View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by AIS Electronic Library (AISeL)

Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2000 Proceedings

Americas Conference on Information Systems (AMCIS)

2000

A Complexity-Based Taxonomy of Systems Development Methodologies

Peter Meso
Kent State University, pmeso@ggc.edu

Gregory Madex
University of Notre Dame, gmadex@nd.edu

Follow this and additional works at: http://aisel.aisnet.org/amcis2000

Recommended Citation
Meso, Peter and Madex, Gregory, "A Complexity-Based Taxonomy of Systems Development Methodologies" (2000). *AMCIS 2000 Proceedings*. 40.
http://aisel.aisnet.org/amcis2000/40

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2000 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

A Complexity-Based Taxonomy of Systems Development Methodologies

Peter Meso, pmeso@bsa3.kent.edu
410A Department of Management and Information Systems
Graduate School of management,
Kent state university, Kent, OH, 44242

Gregory Madey, gmadey@nd.edu
Computer Sciences and Engineering, University of Notre Dame,
Notre Dame, IN 46556

INTRODUCTION

For the last two decades systems developers and researchers have largely assumed that the process of developing business information systems is a well-structured, project-oriented, once in a lifetime undertaking. However, present business models that are intensively reliant on information technology are rendering this perception obsolete. There has been a growing propensity toward increasingly iterative, fast-paced, user-driven systems development methodologies such as Rapid Application Development, Unified Modeling Language, Joint Application Development and the Relationship Management methodology (Hans-Werner1997; Isakowitz, 1995; Shapiro, 1997; Vessey, 1994). At the same time the discipline of information systems has witnessed an increasing awareness of the importance of systems maintenance - with the general proposition that systems development is an ever-proceeding activity (rather than a project-based activity) - becoming the norm (Howard 1990).

The majority of empirical research on systems development, to date, has tested the contributions of different types of knowledge to effective systems development. Much research has also been done on how different methodologies for systems development impact the quality, effectiveness, and efficiency of resultant business information infrastructure. The studies suffer from one or more of the following limitations: (1) They largely perceive information systems evolution via systems development as generally being a slow, linear, structured and continuous process. Current events in the information systems and electronic commerce sectors indicate that systems development is highly dynamic, discontinuous and adaptive in nature – the defining traits of a complex system. (2) Though most of the past studies recognize that systems development is a knowledge intensive activity, rarely is it seen as a primary mechanism by which a firm embeds knowledge into its business information infrastructure's technologies, databases and automated operating procedures. Because almost all business functions and transactions within an electronic commerce enterprise is achieved via the firm's business information infrastructure, enhancement of business knowledge and information within such a firm is expected to be heavily dependent on the enhancements made to that infrastructure via specific systems development approaches. In rapidly evolving environments as characterized by present day electronic commerce, methodologies become a primary means by which the firm continuously updates its knowledge resources hence sustaining or leveraging its competitive advantages. The theory of complexity may contribute to the perception and re-classification of systems development methodologies in such a manner as to provide a clearer understanding of which methodologies are best suited for directing the development and enhancement of business information systems in today's electronic commerce economy. By viewing business information systems as emergent complex adaptive systems, the methodologies employed to derive these systems can be seen as being synonymous to the natural rules that govern the behavior of all natural phenomenon. Thus it enables us to explain what methodologies best match a specific systems development or enhancement tasks allowing for the development of better quality business information systems, especially for electronic commerce applications.

BRIEF SUMMARY OF TAXONOMIES OF SYSTEMS DEVELOPMENT METHODOLOGIES

a) The Software Engineering School

Blum (1994) develops a taxonomy of common methods for designing computer information systems (Figure 1). He perceives design methodologies as being 1) either conceptual or formal and 2) problem oriented or product oriented. While conceptual methods are descriptive in nature, establishing the response to the application-domain need, formal methods are prescriptive and thus set out the behavior of the software to be realized. Methods used also impact the level of validation and verification of the generated solution to varying degrees. While problem oriented methods are excellent at validating the quality of the model derived in the design effort, product oriented methods excel at verification – ascertaining whether or not a formally defined requirement has been met.

	Problem oriented	Product oriented
Conceptual	Structured analysis, Entity relationship modeling Logical construction of systems, Modern structured analysis, and Object oriented analysis	Structured design and Object oriented design
Formal	PSL/PSA, JSD, and VDM	Levels of abstraction, Stepwise refinement, Proof of correctness, Data abstraction, JSP, and Object oriented programming

Figure 1: A classification of design methods (reproduced from Blum, 1994)

Shapiro (1997) outlines the historical development of the discipline of information systems development from the perspective of the software engineering school (Table 1). He enumerates three key systems development paradigms that have governed the development of computer information systems since the 1960s – the process orientation, the data orientation and the object orientation, and outlines key contributors to each of these paradigms. Table 1 provides a summary of the researchers that have contributed to these three information systems development paradigms as reported by Shapiro (1997). Hence Shapiro aptly summarizes development in the software engineering school of information systems development. Barry, Slaughter and Kemerer (1999) study software evolution. They define software evolution to be the dynamic behavior, growth and change of software systems during the course of their productive lives (1999). In particular they identify how software evolution profiles impact important maintenance outcomes such as cost and software errors. Extending Swanson's typology of maintenance types that comprised of corrective, adaptive, and perfective classes into a six class typology made up of data handling, control flow, initialization, user interface, computation and module interface, they found that lifecycle profiles did indeed affect software profiles. Thus the nature of software evolution is influenced by its software maintenance profiles.

Vessey and Conger (1994) apply the same categorization of systems development approaches when they experimentally compare the effectiveness of the process, data and object-oriented approaches in specifying systems requirements. Ahmed and Lochovsky (1998) conduct a comparative case study of three software development environments for database application development. With the aim of identifying if indeed object oriented software development environments reduce development time and improve maintainability, they observe and assess the development of three functionally similar database applications, one developed in an object oriented environment, and the other two in a relational software development environment. Their results indicate that the use of the object oriented environment results in higher quality database applications as measured by number of lines of code, volume, complexity of programs, time used to develop the application, and maintainability.

The Hypermedia Design School

A newer school of information systems development approaches emerged with the advent of the World Wide Web in the early 1990s (Isakowitz, 1995). This school has attracted a significant amount of research with respect to establishing methodologies for web-applications development, comparatively assessing these methodologies to themselves and to those in the conventional software engineering school of systems development, and generation of authoring tools to support these methodologies.

Isakowitz, Stohr and Balasubramanian (1995) propose the Relationship Management Methodology (RMM) for the design and construction of hypermedia applications. They perceive hypermedia design as centered around the management of relationships among objects and acknowledge that by their very nature, hypermedia applications require frequent updating hence necessitating some means of routinizing and automating the initial development and subsequent update processes. The researchers point out those applications that may have irregular structures, those that have dynamic structures, and those that are highly volatile may gain little from the use of the RMM approach. Surprisingly, the majority of present day electronic commerce-applications fall in this category. The volatility of such applications is even the more magnified when viewed from a knowledge management perspective. Hence there is need for improving existing methods to developing and maintaining web-based systems aimed at electronic commerce within the electronic commerce environment. Balasubramanian, Ma, and Yoo, (1995) extend the RMM methodology to reflect the concepts of a cross-entity slice and that of a minimum slice. A cross-entity slice is a combination of elements from different entities into a single window of user-interface display while the minimum slice is the minimal set of an entity's attributes to be included in slices from other entities. They apply the extended RMM to the design of a hypermedia application.

Table 1. Summary of contributors to software engineering school of systems development

Class	Contributions
Process oriented information systems development	Peter Naur (1968):Introduced the term software engineering
	Hoare (1969): Stated that programming was an exact science and as such all its steps could be formalized. Hence fostering the formal methods approach to systems development.
	Niklaus Wirth (1971): Described the process of step-wise refinement
	David Parnas (1972): Articulated the concept of information hiding hence founding the principles of cohesion and coupling.
	Ed Yourdon (Early 1970s): articulated the Yourdon Notation for modeling processes in the analysis and design phases of the systems development life cycle.
	Steven, Myers, Yourdon and Constantin (1973): Combined several basic principles of software development under the label of structured design, formalizing the structured design paradigm of software development.
	Glenford Myers (1973): Described the concept of decomposition which was eventually refined into the concepts of hierarchical and functional decomposition as captured by structure charts, decomposition diagrams, and Warnier -Orr diagrams.
	Dijkstra and Mills (1975): Articulated the three basic control structures sufficient for quality program development as being sequence, selection, and repetition.
	William McKeeman (1975): Described structured programming as being a linear problem-solving process, reinforcing the SDLC and formal methods approach.
	Peter Denning (1975): elaborated the contributions of formal methods in establishing ordered and disciplined thinking leading to clearly structured programs.
	Frank DeRemer and Hans Knor (1975): expanded the meanings of, and distinctions between, intermodule and intramodule complexity hence providing the foundation for the recognition of systems analysis as being distinctly different, and requiring different skill sets, from programming proper (eventually dubbed systems design).
Data oriented information systems development	Barbara Liskov and Stephen Zilles (1975): Explored techniques for specifying data abstractions - spearheading the DATA ORIENTATION approach under the formal methods or structured design paradigm.
	Michael Jackson (mid 1970s): Developed the Jackson method, an approach that centered on the data elements rather than the processes in a software system. Hence spinning off data oriented techniques such as the entity-relationship diagramming, normalization, entity life history diagrams, state transition diagrams.
	Jean Warnier (mid 1970s) developed the Warnier-diagramming notation, later improved into the Warnier-Orr technique by Kenn Orr.
Object oriented information systems development	Brad Cox (1984) and Victor Basili (1985): proposed adding object oriented concepts on top of conventional programming languages.
	Paul Ward (1989): described how to integrate object orientation with structured development methods.
	Russell Abbott (1987: emphasized the crucial role of domain knowledge in effective software development and the potential for object orientation as a technique for articulating such knowledge for future re-uses.
	James Rumbaugh (1991): articulated the Object Modeling technique (OMT)
	Grady Booch (1994): articulated the Booch Object oriented Software engineering (Booch OOSE) Notation.
	Grady Booch, James Rumbaugh and Ivar Jacobson (1995) articulated the Uniform Modeling Language (UML) by building upon and unifying the OMT and Booch OOSE notations. UML is now accepted as the standard for systems specification in object oriented development notation standard.

Hans-Werner, Wicke and Gaedke (1997) propose the term "web-engineering" to refer to the formal approaches for the development and maintenance of web applications. Web applications are characterized as being large scale and distributed in orientation, and as comprising of an increasing number of highly interactive and dynamic components. Therefore, as suggested by the researchers, ad-hoc methods for systems development and maintenance may not be well suited for web engineering. They discuss the modeling of web applications as foundation for web engineering tasks, arguing that the decomposition of web applications into file-based

resources, as evidenced in the RMM and similar methodologies, does not provide the fine granularity required for engineering tasks such as reuse and maintenance. Hans-Werner, Wicke and Gaedke (1997) propose an approach to web engineering, termed WebComposition, which applies an object-oriented philosophy to the development and maintenance of web applications. The approach is embedded in a tool that has the same name.

Three types of web engineering tools exist: tools that support static publication-oriented hypertext, those that support database-centric information systems, and those designed to support highly dynamic and interactive applications. Based on their categorization, full-fledged electronic commerce systems fall into the third category – dynamic interactive applications. While both static links and static pages characterize static hypertext systems, and dynamic page creation but static link structure characterizes database-centric information systems, dynamic interactive applications are characterized both by dynamic page creation and dynamic link structures. Existing methodologies for hypermedia design, specifically the RMM methodology and its techniques are well suited for the first two categories – static hypertext applications, and database-centric information systems. However, they are limited in their ability to address the development and maintenance of dynamic-interactive applications (Hans-Werner, 1997).

Howard (1990) describes the evolutionary approach to systems development which he terms “evolutionary system creation” and distinguishes this from conventional systems development. He states that software evolution is dictated by four rules namely: software evolution: software is not static but always evolving, the process of evolution is slow but continuous, similar systems are related in concept and descend from a common origin, and that systems evolution is the result of a long series of compromises rather than the result of shrewd design. Thus he proposes that firms can capitalize on selective software evolution to leverage their strategic position. This can be achieved via the chunking methodology. Chunking incorporates all of the principles of successful evolution, listed as being: chunks are easier to modify, learning is gradual, users see the results of the requirements, improvements are encouraged, and high return functions are isolated. These principles may apply to the development of software from the complex adaptive systems perspective.

Scharl, and Bauer (1999) use neural networks, to develop and empirically analyze a framework and methodology for analyzing and evaluating commercial business-to-customer electronic commerce systems. The methodology they develop is able to provide for a snapshot, longitudinal, or comparative analysis of electronic commerce systems. Isakowitz, Kamis, A., and Koufaris, (1998) provide an extension to the Relationship Management Methodology (RMM) by introducing the application diagram and demonstrate how it can be used iteratively to refine an application's design. In so doing they provide RMM with capabilities to model a hypermedia system from a Top-down approach while maintaining its Bottom-up approach capabilities. The authors perceive RMM as being employed in a one-time systems development effort (project perspective) where the development of the hypermedia system is done over an established project lifetime beyond which the methodology is no longer required.

The nature of present day electronic commerce is such that the systems development effort lasts for as long as the electronic commerce system remains functional. Hence systems development ceases to be project-oriented in nature and becomes a perpetual or ongoing function of the firm. In such a situation, the methodology used to develop the system needs to allow for the continued re-evaluation, modification, expansion, and even re-engineering of the electronic commerce system in a real-time, zero-down time, environment. Hence the need to reassess systems development methodologies with a view to identifying those best suited for the present day business environment.

OVERVIEW OF COMPLEXITY THEORY

Waldorp, (1992) defines complexity as the “study of spontaneous self-organization and adaptation evident in most multi-agent, multi-reaction natural phenomenon”. Holland (1998) sees it as the ability to increase the intricacy or entanglement evident in systems from permutations and combinations of simple rule-governed models. Complexity theory holds that complex adaptive systems exhibit some common criteria. They consist of multiple agents. They exhibit distributed intelligence and a lack of centralized control. Each agent plays a very specific role or occupies a particular niche within the system and has to be interconnected in some way to the other agents, has to maintain some specific relationships with, or has to interact with other agents in order to satisfactorily perform its role. These systems exhibit a lack of pre-meditated or pre-designed plans of action or maps of direction towards achieving their core objectives. Decision-making in these systems tends to be largely collective. They tend to be perpetually in a state of never ending transformation caused by perpetual dynamic shifts in the state-of-being. In these types of systems, optimization of returns or rents normally lies at some delicate transition point or threshold termed the "edge-of chaos". Extensive collaboration is evidenced in the routine transactions and reactions within the system and the overall behavior of the system EMERGES from the collective collaborative behavior of the multiple agents (Kelly, 1994; Waldrop, 1992; Santosus, 1998; Lissack, 1999).

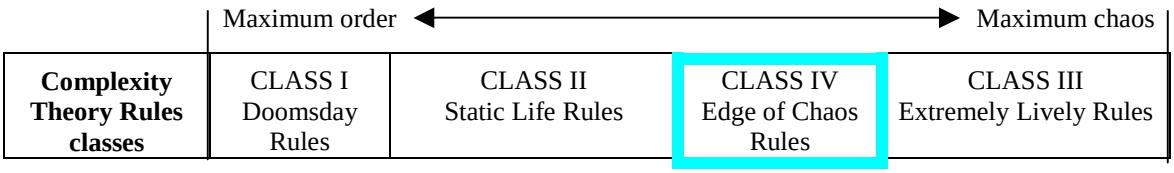

Figure 2: The Four Types of Natural Rules as Defined By Complexity Theory

The theory of complexity classifies all the rules that govern the behavior of complex adaptive systems into one of four classes (Figure 2):

- Class I: Doomsday Rules – those that that propel the complex adaptive system towards maximum order, regardless of the initial state of the system, resulting in the death of the system.
- Class II: Static Life Rules – those that propel the system towards perpetual stagnation, regardless of its initial state, making it static.
- Class III: Extremely Lively Rules – those that propel the system towards maximum anarchy, regardless of its initial state, resulting in maximum chaos within the system.
- Class IV: Edge of Chaos Rules – those that do not maximize chaos or order within the system but instead cause the emergence of coherent structures that propagate, grow, split and recombine in wonderfully complex ways. As the environmental conditions change, these rules perpetually change the structure of the complex adaptive system so that the adaptation of the system to its environment is optimized. Hence the system always remains in synch with its evolving environment (Waldorp, 1992).

A COMPLEXITY PERSPECTIVE OF SYSTEMS DEVELOPMENT METHODS

Perceiving business information systems as being complex adaptive systems leads to the inference of systems development and enhancement approaches as being the rules that govern the systems' adaptation to their environment - the business organization for which they are developed and the larger external environment serviced by this business organization. As such these approaches can be classified as being class I, class II, class III or class IV complexity theory rules based on how they influence the evolution of the business information system, as it adapts to its environment.

Employing the theory of complexity, common systems development approaches can be classified into one of four classes (as depicted earlier in Figure 2). This perception of systems development approaches posits that those methodologies best suited for the perpetual adaptation of a web-based virtual organization and for continued enhancement of knowledge management functions within such an organization, are those with properties closest to the "edge-of-chaos" class of rules. Using the schools of systems development paradigm, all systems development approaches can be classified into two generic classes - software engineering and hypermedia design. By and large, software engineering approaches exhibit higher levels of structure that hypermedia design approaches, and may thus be seen as displaying higher levels of order. Taking the conventional classification of systems development approaches as reported by Shapiro (1997), Vessey and Conger (1994) and Vessey and Glass (1998), the software engineering approaches can be further classified into process oriented, data oriented and object oriented approaches. While most conventional process oriented approaches are highly formalized and linear in orientation, data-oriented approaches and object-oriented approaches are less rigid though still exhibiting relatively high degrees of order (Figure 2).

Past researchers have applied the three orientations – process, data and object – to hypermedia design approaches too. The Relational Management Methodology is an offshoot of the Data Oriented perspective of information systems development (Balasubramanian, 1995; Isakowitz, 1995). WebComposition is an offshoot of the Object Oriented approach (Hans-Werner, 1997). Development languages such as JavaScript and VBScript are mutations of the process and object orientations (December, 1996). Thus we can apply the categorization of systems development approaches into process, data and object orientations to the hypermedia design approaches too. Combining the two provides a complexity-based classification of systems development approaches.

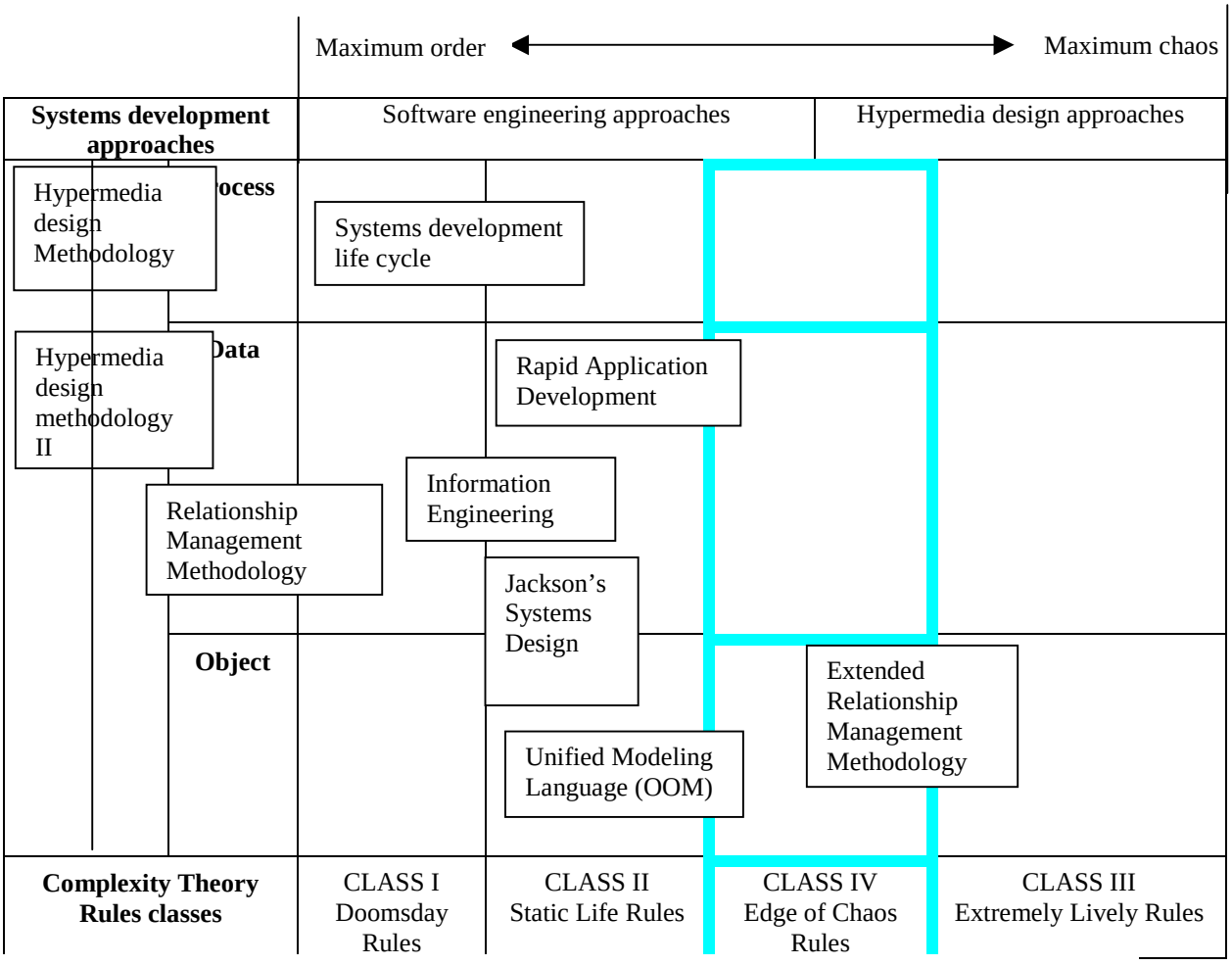

Figure 3: A Complexity-based Taxonomy of Systems Development Methodologies

Most of the software engineering approaches are linear, continuous and highly structured in their disposition to the systems development process. Thus they generally seem to fit the characteristics demonstrated by the Class II: Static Life rules, when perceived from the perspective of Complexity theory. The more rudimentary methodologies such as the Waterfall Model or the Structured Systems Analysis and Design methodology are highly linear and rigidly inflexible. As such they may be closer, in resemblance, to class I rules that they are to Class II rules. Recent software engineering methodologies such as prototyping, the rapid application development methodology and the unified modeling language allow for greater flexibility in the systems development process, and support iterative operations. In this respect they tend towards the class IV rules.

Older hypermedia design methods were highly discontinuous in nature, having very little structure, and embracing the open systems concept. In this respect this group of methodologies - Hypermedia design methodology and enhanced hypermedia design methodology, can be seen as exhibiting the characteristics reminiscent of class III rules. Later hypermedia design methodologies have imposed increasing structure to the development of hypermedia systems while maintaining the flexibility and iterative properties evident in the older methodologies. As such they tend towards class IV rules in their characteristics. These methodologies include The Relationship management methodology and the extended relationship management methodology.

Therefore we are able to map methodologies into one of three orientations - process, data, and object- as elaborated in the systems analysis and design literature. Simultaneously these very methodologies can be mapped into four classes of natural rules as elaborated in the literature on complexity theory. This provides us with a complexity-based taxonomy of systems development methodologies (Figure 3). Given the emergence of electronic commerce, web-based, and virtual organization business models where competition is largely dependent on knowledge resources, and the business environment is increasingly volatile, the taxonomy provides us with a basis for selecting methodologies best suited for the sustenance of corporate information systems with a view

to leveraging the firm's sustainable competitive advantages.

CONCLUSION

This paper had demonstrated the use of Complexity Theory to classify systems development methodologies for electronic commerce. This theory classifies all the rules that govern the behavior of complex adaptive systems into one of four categories: doomsday rules, static life rules, edge of chaos rules and extremely lively rules. It further asserts that the edge-of-chaos rules are responsible for optimizing emergent behavior in a complex adaptive system. Based on the precepts of complexity theory, Hypermedia design approaches, particularly those that take on an object oriented approach to specifying and articulating the problem-solution space may be ideal for systems development in today's electronic commerce economy.

REFERENCES

Balasubramanian, V., Ma, B., and Yoo, J., (1995). A Systematic Approach to Designing a WWW Application. *Communications of the ACM*, 38(8), 47-48

Barry, E. Slaughter, S., and Kemerer, C., (1999). An Empirical Analysis of Software Evolution Profiles and Outcomes. *Proceedings of the 20th International Conference on Information Systems*, 453-458

Blum, B., (1994). A Taxonomy of Software Development Methods. Communications of the ACM, 37(11), 82-94

December, J. and Ginsburg, M., (1996). *HTML 3.2 and CGI*. Indianapolis, IN: Sams.net Publishing

Hans-Werner, G., Wicke, R., and Gaedke M., (1997). WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle. *Computer Networks and ISDN Systems*, 29, 1429-1437.

Holland, J., (1998). *Emergence: From Chaos to Order*. Reading, MA: Perseus Books

Howard W., (1990). Creating an Evolutionary Software System: A Case Study. *Journal of Systems Management*, 41(8), 11-18

Isakowitz, T., Kamis, A., and Koufaris, M., (1998). Reconciling Top-Down and Bottom-Up Design Approaches in RMM. *The Data Base for Advances in Information Systems*, 29(4), 59-65

Isakowitz, T., Stohr, E., and Balasubramanian, P., (1995). RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38(8), 34-44

Kelly, K., (1994). Out of Control. Reading, MA: Perseus Books

Lissack, M., (1999). Complexity: The Science, Its Vocabulary, and Its Relation to Organizations. *Emergence*, 1(1), 110-127

Santosus, M., (1998). Simple, Yet Complex. *CIO*, APRIL 15, 63-67

Scharl, A., and Bauer C. (1999). Explorative Analysis and Evaluation of Commercial Web Information Systems. *Proceedings of the 20th International Conference on Information Systems*, 534-539

Vessey, I., and Conger, S., (1994). Requirements Specification: Learning Object, Process, and Data Methodologies. Communications of the ACM, 37(5), 102-113

Vessey, I., and Glass, R., (1998). Strong Vs. Weak Approaches to Systems Development. *Communications of the ACM*, 41(4), 99-102

Waldorp, M., (1992). Complexity: *The Emerging Science at the Edge of Order and Chaos*. New York, NY: Simon & Schuster Inc.