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 the practice still suffers from poor-quality applications. This paper 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: IT Service Management, Methodology, Software Development Life Cycle, Usability, User Centered Design, 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 & Piott, 2007, p. 11). By one report in 1994, 53% of projects overran original schedules by an average of 222% (Baschob & 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% of all projects delivered planned benefits (Baschob & Piott, 2007). The
situation is such that the IT-business relationship is characterized as hostile in many situations (Agar, et al., 2007; Avison & Gregor, 2009).

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 & Schultze, 2008; Cuyler & 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.

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 & Fitzgerald, 2006), while others view them as skeletal guidelines within which methodologies operate (Conger, 1994). The latter view is taken by this research.
Towards Knowledge Evolution in Software Engineering: An Epistemological Approach
www.igi-global.com/article/towards-knowledge-evolution-software-engineering/38998?camid=4v1a