Over-viewing Development Methodologies in the Context of e-Government

Shawren Singh[1]

ABSTRACT

e-Government will most probably reshape the South African civil society, as is happening in the rest of the world. Government organizations, large corporations, medium and small business, are all now actively trying to establish a presence of some kind on the World Wide Web. This paper describes ongoing research into e-government systems development methodologies.

Keywords*:* Systems development, e-government, design tools and techniques, usability, software engineering

1. Introduction

There are various factors that affect the successful delivery of e-government to the citizens of the particular country. We believe that we should understand the dynamic environment in which the government, citizens and cooperate world operate in. Figure 1 illustrates the critical factors that influences the key role players in e-government being civil society, cooperates and government. It is possible to summarize and identified critical factors (Figure 1) influencing the key role players as follows:

- *Social political issues:* economical forces, political factors, primary and secondary schooling, university policy and cultural forces. *Support structure:* technology, countries infrastructure, university policies and schooling system. *Social clustering:* geography and local-international forces.

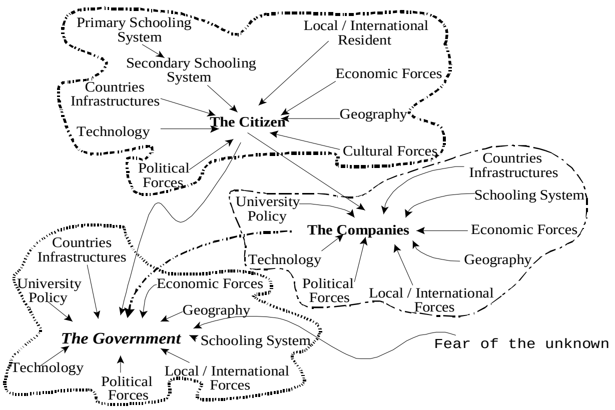

Figure 1: The Operating Environment

[1] Department of Computer Science & Information Systems, University of South Africa, P O Box 392, Pretoria 0003 (Email: singhs@unisa.ac.za Telephone: 012-4296640)

The factors presented above are by no means a definitive list and it is beyond the scope of this paper to fully analyse the identified factors. A holy trinity exists between the civil society, cooperation's, and the government as depicted in Figure 1. The micro and macro environment that these factors belong to affects all three stakeholders in different manners. Consequently, any e-government strategy will inherited a multitude of opportunities, some of these are:

- Integrating administrative functions; Understanding citizens needs; Meeting strategic needs of the country; Transparency in policy development and delivery; Integrating different citizens and cooperate cultures

The greatest challenge may, however, be to manage the other opportunities in such a manner that *fear of the unknown* will be reduced in order to enhance the positive attitudes to an e-government strategy. In the context of this paper we define e-government as an arrangement of *people (civil society),* data, process, information presentation, and information technology that interact to support and improve day-to-day operations in a country, as well as supporting the problem-solving and decision-making needs of cooperates, citizens and government itself. The environment in which modern day government operates is complex and fragile. For example in the South African context the following aspects are beyond the direct control of the government 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 refers to the male sexual organ. What compounds the use of the word is the descriptions added to the word by the car manufacture namely 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 interface development of any e-government information systems. *Levels of technology*: The average South African still does not have general access to the Internet. It is still the domain of a privileged few, and so is access to the latest and greatest computer technology.

Government is driven by civil society, human users of systems, humans making decisions on 'investing' in a government venture, etc. We argue that if these human 'components' could be put to better use or better service with the development of appropriate e-government information systems, a government website would equally empower it users.

This paper addresses this issue by comparing various systems development methodologies. Section 2 of this paper looks at the traditional and contemporary approach to e-government information system development. In Section 3 we discuss comparing information systems development techniques. Section 4 give an overview of the traditional structured systems development methodologies, the object-oriented approaches, as well as development methodologies that focus on the user or human elements and interactive system design for the web. Section 5 assesses the methodologies given the issues identified in Section 2 and Section 6 concludes with our proposals.

2. Systems Development

The traditional approach to eliciting information in an organisation (be it governmental or cooperate), as illustrated in Figure 2, was that a user of certain information would request the information from the information technology department and the information technology department would deliver the information as per request. This system was rather inflexible and largely closed to the human components of the wider organisation or world. In this model the information technology people are seen as the 'knowledgeable elite.' The information technology staff was responsible for finding the information and getting the information to you.

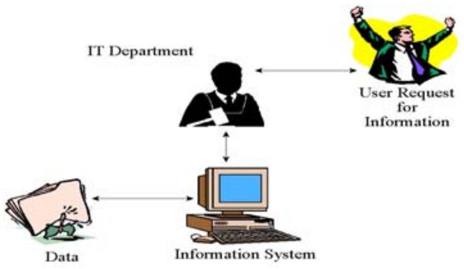

Figure 2: Traditional Approach Shelly *et. al.*(Shelly et al., 2003)

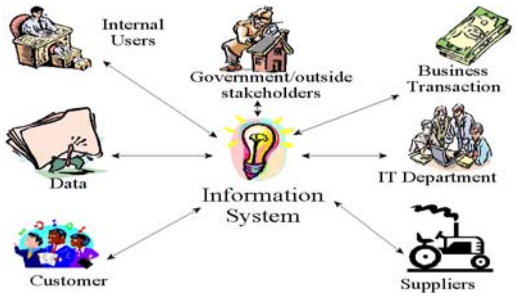

Figure 3: Contemporary Approach (Adapted from Shelly *et. al.* (Shelly et al., 2003))

The more contemporary approach is that the system is opened up to the world and all stakeholders can interact with the system, as illustrated 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 various stakeholders, not only the information technology department. But this more contemporary approach adds a new level of complexity to the systems development, as various levels of experience must be catered for. Browsing a poorly designed e-government portal is difficult. Browsing can be defined from several perspectives. Toms (2000) provides a general description of browsing as "…an activity in which one gathers information while scanning an information space without an explicit objective." Nah and Davis (2002) points out that Toms definition implies that the user's goals may be unclear, or there many be no goals focusing on finding a specific information/product.

Cove and Walsh (1988) distinctively identify three different types of browsing:

- Search browsing – directed search; where the goal is known
- General purpose browsing – consulting sources that have a high likelihood of items of interest
- Serendipitous browsing – purely random

Marchionini (1989) further develops this distinction in designating open and closed tasks. Closed tasks have a specific answer and often integrate subgoals, e.g. go to your local e-government portal and submit your tax returns. Open tasks are much more subject oriented and less specific, e.g. go to your local e-government portal and find information on crime. Browsing can be used as a method of fulfilling either open or closed tasks in e-government tasks. Unusable in terms of the e-government system, most tasks are goal-oriented and most relevant information is usually located within two clicks (Park and Kim, 2000), that is if you know what you want (closed task). Nah and Davis (2002) point out that there are several challengers facing the users in browsing an e-government website, such as the users' inability to navigate web sites and to search of desired information.

3. Comparing Information Systems Development Techniques for e-Government

There are various development methodologies that are used in developing e-government information systems, some more conventional than others. On the conventional side there are two major approaches to systems development methodologies that are used to develop e-government information systems applications: the traditional systems development methodology and the object-oriented development approach. The proponents of human-computer interaction suggest a stronger user focus than the conventional approaches. Section 4 briefly looks at some 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 different 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. 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 e-government needs.

4. Development Methodologies for e-Government

If we consider our definition of e-government (see Section 1), we mentions people as part of their definition. If the key components such as humans and the e-government information systems do not communicate effectively with each other, this e-government website is bound to fail. This section gives an overview of some major groups of development methodologies and the major phases/processes involved. The aim of all these methodologies is to design effective and efficient e-government information systems. 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 Inc, 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 operating 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 Object-oriented methodologies

There is a great deal of diversity within the object-oriented (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 (Coad and Yourdon, 1991), Booch Method (Booch, 1987), Object Modelling Technique (Rumbaugh et al., 1991), IBM approach (IBM, 1990c, IBM, 1990b, IBM, 1990a, IBM, 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 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 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 operating 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.

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 behaviour; 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 4.2.2) summarises our findings and will be placed in context in Section 5.

4.3. Human-Computer Interaction focused life cycle approach

The human-computer interaction (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). 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. 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, 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*. (2002). 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. The Williges *et al*. and Hackos and Redish models are briefly elaborated below.

4.3.1. The Williges et al. HCI focused life cycle approach

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.
- The formative evaluation stage consists of the following four phases: rapid prototyping; user-defined interfaces; user-acceptance testing phase; and an iterative redesign phase.
- 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 introduces the interface design at the inception stage of the system development. 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 modelling.
- 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.

4.4 . Interactive Systems Design for the Web

There are various researches that have suggested different methodologies for designing interactive Internet systems (Pressman, 2001, Avison and Fitzgerald, 2003, Turban et al., 2002, Newman and Lamming, 1995, Carter et al., 2001, Standing, 2001, Vidgen, 2002, Barry and Lang, 2001, Lee et al., 1999, Murugesan et al., 2001, Knight, 2006, Chan, 2005). In this section we will discuss specifically the WISDM methodology methodologies aimed at developing web-based applications.

4.4.1. WISDM

Vidgen *et. al.* (2002) argue that the development of interactive Internet systems require a mix of website development techniques together with traditional information systems development competencies in database and program design. Avison and Fitzgerald (2003) proposed the WISDM methodology, the WISDM methodology places emphasis on design, and interaction design and the user interface.
With the WISDM approach there is no priority ordering of the five aspects of the methodology matrix, each aspect of the matrix can be emphasised alone (or with others), as appropriate during the lifecycle of the project. The five aspects of the methodology matrix are:

- Organisational analysis – represents value creation and stresses that strategic relationships be built and maintained with a broad range of stakeholders. The overall question that is asked is ‘how is the information system supposed to further the aims of the organisation using it?’
- Information analysis – represents the requirements specification. This is a formalised specification of the information and process requirements of the organisation. The overall question that is asked is ‘what is the information processing function that the system is to perform?’ What is of major concern is the web content analysis and management thereof.
- Work design – The classic concern of socio-technical approach of information systems development has been with job satisfaction and genuine user participation in the development process. WISDM attempts to extend this view to incorporate all stakeholders. The overall question that is asked is ‘how can the system be designed to fit into the working lives of the people using the system?’
- Technical design – represents the software model. A formalised model of the software in terms of data structure and program design is needed to support software construction. The overall question that is asked is ‘what are the technical specifications of the system that will come closest to meeting the identified requirements.’

- Interaction design – represents the user interface and is located as an overlapping space in the technical design and work design. The overall question that is asked is 'how can the individual concerned best relate to the computer in terms of operating it and using the output from it?'

We assessed the WISDM 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.

5. Overall assessment of the Methodologies

One of the problems with the traditional methodology for software development and the object-oriented methodologies is that they do not in general clearly identify a role for interaction design in systems development. User interface concerns are 'mixed in' with wider development activities. This may result in one of two problems. Either interaction design is ignored, or it is relegated to the later stages of design as an afterthought. In either case, the consequences can be disastrous. If interaction design is ignored then there is a good chance that problems will occur in the testing and maintenance stages. If interaction design 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.

Table 1 is the methodology matrix representing our comparison of the methodologies discussed. We map the stakeholders of Figure 3 on to this matrix. We then compare the different phases of the development methodologies to see to what extent each phase considers the various stakeholders. For the traditional structured development methodology we use the Dennis and Wixom methodology (Section 4.1.1) as representative methodology. We analysed three object-oriented methodologies: The OMT (Section 4.2.1) and IBM methodology (Section 4.2.2). For the interaction design focused methodologies we analysed the Williges *et. al.* (Section 4.3.1) methodology and the Hackos and Redish methodology (Section 4.3.2), and for interactive systems design for the web we analysis WISDM (Section 4.4.1).

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 Object Modelling Technique (Rumbaugh et al., 1991) there is no special consideration given for the design of the user interface or any of the other components reflected in Figure 3.
- The IBM methodology considers the users in the development of the system, however Figure 3 is still only a partial fit onto this methodology. 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 methodologies. The IBM approach explicitly take into account the interaction design aspect and tend to ignore the other aspects of Figure 3. The Rumbaugh approach is very detailed but still ignored the issue of direct mapping to the final user interface application. Although Object Modelling Technique (Rumbaugh et al., 1991) 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 methodologies and the object-oriented methodologies) do not give adequate attention to the human aspect of the systems development. Both approaches also ignore the greater operating environment as shown in Figure 3. 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 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.

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

When we consider Table 1 specifically with regards to the interaction design focused development methodologies we find that:

- Williges *et. al.* (1987) tries to introduce the usability issues at a much earlier stage of the development process, but this methodology is not widely used.
- The Hackos and Redish (1998) seems to be a most comprehensive methodology we assessed. The shortcoming of this methodology 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 methodology that we investigated that does this. In fact if this is the case the methodology maps onto Figure 3. The usability engineering is done in parallel with the systems development, and integrated throughout.

When we consider Table 1 with regards to methodologies aimed at the web development we find that:

- WISDM makes a valiant attempt to include a wide range of users and tries to introduce usability issues at an early stage. But because there is no priority ordering of the five aspects of the methodology, some aspects many be over emphasised to the determent of the others.

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 object-oriented methodologies fall short on the issue of human aspects and stakeholder involvement. Although we expected the object-oriented methodologies 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. WISDM also has several shortcomings. There has to be a balanced approach to systems development and interaction design development in the overall systems development process.

Table 1: Methodology Matrix

Section No.	**Approach**	**UI Component**	**Internal Users**	**Customers**	**Suppliers**	**IT Department**	**Government**
Traditional systems development methodology							
4.1.1	*The Dennis and Wixom methodology*						
	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	*Object Modelling technique*						
	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 IBM methodology*						
	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
Interaction Design focused life cycle methodologies							
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 methodology*						
	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
Interactive Systems Design for the Web							
4.3.1	*WISDM*						
	Organisational analysis	yes	yes	yes	yes	yes	yes
	Information analysis	yes	yes	yes	attempts	yes	attempts
	Work design	attempts	attempts	attempts	attempts	attempts	attempts
	Technical design	attempts	attempts	attempts	attempts	attempts	attempts
	Human-Computer interaction	yes	yes	yes	attempts	yes	attempts

6. Concluding Remarks

In order for e-government information system 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 e-government information system s, 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 civil society. 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: 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 government; 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.

Acknowledgements

The author acknowledges the help of Prof. P Kotze of the School of Computing, University of South Africa, for her keen interest in this project. The author was involved in various discussions with various government employees and consultants, their discussions lead to the refinement of this paper. The original version of this paper has evolved into this current paper (Singh and Kotze, 2003).

References

5 Avison, D. and Fitzgerald, G. (2003) *Information Systems Development: Methodologies, Techniques and Tool,* McGraw-Hill, London.

6 Bahrami, A. (1999) *Object-Oriented Systems Development: Using the Unified Modeling Language,* Irwin McGraw-Hill, Boston.

7 Barry, C. and Lang, M. (2001) A Survey of Multimedia and Web Development Techniques and Methodology Usage, *IEEE MultiMedia* 8 52-60

8 Belina, F. and Hogrefe, D. (1989) The CCITT-specification and description language SDL *Computer Networks and ISDN Systems,* 16, 311-341.

9 Booch, G. (1987) *Software Engineering with Ada,* Benjamin-Cummings, Menlo Park, CA.

10 Carter, R. A., Antón, A. I., Dagnino, A. and Williams, L. (2001) Evolving Beyond Requirements Creep: A Risk-Based Evolutionary Prototyping Model Proceedings of the Fifth International Symposium on Requirements Engineering IEEE: Computer Society

11 Chan, S. T. (2005) Constructing a Globalised E-Commerce Site In *Encyclopedia of Multimedia Technology and Networking,* Vol. 1 (Ed, Pagani, M.) Idea Group Reference, Hershey.

12 Coad, P. and Yourdon, E. (1991) *Object-Oriented Analysis,* Yourdon Press, Englewood Cliffs, New Jersey.

13 Cove, J. F. and Walsh, B. C. (1988) Online text retrieval via browsing *Information Processing and Management,* 24, 31-37.

14 Dennis, A. and Wixom, H. B. (2000) *System Analysis and Design,* John Wiley & Sons, Inc, New York.

15 Hackos, T. J. and Redish, C. J. (1998) *User and Task Analysis for Interface Design,* Wiley, New York.

16 Hartson, H. R. and Hix, D. (1989) Human-Computer Interface Development: Concepts and Systems for Its Management ACM Computing Surveys 21 5-92

17 IBM (1990a) IBM International Technical Support Center, Raleigh, North Carolina.

18 IBM (1990b) IBM International Technical Support Center, Raleigh, North Carolina.

19 IBM (1990c) IBM International Technical Support Center, Boca Raton, Florida.

20 IBM (1999) IBM International Technical Support Centers, Raleigh, North Carolina.

21 Iivari, J. (1991) Object-oriented information systems analysis: A framework for systems analysis *IEEE Transaction*, 205-218.

22 Jackson, M. (1983) *Systems Development*, Prentice-Hall, Englewood Cliffs, NJ.

23 Jacobson, I. (1994) *Object-Oriented Software Engineering: A use Case Driven Approach,* Addison-Wesley, Reading, MA.

24 Knight, J. (2006) Design Frameworks In *Encyclopedia of Human-Computer Interaction* (Ed, Ghaoui, C.) Idea Group Reference, Hershey.

25 Lee, S. D., Yang, Y. J., Cho, E. S., Kim, S. D. and Rhew, S. Y. (1999) COMO: A UML-Based Component Development Methodology Sixth Asia-Pacific Software Engineering Conference (APSEC'99) 54

26 Mandel, T. (1997) *The Elements of USer Interface Design,* John Wiley & Sons, NY.

27 Marchionini, G. (1989) Information-seeking strategies of novices using a full-text electronic encyclopedia *Journal Amer. Soc. Inform. Sci,* 40, 54-66.

28 Mayhew, D. J. (1999) *The Usability Engineering Lifecycle: a practitioner's handbook for user interface design,* Morgan Kaufmann, San Francisco.

29 Monarchi, D. E. and Puhr, G. I. (1992) A Research Typology for Object-Oriented Analysis and Design *Communication of the ACM,* 35, 35-47.

30 Murugesan, S., Deshpande, Y., Hansen, S. and Ginige, A. (2001) *Web Engineering : Software Engineering and Web Application Development,* Springer Berlin, Heidelberg.

31 Nah, F. F. and Davis, S. (2002) HCI Research Issues in E-Commerce *Journal of Electronic Commerce Research,* 3, 98-113.

32 Newman, W. M. and Lamming, M. G. (1995) *Interactive System Design,* Addison Wesley, Wokingham.

33 Park, J. and Kim, J. (2000) *Effects of Contextual Navigation Aids on Browsing Diverse Web Systems,* Chi 2000, The Hague, Amsterdam.

34 Paroz, R. A. (1988) *Southern Sotho English Dictionary,* Morija Sesuto Book Depot, Morija, Lesotho.

35 Preece, J., Rogers, Y. and Sharp, H. (2002) *Interaction Design: Beyond human-computer interaction,* John Wiley & Sons, New York, USA.

36 Pressman, R. S. (2001) *Software Engineering: A Practitioner's Approach,* McGraw Hill, London.

37 Ross, D. (1985) Application and extension of SADT *Computer,* 18, 25-35.

38 Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991) *Object-Oriented Modeling and Design,* Prentice-Hall, Englewood Cliffs, New Jersey.

39 Schach, S. R. (2002) *Object-Oriented and Classical Software Engineering,* McGraw Hill, Boston.

40 Shelly, B. G., Cashman, J. T. and Rosenblatt, J. H. (2003) *Systems Analysis and Design,* Thomson: Course Technology, Australia.

41 Singh, S. and Kotze, P. (2003) An Overview of Systems Design and Development Methodology with Regard to the Involvement of Users and Other Stakeholders SAICSIT 2003: IT Research in Developing Countries Fourways, South Africa 37-47

42 Standing, C. (2001) The Requirements of Methodologies for Developing Web Applications Global Co-Operation in the New Millennium: The 9th European Conference on Information Systems Bled, Slovenia548-557

43 The Object Agency (1993) A Comparison of Object-Oriented Development Methodologies The Object Agency 2 August 2003 http://www.toa.com/pub/mcr.pdf

44 Toms, E. (2000) Understanding and Facilitating the Browsing of Electronic Text *International Journal of Human-Computer Studies,* 52, 423-452.

45 Turban, E., King, D., Lee, J., Warkentin, M. and Chan, H. M. (2002) *Electronic Commerce 2002: A Managerial Perspective,* Prentice Hall, New Jersey.

46 Vidgen, R. (2002) Constructing a web information system development methodology *Info Systems,* 12, 247–261.

47 Vidgen, R., Avison, D., Wood, R. and Wood-Harper, A. (2002) *Developing Web Information Systems,* Butterworth-Heinemann, London.

48 Williges, R. C., Williges, B. H. and Elkerton, J. (1987) Software Interface Design In *Handbook of Human Factors*(Ed, Salvendy, G.) John Wiley & Sons, New York, pp. 1414-1449.

49 Yourdan Inc (1993) *Yourdan(tm) System Method: Model-Driven Systems Development,* Prentice-Hall, Englewood Cliff, NJ.

About the Author

Shawren Singh is a Lecturer at the University of South Africa in the School of Computing. Shawren is actively involved in research relating to E-commerce, Human Computer Interaction (HCI), Open Source, Internet Security, Web-based Courseware Tools, Internet Applications, Web-based Education and Accounting Information Systems. He is a member of SAICSIT (South African Institute for Computer Scientists and Information Technologists, *SAICSIT*) and SACLA (South African Computer Lecturers Association).