December 2006

Toward an Ontology in the Domain of Information Systems Delivery and Evolution

Lila Rao
The University of the West Indies- Jamaica

Gunjan Mansingh
The University of the West Indies- Jamaica

Evan Duggan
University of Alabama- U.S.A.

Toward an Ontology in the Domain of Information Systems Delivery and Evolution

Lila Rao Graham
The University of the West Indies, Jamaica
lila.rao@uwimona.edu.jm

Gunjan Mansingh
The University of the West Indies, Jamaica
gunjan.mansingh@uwimona.edu.jm

Evan W. Duggan
University of Alabama, U.S.A.
eduggan@cba.ua.edu

ABSTRACT

In this study we examine the relevant information systems delivery and evolution literature in order to develop an ontology for helping to reduce the fuzziness and ambiguity that often arises from inconsistent use of terminology in this domain. Such an undertaking necessarily encompasses a broader set of concepts that can be accommodated in this paper. We have therefore narrowed the scope of this ontology to the more controversial terms used at the heart of the information systems life cycle. We hope to extend knowledge by contributing to a standard vocabulary, classifying concepts and terms, identifying synonyms and homonyms, and providing clarifying examples (instances) in order to facilitate scholarly activities that require a deeper understanding of the structure of information in this domain. We use a frame-like representational scheme for the ontology and set the stage for validating the results using brainstorming and card sort to evaluate it effectiveness.

Keywords

Ontology, information systems delivery, frame-based representation, IS classification, system life cycle

INTRODUCTION

In comparison to business disciplines such as accounting, finance, management, and marketing, the information systems (IS) field is relatively new and there is not yet convergence of thought on, or standard definitions for many of the important concepts that are often discussed and researched (Duggan and Reichgelt, 2006). While we await this consensus, the field is increasingly plagued with this ambiguity (Wynekoop and Russo, 1995). This is true for many IS subfields including the information system life cycle, where many researchers and practitioners seek definitive insights to improve IS quality and reduce the impact of what some have called the software crisis (Brynjolfssen, 1993; Gibbs, 1994).

This absence of a common vocabulary gives rise to inconsistent use of terms in the IS literature which curtails the progress of cumulative IS research in several ways: (1) It contributes to inappropriate operationalization of constructs (Barki, Rivard and Talbot, 1993); (2) it restricts our ability to replicate and interpret research results, and infer logically coherent relationships among phenomena – a theory-generating exercise; (3) It impedes literature search and guarantees low precision -- defined as the ratio of the number of relevant documents retrieved to the total number of documents retrieved (Liu and Shih, 2004). Debates often rage about the legitimacy of research findings because of divergent interpretations of the constructs and variables analyzed (Roberts, Gibson, Fields and Rainer, 1998; Roberts, Gibson, Rainer and Fields, 2001; Yadav and Shaw, 2001)

The divergence of usage of IS terms and misclassification of concepts have also affected other constituents. The misapplication of these terms has presented much difficulty for practitioners who are often confused by the inconsistency, which promotes miscommunication (Yadav and Shaw, 2001). Teachers and students in academic institutions are sometimes

ambivalent and sometimes unable to decipher meaning and effectively assimilate literature in this domain and new entrants to the field are sometimes discouraged by the lack of clarity (Iivari, Hirschheim and Klein, 2000/2001).

We seek to address this problem in this paper and hope to extend knowledge in this domain by contributing the ontology, cognizant that such an undertaking encompasses a broader set of concepts than can be addressed here. We have therefore begun this cumulative process by focusing on a standard vocabulary and classification of terms, providing clarifying examples (instances), and identifying synonyms and homonyms in the area of systems delivery – analysis, design, and construction – at the heart of the systems life cycle, which accounts for many of the controversial terms.

We intend to complete other components of the ontology ourselves in the systems delivery domain but we also offer the immediate results to other researchers in order to facilitate scholarly activities that require a deeper understanding of the structure of information in this domain. In the rest of this paper we provide an analysis of the issues that further elaborate on the value of this research, present the ontology using a frame-like structure, and offer our conclusions including our plans for advancing this research.

ONTOLOGY IN THE SYSTEMS DELIVERY DOMAIN

A superordinate goal of every discipline is to define and categorize terms, concepts and phenomenon of interest to that discipline in order to create a common language for advancing knowledge in that discipline (Kishore, Sharman and Ramesh, 2004). The systems life cycle literature consists of a plethora of terms such as information systems development, software engineering, IS delivery, systems development, systems implementation, systems deployment, development models, and development paradigms, among others. Although there is no consensus on the accepted definitions of these concepts, some authors compound the problem by using these terms without defining them, assuming away the ambiguity that exists. This makes effective communication and the sharing of domain knowledge extremely difficult (Yadav and Shaw, 2001), which justifies the ontology call.

Noy and McGuinness (2001) defines ontology in this context as the formal explicit description of concepts in a domain of discourse. According to them, developing an ontology to make domain assumptions explicit offers several benefits such as (1) permitting the sharing of a common understanding of the structure of information among stakeholders in a discipline (2) facilitating more effective communication and idea-sharing (3) assisting new entrants in a field to quickly assimilate important domain concepts and knowledge and (4) generally supporting the analysis of domain knowledge (Noy and McGuinness, 2001). These benefits offer advantages to research, pedagogy, and practice.

Barki et al. (1993) identified and responded to the ongoing need for the explication of terminology in our discipline and updated their previous classification scheme for IS keywords to contribute to a common language for enabling cumulative research for the field's development. They claimed that this general classification of IS terms helped to define the field in some detail, promoted a common vocabulary, and facilitated the study of the evolution of IS research. The dynamic nature of our discipline demands such an ongoing effort to which we now contribute. However, we have restricted our focus to the language of the systems lifecycle where we believe the need for a common vocabulary is most glaring. Following, we delimit the scope of our immediate effort within the framework of the wider need, and set the stage for the development of the ontological structure.

Figure 1 provides a domain map which highlights the ultimate scope of this effort. The systems life cycle consists of two generic activities: (1) Systems delivery, the activities required to deliver an information system into production; "delivery" is preferred to "development" to account for all the possible methods (in-house development, purchase, outsourcing, etc.) of sourcing an information system. (2) Systems evolution, post deployment support activities as the systems evolve through corrective, adaptive, and perfective maintenance. The shaded areas identify our immediate area of concentration. We will also clarify the concepts of methodology – systems development methodology (SDM) and method – software production method, which are often used interchangeably within the system life cycle domain.

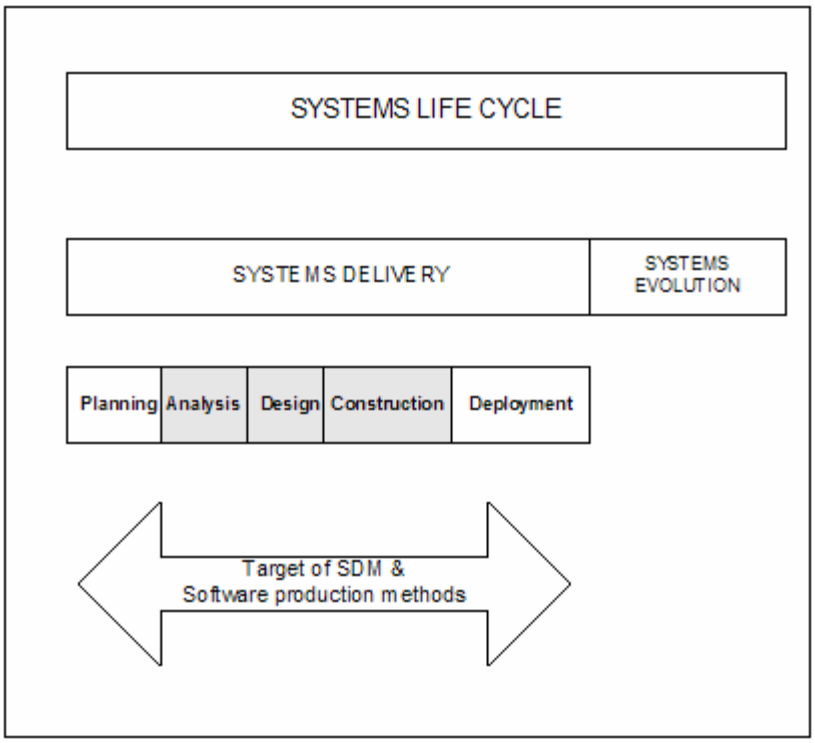

Figure 1. Systems Life Cycle

The formal ontology will be implemented using a frame-based knowledge representation language as recommended by (Fikes and Kehler, 1985; Reichgelt, 1991; Sharman, Kishore and Ramesh, 2004). A formal ontology is one where the conceptualization is rigorously specified using a specification or programming language. In such frame-based languages, knowledge is stored in larger chunks as a set of conceptual entities with associated descriptions. The network of chunks, which are representational structures referred to as frames contains descriptions called slots (Reichgelt, 1991).

The use of frames as a representation language offers several advantages. Frame-based knowledge representation languages capture the way in which domain experts typically think about their knowledge (Fikes and Kehler, 1985). This arrangement improves the ease with which knowledge is grasped by an expert. Entities are described by specialization, that is by relating an entity to others for which information is known, a feature that is well liked by domain experts (Fikes and Kehler, 1985). Frames use inheritance, which is very powerful for capturing inferences that humans naturally draw. Their hierarchical structure supports default reasoning, a feature of human analysis that accepts some value for an object until there is evidence to the contrary (Reichgelt, 1991).

THE STRUCTURE OF THE ONTOLOGY

We have developed the ontology by synthesizing from existing literature the definitions of commonly used but controversial or ambiguous terms within our scope. We then classified these terms at two levels, using the combination development process recommended by (Noy and McGuinness, 2001), which consists of a domain level and an instance level. The domain level is developed in a top-down manner and the instance level is developed using the bottom-up process. In our frame-based knowledge representation scheme, each term is represented as a frame in the hierarchy and each frame consists of a term, its definition and classification, the synonyms that are associated with it, and instances (or examples). This representation allows us to associate similar characteristics of terms in one frame with frames higher up in the hierarchy (Fikes and Kehler, 1985; Reichgelt, 1991; Sharman et al., 2004). The frames lower in the hierarchy inherit properties from the higher frames and their values can also be overwritten if necessary. A frame is similar to a class and the properties of these classes are represented as slots (Noy and McGuinness, 2001).

Domain Ontology

Duggan (2006) presented four major categories of systems delivery concepts:

1. Systems delivery paradigm - common distinguishing features of a family of life cycle approaches (e.g., sequential, incremental/iterative and reuse).
2. Systems development methodology - a comprehensive set of practices and principles for structuring the delivery process (e.g., Information Engineering, Method 1, Navigator, and RUP).
3. Software production method – a set of principles and objectives for executing and managing the production of software (e.g., RAD, extreme programming, cleanroom software engineering, object-oriented development, and component-based development).
4. Systems delivery tool – facilities that provide specific implementation support for a variety of methods (e.g., CASE, data flow diagrams, use cases)

We will use these definitions; however, we will further decompose tools into practices, techniques, and models for classification clarity. Figure 2 shows the relationships between the various terms.

§ Systems development methodology
A systems development methodology provides the strategy for advancing through the stages of systems delivery (Duggan, 2006). It specifies the deliverables, the quality standard, the roles of people and the techniques and tools that are supported in the software delivery process (Roberts and Hughes, 1996; Wynekoop and Russo, 1995). A methodology addresses the consistency and repeatability in IS delivery processes to reduce variability and ultimately contributes to better outcomes (Roberts et al., 2001).
Synonym (s): project management methodology

§ Systems delivery paradigms
Systems delivery paradigm refers to the underlying objectives of related delivery methods. Duggan (2006) and Robey et al. (2001) define it as, the common distinguishing features of a family of life cycle approaches. Several life cycle methods exist and there are properties which are common between them, to have some order in classification there is a need to group the common properties and describe them at as higher level of abstraction. This allows the shift in focus from individual methods to a level where their common features can be discussed (Iivari, Hirschheim and Klein, 1999).
Synonym(s): approaches

§ Software production methods
Production methods can be described as a set of principles and objectives for guiding the software delivery process (Duggan, 2006). A process model is a representation of the sequences of stages through which a system evolves (Wynekoop and Russo, 1995). Based on these definitions we define software development methods or process models as a set of principles and activities for guiding the software delivery process through its life cycle.
Synonym(s): development method, method, process model

§ Techniques
A number of researchers define techniques as a procedure possibly with a prescribed notation, to perform a development activity (Brinkkemper, 1996; Huisman and Iivari, 2003; Wynekoop and Russo, 1995). According to Iivari (2000/2001) techniques consist of a well-defined sequence of elementary operations that more or less guarantee the achievement of certain outcomes if executed correctly. We concur with the first definition and consider techniques to perform either a single or multiple activities within the methods. Techniques are not unique to a particular method and a method can use different techniques in the different phases of delivery.

§ Practices
Standard activities and policies that apply to methods or methodologies (MacCormack, Kemerer, Cusumano and Crandall, 2003)

§ Tools
Artefacts (which could include software) that supports a particular technique to carry out the activities specified by the method (Duggan, 2006; Hardgrave, Davis and Riemenschneider, 2003; Robey, Welke and Turk, 2001).

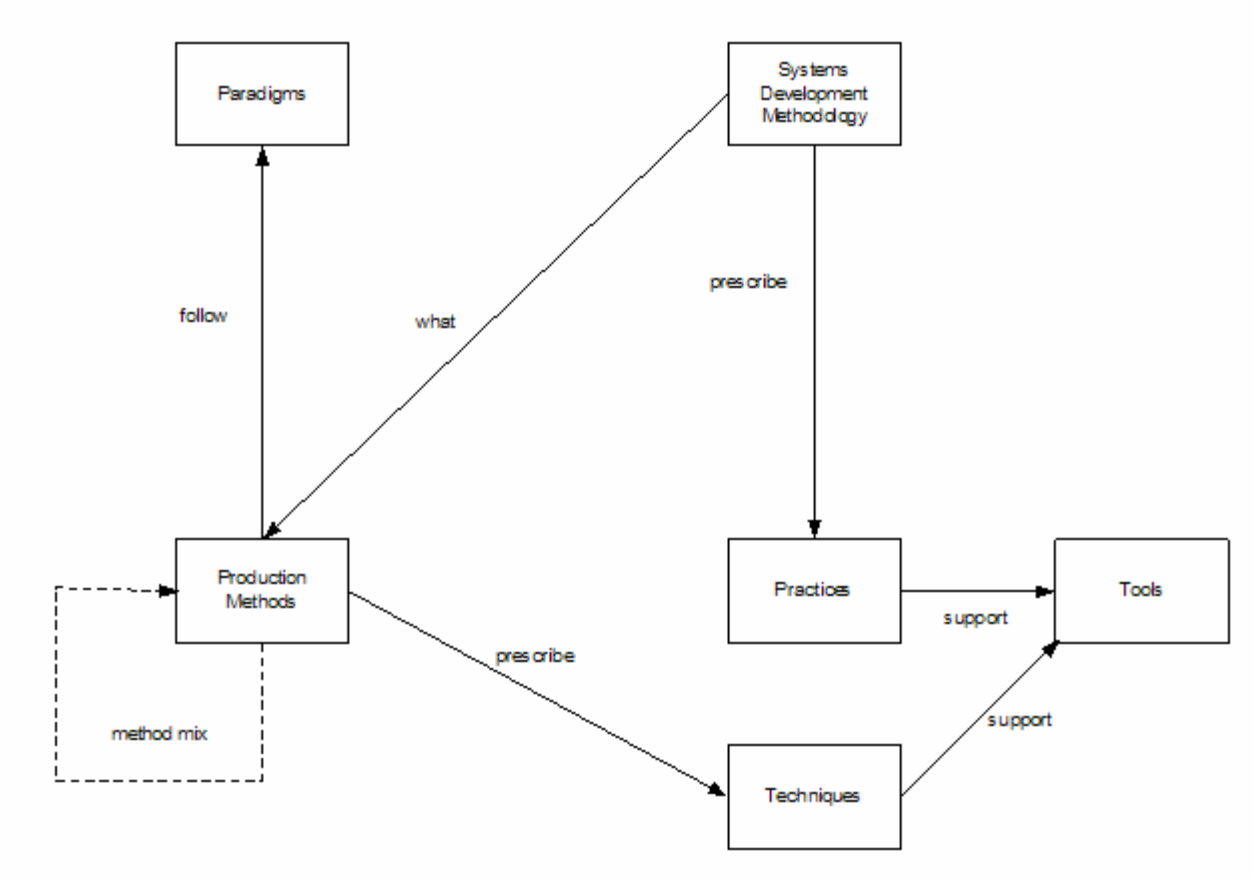

Figure 2. Domain Ontology for Information Systems Delivery

Instance Ontology

The instance ontology is represented as frames in which an inheritance-based reasoning mechanism is used. The frames lower in the hierarchy inherit properties from the higher frames and their values can be overwritten. In Figure 3, the three main software development paradigms, the production methods and their instances have been represented as frames. Some of slots have been identified but the list is not comprehensive and further work will be done to develop a complete representation of structural knowledge within this domain.

Conclusions and Future Work

We have used a frame-based representational structure to create a domain- and instance-level ontology in order to clarify frequently used but sometimes misused terms that describe systems life cycle activities. This initial effort concentrated on software production processes and methods. However, our ultimate goal is the entire life-cycle, which will be addressed progressively in future steps to extend, refine, deepen, and broaden our ontology. Because ontologies are reusable, our effort may also be extended by other researchers by combining (or modifying) our results to extend the scope to other IS areas that require similar clarification.

The mere completion of an ontology does not of itself guarantee the removal of semantic confusion; it requires validation. Our intention is to continue this research effort towards such validation by using card sort to confirm representational accuracy. Card sort is a knowledge elicitation technique that has been used to identify different concepts and the relationships between these concepts. Ultimately, we believe, this cumulative approach will help to solve the longstanding and increasingly bothersome problems of ambiguity and the absence of a common language in our field. This solution will benefit both researchers and practitioners.

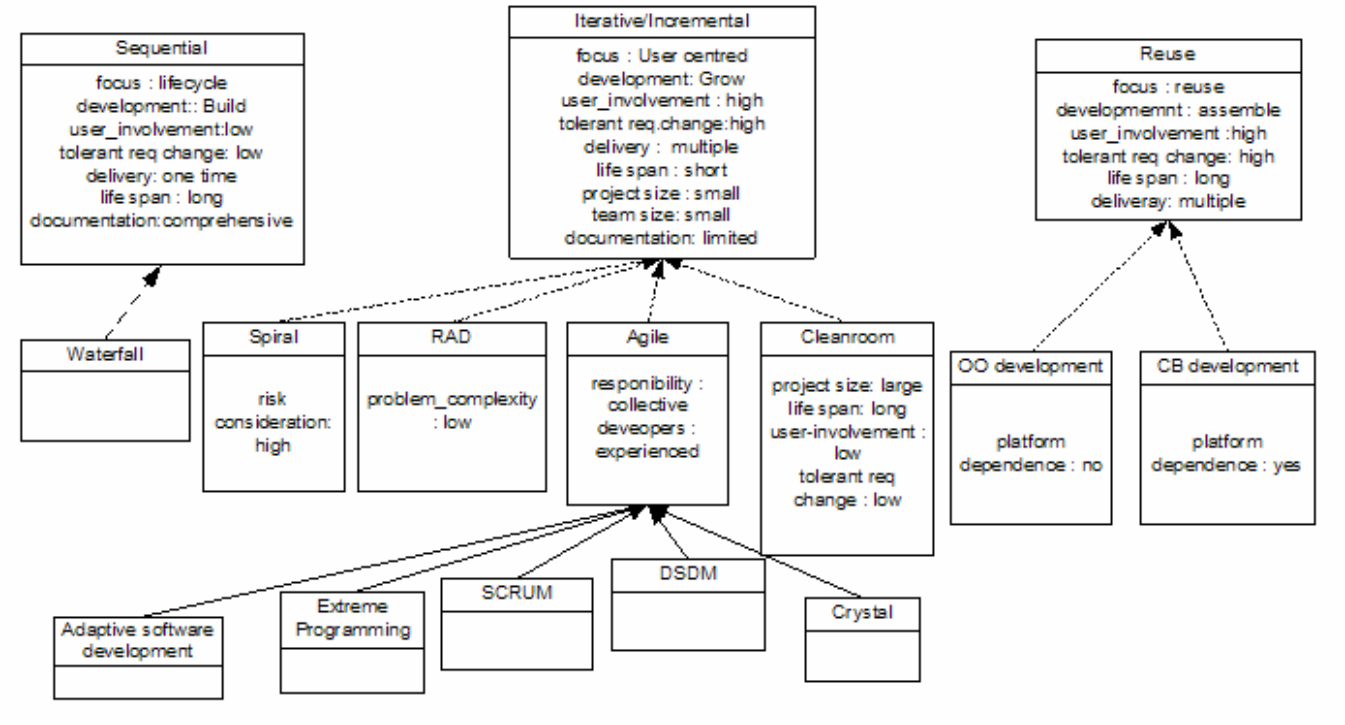

Figure 3. Instance Ontology for Paradigms and Methods

REFERENCES

1. Barki, H., Rivard, S. and Talbot, J. (1993). A Keyword Classification Scheme for IS Research Literature: An Update. *MIS Quarterly, 17*(2), 209.
2. Brinkkemper, S. (1996). Method Engineering: Engineering of Information Systems Development Methods and Tools. *Information and Software Technology, 38*, 275-280.
3. Brynjolfssen, E. (1993). The Productivity Paradox of Information Technology. *Communications of the ACM, 36*(12), 67-77.
4. Duggan, E. W. (2006). Tranquilizing the Werewolf that Attacks Information Systems Quality. In *Advanced Topics in Information Resources Management*.
5. Duggan, E. W. and Reichgelt, H. (2006). The Panorama of Information Systems Quality in. In E. W. Duggan & H. Reichgelt (Eds.), *Measuring Information Systems Delivery Quality*.
6. Fikes, R. and Kehler, T. (1985). The Role of Frame-Based Representation in Reasoning. *Communications of the ACM, 28*(9), 904-920.
7. Gibbs, W. (1994). Software's Chronic Crisis. *Scientific American, 271*(3), 86-95.
8. Hardgrave, B. C., Davis, F. D. and Riemenschneider, C. K. (2003). Investigating Determinants of Software Developers' Intentions to Follow Methodologies. *Journal of Management Information Systems, 20*(1), 123-151.
9. Huisman, M. and Iivari, J. (2003). *Systems Development Methodology Use in South Africa.* Paper presented at the Ninth Americas Conference on Information Systems.
10. Iivari, J., Hirschheim, R. and Klein, H. K. (1999). *Beyond Methodologies: Keeping up with Information Systems Development Approaches through Dynamic Classification.* Paper presented at the 32nd Hawaii International Conference on System Sciences.
11. Iivari, J., Hirschheim, R. and Klein, H. K. (2000/2001). A Dynamic Framework for Classifying Information Systems Development Methodologies and Approaches. *Journal of Management Information Systems, 17*(3), 179.
12. Kishore, R., Sharman, R. and Ramesh, R. (2004). Computational Ontologies and Information Systems: 1. Foundations. *Communications of the Association for Information Systems, 14*, 158-183.

13. Liu, D.-R. and Shih, Y.-Y. (2004). Integrating AHP and data mining for product recommendation based on customer lifetime value. *Information & Management*.
14. MacCormack, A., Kemerer, C., Cusumano, M. and Crandall, B. (2003). Trade-offs between Productivity and Quality in Selecting Software Development Practices. *IEEE Software*.
15. Noy, N. F. and McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.*
16. Reichgelt, H. (1991). *Knowledge Representation: An AI Perspective*: Ablex Publishing Corporation.
17. Roberts, T., Gibson, M., Fields, K. and Rainer, K. (1998). Factors that Impact Implementing a Systems Development Methodology. *IEEE Transactions on Software Engineering, 24*(8), 640-649.
18. Roberts, T., Gibson, M., Rainer, K. and Fields, K. (2001). Response to "Comments on Factors that Impact the Implementation of a Systems Development Methodology". *IEEE Transactions on Software Engineering, 27*(3), 282-286.
19. Roberts, T. and Hughes, C. (1996). Obstacles to implementing a system development methodology. *Journal of Systems Management, 47*(2), 36-40.
20. Robey, D., Welke, R. and Turk, D. (2001). Traditional, Iterative, and Component Based Development: A Social Analysis of Software Development Paradigms. *Information Technology and Management, 2*(1), 53.
21. Sharman, R., Kishore, R. and Ramesh, R. (2004). Computational Ontologies and Information Systems: 11. Formal Specification. *Communications of the Association for Information Systems, 14*, 184-205.
22. Wynekoop, J. L. and Russo, N. L. (1995). Systems Development Methodologies: Unanswered Questions. *Journal of Information Technology, 10*, 65-73.
23. Yadav, S. and Shaw, N. (2001). Comments on "Factors that Impact Implementing a System Development Methodology. *IEEE Transactions on Software Engineering, 27*(3), 279-281.