Chapter VII
On the Alignment of Organizational and Software Structure

Cleidson R. B. de Souza
Universidade Federal do Pará, Brazil

David F. Redmiles
University of California, Irvine, USA

ABSTRACT

This chapter reviews the socio-technical relationship between organizational and software structure. It describes the early theoretical work about this relationship, the empirical studies that identified this relationship in practice and, more importantly, identifies two main approaches for exploring this relationship. The first one is based on the construction of tools to facilitate software development, while the second is a more theoretical one aimed at investigating the consequences of this relationship in the work of software developers. Furthermore, the authors hope the theoretical background presented in this chapter will not only motivate other researchers to study software development as a socio-technical endeavor, but also assist practitioners in the understanding of the aspects necessary to make software development succeed.

INTRODUCTION

Software development is a typical socio-technical endeavor. Any non-trivial software development effort requires both technical skills and the ability to efficiently coordinate the work of hundreds of people (Brooks, 1974). Researchers and practitioners of software engineering have recognized this relationship for more than 30 years. Conway (1968), for instance, postulated that the structure of a software system would reflect the communication needs of the people performing the work, a relationship that became known as Conway's Law. Later, Parnas (1972) suggested that by reducing technical dependencies between software modules, it was possible to reduce the communication needs

of software developers, thus creating a managerial advantage. As postulated by Conway and Parnas, the socio-technical relationship occurs between the organizational structure and the software structure. This has been validated by several different empirical studies. For instance, in a seminal study, Curtis et al. (1988) recognized that "occasionally, the partitioning [of software components] was based not only on the logical connectivity among components, but also on the social connectivity among the staff." There are several qualitative studies with similar results (de Souza & Redmiles, 2008; de Souza, Redmiles, Cheng, Millen, & Patterson, 2004; Rebecca E. Grinter, 1998; Staudenmayer, 1997), as well as quantitative studies (Cataldo, 2007; Cataldo, Wagstrom, Herbsleb, & Carley, 2006; Sosa, Eppinger, Pich, McKendrick, & Stout, 2002).

This socio-technical relationship between organizational and software structure is relevant to both researchers and practitioners because it impacts the coordination of software development efforts. In other words, this relationship can also be understood as a relationship between coordination of software development efforts and software architecture[1]. Despite this long-term interest, it had not been sufficiently explored to understand or facilitate software development activities until recently (Cataldo, 2007; Cataldo et al., 2006; de Souza, 2005; Trainer, Quirk, de Souza, & Redmiles, 2005; Valleto et al., 2007). This chapter reviews the research literature about this relationship presenting the theoretical arguments, empirical studies, tools and approaches that build on this relationship.

The rest of this chapter is organized as follows. We begin by presenting a literature review focusing on both the theoretical arguments and the empirical studies on the socio-technical relationship between coordination of software development activities and software architecture. Two approaches have been adopted by researchers to explore this relationship. The first one is based on the construction of software tools to provide useful information for software developers and is described in Section 3. And the second one is a more theoretical approach aimed at investigating the consequences of this relationship in the work of software developers. This approach, called socio-technical congruence, is described in Section 4. We conclude with final remarks in Section 5.

LITERATURE REVIEW

The relationship between the coordination of software development efforts and software architecture has been studied in two different research areas: software engineering and computer-supported cooperative work (CSCW). Software engineering researchers are concerned primarily with dependencies in the software architecture and their impact on both the quality of the software being developed and the process of developing it (Sommerville, 2000). CSCW researchers' main concerns are with the coordination of collaborative work and how computational tools can support this task (Schmidt & Bannon, 1992). In this section we review both the software engineering and the CSCW literature to properly understand the relationship of interest. We will start reviewing approaches for handling software architecture (i.e., components and their dependencies) between software development artifacts.

Software Dependencies

The first approach aimed to handle software dependencies was Parnas' information hiding (Parnas, 1972). When Parnas proposed this principle, he also suggested that it would bring a managerial advantage: reducing dependencies between software modules would also reduce developers' dependencies on one another, therefore reducing communication needs and facilitating the coordination.

Software engineers have created additional techniques, tools, and principles to deal with dependencies. One of the most influential approaches adopted by software engineers is based on the notion of cohesion and coupling. While cohesion measures the degree of dependencies that occur within a module, the term coupling is used as a