of capturing requirements and consulting with the requestor should continue throughout analysis. The analysis phase has the following sub-phases: write or obtain an initial description of the problem (problem statement); build an object model; develop a dynamic model; construct a functional model; and verify, iterate, and refine the three models.

— During the systems design phase, the high-level structure of the system is chosen. There are several canonical architectures that can serve as a suitable starting point. According to Rumbaugh et al. [1991] the OO paradigm introduces no special insight into system design.

— During object design the analysis model is elaborated and a detailed basis for implementation is provided. Decisions are made that are necessary to realise the system without descending into the particular details of an individual language or database system. This phase has the following sub-phases: obtain operations for the object model from the other models; design algorithms to implement operations; optimize access paths to data; implement software control by fleshing out the approach chosen during system design; adjust class structure to increase inheritance; design implementation of associations; determine the exact representation of object attributes; and package classes and associations into modules.

We assessed the Rumbaugh approach against each of the components depicted in Figure 3. Table 1 (section 4.2.1) summarises our findings and will be placed in context in Section 5.

4.2.2 The Coad and Yourdan Model

The Coad and Yourdon [1991a, 1991b] model consists of two phases, the OO analysis and the OO design phases:

— The OO analysis phase identifies and defines classes and objects which directly reflect the problem domain and the system's responsibilities within it. The analysis phase has the following sub-phases: identify objects; identify structures; identify subjects; define attributes; define services.

— The OO design phase identifies and defines additional classes and objects, and reflects on the implementation of the requirements. It has the following sub-phases: design the problem domain component; design the human interaction component; design the task management component; and design the data management component.

In the Coad and Yourdon [1991b] model there is therefore an active component for the design of the interaction between human and machine. This component, the human interaction component, consists of the following activities: classify the humans; describe the humans and their task scenarios; design the command hierarchy; design the detailed interaction; continue to prototype; design the human interaction component classes; and design and account for graphical user interfaces (GUIs) (when applicable).

We assessed the Coad and Yourdan approach against each of the components depicted in Figure 3. Table 1 (section 4.2.2) summarises our findings and will be placed in context in Section 5.

4.2.3 The IBM model

The IBM model [1990a, 1990b, 1990c, 1999] consists of two phases, the OO design phase and the design the business model phase:

— The OO design phase has the following sub-phases: define business transactions; capture the user's model; specify the objects; build the view; and code and test the view.

— The business model design phase has the following sub-phases: find the technical model objects; implement the model objects; code and test the classes; and build the objects.

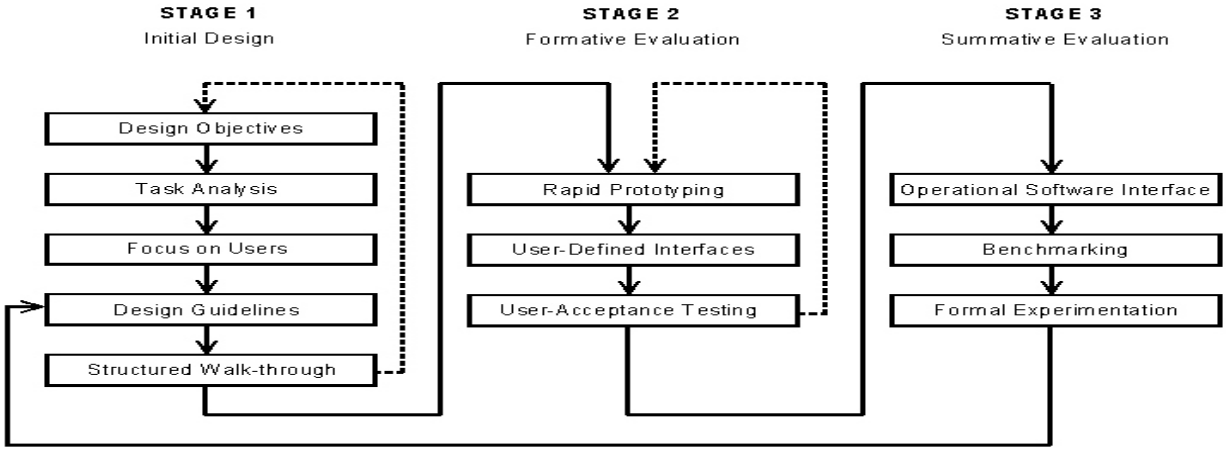

Figure 4. *HCI Life Cycle Model [Williges et al. 1987]*

The model therefore includes two specific user-oriented phases, i.e. the capture the user's model phase and build the view phase. These two phases include the following sub-components:

— Model the user's objects; model the user's behavior; enhance the user's model; provide natural interaction techniques; extract the object; document the user's model; and validate and iterate.
— Design direct manipulation actions; decompose the model objects; define instance variables; and design the windows.

We assessed the IBM approach against each of the components depicted in Figure 3. Table 1 (section 4.2.3) summarises our findings and will be placed in context in Section 5.

4.3 HCI focused life cycle approach

The HCI proponents aim to focus more on the human and end-user aspects. There are two types of users for most computer systems: those with experience with the system and those without (or with little experience). With the widespread introduction of information and communication technology in our everyday lives, most computer users today have limited computer experience, but are expected to use such systems.

Usability is a measurable characteristic of a product user interface that is present to a greater or lesser degree. One broad dimension of usability is how easy to learn the user interface is for novice and casual users [Mayhew 1999]. The vast majority of novice users are not prepared (or do not have the necessary background) to learn the computer-oriented details typically required of experienced users. Often, novice users expect to walk up and use an application as easily as they use the telephone or a car [Turban 1995]. Novice users are most concerned with the ease of learning, i.e. how quickly they learn new systems. Another usability dimension is how easy to use (efficiency, flexibility, powerfulness, etc.) the user interface is for frequent and proficient users, after they have mastered the initial learning of the interface [Mayhew 1999]. Expert users are therefore usually most concerned with ease of use [Dennis and Wixom 2000]. Systems and user interfaces should be designed with both types of users in mind.

Williges et al. [1987] have produced an alternative model of systems development, as illustrated in Figure 4, to rectify the problems in the traditional software development models. In their model interface design drives the whole process. Preece et al. [2002] suggest a simple lifecycle model, called the Interaction Design Model, consisting of identifying needs/establish requirements; evaluate; build an interactive version; and (re)design. Mandel [1997] suggest a similar development model to that of Preece et al. Other lifecycles models that focus on HCI aspects include the Star Model of Hartson and Hix [1989], the Usability Engineering Lifecycle of Mayhew [1999], and Hackos and Redish's model [1998]. These methods also introduce various strategies for the development of effective user interfaces.

Although varying greatly as far as individual phases are concerned, they are all based on an iterative system development approach and essentially contain more explicit main phases than the structured and OO systems development approaches. The argument is that by emphasising user requirements early in the development cycle, there will be less of a demand for code regeneration and modification in the latter part of systems development. Only time will tell whether or not this is a cost-effective strategy. In the meantime, it is important to understand the various techniques that can be used to support interface development early in the life cycle of a project. The Williges et al. and Hackos and Reddy models are briefly elaborated below.

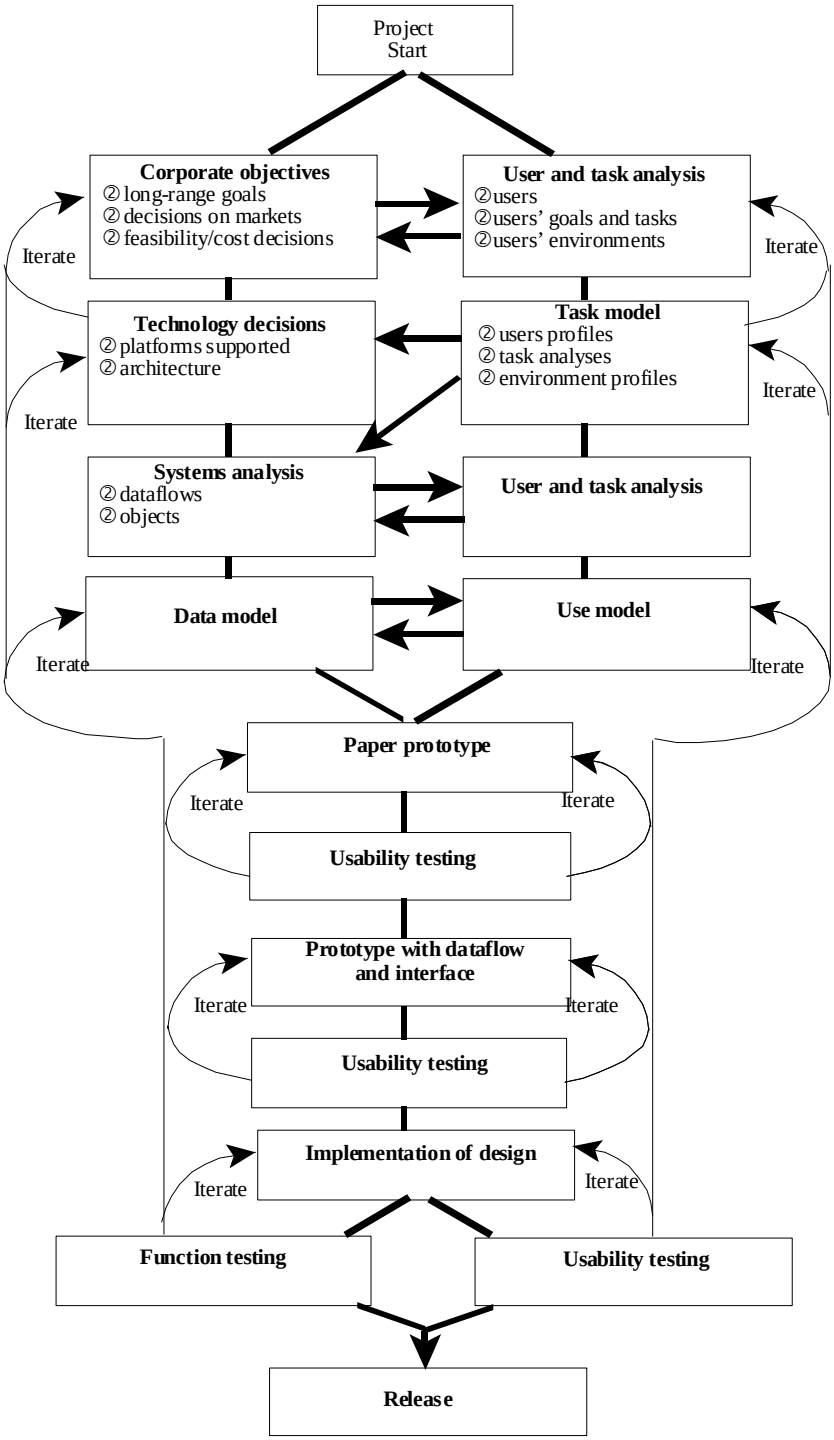

Figure 5. *Interface and Systems Development Hackos and Redish [1998]*

4.3.1 The Williges et al HCI focused life cycle approach

Figure 4 depicts a generalised flow diagram of the iterative design process for the development of the software interface, as proposed by Williges et al. [1987]. The model consists of three phases: the initial design stage in which the software interface is specified; a formative evaluation stage during which the interface evolves; and a summative evaluation stage in which the resulting system of the formative evaluation stage is tested:

— The initial design stage consists of the following six phases: determining design objectives; task-function analysis; focus on users; dialogue design guidelines; structured walk-trough; and initial design modifications (a feedback loop to refine the design (dotted line in Figure 4)).

— The formative evaluation stage consists of the following four phases: rapid prototyping; user-defined interfaces; user-acceptance testing phase; and an iterative redesign phase.

Section No.	Approach	UI Component	Internal Users	Customers	Suppliers	IT Department	Government
Traditional systems development approaches							
4.1.1	*The Dennis and Wixom Approach*						
	Planning	no	yes	not involved	not involved	actively part of	not involved
	Analysis	no	yes	not involved	not involved	actively part of	not involved
	Design	yes	yes	not involved	not involved	actively part of	not involved
	Implementation	yes	yes	not involved	not involved	actively part of	not involved
The OO methodologies							
4.2.1	*The Rumbaugh model*						
	Analysis phase	attempts	attempts	not part of	not part of	actively part of	not part of
	System design	no	not involved	not part of	not part of	actively part of	not part of
	Object design	no	not involved	not part of	not part of	actively part of	not part of
4.2.2	*The Coad and Yourdan Model*						
	Analysis	no	attempts	not part of	not part of	actively part of	not part of
	Design	yes	attempts	not part of	not part of	actively part of	not part of
4.2.3	*The IBM model*						
	OO design phase	yes	attempts	not part of	not part of	actively part of	not part of
	The design the business model phase	no	not part of	not part of	not part of	not part of	not part of
HCI focused life cycle approach							
4.3.1	*Williges et al*						
	Initial Design	yes	yes	attempts	attempts	actively part of	attempts
	Formative Evaluation	yes	yes	attempts	attempts	actively part of	attempts
	Summative Evaluation	yes	yes	attempts	attempts	actively part of	attempts
4.3.2	*Hackos and Redish Approach*						
	Systems development	no	attempts	attempts	attempts	actively part of	attempts
	Interface design	yes	yes	attempts	attempts	actively part of	attempts
	Design and implementation	yes	yes	attempts	no	actively part of	no
	Testing phase	yes	yes	attempts	no	actively part of	no

Table. 1. Methodology Matrix

— The summative evaluation stage consists of the following phases: operational software interface; benchmarking; formal experimentation; and a feed-forward results phase.

We assessed the Williges approach against each of the components depicted in Figure 3. Table 1 (section 4.3.1) summarises our findings and will be placed in context in Section 5.

4.3.2 Hackos and Redish Approach

The Hackos and Redish [1998] approach (see Figure 5) introduces the interface design at the inception stage of the system development. Figure 5 shows how the usability analysis fits into both the design of the interface and the design of the underlying system. The Hackos and Redish approach consists of the systems development phase, the interface design phase, the design and implementation phases, and the testing phase. The systems development phase and interface design phase are conducted in parallel:

— The systems development phase is concerned with the overall system and interaction with outside stakeholders. The systems development phase has the following sub-phases: corporate objectives; technology decisions; systems analysis; and data modeling.
— The interface design phase is an in-depth analysis of the design of the interface for the proposed system. The interface design phase has the following sub-phases: user and task analysis; task model; user and task analysis; and use model.
— The design and implementation phase is a marriage of the above two phases to produce the final system. The design and implementation phase has the following sub-phases: paper prototyping; usability testing; prototyping with dataflow and interface; usability testing; and implementation of design.

The testing phase tests functionality and usability of system. There are two sub-phases: function testing and usability testing.

We assessed the Hackos and Redish approach against each of the components depicted in Figure 3. Table 1 (section 4.3.2) summarises our findings and will be placed in context in Section 5.

5. OVERALL ASSESSMENT OF METHODOLOGIES

One of the problems with the traditional model for software development and the OO approaches is that they do not in general clearly identify a role for HCI in systems development. User interface concerns are ‘mixed in’ with wider development activities. This may result in one of two problems. Either HCI is ignored, or it is relegated to the later stages of design as an afterthought. In either case, the consequences can be disastrous. If HCI is ignored then there is a good chance that problems will occur in the testing and maintenance stages. If HCI is relegated to the later stages in the development cycle then it may prove very expensive to ‘massage’ application functionality into a form that can be readily accessed by the end-user and other stakeholders. In either case the cost of introducing ‘usability’ issues will rise, the later one postpones it in the development cycle.

If we examine the methodology matrix (Table 1) in more detail with regards to all the components reflected in Figure 3, we find the following with regards to the structured development and OO development methodologies:

— In the Dennis and Wixom approach interface design is only considered in the later stages of development. The components of Figure 3 only partially maps on to this approach, with no reference to the customers, suppliers, the IT department specifically, or the governmental issues.
— In the Rumbaugh et al. model [1991] there is no special consideration given for the design of the user interface or any of the other components reflected in Figure 3.
— In the Coad and Yourdon model the human interaction component includes the actual displays and inputs needed for effective human-computer interaction. This model is a partial fit onto Figure 3. While the internal users of the system are well catered for the other stakeholders are not actively involved in the process at all.
— The IBM model considers the users in the development of the system, however Figure 3 is still only a partial fit onto this model. The internal users are considered in the development of the system, but the external users and other stakeholders are sidelined.

It is clear from the above that there are several missing components in all these SDLCs. The Coad and Yourdan approach explicitly take into account the HCI aspect and tend to ignore the other aspects of Figure 3. The same applies for the IBM approach, but the process is much shorter. The Rumbaugh approach is very detailed but still ignored the issue of direct mapping to the final user interface application. Although Rumbaugh actively employs use case scenarios, get users involvement in systems design, it does not map directly into the system user interface design.

The root cause of this poor communication is that all the conventional development methodologies (including the traditional development approach and the OO approach) do not give adequate attention to the human aspect of the systems development. Both approaches also ignore the greater business environment as shown in Figure 1. Many researchers have proposed ways of improving the systems interface, but most of this have not been integrated into the development techniques.

Our findings are a confirmation of the work of Monarchi and Puhr [1992]. They compared 23 object-oriented analysis and design methodologies to identify common themes, and strengths and weaknesses in object-oriented analysis and design methods in general, and found that:

— There is a great deal of current literature on object-oriented (OO) systems development. But there is, as yet, little consensus or standardization among these new development techniques.
— The user interface is typically part of the solution to a problem, rather than a part of the problem itself.
— Interface objects are associated with the user interface. They are not directly a part of the problem domain; instead they represent the users' view(s) of the semantic objects.
— User interfaces are a prevalent and integral part of systems development today and should be a part of any software analysis and design methodology. Interface objects are rarely addressed unless the system being modelled is one in which the interface objects are part of the problem domain (automated teller machines, for example). Iivari [1991] recognizes the absence of interface objects in other analysis methods and includes them in his framework for identifying objects. A number of other authors mention that user interfaces are an element of systems, but they offer little in the way of identifying or modelling interface objects.

Eleven years on this still applies. There is still no unified process that marries the development approaches with the usability issues. The HCI focused models attempt to do this, but they all also still suffer major shortcomings as illustrated in Table 1. Certain role players are not part of the design process such as suppliers, and the government or legislative issues.

When we consider Table 1 with regards to the HCI focused development models we find that:

— Williges et al. [1987] tries to introduce the usability issues at a much earlier stage of the development process, but this model is not widely used.
— The Hackos and Redish model [1998] as depicted in Figure 5, seems to be a most comprehensive model we assessed. The shortcoming of this model is, however, that it still ignores the outside stakeholders, unless the corporate objectives phases states that the organisation should give special consideration to the external users, such as customers, suppliers and government. Hackos and Redish are however silent on this issue and does not elaborate what they mean by corporate objectives. If the corporate objectives do include the outside stakeholders, this is the only model that we investigated that does this. In fact if this is the case the model maps onto Figure 3 and considers most of the aspects addressed in Figure 1. The usability engineering is done in parallel with the systems development, and integrated throughout.

The shortcomings of all the methodologies are therefore related to the complexity of the wider environment introduced by the issues highlighted in Figures 1 and 3, and how these aspects should inform the systems development process. None of the development methodologies addressed the human component as well as the issue of other stakeholders sufficiently. Both the traditional SDLC and OO approaches fall short on the issue of human aspects and stakeholder involvement. Although we expected the OO approaches to fair better on these issues, the results above clearly illustrate that these methodologies still have a long way to go to fully integrate environmental issues. Which one faired the best? Although the Williges et al [1987] and Hackos and Redish [1998] approach focussing on the user goes a long way in achieving this, several shortcomings can still be identified. There has to be a balanced approach to systems development and HCI development components in the overall systems development process.

6. AN EXAMPLE OF SYSTEMS DEVELOPMENT GONE WRONG

The case below is just one of many examples of poor mapping from the initial well thought out plan to the final product. The authors of this paper recently attempted to submit a paper to an international conference. Being in the business of computer technology the conference submission process was totally electronic. The instructions were downloaded from the conference web site. The instructions were very clear on the process that should be followed in the uploading of the final paper. Upon arrival of the closing date for submission of papers, the authors in true fashion (like most of the other authors) left it to the last minute. The instructions asked for two versions of the paper, one anonymous and the other with author details. The authors followed all the instructions religiously, managed to create a user account, password and then it was time to upload the paper. Two versions of the paper were asked for but upload provision was made for only version. The instructions clearly stated that papers that did not meet all the criteria, including the two versions of the paper will be disqualified.

Which paper do you upload? Some users created two accounts, one to upload the anonymous version and the other with the full details and affiliations of the authors included. It was too late when the organisers realized what had happened and had to e-mail all corresponding authors with apologies, and change the business rules midway.

What came out very clear in this experience is that there was a well planned out strategy and business rules, but the plan did not map out to the implemented system. Some way in the development processes something went wrong. There was seemingly no task analysis conducted – affecting the end-users of the system. The supplier provided an off-the-shelf system and did not customise the data model. This could have been prevented if the supplier were involved in setting up the original business rules. The question on all developers' lips is: which development methodology did they follow? Nobody is saying at this stage, and given our investigation it is hard to tell which would have been the most beneficial, and solved the problem before implementation.

7. CONCLUSION AND PROPOSAL

In order for IS development to stay relevant and deliver systems fitting the demand of the current business environment, our research indicates that there is a dire need for the establishment of a unified process to the development of ISs, including all of the components identified in Figure 3. In order to meet these demands the exiting models should be enhanced to bridge the usability gap and map seamlessly to contemporary business environments.

We believe that many of the shortcomings of the development models could be catered for by making the end-user of the system a primary element in the entire process, and include explicit guidelines for the inclusion of other external issues such as laws and regulations, human rights issues (including accessibility), the abilities and skills of the human resource complement of the IT department, the supplier chain and availability of technology, et cetera. We argue that in the requirements/analysis and design phases the following issues should, for example, as a minimum consideration be catered for: Corporate issues such as competition, local economic climate and local domestic environments; political climate both locally and internationally; legal issues both on national and international level; acts and regulations that could affect the business; human rights issues e.g. accessibility laws/standards, access to information, etc.; the user context (geographical, cultural, socio-economic, educational, etc.); procurement issues; et cetera.

We further argue that most of the stakeholders identified in Figure 3, should ideally be involved in the formative and summative evaluation of the proposed and delivered system. After all, if a system does not meet regulatory standards, violates human rights issues, does not meet the exact requirements of the customers, needs very specialised equipment not readily available from suppliers, is not cost effective in terms of business transactions, do not meet the data or information requirements it is intended for, and cannot be developed by means of available skills or technology, how can it be successful? If these aspects are important, they should be explicitly catered for in the systems development model. Extensive further research would be required to establish processes and procedure to achieve this.

8. REFERENCES

BAHRAMI, A., 1999. Object-Oriented Systems Development: Using the Unified Modeling Language.Irwin McGraw-Hill, Boston.

BELINA, F AND HOGREFE, D. 1989. The CCITT-specification and description language SDL. Computer Networks and ISDN Systems, 16, pp 311-341.

BOOCH G. 1987. Software Engineering with Ada, 2nd ed. Menlo Park, CA: Benjamin-Cummings.

COAD P. and YOURDON E .1991a Object-Oriented Analysis. Englewood Cliffs, New Jersey: Yourdon Press.

COAD P. and YOURDON E. 1991b. Object-Oriented Design. Englewood Cliffs, NJ: Yourdon Press Computer Series.

DENNIS, A. and WIXOM, H. B. 2000. System Analysis and Design, John Wiley & Sons, Inc, New York.

VANDERDONCKT J and HARNING M.B. 2003. Closing the Gaps: Software Engineering and Human-Computer Interaction. Interact 2003 Workshop. http://www.interact2003.org/workshops/ws9-description.html.

HACKOS, T. J., and REDISH, C. J. 1998. User and Task Analysis for Interface Design. New York: Wiley.

HARTSON, H.R, AND HIX, D. 1989. Human-Computer Interface Development: Concepts and Systems for Its Management. ACM Computing Surveys 21(1): 5-92.

IBM. 1990a. Developing a CUA Workplace Application-Document GG24-3580-00. Raleigh, North Carolina: IBM International Technical Support Center.

IBM. 1990b. Object-Oriented Analysis of the ITSO Common Scenario-Document GG24-3566-00. Raleigh, North Carolina: IBM International Technical Support Center.

IBM. 1990c. A Practical Introduction to Object-Oriented Programming-Document GG24-3641-00. Boca Raton, Florida: IBM International Technical Support Center.

IBM. 1999. Object-Oriented Design: A Preliminary Approach-Document GG24-3647-00. Raleigh, North Carolina: IBM International Technical Support Center.

IIVARI, J. 1991. Object-oriented information systems analysis: A framework for systems analysis. IEEE Trans. pp 205-218.

JACOBSON, I. 1994. Object-Oriented Software Engineering: A use Case Driven Approach. Reading, MA: Addison-Wesley, Object Technology Series.

JACKSON, M. 1983. Systems Development. Englewood Cliffs, NJ: Prentice-Hall.

MANDEL, T. 1997. The Elements of user interface design, Wiley, NY.

MAYHEW, D. J. 1999. The Usability Engineering Lifecycle: a practitioner's handbook for user interface design, Morgan Kaufmann, San Francisco.

MONARCHI, DE and PUHR I.G. 1992. A Research Typology for Object-Oriented Analysis and Design. *Communications of the ACM*, Vol. 35, No. 9, September 1992, pp. 35 - 47.

PAROZ. R.A. 1988. Southern Sotho English Dictionary. Morija Sesuto Book Depot. Morija. Lesotho.

PREECE, J., ROGERS, Y. and SHARP H. 2002. Interaction Design: Beyond human-computer interaction, John Wiley & Sons, New York, USA.

RICKS, A. D. 1993. Blunders in International Business, Blackwell Publishers, Cambridge, Massachusetts.

ROSS, D., 1985. Application and extension of SADT. Computer 18, 4, pp 25-35.

RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F., AND LORENSEN W. 1991 Object-Oriented Modeling and Design. Englewood Cliffs, NJ: Prentice-Hall.

SATZINGER, J. W., JACKSON, R. B. and BURD, S. D. 2002. Systems Analysis and Design in a Changing World, Thomson Learning, Canada.

SCHACH, R.W. 2002. Object-Oriented and Classical Software Engineering, McGraw Hill, Boston.

SHELLY G,B, CASHMANT.J and ROSENBLATTH.J. 2002. Systems Analysis and Design. Thomson Learning, Canada.

THE OBJECT AGENCY. 1993. A Comparison of Object-Oriented Development Methodologies. http://www.toa.com/pub/mcr.pdf .

TURBAN, E. 1995. Decision Support and Expert Systems: Management Support Systems., Prentice-Hall, USA.

WHITTEN, L. J., BENTLEY, D. L. and DITTMAN, C. K.. 2001. Systems Analysis and Design Methods, McGraw-Hill Irwin, Boston, USA.

WILLIGES, R. C., WILLIGES, B. H. and EIKERTON, J. 1987. In Handbook of Human Factors(Ed, Salvendy, G.) John Wiley & Sons, New York, pp. 1414-1449.

YOURDAN, INC. 1993. Yourdan™ System Method: Model-Driven Systems Development. Prentice-Hall, Englewood Cliff, NJ.