8-25-1995

A Socio-technical Approach to CASE and the Software Development Process

Ravi Patnayakuni
Southern Illinois University, ravip@siu.edu

Nainika Patnayakuni
Southern Illinois University

A Socio-technical Approach to CASE and the Software Development Process

Ravi Patnayakuni and Nainika Patnayakuni
Department of Management, Southern Illinois University
Carbondale, IL 62901
Phone: 618/ 453-3307 Fax: 618/453-7835 Email: ravip@siu.edu

Introduction

An increasing number of organizations are adopting computer aided software engineering (CASE) technology to support their system development process. Proponents of CASE predict that its use will improve development productivity, reduce backlog, and improve software quality (Forte and Norman, 1992). At the same time, there is evidence to indicate that CASE technology may not meet these expectations (Orlikowski, 1992). One of the reasons identified has been the substantial changes required to the development process for implementing CASE (Fichman and Kemerer, 1993). The paper adopts a socio-technical-system based framework for examining CASE in the context of software development process. Analysis of the socio-technical context of CASE adoption and the software development process could suggest implications of different change management strategies and extend our understanding of existing research results.

Socio-Technical System Approach to Systems Delivery

Research has approached technology from various perspectives. One approach has been to consider technology as "hardware"; that is, as equipment, machines, and instruments that humans use in productive activities, whether industrial or informational devices (Blau et. al., 1986; Woodward, 1958; Zuboff, 1988). The approach focuses on the ability of the technology to convert raw materials into finished goods in a efficient and effective manner. In contrast is the concept of "social technologies" which includes besides hardware, generic tasks, techniques, and knowledge utilized when people engage in any productive activity (Eveland, 1986; Perrow, 1967; Thompson, 1967).

Introduction of a new technology, particularly complex technologies such as CASE, invariably exerts a pressure on the organization to change, adjust, or adapt to the new technology. The socio-technical systems approach has become an increasingly popular approach for examining the changes brought about by technology. The success of an enterprise depends upon the compatibility between its social and technical subsystem.

In this paper a socio-technical system based framework (Shani et. al. 1992) is adapted to analyze social and technical attributes associated with usage of CASE tools at different levels of software development process maturity (Humphrey, 1988). The socio-technical systems perspective considers every organization to be made up of social subsystem of people using the technical subsystem comprising of tools techniques, and knowledge to produce products or services (Trist, 1982).

CASE tools differ in their functionality and extent of support offered for different activities in the software development process. Some tools support a particular phase, such as requirement analysis or code generation, while others provide an integrated software development environment. Front-end CASE tools support design activities such as diagramming, creating and maintaining data dictionaries/ repositories, designing reports, forms and screens. Back-end CASE tools are used to analyze program and database structures to generate code that can be used for unit and system testing. In addition, there are full life cycle products that support and integrate all the system development functions from the initial conceptualization to code generation and testing for application systems.

Software Development Process

Organizations approach their systems development function from a variety of methodological platforms, and often subscribe to no particular methodology or they subscribe to several methodologies. Organizations may be characterized in terms of the maturity of their software development process (Humphrey, 1988). Software Process Capability maturity model (CMM) (Humphrey, 1988) is one of the models developed to characterize the system development process of an organization. The CMM is five level model in which higher levels are indicative of greater process maturity.

In the model the initial level (level 1) represents crisis driven, ad-hoc development. The organization at this level of process maturity would primarily aim at estimating project effort such as size, resources required, schedules and measures such as lines of code and errors. The IS organization would be unable to function in a tightly integrated CASE environment and it would probably deploy stand alone CASE tools or at lower levels of functionality.

The next level of maturity (level 2) is when the development process, while still intuitive and dependent on individuals, exhibits regularity in repeating previously mastered tasks. The process is still largely undefined. Here the metrics needed by the organization would largely be similar to the previous level Level three is labeled as a defined process. The development process is specified and institutionalized, no longer dependent on individuals. Level four is characterized as a managed process. The development process is measured and controlled , in the sense that relationship between activities are understood quantitatively. The IS organization evaluates both key process activities and major product properties for assessing the software development process. Changes are made to software process specifications based on the results of analyzing this data. This level of maturity is ideally positioned to exploit the capabilities of the CASE tool to improve software quality and productivity. The approach is more akin to an engineering approach where CASE technology is used as a tool for manufacturing software. A more systemized approach would ideally be able to work on an restricted CASE tool.

Finally, the highest level of maturity provides not only for the management of a defined process using automatic data collection, but also for change and optimization of the process itself. At this level, the CASE tool would be used. Each of the levels is shown in

the table below, along with technical and social system attributes. Automated data collection would require that the process be well defined and standardized for providing a common platform for comparison. CASE would be viewed by the organization not only as a tool for the development process but also as an important agent of software quality improvement. The organization would be in a position to emulate the principles of a software factory and would be in a position to exploit an integrated CASE environment for software development.

Technical System

The deployment of functionality and need for an integrated CASE environment is likely to increase with increasing level of maturity. At this level of maturity the technical system of the organization is characterized by low complexity due to the use of stand alone tools. Complexity of the software development process increases at level 5 as organizations begin to use I-CASE tools and integrated development environments. Interfaces between tools from multiple vendors and automated data collection and usage are critical aspects of the complexity of software development process as organizations increase in process maturity.

Innovation at lower levels of software process maturity is characterized by automation of stand alone tasks such as diagraming and designing reports and screens. As the maturity of software development process increases, organizations begin to employ CASE tools for automatic data gathering, and metrics have a process improvement focus. With a total engineering approach, productivity and quality would be emphasized not only for the process but for the entire development life cycle. Development of reusable components is encouraged requiring adherence to integrity and consistency at all times. The technology would be integrated with metrics and a suitable software process model to generate management information suitable for optimizing the process. Thus, increasing levels of software process maturity are associated with increasing process innovation. This is supported in research which argues that implementing full capabilities of CASE is a prime case of business process redesign (Rai and Howard 1993). Use of re usable components and a software factory-based approach also results in increasing product innovation.

Work Design

We discuss work design in terms of task design, human resource practices, skill and role requirements, information flow, organization structure and control systems. As organizations move to higher levels of software development maturity, the task design changes from isolated development to working in teams in an integrated ICASE environment. Developers are increasingly required to interact with one another and with users and clients. Such interaction could consist of requirements definition, developments of components for reuse and working in autonomous teams for application development.

At lower levels of software development maturity developers need to have specialized skills in their specific job/role, and not all developers may need to have business skills as

these skills may themselves constitute a specialized area. However, at level 5 developers need to be multi-skilled, and need to have business knowledge and team working capabilities. Organizations may want to modify their reward systems to reinforce team-based functioning in the organization by relying on group-based and team-based rather than individual rewards.

Information flow across task units is manual in a stand-alone CASE context. Higher levels of process maturity are defined by automated information flow across task units. This automation in information flow requires developers and analysts to adhere to certain guidelines for systems development. An increase in systems development maturity is accompanied by increasing formalization of the software development process in the organization. Organizations at a higher level of process maturity have a total quality approach to systems development, and such an approach requires the participation and involvement of developers for its success. Organizations are likely to balance the demands of increasing formalization by making the control systems more self-regulated by work groups involved in systems development. Such an approach would be conducive to the maintenance of developers' motivation in what they perceive as a increasingly rigid systems development context.

(References available upon request)

Table 1. A Socio-technical System Based Comparative Examination of Five Levels of Software Process Maturity

Software Process Maturity Levels	Level 1	Level 2	level 3	Levell 4	Level 5
	Ad-hoc	Repeatable	Defined	Managed	Optimized Process
CASE Tools	Stand Alone--	--------	--------	--------	--Integrated
Key Socio-technical Elements					
Technical System					
Level of Complexity	Low	Low	Moderate	High	High
Innovation	Process automation	Limited Scope	Moderate Scope	Extended Innovation	High Process

		Process Redesign	Process Redesign	in Process, Limited Product Innovation	Innovation, Moderate Product Innovation
Social System: Work Design					
Task Design	Individual Stand Alone Tasks	Limited Local Task Integration	Moderate Task Integration	High Task Integration with Semi-Autonomous Work Groups	High Task Integration with Autonomous Work Groups
Skill Requirements	High Specialization	Medium Specialization	Multiple Skill Requirements	Increasing Emphasis on Business and Team Working Skills	Multiple Skills including Business and Team Working Skills
Reward Practices	Individual Based	Individual Based	Individual and Group Based	Group Based	Group Based
Information Flow	Manual Restricted	Manual, extended Scope	Partly Automatic	Partly Automatic, extended Scope	Automatic Information Transfer, Wide Scope
Organization Structure	Low Formalization	Low formalization	Moderate Formalization	Moderate Formalization	High Formalization
Control	Centralized Control	Decreasing Centralized Control	Moderate Group Self-Control	High Group Self-Control	High Group Self-Control