Chapter VI

Information Systems as Social Systems

N. F. du Plooy
University of Pretoria, South Africa

ABSTRACT

Since the very early days of computing, information systems were regarded as “man-machine” systems. This definition is of far greater importance to the teaching of information systems as well as to the profession of information systems developers than is often recognised. The majority of teachers, and textbooks, are still caught in the paradigm that information systems development is a technical “art” rather than a field that concerns sociological, rather than technical problems. This chapter argues that the “human” or sociological side of information systems is of such importance that it should be seen as the core of the discipline, and that information systems are best understood when viewed as social systems.

Examining the nondeterministic nature of information systems and applying Ashby’s Law of Requisite Variety are helpful to describe and explain these systems as social systems. The paper also refers to current thinking on systems (especially soft systems methodology) and its place in supporting information systems in a constantly changing environment.

Finally, the implications of viewing information systems as social systems are discussed with respect to teaching and research, the impact of information systems on organizations, and the shortcomings of current software engineering methodologies.
INTRODUCTION

I wish to make clear right at the onset of the chapter, that when we refer to information systems, we have in mind systems that are used in an organizational setting (typically a business or a State department), supporting transactional activities as well as managerial decision making.

In 1968, a conference was held in Europe to discuss the so-called “software crisis.” During that conference, much was said about budget overruns, systems being late, systems being produced but never used, a growing backlog of applications, systems being unmanageable, etc. The conference concluded that there is an “...urgent need for techniques and methods which allowed the complexity inherent in large software systems to be controlled” (Sommerville, 1989).

This goal seems to be somewhat elusive, since more than two decades later, Ng and Yeh (1990) stated that “…the average application backlog in large development shops has increased from 19 months to 27 months in just three years.” However, they confidently continue that “...a solution to this... is to automate the software development and maintenance process...”

But the quest for techniques and methods to reduce the problems of software development seems to have been less than successful, because nearly 35 years after the initial conference, the “software crisis” is clearly still with us. Kautz and Larsen (2000) state that: “…unfinished projects, project overruns, systems failures and missing functionality is still the norm (in software development)....”

A starting point in understanding the mainly technological response to software development problems and failures could be in, inter alia, an older paradigm of computer science, namely, that the basic question underlying all of computing is: “What can be (efficiently) automated” (Denning et al, 1989). The implication of this statement is that in order to increase the efficiency of an organization, we should automate all or part of it, and similarly, in order to increase the productivity of the systems developers, we must also automate the automation process! Such arguments are quite commonplace among the computing fraternity, being one of the driving forces behind the search for a shared information systems development methodology, a methodology that will be repeatable and that will enable us to develop systems that are efficient and effective and that are completed and used. It could also very well be that this same sentiment underlies the technological hope that a kind of “generic” information system (e.g., ERP systems) that fits all organizations could be found.

But possibly, there exists another view of this situation and that is that the actual nature of information systems is not fully understood by its developers. The fact that information systems are sociotechnical systems is often ignored. This will be the basis of the argument for information systems as social systems in this chapter. I intend to show that, if we view information systems as social systems, the process
Ethical Negotiations: A Trust-Building Approach to International Negotiations
www.igi-global.com/article/ethical-negotiations/227743?camid=4v1a