1-10-2013

Software Development Life Cycles and Methodologies:Fixing the old and adopting the new

Sue Conger
University of Dallas, sconger@gsm.udallas.edu

芽|Sprouts Working Papers on Information Systems ISSN 1535-6078

Software Development Life Cycles and Methodologies: Fixing the old and adopting the new

Sue Conger
University of Dallas, USA

Abstract
Information Systems as a discipline has generated thousands of research papers yet practice still suffers from poor-quality applications. This research evaluates the current state of application development, finding practice wanting in a number of areas. Changes recommended to fix historical shortcomings include improved management attention to risk management, testing, and detailed work practices. In addition, for industry's move to services orientation, recommended changes include development of usable interfaces and a view of applications as embedded in the larger business services in which they function. These business services relate to both services provided to parent-organization customers as well as services provided by the information technology organization to its constituents. Because of this shift toward service orientation, more emphasis on usability, applications, testing, and improvement of underlying process quality are needed. The shift to services can be facilitated by adopting tenets of IT service management and user-centered design and by attending to service delivery during application development.

Keywords: Software development life cycle, methodology, IT service management, user centered design, usability, user satisfaction

Permanent URL: http://sprouts.aisnet.org/10-172

Reference: Conger, S. (2010). "Software Development Life Cycles and Methodologies: Fixing the old and adopting the new ," . *Sprouts: Working Papers on Information Systems*, 10(172). http://sprouts.aisnet.org/10-172

Software Development Life Cycles and Methodologies: Fixing the old and adopting the new

Sue Conger
University of Dallas

ABSTRACT

Information Systems as a discipline has generated thousands of research papers yet practice still suffers from poor-quality applications. This research evaluates the current state of application development, finding practice wanting in a number of areas. Changes recommended to fix historical shortcomings include improved management attention to risk management, testing, and detailed work practices. In addition, for industry's move to services orientation, recommended changes include development of usable interfaces and a view of applications as embedded in the larger business services in which they function. These business services relate to both services provided to parent-organization customers as well as services provided by the information technology organization to its constituents. Because of this shift toward service orientation, more emphasis on usability, applications, testing, and improvement of underlying process quality are needed. The shift to services can be facilitated by adopting tenets of IT service management and user-centered design and by attending to service delivery during application development.

Keywords: Software development life cycle, methodology, IT service management, user centered design, usability, user satisfaction

INTRODUCTION

Information Systems as a discipline is over 60 years old. Over that time, practices have been created and forgotten almost as fast as the technology has changed. An enormous amount of research has produced thousands of research papers relating to information systems development, with many seminal breakthroughs by luminaries such as Avison, Bjorn-Anderson, Boehm, Booch, Brooks, Checkland, Codd, Date, De Marco, Dijkstra, Fitzgerald, Gregor, Hoare, Jackson, Lyytinen, Martin, Mumford, Osterweil, Parnas, Rumbaugh, Schneiderman, Weber, Yourdon and many others.

Even with the thousands of research projects, the track record of information technology (IT) in organizations is dismal. The “IT Department is a source of tremendous frustration, missed opportunity, and inefficiency in companies" (Baschob and Piott, 2007,

p. 11). By one report in 1994, 53% of projects overran original schedules by an average of 222% (Baschob and Piott, 2007). In addition, 31% of projects were cancelled. Completion of projects on time and within budget in large companies was 9% and only 42% or all projects delivered planned benefits (Baschob and Piott, 2007). The situation is such that the IT-business relationship is characterized as hostile in many situations (Agar, et al., 2007; Cuyler and Schatzberg, 2003).

Even with the huge body of research, some IT failures are due to goals that outstrip the techniques and technology of the time. The desire for greater software integration across enterprises, use of leading-edge technologies, and increasing complexity of IT operations technology all have contributed to project failures (Boehm, 2006).

Accompanying the technological aspects of applications that continuously change and get more complex, business too is changing. The current changes business is undergoing are to servitize business operations such that physical products are accompanied by, or embedded in, revenue-generating services. The move to services in the U.S. economy alone is such that over 85% of the economy is involved in service delivery of some type (Gallagher, et al. 2005). As a result, IT that supports business service delivery has become desirable.

At the same time that service orientation is becoming important in business, IT Departments are under pressure to demonstrate their value to their organizations. Statements like, ‘do more with less,’ ‘learn to run IT like a business,’ and ‘join the rest of the company’ demonstrate the pressures on IT organizations (Conger and Schultze, 2008; Cuyler and Schatzberg, 2003). This confluence of pressures, change of emphasis, and history of failures is useful to force self-reflection on the profession to determine its next steps to develop a better rapport with its customers, improve the quality of its offerings, and demonstrate its value to its parent organization.

This paper reflects on the history of software development and its role in the present state of IT in organizations. The discussion focuses on software development life cycles (SDLC) and methodologies and their roles and outcomes as contributing to the pervasive failing state of IT. Key successes and failings are identified to establish a baseline for discussion of how to remedy past weaknesses and improve to address current needs. Then, tenets of design science are adapted to application development issues to discuss needs for changes in practice to adapt to the business shift to services. The outcome is a series of recommendations for academics and professionals to reinvent IT to develop holistic IT services to align more closely to the business services they support.

SDLCs and Methodologies

The most common way of thinking of the SDLC is the waterfall model within which phases of activity are defined based on the thought processes required to conduct the activities (see Figure 1) (Royce, 1972). Output of each phase is input to the next phase. Phases historically included the following with the key focus in parentheses: feasibility (readiness), analysis (what), design (how), detailed design (how), coding and unit testing (technology), testing (correctness), and implementation (transition to operation). On-going maintenance accounts for about 80% of an application's life cycle cost and follows each phase but with a narrower scope than the whole application. In this model, application development ceases at implementation with little attention to use of the application in its various contexts.

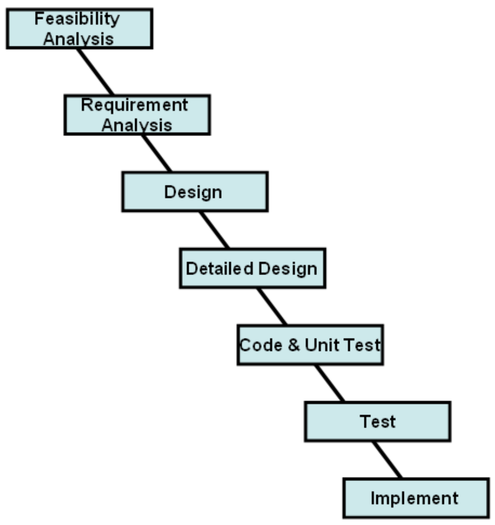

Figure 1. Waterfall software development phases (Adapted from Royce, 1970)

The traditional waterfall outcome is an entire application. Waterfall alternatives are iterative, non-sequential ways of performing the work such as spiral, prototype, and agile (Boehm, 1998; Beck, et al., 2001). Waterfall alternatives are non-sequential development sequences, by which waterfall steps are done on partial functionality with iterations until all functionality is automated. Both of these views of application development focus on application functionality, as opposed to other aspects of the application such as its operational environment, its usability, or its social context. Some authors consider SDLC and prototyping as methodologies (e.g., Avison and Fitzgerald, 2006), while others view them as skeletal guidelines within which methodologies operate (Conger, 1994). The latter view is taken by this research.

A methodology is the tenets, tools, philosophy, and so on about how to approach problem analysis and design. Within a life cycle stage, a methodology guides the work via tools and techniques, focusing analysis on a specific aspect of the work (See Figure 2). Commonly used methodologies foci include process (DeMarco, 1978; Yourdon and Constantine, 1975), data (Jackson, 1975; Martin, 1991), objects (Jacobson, et al., 1999), or stakeholders and the social context (Checkland, 1981).

Criticisms of all of these life cycles and methodologies abound. The most condemning statement is that they appear to make no difference to the resulting quality of an application (Avison and Fitzgerald, 2006). Another is that every focus on one aspect of an application results in ignoring, constraining, or assuming other aspects of the application (Boehm, 2006; Suchman, 1983).

Research on application and software development, methodologies, and SDLC, has led to many discussions of what is wrong with life cycles and methods and invariably, what is next (Avison and Fitzgerald, 2006; Fitzgerald and Fitzgerald, 1999). One answer to that issue is the addition of service perspective to parallel the economic changes to service orientation. Yet, to add new requirements on top of failing work is illogical. Therefore, further assessment of the successes and failures of SDLCs and methodologies is needed to determine what is needed to improve application quality.

Figure 2 summarizes the SDLCs and methodologies to identify their focus and perspective as these constrain how the problem is perceived and, therefore, how the problem is automated. Followers of the waterfall life cycle develop whole applications, decomposing the problem into phases that reflect the thinking for each phase. In contrast, iterative SDLCs focus on chunks of an application and the current period's functionality. By taking a piecemeal view of applications, the iterative SDLCs often result in partially built software that experiences difficulty with integration of later-developed functionality (Abrahamsson, et al., 2002; Boehm, 2006).

Soft Systems methods originate from Checkland (1981) and are expanded by Wood-Harper and others (Doyle, et al., 1993). The focus of Soft Systems is the social system as a basis for change that results in an application. The Soft Systems approach views application development as a cultural activity inclusive of as many stakeholders as can be accommodated, and therefore, can drag on without progress for long periods. Contradictions arise when different groups air their priorities and the contradictions can be difficult to resolve (Mathiassen and Nielsen. 1989). Once complete, Soft Systems applications result in high levels of user satisfaction (Checkland, 1981). Soft Systems highlights the importance of situated work that requires attention by IT of both the automated and non-automated aspects of the work (Suchman, 1983).

	Life Cycles		Methodologies	
Charact-eristic	Sequential SDLC	Iterative SDLC-- - Prototype, Agile	Soft Systems Methods –	Process, Data, Object Method –
Purpose	Design and implementation of work support systems	Focus on functionality and/or timing of delivery	Focus on contexts and stakeholder rights	Focus on area of interest
Goal	Complete functionality	On time, short-term delivery of partial functionality	Contextualized design	Focus on area of complexity to ensure its correctness
Perspective	Design thought processes	This period's functionality	Organization, information, technology, and socio-technical aspects of the problem	Functional quality of the most complex aspect

Figure 2. Perspectives from Life Cycles and Methodologies

Process, data and object methods are grouped because they all focus attention on a key area of complexity in the application as functionality, data, or objects, respectively. Object methods have matured somewhat and morphed into service oriented architectures (SOA) but object concepts and focus do not change in SOA. As a focusing mechanism, these methodologies function as intended. However, these methodologies constrain thinking in the same way as the SDLCs and other methods through the very focus they seek. By focusing on functionality, the social system, interface design, or other aspects of an application may be ignored.

All of the SDLCs and methodologies in Table 2 have shortcomings as a group in that they provide tools and techniques without providing an overall checklist of what should be evaluated and considered within the context of applications development. Moreover, the SDLCs and methodologies alone do not give clues about how to fix the failures of application development let alone how to improve it to deal with today's application needs. The next section looks at successes and failures in application development practice to determine the characteristics most needed in successful applications.

APPLICATION DEVELOPMENT

With all of the failures of information systems, we sometimes forget that there are also impressive successes. The aerospace and defense industries have sent and returned people to the moon and kept bombs from exploding before their time. Virtually every home device has imbedded computer chips, which run appliances and simplify our lives. These successes have many characteristics in common. These characteristics may vary by type of application but some characteristics cross application types.

Successes in Application Development

Systems success is best summarized by the DeLone and McLean success model (1992; 2003; and Petter, et al., 2008), which found the following constructs of importance:

Key Driver	Sub-Characteristics
Systems quality	Adaptability Availability Reliability Response time Usability
Information quality	Completeness Ease of understanding Personalization Relevance Security
Service quality	Assurance Empathy Responsiveness

Use	Nature of use Navigation patterns Number of site visits Number of transactions executed
User satisfaction	Repeat purchases Repeat visits User surveys
Net benefits	Cost savings Expanded markets Incremental additional sales Reduced search costs Time savings

Figure 3. Key Drivers of Successful Information Systems (DeLone and McLean, 2003, p.26)

DeLone and McLean built on hundreds of other research projects to develop both a parsimonious list of critical factors that generally fits all applications. The details of each characteristic is beyond the scope of this paper, but the key drivers are of interest because they span applications types in some form with many sub-factors seeming to be universal, as well. Three types of quality are expected of successful applications: System, information and service. Systems quality refers to the application in its operational environment and the extent to which it performs at the time needed and in the manner expected. System quality is important because inattention to system quality early in the development cycle can easily result in poor quality upon implementation.

Information quality refers to the suitability and usefulness of the data provided to the user. Information quality in any transactional system needs to be complete and accurate. Similarly, relevant, secure data seem to be universal in their appropriateness across application types.

Service quality also may be appropriate for all applications but in a different sense than expressed by the sub-factors provided here. The sub-factors in the De Lone and McLean list are from SERVQUAL, a well researched model of service quality in an online environment (Parasuraman, et al., 1988). SERQUAL needs additional research to determine characteristics that fit other arenas of IT support. For instance, extensions to SERVQUAL to adapt measures of quality from the total quality movement might be appropriate. In a broad services context, service quality refers to overall quality provided by the 'system' and can include the application, help desk, maintenance staff, and others in the IT Department who might interact with users for some reason. Specifics of services are not yet incorporated into service quality research or measures. Thus, a more general view of services, which is consistent with servitizing tenets (Van Bon, 2007) indicates a need for expansion of SERVQUAL for IT services management quality. Gap analysis to evaluate expectations versus attributes of objective product, specific characteristics of service quality (e.g., help desk resolves problem during first contact), definition of customer benefits, and usefulness are other potential additions to SERVQUAL that may improve its applicability to information systems (Chen and Sorenson, 2007). Further, contextualizing service concepts may lead to more accurate measures of service. For instance, in e-commerce, service and system quality are interwoven and no known research has teased out the nuances of their differences.

User satisfaction also is a well-researched area but it has little research relating user satisfaction across application types. The complexity of attitudes and the nature of the application types, designs, and possibly other factors may impact user satisfaction (Melone, 1990). Therefore, while the concept seems relevant across all applications, the details of its measurement as presently operationalized need further contextualization.

The final component of applications success, net benefits, also seems to apply across the board to all applications. The concept of net benefits in terms of evaluating business outcomes is not new but has been elusive and difficult to quantify (Brynolffson and Hitt, 2003). Research on how individual IT efforts relate to, support, and ultimately contribute to business outcomes is critical as IT struggles to remain relevant to its parent organization (c.f., Cuyler and Schatzberg, 2003).

Thus, even though De Lone and McLean's success model and SERVQUAL measures appear to have significant carryover across application types, more research is needed to contextualize their constructs (Petter, et al., 2008).

Failures in Application Development

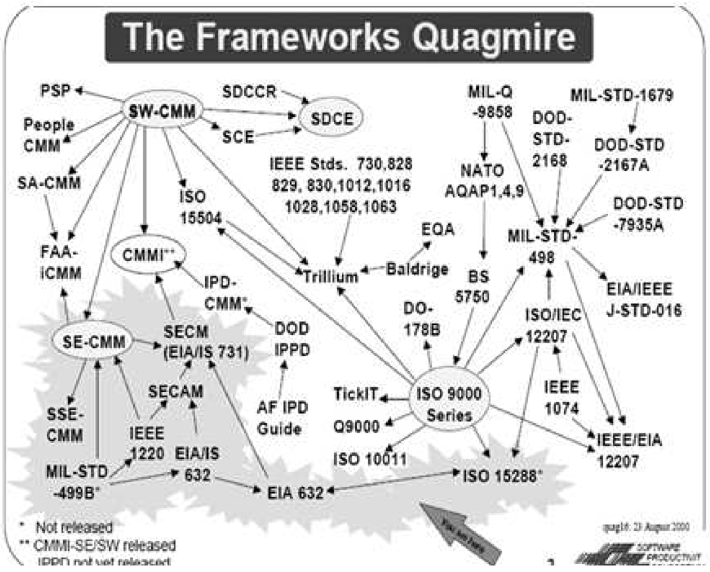

Figure 4. Software development frameworks c. 2000 (Doran, 2000, p. 3)

By examining SDLC and methodological failures, we can back into a definition of what leads to successful implementations. The shortcomings are not simple however, as SDLCs and methodologies are not the only issues. This section examines failings of IT development and acquisition organizations, and thereby, determine what aspects, if done some other way, could contribute to success. In addition, research on information systems risks also is relevant to failure discussions because risks not attended to are likely to lead to failures of the resulting information systems.

Confusion about SDLCs and Methodologies

From a standards perspective, there are simply too many standards relating to SDLCs and methodologies. By one count, there are over 1,000 methodologies alone (Avison and Fitzgerald, 2006). This quagmire of differing descriptions of essentially the same things, all with different breadth, depth, and focus, is a source of significant confusion. Figure 4 shows just standards of the International Standards Organization (ISO), the Institute of Electrical and Electronics Engineers (IEEE), and U.S. Department of Defense and their intellectual linkages.

Figure 5 shows one description of the full extent to which whole bodies of knowledge relating to many hundreds of methodologies and life cycles proliferate

(Boehm, 2006). It also shows the development of information systems as a profession that has adapted and changed to deal with the overriding complexity of each decade. For instance, the craft of programming gave way to structured methods, which morphed into productivity-oriented frameworks, that then needed to deal with concurrency, increased pressures for productivity, and eventually, global connectivity.

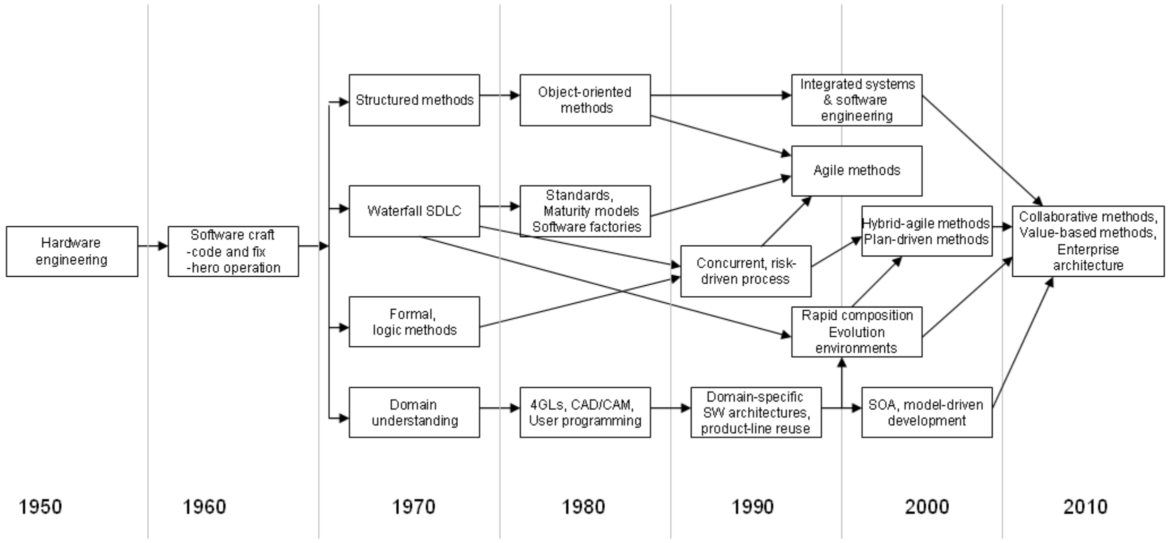
Hardware engineering
Software craft -code and fix -hero operation
Structured methods
Object-oriented methods
Integrated systems & software engineering
Agile methods
Waterfall SDLC
Standards, Maturity models Software factories
Concurrent, risk-driven process
Hybrid-agile methods Plan-driven methods
Collaborative methods, Value-based methods, Enterprise architecture
Formal, logic methods
Rapid composition Evolution environments
Domain understanding
4GLs, CAD/CAM, User programming
Domain-specific SW architectures, product-line reuse
SOA, model-driven development
1950
1960
1970
1980
1990
2000
2010

Figure 5. Progressive development of methodologies and life cycles (Adapted from Boehm, 2006, p. 16)

As these figures depict, the linkages and profusion of frameworks foster confusion more than understanding. Companies trying to determine which, if any, method or SDLC is right for a single project often abandon the search when faced with the variety of available choices. Some authors recommend evaluating the suite of alternatives to develop the set of techniques, tools, life cycle, and methods that best fit the problem (Brinkkemper, 1996). But, as a result of confusion relating to the plethora of tools, techniques, methods, and so on, companies that do use methodologies often select one, using it as the guiding outline for all project work. This practice leads to the second major shortcoming: Practice failings.

Practice Failings

Several practice failings are discussed in this section. First, the use of a single methodology to guide all project work is a failing because there is ‘no silver bullet’ and no one SDLC or methodology can usefully guide the variety of work done in a typical IT development department (Brooks, 1975, 1987).

Second, practitioners do not do a good job of practicing what is taught or researched. As many as 50% of programmers have less than four years of college, are overwhelmed by their work, and do not use good software or design practices (Boehm, 2006). The same applies to newer disciplines, such as user-centered design (Høegh, 2006; Mai, 2005)

Third, many risks attendant on development projects are ignored. Major project practice risks relate to realism of schedule and budgets (Boehm, 1981; insufficient user involvement (Dodd and Carr, 1994); insufficient attention to functional complexity (Boehm, 2006; Ewusi-Mensah, 2003); inability to learn from past failures (Lyytinen and Robey, 1999); insufficient attention to user interface (Keil and Carmel, 1995); problem avoidance (Keil, 1995; Sherman, et al., 2006); inability to control project scope (Boehm, 1991; Ewusi-Mensah, 2003; Markus and Keil, 1994); and lack of adequate technical skills (Boehm, 2006; Ewusi-Mensah, 2003; Sumner, 2000).

Development practices and failure to manage risks are not the only failing. Most companies do not follow any methodology or life cycle. They simply use the same tools and practices they have used in the past, much like using a hammer to fit a screw because it is the tool that is known. Such uses of methods that do not fit the problem are known to contribute to project failures (Boehm, 2006; Brinkkemper, 1996; Mai, et al., 2005).

Agile has recently been touted as a life cycle that provides productivity with less formality than past methods and life cycles. It provides a useful example of the shortcomings that are present to greater or lesser degrees in other methods and life cycles. Many practitioners of the current fad Agile do little or no requirements definition before beginning to code (Abrahamsson, et al., 2002). In addition, there are several different methods within the 'agile' life cycle and each is limited in some way. For instance, agile spreadsheet development (ASD) focuses on concepts and culture rather than on functionality and correctness; extreme programming (XP) develops no overall view, making integration of final products difficult; rational unified process (RUP) does not provide details on how to obtain requirements or how to tailor its methods for a given project type; and Scrum details 30-day release cycles but provides no integration or acceptance testing in its methodology descriptions (Abrahamsson, et al., 2002). In addition, many practitioners of agile methods select simple, easily implemented functionality as the early project work to provide fast turnaround and build rapport with their clients (Boehm, 2006). However, they then miss the complexity of later functionality and experience difficulty integrating complex functions after-the-fact (Boehm, 2006). When this functionality affects the user interface, projects are more likely to be cancelled (Markus and Keil, 1994).

Application Development Management Issues

Developers are not alone in their application development failings. Managers also are less attentive to application development than needed to ensure their success (Sumner, 2000). The role of a project manager traditionally has been as the most senior technical person who also has managerial duties for the project (Conger, 1994). For instance, the project manager and key technical staff decide the methodology, the life cycle, the tools, and the resources needed for the project. In addition, the project manager, with key staff, develops the work breakdown, project plan, and skills desired for each task. The project manager is the main client liaison. In this role, the project manager attends the requirements elicitation meetings, sometimes as the analyst, gaining the understanding of the required functionality. In addition, the project manager is the official communicator of project status, problems, and work. Thus, the role has many gate-keeping functions that provide for filtering information (Keil, 1995), gaining commitment of other managers and user management (Sumner, 2000), and hiring or firing employees from a project (Conger, 1994; Sumner, 2000).

Risks associated with the managerial roles include scheduling, budgeting, assignment of personnel, management of personnel, acquisition of sufficient IT resources, dealing with training needs of assigned staff, ensuring sufficient user involvement dealing with problems as they arise, and controlling scope creep (Boehm, 2006; Ewusi-Mensah, 2003; Markus and Keil, 1994; Sherman, et al., 2006; Sumner, 2000). To the extent that these risks are not attended to, project success becomes less likely.

Thus, from analysis of failures, if the wrong people do the wrong things, use the wrong methods and techniques, and do not attend to the necessary variety of complexity, application success is unlikely. Fixing these problems sounds like a simple matter of attention to details but there is an elusive 'sweet spot' of project contextualizing that needs further research to become fully articulated (Conger, 2010c).

KEY ISSUES IN FUTURE APPLICATION DEVELOPMENT

This section takes a design science perspective of the future needs in IT systems design to address the shortcomings and incorporate the positive aspects of application development from the previous section (Hevner et al., 2004). By adapting the seven guidelines from Hevner, et al. (2004) all aspects of future systems design are evaluated to identify repetitive themes of application development. The themes are used to develop the key issues for future systems design.

Systems Artifacts

Application systems are the key artifacts that derive from the development process (Guideline #1, Hevner, et al., 2004). However, contrary to what is taught in most systems analysis and design (SAD) texts, the system should not be the sole focus of development.

The perspective needs to shift from application-as-end to application-as-imbedded component within work service systems (see Alter, 2010). The two work systems of interest are the one that serves the main business purpose and the one that supports the operational application within IT. One way of altering the SDLC is to review each area of operational support needs during each phase of the chosen life cycle to determine the applicability of the various services activities (Gupta, 2008). In particular, during requirements elicitation, the non-functional requirements should be defined for security, reliability, accessibility, application support, and capacity, to name a few. The purpose of application development then shifts to become the delivery of IT-based work support capabilities that provide measurable business value within a services delivery context.

ISO/IEC 15288:2002 for application development is appropriate to initiate this shift (ISO/IEC, 2002). The standard identifies not only the functional application requirements for its focus but also advocates consideration of key operational aspects of applications during development. For instance, the phases in the standard include concept, development, production, utilization, and support (ISO/IEC, 2002). Each phase contains activities that look forward to the ability to operate the application as shown in Figure 6.

Phase	Application Activities	Operational Activities
Concept	"The preparation and baselining of stakeholder requirements and preliminary systems requirements (technical specifications for the selected system concept and usability specifications for the envisaged human-system interactions)" (p. 44)	Initial specification of infrastructure (p. 44)
Development	Technical data package, including as appropriate: 1) hardware diagrams, simulations; 2) software design documentation; 3) production plans training manuals for operators; and 6) maintenance procedures (p. 45)	Refined objectives for the production, utilization, support, and retirement (p. 45)
Production	It is presumed that the organization has available the production infrastructure, consisting of production equipment, tools, procedures and competent human resource (p. 45) to operate the application	Outcome packaged product transfer to distribution channels or customers (p. 46)
Utilization	The application is "installed and used at the intended operational sites" (p.46).	The application is "installed and used at the intended operational sites" (p. 46).
Support	"The Support Stage begins with providing maintenance, logistics and other support for the system operations and use" (p. 47)	Support includes " Maintained system product and services and the provision of all related support services " and " logistics, to the operational sites" (p. 47)

Figure 6. Application and Operations Activities (ISO/IEC, 2002)

The ISO/IEC 15288 standard is too generic to guide all activities but it does provide a checklist of major items for consideration during each phase of development. If coupled with ISO/IEC 20000-1, the standard for IT service management, anticipating the needs of the operational environment at each stage makes application compatibility with the service in which it is imbedded more likely (ISO/IEC, 2005).

Problem Relevance

In this discussion, relevance (Guideline #2, Hevner, et al., 2004) relates to the business need for the application and the extent to which the need is met. This broad definition moves focus from the application artifact to its situated operational context and includes all aspects of support for applications use in addition to its development quality.

Financial Relevance

A cost-benefit analysis of the application that includes risk assessment and mitigation strategies, work breakdown and project plan, and an analysis of the expected financial payback are assumed. As many as 80% of projects are conceived and begun without any planning beyond what is due in a given time frame (Eberlein and Sampaio do Prado Leite, 2002). Without expected benefits, application relevance can easily be sidetracked.

In addition to developing application expectations, post-implementation audits and performance measures should be conducted to determine that the payback is, in fact, gained. However, 80% of U.S. companies have no post-implementation audit (Levinson, 2003) and 84% of U.S. companies do not report metrics on financial performance. One study of seven countries found that at least 67% of companies did not measure IT value of any kind (Infosecurity.com, 2009).

Business Process Relevance

The relationship between business processes and automation that supports them is not a well researched area. By focusing on application artifact development and ignoring its operational context, the solution is likely sub-optimal (Conger, 2010b; Checkland, 1981). In addition, automation without process management is likely to yield no payback to the parent organization while process design preceding automation can yield a 20% return (Dorgan and Dowdy, 2004).

Processes are the heart of services; they are "interface between the strategy and its execution" (Goldenstern, 2010, p. 6). With this crucial role, Goldenstern recommends that software should conform to an optimized process, interfaces should be simple and managed, reliance on time and resolution in support actions, task training, and service training all should be developed. Outcomes of these efforts are rewarded with an average 18% reduction in incident resolution times and a focus on providing customers the 'best' service (Reichheld, 2003), improved customer satisfaction and loyalty, and sales (Goldenstern, 2010). In addition, process "standardization truly enables leverage," leading to reduced cost of creating applications by 50% to 80% while boosting companies' ability to bring new products to market faster (King, 2009, p. 1). Process standardization can generate repeatable outcomes at a defined level of quality. Processes need to be viewed, not as stand-alone any more than an application is stand-alone, but as embedded within a service context that delivers value to the organization's customers. The notion of process as embedded in a service is discussed in the section on contribution.

A focus only on the business process of an application means ignoring the support processes needed by IT staff. Some authors argue for addition of user interaction analysis, non-functional requirements, and change management to improve software quality (Conger, 2010b; Eberlein and Sampaio do Prado Leite, 2002; Gupta, 2008; Pollard and Cater-Steel, 2009). For example, standardized messages that identify failings in an application should be designed and used across applications to simplify help desk outage resolution (Gupta, 2008). This implies design of two types of error messages -- those for business users and those for IT users. In addition to these simple changes, definition of standard processes for the IT function that incorporate services perspectives should lead to improved application quality both for the business function and for IT operations support functions.

Development Rigor

Rigor in Hevner, et al. (Guideline #5, 2004) refers to research rigor while herein the rigor is directed at application development and its operational instantiation. System quality is the focus of this discussion.

System quality has been viewed from several perspectives relating to the overall system, application, and its information. System quality in terms of operations refers to reliability, availability, accessibility, security, and compliance (Gorla and Lin, 2010; Van Bon, 2007). Application quality relates to effective development and deployment of applications (Arnott and Pervan, 2008); reliability, ease of use, and usefulness (Gorla and Lin, 2010); and completeness, consistency, simplicity of learning, flexibility, sophistication, reliability, customizability, and functionality (Guimaraes, et al., 2009; Petter, et al., 2008). Information quality characteristics relate to accuracy, completeness, currency and format (Nelson, et al., 2005).

System quality research is an expansion of application quality that includes characteristics of operational, information, and service quality as contributing to overall quality perceptions (Arnott and Pervan, 2008; Gorla and Lin, 2010; Petter, et al, 2008). Key facets of application context are omitted by failing to evaluate the human-computer interface or the variety of users from business users to IT operations users and Help Desk staff (cf., Guimaraes, et al., 2009). Yet, no comprehensive definition of system quality in all of its contexts has emerged. Operational quality present in, for instance, the IT Infrastructure Library (ITIL) (Van Bon, 2007), is not discussed in texts on systems analysis and design. Nor do the frameworks and standards that include operational quality describe how best to design applications for operational or service quality. These are areas for future research. As a result, system quality needs careful definition for each application context to ensure that the development activities address all requirements.

Systems as Search Process

Thinking of a system as a search process (Guideline #6, Hevner, et al., 2004) leads to discussion of innovation and improvisation in the application development activity.

Innovation

Innovation relates to the introduction of processes, artifacts, tools, techniques, or technology that is new to an organizational setting (Prescott and Conger, 1995). Innovation is a key CIO priority (CIO, 2009). Innovation is viewed as integral to information systems since the IT function is generally tasked with bringing new technologies into the organization. Innovation research relating to IT usually refers to the adoption of technology. Most studies relate to organizational adoption that omits or minimizes the role of IT organizations in the adoption process (Prescott and Conger, 1995).

Innovations in IT units can be either technology or process related. Of six such studies, five relate to individual adoption of a technology and one relates to general database machine innovation (Prescott and Conger, 1995). One shortcoming of research on IT innovation is that research on adoption and use of new techniques, methods, design ideas, frameworks and other process-related innovations is lacking. As a result, innovation impacts on the IT organization remain largely unknown.

Changes to life cycles for innovation are also mostly missing with the exception of environmental innovations. Environmentally sustainable innovations are the "IS-enabled organizational practices and processes that improve environmental and economic performance" (Melville, 2010, p. 1). Evaluation of outsourcing, co-production, and environmentally improved technology for any new application can reduce its environmental impacts (Conger, 2010b). Altering application development to include a life cycle analysis of the application's environmental impacts and mitigating or negating the impacts to the extent possible is also suggested (Melville, 2010). Such altering of the life cycle might be done for any innovation, but the environmental innovation recommendations demonstrate opportunities to develop innovation adoption research and practice for IT applications beyond its present state.

Improvisation

Improvisation is comprised of extemporaneous processes based on expertise that serve as coping mechanisms (Ciborra, 1996, 1998). Improvisation is important in information systems development because regardless of how standardized a process is, unexpected events, outcomes from prior decisions, and actions by project members require constant evaluation of impacts and adjustment of schedule, outcome definition, or budget, as needed.

While improvisation is needed, the result still needs the requisite discipline of any planned activity (Ciborra, 1998). The balance between improvisation and standardization is precarious but the outcomes of both require knowledge and discipline to develop purposefully designed artifacts (Hevner, et al., 2004). More research on the nature, idiosyncrasy, and manageability of improvisation is needed to understand how it works in IT applications sourcing.

Design Evaluation

This section discusses design evaluation for application systems in terms testing and walkthroughs (Guideline #3, Hevner, et al., 2004)

Walkthroughs are structured meetings for finding errors in requirements, designs, code, test plans, or other system artifacts (conger, 1994). Walkthroughs are successful at finding significant errors and, by having the errors corrected during the development process, walkthroughs significantly reduce the cost of the application. The estimated annual cost of software defects is $59 billion, of which $22 billion could be avoided through walkthroughs (Rombach, et al., 2008). Only about 35% of companies practice any type of walkthrough, providing a significant opportunity for its adoption (Rombach, et al., 2008).

Testing is the art of finding problems in code (Myers, 1979). Testing as an area of application activity can focus on everything from individual code modules to stress testing to find limits of an application's use. Problems can relate to functionality, formatting, lack of relationship to requirements, limits or constraints, security, usability, and performance, to name a few (Myers, 1979; Kaner, 2001, 2003). Many organizations have a quality assurance function that develops acceptance tests as a gate keeping function for the client organizations.

Testing failures are well known and some of those failures lead to tragedy. Between 2008 and 2010, "system vendors reported 260 system malfunctions that caused 44 injuries and six deaths" in a single application (Brewin, 2010, p. 1). Most applications enter their production state with known errors and many applications experience errors throughout their productive lives (Baschob and Piott, 2007).

There is little agreement on many issues in testing, including the following. What constitutes testing? Are there testing 'best practices'? Is all testing contextual and unique? Should waterfall or agile be used as the overall model for when testing should be done? Should testing focus on functionality or usability or something else? Are scripts the best method for testing (Kaner, 2001, 2003)? The ultimate goal of testing research is fully automated testing but that remains an elusive dream at present (Bertolino, 2007). In addition to needing more research, testing is a subject often left out of programming classes beyond getting syntax and logic of simple programs to work. As a result, while testing sophistication has increased measurably in the last ten years, most practitioners do not know about that progress (Bertolino, 2007).

Organizational Contribution

While Hevner, et al. (Guideline #4, 2004) address research contribution, in the context of application quality, thinking of organizational contribution is more appropriate. Completing an application is insufficient to develop a contribution. Rather, the application in use, must comply with all of its needs. The irony of the prior statement is that application developers tend to think of 'needs' as only functional requirements. Rather functional and non-functional requirements are necessary, as are requirements for more ephemeral aspects of contribution such as simplicity, learnability, and so on (Nielsen, 2000). To determine value added to an organization, IT must measure and manage its activities, particularly those that determine organizational success. Current thinking on these operational activities is that taking a services orientation that mirrors the services orientation the organization seeks to perfect, will lead to value-adding outcomes for IT. This section develops the concepts of service orientation and discusses it in the context of the IT operations environment.

A 'service orientation' is one in which the organization provides intangible service thus, generating value to its customers. Value includes many characteristics for instance, need satisfaction, prompt and friendly interactions, and minimal clicks on a web site (Conger, 2010c; Deloitte, 2002). A service design takes a defined process and situates it in a governance and management structure, defines number and nature of work for multiple locations, defines software, data, and IT resource support for the functions and roles, and defines service levels for customer delivery including response time, service desk response time, and so on (Conger, 2010a). This differs from typical application design by defining the application plus its customer context, plus its IT contexts for on-going operation. Services are composed of key components for utility and warranty. Utility addresses the traditional functional aspects of applications and conduct of work (Conger, 2010a). Warranty addresses the non-functional, but increasingly important aspects of IT work. Examples of warranty include computing availability and reliability, response time for a service request, response time for simple outages, etc. Services have a life cycle that parallels the business product life cycle, beginning with business strategy, progressing to initiatives, tactics, processes and products, and production. ITSM life cycle mirrors this business life cycle and should be fully integrated and part of each step of the business service life cycle, from strategy formulation through retirement (Conger, 2010a).

Moving to a service orientation is not without cost. Some of the key costs relate to training, travel, and communications for project team members involved in design and implementation of the services efforts. Understanding and communicating semantic nuances of terminology and getting to an understanding of what it means to deliver a service is an early challenge (Winniford, et al., 2009). Training and communications costs extend to anyone touched by or managing services changes. Changing culture to a service-orientation is a difficult aspect of services adoption and also adds to service adoption costs (Conger and Picus, 2009).

ITSM innovation requires management of tradeoffs – development of an ITIL bureaucracy versus standardizing but remaining Spartan, blind adoption of all of ITIL or ISO/IEC 20000 versus adoption of selected processes and services based on need and value-adding potential, and rote versus contextualized adoption of processes and service (Cater-Steel, et al., 2008, Conger and Schultze, 2008; Marrone and Kolbe, 2010a, 2010b).

Many benefits have accrued to companies that successfully implement services. Examples of benefits include missed service level agreement target penalty reductions of as much as 80% in two years (Conger and Picus, 2009), increases in service quality, global process standardization and resulting reduced expenses and increased customer satisfaction, reduced outages and related downtime of operations, improved staff mobility, improved financial control, and improved IT morale (Cater-Steel, et al., 2008; Conger and Picus, 2009; Conger and Schultze, 2008; Dubie, 2002; Hochstein, et al., 2005; Lynch, 2006; Marrone and Kolbe, 2010a, 2010b; Pollard and Cater-Steel, 2009; Potgeiter, et al., 2005).

Though services provide significant benefits upon adoption and maturation of practice, issues with ITSM adoption exist. Challenges of adopting ITIL include the need for executive sponsorship, the need but business understanding of ITIL objectives, adequate resources, time, people with ITIL and change management knowledge and skills, funding for training, travel, certification if needed, and implementation activities, maintenance of momentum toward changes (Marrone and Kolbe, 2010). The demonstration of results after a short period of ITIL use is important to silencing change critics (Hochstein et al. (2005). Yet, virtually every project reports resistance even with quick results that must be successfully countered to ensure project success (Cater-Steel, et al., 2008; Conger and Schultze, 2008; Conger and Picus, 2009; (Marrone and Kolbe, 2010).

The risk-reward payoff is significantly weighed in favor of rewards for successful ITSM projects (Cater-Steel, et al., 2008; Conger and Picus, 2009; Conger and Schultze, 2008; Potgeiter, et al., 2005). However, two aspects of services are important to consider for organizational contribution. First, is the application as imbedded in its business service function and the value that accrues to the organization as a result of the application. There is little research on this area but it is a crucial aspect of an application that determines its importance to the business. Second, is the application's operational environment and how process-driven and smoothly it operates in both normal and outage situations. There is also little research on this area beyond case studies. Thus, both areas need further research to describe how best to accomplish service embeddedness and its contribution to the business.

Systems as Communication

The concept of systems as communication, adapting from Hevner, et al. Guideline #7 (2004), is not well articulated. One conception is that of how information accessibility is a form of communication between the application and the user (Culnan, 2007). From this perspective, communication occurs from physical access to the source, the interface to the source, and the ability to physically retrieve potentially relevant information (Culnan, 2007).

A different perspective is that the human interface is a form of communication between the developers (and management) to the application users (Nielsen, 2000). From this perspective, application usability and user experience are key outcomes of the communication.

In both senses of the term communication, application usability refers to incorporation of both needed functionality to accomplish a goal and characteristics such as effortless learning and remembering, usage efficiency, eliciting few errors, and subjectively pleasing use (Nielsen, 2000). Usability is an application feature that has a long history in terms of human-computer interaction (HCI) research with seminal works by, for instance, Ben Shneiderman (1997). Low usability relates to non-use of applications (Markus and Keil, 1994). However, usability is measured as a component of information quality, implying that the only usability is for data generated by an application (Petter, et al., 2008). Usability should also be a feature of application quality to develop measures of the extent to which the interface engages and is useful to its users (Nielsen, 2000, 2005).

User experience refers to the feelings and attitudes developed by users of an application and embodied in the application characteristic usability. The term user experience is more general than many related, constituent predecessor terms such as user satisfaction, information system effectiveness, performance, and so on (Melone, 1990).

Product usability and user experience are related because they evaluate different aspects of the same phenomena. The phenomenon under study ultimately is the user experience. The assumption is that the more enjoyable and satisfying the experience, the more likely the user is to use a system. Melone (1990) analyzes outcomes while the research conducted by Nielsen (2000) analyzes characteristics that lead to the outcomes. Nielsen articulates characteristics to be designed into an IT artifact, which ultimately is the goal of application development and the approach that will be discussed here.

Key components of usability are ease of learning, ease of remembering, usage efficiency, minimal error elicitation, and usage esthetics (Nielsen, 2000). Note that functionality is still important in terms of practical acceptability but that usability focuses on user perceptions and ability to actually use the application. Learnability and memorability both have aspects of design for experts and novices in either the knowledge domain or in use of computer interfaces. Learnability refers to the length of time and amount of effort required to learn the software. Memorability refers to the extent to which the software is easily memorized. At best, a usable interface is intuitive, requiring little or no learning and little effort. One problem with usability is that the user is defined as the end user, who will be the daily user of the interface. However, little attention is given to the Help Desk staff that must also interface with the application whenever it exhibits problems. Similarly, there is little thought given to error messages. For instance, "Bad data" often seen as an error message, however, the name of the data field, its location in the program, the exact error, and guidelines on how to fix the error all are missing. If provided, the time to locate and remedy bugs can be cut by orders of magnitude (Gupta, 2008).

Efficiency relates to user development of a consistent, steady-state of performance over time that does not require extraneous, non-value adding activities. Efficiency, too, is viewed from the perspective of the business end user. If Help Desk efficiency were also considered during design, resolution time user and system problem would be reduced (Gupta, 2008). With poor error messages, no learning can take place beyond how to locate a problem in this program, and therefore, no efficiencies can be gained.

Satisfaction relates to game-like qualities that allow a user to develop a state of flow such that they become engaged in the application and derive satisfaction from its use. Most applications ignore this aspect of design for all users, not just IT support. While there is high quality research on interface design and usability, there is no known research that links all of the characteristics to user experience (e.g., Norman, 2002; Shneiderman, 2004). Most application research links usability characteristics to application usage or generic user satisfaction. There are few best practices that identify all aspects of all of the components in a single publication or that are universally applicable across application areas, cultural contexts, or user types (Nielsen, 2000). As a result the application developer must read a significant body of work (c.f., Jokela, et al., 2003; Jones, 1992; Kaikkonen, et al., 2005; Lewis, 1995; Nielsen, 2000, 2005; Norman, 1998; Park, 1997; Shneiderman, 2000, 2004) to develop even an inkling of the global thought on usability and the parent field of research on human computer interaction (HCI) (Zhang, et al., 2007).

Early ISO standards relate to usability – ISO/IEC 13407 and ISO/IEC 9241-11 (ISO/IEC, 1999; Jokela, et al., 2003). ISO 13407 defines user-centered design as the "level of principles, planning, and activities" while ISO 9241-11 approaches usability from a goal-oriented perspective to achieve "effectiveness, efficiency, and satisfaction" (Jokela, et al., 2003, p.54). Both are replaced by ISO 9241-210:2010, part of a comprehensive standard that includes 28 sub-standards relating to every area of human interaction (ISO 9241-210:2010, 2010). However, all of the standards are generic, non-specific, and oriented toward a process for involving users in the development of interfaces. This approach, while useful, ignores the characteristics of usability and, as a result, is too abstract to guarantee any usability outcomes.

User-centered design methods, based on the ISO standards developed to deal with usability issues and ensure that user needs are included in interface design (Mao, et al., 2005; Thayer and Dugan, 2009). User-centered design has grown in practice but its practice is has no standard method for its conduct (Alonso-Rios, et al., 2010; Mai, et al., 2005; Thayer and Dugan, 2009). Even with all of the standards and methods, user-centered design has not found its way into mainstream industry practice and is used by under 40% of projects (Mai, et al., 2005; Thayer and Dugan, 2009).

Finally, much usability research is nonspecific, fragmented, not linked to user experience and not universally applicable. Usability has no agreed on definition and is studied with many interpretations (Alonso-Rios, et al., 2010). In addition, systems analysis and design texts generally cover interface design in chapters that provide information at the level of the ISO standards (cf. Valacich, et al., 2009). Few programmers learn anything beyond rudimentary rules of thumb for interface design and, as a result, user satisfaction with custom-developed software because of poor interface design tends to be very low (Norman, 2002).

To summarize, this section has evaluated the state of application development from the perspective of design research. Practice has narrowed over the years to focus on only the aspects of applications that are articulated in SDLCs and methodologies. As a result, key aspects of applications are missing or insufficient for their purpose. These aspects include usability, quality, operatability, and attention to all user communities. Each area discussed in this section provides many opportunities for future research and improved integration in pedagogy and practice.

LIMITATIONS AND FUTURE RESEARCH

This paper provides a necessarily abbreviated discussion of the history, state, and issues with SDLC and software analysis and design methodologies to determine future needs to improve quality and usefulness throughout the organization.

Future research was identified and discussed in the following areas: A need to define the relative importance of key drivers of successful applications, specific techniques and processes for developing usable interfaces, best practices in servitizing applications development, SERVQUAL modifications to include IT services evaluation and to tease out the nuances between system and service in web sites, application use and satisfaction relationship elaboration, common methodological checklists of items for application development consideration, methods to move new techniques into industry practice, checklists for managerial roles in applications development, usability and user experience, testing and system characteristics such as ease of use, the role of process in application development, the extent to which process standardization can contribute to a higher quality IT product, innovation driven by IT, innovation within IT, the extent to which improvisation can be institutionalized, uses of improvisation, measurement of application business value, and communication aspects of applications.

CONCLUSION

This paper evaluates application of methodologies to design systems artifacts and the challenges of the process. Through this analysis a series of changes to current practice and needs for future research and practical adaptation are identified. When these changes, additions, and future needs are examined, they do not differ substantively from recommendations of many research projects in the related areas. As a profession, we seem to forget our roots by omitting traditional activities that have led to past successes. Some of these activities include interface usability design, testing, product quality, and risk management. If collective forgetting continues, we are forever doomed to repeat past failings in a never-ending redevelopment of basic tenets. However, if we return to our roots and begin to identify and hone enduring practices, we improve the probability of future success in application design and development processes and as a result we also improve the potential for organizational contribution and relevance. More complex life cycles or methodologies do not necessarily result. Rather, checklists of issues to be considered and factored into application development, as needed, are required.

A move toward development of usable applications embedded within organizational services requires some changes. A services orientation requires understanding that no application is an end of itself. Rather the application is embedded in an organizational setting, is used by humans in the course of their work, and should add value to that work. The 'application user' includes all users, not just those in the non-IT community. The value adding aspects of applications include their ability to decrease cycle times, increase quality of services supported, and improve the work life of the application user. Remedying problems of application development and attending more to needs for usable services and should reduce costs of in-house development, increase user satisfaction, and provide clearer value contribution to business success.

REFERENCES

Abrahamson, P., Salo, O., Ronkainen, J. and Wartsa, J. (2002). Agile Software Development Methods: Review and analysis. Oulu, Finland: VTT Electronics.

Agar, M., Ali, F., Bhasin, S., Kota, N., Landa, R., Linares, G.L., Conger, S. and Landry, B.J.L. (2007). *IT Assessment: Final Report*. University of Dallas Capstone Consulting Report to Pattonair U.S.

Alonso-Ríos, D., Vázquez-García, A., Mosqueira-Rey, E., & Moret-Bonillo, V. (2010). Usability: A Critical Analysis and a Taxonomy. *International Journal of Human - Computer Interaction*, 26(1), 53-63.

Alter, S. (2010). Viewing Systems as Services: A Fresh Approach in the IS Field. *Communications of the Association for Information Systems*: 26, Article 11. Downloaded June 23, 2010 from http://aisel.aisnet.org/cais/vol26/iss1/11

Arnott, D. and Pervan, G. (2008). Eight key issues for the decision support system discipline. *Decision Support Systems*, 44(3), pp. 657-672.

Avison, D.E. and G. Fitzgerald (1988, October) Information systems development: Current themes and future directions *Information and Software Technology,* 30(8), pp. 458-466.

Avison, D.E. and Fitzgerald, G., (2003). Where now for Development Methodologies? *Communications of the ACM*, 46(1), pp. 79-82.

Avison, D.E. and Fitzgerald, G., (2006). Developing and Implementing Systems. W. Currie, and R. Galliers, Editors. *Rethinking MIS*. Oxford, England: Oxford University Press, pp. 250-278.

Avison D.E. and Gregor, S. (2009). An exploration of the real or imagined consequences of information systems research for practice. In *Proceedings of the 17th European Conference on Information Systems* S. Newell, E. Whitley, N. Pouloudi, J. Wareham, L. Mathiassen, eds., pp. 1780-1792, Verona, Italy.

Baschab, J. and Piott, J. (2007) *The Executive's Guide to Information Technology, 2nd Edition*, NY: Wiley.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland J., and Thomas, D. (2001). The Agile Manifesto. Downloaded on May 1, 2010 from http://agilemanifesto.org/

Bertolino, A. (2007, May 23 - 25). Software Testing Research: Achievements, Challenges, Dreams. In *2007 Future of Software Engineering -- International Conference on Software Engineering*. IEEE Computer Society, Washington, DC, 85-103.

Boehm, B. (1981). *Software Engineering Economics,* NJ: Prentice Hall.

Boehm, B., (1988, May) A Spiral Model for Software Development and Enhancement. *Computer,* 21(5), pp. 61-72.

Boehm, B. (2006, May, 20-28). A View of 20th and 21st Century Software Engineering. ACM Proceedings of the International Conferences on Software Engineering '06, Shanghai, China, pp. 12-30.

Booch, G., Rumbaugh, J. and Jacobson, L. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley Longman Inc.

Brewin, B. (2010, March 4) Glitch prompts VA to Shut Down e-health data exchange with Defense. NextGov.com. Downloaded on April 27, 2010. URL= http://www.nextgov.com/site_services/print_article.php?StoryID=ng_20100304_9977

Brinkkemper, S. (1996). Method Engineering: Engineering of information systems development methods and tools. *Information and Software Technology*, 38, pp. 275-280.

Brooks, F. P. (1987). No silver bullet: Essence and accidents of software engineering. *IEEE Computer*, 20(4), pp. 10-19.

Brooks, Jr., F.P. (1975) *The Mythical Man-Month*. Reading, MA: Addison-Wesley.

Brynjolfsson, E. and Hitt, L. (2003) Computing Productivity: Firm-level Evidence, *Review of Economics and Statistics*, Vol. 84, No. 4.

Cater-Steel, A. and Toleman, M. (2007). Education for IT service management standards," *International Journal of IT Standards and Standardization Research*, 5, 27-41.

Checkland, P. (1981) *Systems Thinking, Systems Practice*. London: Wiley.

Chen, X. and Sorenson, P. (2007, November). Toward TQM in IT Services. *Proceedings of the ASE Workshop on Automating Service Quality*. Atlanta, Georgia, USA. pp. 42-48.

Ciborra, C. (1998). Notes on Improvization and time in Organizations. *Journal of Accounting, Management and Information Technology*, 9, pp. 77-94.

Ciborra, C. (1996, Mar/Apr). The Platform Organization: Recombining Strategies, Structures, and Surprises. *Organization Science*. 7(2), pp. 103-118.

Conger, S. (1994). *The New Software Engineering*, NY: Thomson Publishing.

Conger, S. (2009). Information Technology Service Management and Opportunities for Information Systems Curricula. *International Journal of Information Systems in the Service Sector (IJISSS)* Special Issue on The Engineering, Management, and Philosophy of Service-Oriented Information Systems, 1(2), pp 58-68.

Conger, S. (2010a). IT Infrastructure Library ITIL v3. *The Handbook of Technology Management Volume I*, Hossein Bigdoli, Editor. NY: John Wiley & Sons. pp. 244-256.

Conger, S. (2010b). *Process Mapping and Management*, NY: Business Expert Press.

Conger, S. (2010c). Finding the Sweet Spot in ITIL Implementation. Accepted for presentation at Pink Elephant IT Management Conference, Las Vegas, NV, February 20-23, 2011.

Conger, S. (2011). Finding the sweet spot in IT service management implementation. Accepted for presentation at Pink Elephant Annual Conference, Las Vegas, February, 2011.

Conger, S. and Landry, B.L.J. (2009). Problem Analysis: When established techniques don't work. *Proceedings of 2nd Annual Conf-IRM Conference*, Al-Ain, UAE, May 19-24.

Conger, S. and Picus, B. (2009). Sustainable Certification using ISO/IEC 20000," American Society for Quality's *Quality Management Forum*, Spring, pp. 14-19.

Conger, S. and Pollard, C. (2009). Servitizing the Introductory MIS Course. *Proceedings of the AIS Special Interest Group on Services (SIG SVC) Workshop*, Phoenix, AZ. December 14, 2009.

Conger, S. and Schultze, U. (2008). IT Governance and Control: Making sense of Standards, Guidelines, and Frameworks," Chicago, IL: The Society for Information Management International, Advanced Practices Council.

Conger, S., Venkataraman, R., Hernandez, A., and Probst, J. (2009). Market Potential for ITSM Students: A Survey. *Information Systems Management*, Special Issue on IT Service Management, Aileen Cater-Steel (Ed.). 26(2), pp.176-181.

Culnan, M.J. (2007). The dimensions of perceived accessibility to information: Implications for the delivery of information systems and services. Journal of the American Society for Information Science. 36(5), pp. 302-308.

Cuyler, T. and Schatzberg, L. (2003). Customer service at SWU's Occupational Health Clinic. Journal of Information Systems Education, 14(3), 241-246

De Marco, T. and Plauger, P. J. (1979). *Structured Analysis and System Specification*, Upper Saddle River, New Jersey: Prentice Hall.

Deloitte. (2002). Achieving, Measuring, and Communicating IT Value. Deloitte & Touche Consulting, referenced through *CIO Magazine*. URL= http://www.cio.com/sponsors/041503dt/complete.pdf

DeLone, W. H., and McLean, E.R. (1992, March). Information Systems Success: The Quest for the Dependent Variable, *Information Systems Research* 3(1), pp. 60-95.

DeLone, W. H., and McLean, E.R. (2003, Spring). The DeLone and McLean Model of Information Systems Success: A Ten-Year Update, *Journal of Management Information Systems* 19(4), pp. 9-30.

DeMarco, T. (1978) *Structured analysis and system specification*. Prentice Hall.

Dodd, J. L, & Carr, H. H. (1994). Systems development led by end-users. *Journal of Systems Management*, 45(8), 34.

Doran, T. (2000, October 25). Compliance Frameworks: Software Engineering Standards, Washington, D.C.: NDIA Systems Engineering & Supportability Conference. URL=http://sce.uhcl.edu/helm/SENG_DOCS/compliance_framework.pdf

Dorgan, S. J. and Dowdy, J. J. (2004, November). When IT lifts productivity. *The McKinsey Quarterly*, 4, pp. 13-5.

Doyle, K. G., Wood, J.R.G. and Wood-Harper, A.T. (1993). Soft systems and systems engineering: on the use of conceptual models in information system development. *Information Systems Journal* 3(3), pp.187-198

Dubie, D. (2002, October 1). Procter and Gamble touts IT services model, saves $500 million. *ComputerWorld Management*.

Eberlein, A. and Sampaio do Prodo Leite, J.C. (2002, September). Agile requirements definition: A view from requirements engineering. In *Proceedings of the International Workshop on Time-Constrained Requirements Engineering (TCRE'02)*.

Ewusi-Mensah, K. (2003). *Software development failures: anatomy of abandoned projects*, Boston, MA: MIT Press.

Fitzgerald, B. and Fitzgerald, G. (1999). Categories and Contexts of Information Systems Development: Making Sense of the Mess. In *Proceedings of the Seventh European Conference on Information Systems* (Pries-Heje J, Ciborra CU, Kautz K, Valor J, Christiaanse E, Avison D, Heje C eds.), Copenhagen Business School, Copenhagen, June 23-25, pp. 194-211.

Gallagher, M., Link, A., and Petrusa, J. (2005). *Measuring Service Sector Research and Development*. Washington, D.C.: National Institute for Science and Technology, March. Downloaded May 1, 2010 from http://www.nist.gov/director/prog-ofc/report05-1.pdf

Galup, S., Dattero, R., Quan, J. and Conger,S. (2009). An Overview of IT Service Management," *Communications of the ACM*, 52(5), pp 124-128.

Goldenstern, C. (2010). Closing the 21st Century Service Capability Gap. Kepner-Tregoe, p. 4. Downloaded on April 28, 2010. URL=http://www.tsia.com/secure/whitepapers/Closing_the_21st_Century_Service_Capability_Gap.pdf

Gorla,N. and Lin, S.-C. (2010). Determinants of software quality: A survey of information systems project managers. *Information and Software Technology*. Amsterdam, 52(6), p. 602.

Guimaraes, T., Armstrong, C.P., and Jones, B.M. (2009). A New Approach to Measuring Information Systems Quality. *The Quality Management Journal*. Milwaukee, 16(1), pp. 42-55.

Gupta, D. (2008, September 14-16) Servitizing Applications. Presentation at the 3rd Academic Forum at the itSMF-USA Conference, San Francisco, CA.

Hevner, A.R., March, S.T., Park, J. and Ram, S. (2004). Design Science in Information Systems Research, *Management Information Systems Quarterly*, 28(1), pp. 75-105.

Høegh, R. T. (2006, November 20 – 24). Usability problems: do software developers already know? In *Proceedings of the 18th Australia Conference on Computer-Human interaction: Design: Activities, Artifacts and Environments* (OZCHI '06), J. Kjeldskov and J. Paay, Eds. Sydney, Australia. 206, pp. 425-428.

Hochstein, A., Tamm, G., and Brenner, W. (2005, May 26-28) Service-Oriented IT Management: Benefit, Cost and Success Factors. In *Proceedings of the Thirteenth European Conference on Information Systems* (Bartmann D, Rajola F, Kallinikos J, Avison D, Winter R, Ein-Dor P, Becker J, Bodendorf F, Weinhardt C eds.), pp. 911-921, Regensburg, Germany.

InfoSecurity.com. (2009). Companies Invest in IT but Do Not Measure It. Infosecurity.com. Downloaded on May 1, 2010 from http://www.infosecurity-us.com/view/3046/companies-invest-in-it-but-do-not-measure-it-value/

ISO/IEC (1995) *ISO/IEC 12207:1995 Standard for Information Systems Life Cycle Processes.* International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC). Washington, D.C.

ISO/IEC (2002) *ISO/IEC 15288:2002. Standard for Systems Engineering.* International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC). Washington, D.C.

ISO/IEC (2005) *ISO/IEC 20000-1: 2005 Standard for Information Technology – Service Management, Part 1: Specification.* International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC). Washington, D.C.

ISO/IEC (2008) *ISO/IEC 12207:2008 Systems and software engineering -- Software life cycle processes*. International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC). Washington, D.C.

ISO/IEC (2010) ISO/CD 9241-210:2010 (2010) Ergonomics of human- Part 210: Human-centred interactive systems International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC). Washington, D.C.

Jackson M. A. (1975). *Principle of Program Design*. NY: Academic. Press.

Jacobson, I., Booch, G. and Rumbaugh, J. (1999). *Unified Software Development Process*, Reading, MA: Addison-Wesley.

Jokela, T., Iivari, N., Matero, J., and Karukka, M. (2003.) The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11. In *Proceedings of the Latin American Conference on Human-Computer interaction* (Rio de Janeiro, Brazil, August 17 - 20, 2003). CLIHC '03, vol. 46. ACM, New York, NY, pp. 53-60. Downloaded on May 1, 2010 from http://portal.acm.org/citation.cfm?id=944519.944525

Jones, J. C. (1992). *Design Methods*, Second Edition. NY: John Wiley & Sons.

Kaikkonen, A., Kallio, T., Kekäläinen, A., Kankainen, A., and Cankar, A. (2005). Usability Testing of Mobile Applications: A Comparison between Laboratory and Field Testing. *Journal of Usability studies*, 1(1), pp. 4-16.

http://en.wikipedia.org/wiki/Software_testing - cite_ref-34Kaner, C. (2001). NSF grant proposal to "lay a foundation for significant improvements in the quality of academic and commercial courses in software testing. Testingeducation.org. Downloaded on April 27, 2010 URL=http://www.testingeducation.org/general/nsf_grant.pdf.

Kaner, C. (2003). Measuring the Effectiveness of Software Testers. Star East. Downloaded on April 27, 2010 URL=http://www.testingeducation.org/a/mest.pdf.

Keil, M. and Carmel, E. (1995). Customer-developer links in software development. *Communications of the ACM* 38, 5 (May. 1995), 33-44.

Keil, M. (1995). "Pulling the Plug: Software Project Management and the Problem of Project Escalation," *Management Information Systemns Quarterly*, 19(4).

King, J. (2009, December 7). IT's top tier: Strong and steady leadership. *Computerworld* Downloaded on June 23, 2010 from https://www.computerworld.com/s/article/print/344381/IT_s_top_tier_Strong_and_steady_leadership?taxonomyName=Management&taxonomyId=14.

Levinson, M. (2003, October 1). How to Conduct Post-Implementation Audits. *CIO Magazine*. Downloaded March 10, 2010 from http://www.cio.com/article/29817/How_to_Conduct_Post_Implementation_Audits

Lewis, J.R. (1995). “IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use,” International Journal of Human-Computer Interaction, 7(1):57–78

Lynch, C. G. (2006, March 6). Most Companies Adopting ITIL® Practices," *CIO Magazine*.

Lyytinen, K. and Robey, D. (1999). Learning Failure in Information Systems Development. *Information Systems Journal 9, pp. 85-101.*

Mao, J-Y., Vredenburg, K., Smith, P.W., and Carey, T. (2005, March). The state of user-centered design practice. *Communications of the ACM*. 48(3), pp. 105-109.

Markus, M.L., and Keil, M. (1994). If We Built It, They Will Come: Designing Information Systems that People Want to Use. *Sloan Management Review.* 35(4) , pp. 11-25.

Marrone, M. and L. M. Kolbe. (2010a, February 23-25). Providing more than just operational benefits: An empirical research. in Proceedings of Multikonferenz Wirtschaftsinformatik 2010, M. Schumann, Ed. Göttingen, Germany, pp. 61-63.

Marrone, M. and L. M. Kolbe. (2010b, June 6-9). ITIL and the creation of benefits: An empirical study on benefits, challenges and processes. Proceedings of the European Conference on Information Systems (ECIS), Pretoria, S. Africa.

Martin, J. (1991). *Rapid Applications Development*. NY: Macmillan.

Mathiassen, L. And Nielsen, P. A. (1989). Soft Systems and Hard Contradictions - Approaching the Reality of Information Systems in Organizations.: *Journal of Applied Systems Analysis*, Vol. 16.

Melone, N.P. (1990). A Theoretical Assessment of the User Satisfaction Construct in Information Systems Research. *Management Science*. 36(1), pp. 76-86.

Melville, N.P. (2010). Information Systems Innovation for Environmental Sustainability, *MIS Quarterly*, 34(1), pp. 1-21.

Myers, G. J. (1979). *The Art of Software Testing*. John Wiley and Sons.

Nelson, R.R., Todd, P.A., and Wixom, B.H. (2005). Antecedents of information and system quality: an empirical examination within the context of data warehousing, *Journal of Management Information Systems.* 21 (4), pp. 199–235.

Nielsen, J. (2000). *Usability Engineering*. San Diego, CA: Kaufmann.

Nielsen, J. (2005). Ten Usability Heuristics. Downloaded on April 27, 2010 from http://www.useit.com/paper s/ heuristic/heuristic_list.html

Nielsen, J. (1994). Using discount usability engineering to penetrate the intimidation barrier. Retrieved June 8, 2010 http://www.useit.com/papers/guerrilla_hci.html

Norman, D. A. (1998): The Design of Everyday Things. NY: Basic Books.

Parasuraman, A. Zeithaml, V., and Berry, L.L. (1994). Reassessment of Expectations as a Comparison Standard in Measuring Service Quality: Implications for Further Research. *Journal of Marketing*, 58(1), pp. 111-125.

Parasuraman, A. Zeithaml, V., and Berry, L.L. (1988). SERVQUAL: A Multiple-item Scale for Measuring Consumer Perceptions of Service Quality. *Journal of Retailing*, 64(1), pp. 12-37.

Park, K. S. (1997). Human Error. In Gavriel Salvendy (Ed.), *The Handbook of Human Factors and Ergonomics*, (2nd Edition). John Wiley & Sons.

Petter, S., Delone, W. and Mclean, E. (2008). Measuring information systems success: models, dimensions, measures, and interrelationships. *European Journal of Information Systems*. Basingstoke: 17(3), pp. 236-263.

Pollard, C and Cater-Steel, A. (2009). Justifications, Strategies, and Critical Success Factors in Successful ITIL Implementations in U.S. and Australian Companies: An Exploratory Study. *Information Systems Management*, 26(2), pp. 164-172.

Potgieter, B.C., Botha, J.H., and Lew, C. (2005, July 10-13). Evidence that use of the ITIL framework is effective. *Proceedings of the 18th Annual Conference of the National Advisory Committee on Computing Qualifications*, Tauranga, NZ.

Prescott, M. and Conger, S. (1994). Information technology innovations: A Classification by IT locus of impact and research approach. *Database*. 26(2-3), pp. 20-42.

Reichheld, F.F. (2003 – December 1). The One Number You Need to Grow. *Harvard Business Review*. Downloaded on April 28, 2010 from http://harvardbusinessonline.hbsp.harvard.edu/b02/en/common/item_detail.jhtml?id=R0312C&referral=2340.

Rombach, D., Ciolkowski, M., Jeffery, R., Laitenberger, O., McGarry, F., and Shull, F. (2008, October). Impact of research on practice in the field of inspections, reviews and walkthroughs: learning from successful industrial uses. *SIGSOFT Software Engineering Notes* 33(6). pp. 26-35.

Sherman, D.K., Mann,T., and Updegraff, J.A. (2006). Approach/Avoidance Motivation, Message Framing, and Health Behavior: Understanding the Congruency Effect. *Motivation and Emotion* 30(2), pp. 165–169.

Shneiderman, B. (2004). *Designing the User interface: Strategies for Effective Human-Computer Interaction, 4th Edition*. Reading, MA: Addison-Wesley.

Shneiderman, B. (2000, May). Universal Usability. *Communication of the ACM* (43:5), pp 84-91.

Suchman, L. A. (1983, October). Office procedure as practical action: models of work and system design. *ACM Transactions on Information Systems* 1(4), pp. 320-328.

Sumner, M. (2000). Risk factors in Enterprise Wide Information Management Systems. *Proceedings of the AMC SIG CPR Conference*, Evanston, IL. 2000, pp. 180-188.

Thayer, A. and Dugan, T.E. (2009, July 19- 22). Achieving design enlightenment: Defining a new user experience measurement framework. *Proceedings of the 2009 IEEE International Professional Communication Conference*, Waikiki, HI. pp.1-10.

Valecich, J., George, J., and Hoffer, J. (2009). *Essentials of Systems Analysis and Design*, Third Edition, Upper Saddle River, NJ: Prentice-Hall.

Van Bon, J. (2007). *IT Service Management: An Introduction*, London: itSMF International.

Winniford, M.A., Conger, S., and Erickson-Harris, L. (2009). Confusion in the Ranks: IT Service Management Practice and Terminology. *Information Systems Management*, Special Issue on IT Service Management, Aileen Cater-Steel (Ed.), 26(2), pp. 153 – 163.

Yourdon E. and Constantine, L. L. (1975). *Structured Design*. NY: Yourdon Press.

Zhang, P., Galletta, D., Li, N., and Sun, H. (2007). Human-Computer Interaction, in Wayne Huang (ed.), *Management Information Systems*, Beijing, China: Tsinghua University Press.