There are well established paradigms for information systems development but the methods used for Information Systems Development (ISD) have not been tied to most of these paradigms. Researchers have attempted to document the assumptions underlying different paradigms with the goal of making systems developers become aware of the assumptions and beliefs that they employ for a system development task. However, a number of Information Systems that have failed are as a result of lack of awareness by the information systems developers of some methods that should be used when dealing with an ISD problem. Information Systems researchers have not related ISD methods that can be used by practitioners to the identified ISD paradigms. In this paper, ISD methods have been classified under some of the major paradigms in form of a matrix. It is hoped that such classification will enable IS Developers easily identify methods they should use which in turn should lead to better quality of Information Systems, and a reduction in time and costs in Information Systems Development.

Category: H. [Information Systems].
Key words and phrases: Information Systems Development, ISD paradigms, ISD methods

1. INTRODUCTION

Information is one of the most valuable assets of modern corporations. However, development of Information Systems (ISs) faces many problems. Some traditional problems include [Tolvanen 1998; Lycett et al. 2007]: an inadequate alignment of ISs with business needs, low productivity, and a large number of failures. In this paper, we attempt to relate some of the major IS development methods to major IS development paradigms with the aim of addressing some IS development problems.

The need to ground ISD methods in different paradigms has long been sought for. Before looking at ISD methods, Hirschheim and Klein [1988] attempted to document the relationship between systems development methodologies to paradigms, however, they found the process to be complicated because different methodologies such Soft Systems Methodology had properties of more than one paradigm. This should have been obvious to Hirschheim and Klein since the methodologies that they considered comprised of methods based on different assumptions. In the field of information systems development, Hirschheim [1989] proposed grounding existing ISD methods by analyzing them with regard to different paradigms. According to Hirschheim [1989], “a paradigm is a set of assumptions adopted by a professional community that allows its members to engage in commonly shared practices”. It is these assumptions that play a central role in
guiding information systems development. Several researchers in the IS field later on
discovered that although ISD methods were used extensively, they were actually not
widely applied [Hardy et al. 1995; Wynekoop and Russo 1993]. To make it easier to
adopt suitable and specific ISD methods, Iivari et al. [1999] proposed a way of
classifying ISD methodologies based on their features leading to comprehensible
approaches that organizations could use to meet their business needs.

In this paper, we adapt the classification method used by Iivari et al. [1999] to relate a
set of ISD methods to ISD paradigms. The paper is organized as follows: In section 2 we
review the four traditional paradigms for ISD; in section 3 we review a set of well known
ISD methods and summarize their characteristics in a table; in section 4, we map the ISD
methods presented in section 3 to ISD paradigms to generate a relation as described in
section 5. Section 6 concludes the paper and proposes future work.

2. INFORMATION SYSTEMS DEVELOPMENT PARADIGMS
There exist different definitions for the term ‘paradigm’; however, the most popular is
that of Kuhn [1962]: “A paradigm is a set of common beliefs and agreements shared
between scientists about how problems should be understood and addressed”. This
definition is clearly appealing to the aim of the work reported in this paper. Hirschheim
and Klein [1989] also defined a paradigm as “the most fundamental set of assumptions
adopted by a professional community that allows its members to share similar
perceptions and engage in commonly shared practices”. Hirschheim [1989] proposed four
paradigms for Information Systems Development which were initially used in
organizational and social research [Burrell and Morgan 1979]. The paradigms (figure 1)
include [Hirschheim, 1989]: functionalism, social relativism, radical structuralism, and
neo-humanism. These paradigms are used in this paper as a starting point for classifying
ISD methods. Table 1 summarizes the major emphasis and underlying assumptions of the
four paradigms.

![Figure 1. Information Systems Development paradigms (Adapted from Burrell and Morgan 1979).](image-url)
Table I: Features of ISD Paradigms (Adapted from [Hirschheim 1989])

<table>
<thead>
<tr>
<th>Paradigms</th>
<th>Functionalism</th>
<th>Radical Structuralism</th>
<th>Social Relativism</th>
<th>Neohumanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emphasis</td>
<td>Explains status quo, social order, social integration, consensus, needs satisfaction and rational choice</td>
<td>Emphasizes the need to overthrow or transcend the limitations placed on existing social and organizational arrangements. Focus primarily on the structure and analysis of economic power relations</td>
<td>Seeks explanation within the realm of individual consciousness and subjectivity and within the framework of reference of the social actor as opposed to the observer of the action</td>
<td>Seeks radical change, emancipation and potentiality and stresses the role that different social and organizational forces play in understanding change. It focuses on all forms of barriers to emancipation (ideology, power, and psychological comparisons)</td>
</tr>
<tr>
<td>Assumptions</td>
<td>Epistemology: developer gains knowledge about the organization by searching for measurable cause-effect relationships. Ontology: it is believed that there exists an independent, empirical organizational reality</td>
<td>Epistemology: dialectic inquiry in the specific form of a materialistic view of history and society. Ontology: realism reflecting the belief in a pre-existing empirical reality</td>
<td>Epistemology: anti-positivism (that is, will and need to make sense of oneself and the situation. Ontology: Reality is not given, but socially constructed</td>
<td>Epistemology: Positivism for technical control, anti-positivism for mutual understanding and emancipation. Ontology: realism for technical control, social construction for mutual understanding and emancipation</td>
</tr>
</tbody>
</table>

3. INFORMATION SYSTEMS DEVELOPMENT METHODS

Generally, a method is defined as a systematic way of doing something. In Information Systems literature, a development method is considered to be comprised of a set of procedures, techniques, tools, and documentation aids that help in developing an information system [Smolander et al. 1990]. Olle et al. [1991: 1-2] also define a development method as “a methodical approach to information systems development used by one or more persons to produce a specification or design product by performing a design process”. A method may be broken down into phases, and the phases into sub-phases that help system developers choose procedures, techniques, tools, etc. that are appropriate at each stage in a systems development project [Avison and Fitzgerald 2003: 20].

The necessity to classify ISD methods under ISD paradigms is due to the fact that in using a particular method, one is making an implicit or explicit assumption about the nature of the world and knowledge. As we have already seen, it is these assumptions that are associated with a given paradigm. Examples of ISD methods include Structured
Analysis and Design [Yourdon 1989] and object-oriented methods of Booch [1991] and Rumbaugh et al. [1991]. In the next subsections, we identify some of the ISD methods in existence and construct a table summarizing their features which are used for relating to ISD paradigms.

3.1 Structured Methods
Structured development methods arose out of the information systems community to help deal with various problems associated with the mismatch between information systems developed and the business requirements the systems were supposed to meet. The most common example is the Structured Systems Analysis and Design Methodology (SSADM) [Yourdon 1989]. These methodologies formalize the requirements elicitation process to reduce chances of misunderstanding the requirements and use best practice techniques to the analysis and design process.

3.2 Object-oriented Methods

3.3 Rational Unified Process
The Rational Unified Process (RUP) is as a result of merging different object-oriented methods including the objectory process from Ivar Jacobson and those of Booch [1991] and Rumbaugh [1991]. The main idea behind the Rational Unified Process is to start small and iteratively build on that. The three main building blocks include: roles, work products, and tasks. More details about RUP can be found in the Rational Software White paper [Rational 2001].

3.4 Agile Development Methods
Agile development methods were initially aimed at developing software in a lighter, faster, and more people centric way [Beck et al. 2001]. Based on the goals identified, a number of principles were put forth concerning agile software development. Schuh
[2004] gives examples of the best known of a growing number of software development methodologies that are based on the principles underpinning agile development: Adaptive Software Development (ASD); the Crystal methodologies; Dynamic Systems Development method (DSDM); Extreme Programming (XP); Feature-Driven Development (FDD); Lean Software Development.

3.5 Scenario Requirements Analysis Method (SCRAM)
The SCRAM is a requirements elicitation method which employs scenarios and prototypes to help users relate a design to their work or task context and thereafter develop requirements [Shin et al. 2003]. Scenarios relate to real world experiences and can be expressed in different ways including using natural language, pictures, or other media [Gough et al. 1995; Caroll 2000]. The main goal of the method is to have a requirements specification in which user preferences for different design options are expressed [Sutcliffe 1997].

3.6 Summary of the characteristics of selected ISD methods
Table II shows a summary of the characteristics of some of the major ISD methods considered for classification under ISD paradigms.

4. METHOD FOR RELATING ISD METHODS TO ISD PARADIGMS
The method we use for obtaining the relating ISD methods to ISD paradigms is an adaptation of a procedure used by Iivari et al. [1999] for classifying ISD methodologies under ISD approaches. The basic idea of the method is to identify ISD paradigms of which an ISD method is an instance. We check whether and ISD paradigm fits under an ISD paradigm by analyzing the correspondence of the features of the ISD method and paradigms through tables I and II respectively. If there is no correspondence that can be identified for a given ISD method, then we check if an existing paradigm can somehow be modified or generalized to assimilate the ISD method. The features of the new modified paradigm must correspond to the features of the ISD method before a relationship can be established between the method and paradigm. If at the end of modifying a given paradigm, there still exists no correspondence between the features of the method and the modified paradigm, a new paradigm has to be proposed associated with the ISD method. However, we do not attempt proposing new paradigms in this paper. Otherwise, the ISD method would first be developed to an ISD methodology and later to a specific ISD paradigm.
Table II. ISD Methods and their characteristics

<table>
<thead>
<tr>
<th>Methods</th>
<th>SSADM</th>
<th>Object-Oriented methodology</th>
<th>Agile Development Method</th>
<th>Soft Systems Methodology</th>
<th>SCRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong></td>
<td>To provide methods which help high quality (reliable and maintainable) software in a productive way</td>
<td>To provide a method which helps to ensure that the products are delivered to the user on time and within budget, that the products meet user requirements, that user request to modify the system and/or fix bugs are responded to in a timely fashion that increasingly sophisticated products are offered so as to keep competitive edge that the changes in standards and delivery technology are kept up and the project team feels motivated and successful</td>
<td>To provides methods for building applications in a very short amount of time; traditionally with compromises in usability, features and/or execution speed.</td>
<td>To provide learning methods to support debate on desirable and feasible changes</td>
<td>Requirements Acquisition, Modeling and Analysis [Zhu and Jin 2004]</td>
</tr>
<tr>
<td><strong>Principles and Assumptions</strong></td>
<td>Separation of the essential model from the implementation model; Careful documentation to make the development process visible; Graphical notations; Top-down partitionable transformation / process models to hide complexity; unambiguous minimum redundant graphic specification; Balancing of models: Design modules with high cohesion and weak coupling</td>
<td>Seamless analysis, design and Implementation; Encapsulation; Information (implementation) hiding</td>
<td>Active user Involvement; Teams must be empowered to make Decisions; Focus on Frequent Delivery; Fitness for Business is Criterion for Accepted Deliverables; Iterative and Incremental Development is Mandatory; All Changes During Development Must Be Reversible; Requirements are Baselined at High-Level; Testing is Integrated throughout the Lifecycle; Collaborative and Co-operative Approach</td>
<td>Use of notional system modules called ‘human activity systems’ to illuminate different philosophies which may be applied to any social system; An information system is a system to support the truly relevant human activity system</td>
<td>Active User involvement for acquisition of requirements.</td>
</tr>
<tr>
<td><strong>Fundamental concepts</strong></td>
<td>Essential model vs. Implementation model; Transformation; Data flow; Data Store; Terminator; Module; Cohesion; Coupling</td>
<td>Problem domain vs. Implementation domain; Object and Class; Encapsulation; Information hiding; inheritance; Polymorphism; Communication between objects</td>
<td>Philosophy; Human Activity Systems; Root definition; Relevant system</td>
<td>A scenario is considered as a continuum of the real world descriptions and stories to models and specifications</td>
<td></td>
</tr>
</tbody>
</table>
The classification method used by Iivari et al. [1999] follows the logic of a single hierarchy. First, the method checks whether any of the candidate ISD paradigms share a strict subset of the features with a new paradigm abstracted from an ISD methodology. If it does, then the new paradigm is inserted as a sub-paradigm. Next the method checks whether any of the candidate paradigms can be generalized or modified so that the resultant paradigms share a strict subset of features of the new paradigm formed from the ISD methodology. If so, the new paradigm formed from the ISD methodology is inserted in the resultant paradigm. For our case, we only associate ISD methods to ISD paradigms by looking at features common to both, and use a scoring scheme to indicate the level of relationship between a given ISD method and paradigm. We specifically find relationships with regard to ISD methods and not ISD methodologies since some of the ISD methods can be constituents of a larger ISD methodology and yet have completely different features. One obvious advantage that results from this is that establishing the degree of relationship between ISD methods and ISD paradigms simplifies establishing the degree of relationship between various ISD methodologies and ISD paradigms.

5. RESULTS

From an analysis of the characteristics of the ISD methods shown in table II, and the assumptions and emphases of ISD paradigms shown in table I using the method introduced in section 4, we related ISD methods to paradigms as shown in table III.

Table III. Relationship between ISD methods and paradigms

<table>
<thead>
<tr>
<th>Methods</th>
<th>Function</th>
<th>Radical</th>
<th>Social</th>
<th>Neohumanism</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Structuralism</td>
<td>Relativism</td>
<td></td>
</tr>
<tr>
<td>SSADM</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Object-Oriented Method (OOM)</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Rapid Application Development</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>(RAD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Soft Systems Method (SSM)</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Scenario Requirements Analysis</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Method (SCRAM)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agile Method (Dynamic Systems</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Development Method - DSDM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Key

✔️✔️✔️ More fitting; ✔️ ✔️ Averagely fitting; ✔️ Less fitting; None Not Applicable
Table III shows the level of relationship between ISD methods and ISD paradigms as specified by the Key. Given a new method and a new paradigm the same method used for obtaining the relationships can be used. An example to explain the results in table III follows. SSADM fits more under Functionalism and Radical Structuralism paradigms because most of its characteristics are found to fit within the assumptions and emphases of the two paradigms. SSADM, however is found to be less suitable for use under the Social relativism and Neohumanism paradigm, averagely fitting when used under the Social relativism paradigm, and less fitting when used under Functionalism and Radical Structuralism paradigm. This explanation can be used for the rest of the ISD methods.

5. CONCLUSION
We have determined the level of usability of some ISD methods under different ISD paradigms. Not all ISD methods and methods were analyzed in this paper. However, a similar approach can be used for all the established ISD paradigms and widely used ISD methods. The approach we adapted for obtaining the levels of relationships between ISD methods and paradigms takes into consideration emerging new ISD methods and the possibility of proposing new paradigms. Obtaining relationships between ISD methods and paradigms also serves to enrich on the information that is required to map ISD methodologies or approaches to ISD paradigms. As future work, we hope that a general framework for categorizing methods and paradigms can be developed.

REFERENCES


AUTHOR BIOGRAPHIES

Peter NABENDE is a member of academic staff at the Faculty of Computing and Information Technology, Makerere University, Kampala. Currently, he is a PhD student in the Computational Linguistics Research group, University of Groningen. His interests include: Information Systems Theory, Development, Processing and Management; Transliteration and Translation generation, and Cross Lingual Information Extraction; Machine Learning and Artificial Intelligence application to Information Systems.

Benjamin AHIMBISIBWE is a PhD student at the Faculty of Computing and Information Technology, Makerere University. He holds a MSc. in Information Science and Bachelors in Library and Information Science from Makerere University.

Jude T. LUBEKA is a member of academic staff in the Department of Information Technology, Faculty of Computing and Information Technology (FCIT), Makerere University. Currently, he is the Acting Deputy Dean (Graduate Studies and Research) at FCIT. His research interests include: E-Learning (Tracking, Feedback, Assessment, Repositories, Instructional Design, Multimedia), E-Commerce, Data Warehousing (Data Mining, OLAP), Software Agents and Business Modelling.