Int. J. Human-Computer Studies 111 (2018) 1–11

Contents lists available at ScienceDirect

International Journal of Human-Computer Studies

journal homepage: www.elsevier.com/locate/ijhcs

Engendering cohesive software development teams: Should we focus on interdependence or autonomy?

Adarsh Kumar Satindarlal Kakar

Alabama State University, 915 S Jackson St, Montgomery, AL 36104 United States

ARTICLE INFO

Keywords:
Team cohesiveness
Autonomy
Task interdependence
Outcome interdependence

ABSTRACT

Empirical evidence suggests that both autonomy and interdependence are important considerations in team design. But how do interdependence and autonomy affect team cohesiveness, an important antecedent of team performance? The results of this multi-year study with software development projects show that task interdependence and task autonomy have both synergistic and antagonistic impacts on team cohesiveness. At high levels of outcome (goal) interdependence, task autonomy and task interdependence have a synergistic impact on team cohesion, while at low levels of outcome interdependence, task autonomy and task interdependence have an antagonistic impact on team cohesion. Further, taylorist teams showed lower cohesiveness compared to agile teams.

1. Introduction

Cohesion reflects the degree of attraction among group members (Shaw, 1981). A study of cohesiveness is considered essential for understanding group dynamics in teams (Zander, 1979). Cohesiveness is critical for social integration and sustenance of groups (O'Reilly et al., 1989). Studies have shown that cohesive groups coordinate better and display more altruistic behaviors (Berkowitz, 1954; Hogg, 1992; Shaw, 1981). Team members of cohesive groups willingly assist others, participate in group activities and consider group goals as their own (Henry, Arrow, and Carini, 1999; Moreland & Levine, 1984).

However, engendering cohesive teams can be tricky. Cognitive Evaluation Theory (CET) (Deci and Ryan, 1985) suggests that both autonomy and relatedness are fundamental human needs that should be met to enhance intrinsic motivation of team members. Teams are therefore designed to provide high levels of task interdependence as well as autonomy to its team members (Cannon-Bowers, Oser, and Flanagan, 1992; Sundstrom et al., 1990). However, this creates a dilemma as high interdependence may constrain autonomy and high autonomy may undermine interdependence (Janz, Collquitt, and Noe, 1997). Further, while high interdependence is likely to enhance group cohesion, high autonomy might hamper it.

Autonomy is defining as little as possible how tasks should be performed, practicing the principle of self-management (Herbst, 1974; Morgan, 1986), and allowing as much independence in terms of work pace and work methods as possible for performing tasks (Hackman and Oldham, 1976). However, for group tasks requiring high interdependence, such as software development, this can lead to frequent adjustment of tasks, compromises, and conflicts (Niepce and Molleman, 1998). Keeping this context in view, this study investigates the interplay between task interdependence and autonomy in engendering team cohesiveness. The aim is to get answers to the following questions: What is the trade-off in deciding between task interdependence and autonomy for enhancing team cohesion? When should one focus on autonomy versus interdependence?

These questions are relevant to all teams but particularly those engaged in knowledge work such as software development. Depending on the philosophy they subscribe to, the work in software development is designed to provide different levels of task autonomy and interdependence providing the variation necessary to comprehensively investigate the research questions. Plan-driven software development approaches prefer a hierarchical control and command team structure. Managers not only assign tasks to the team members but also specify how (process) they should be performed and by when (schedule) they should be completed (Melnik and Maurer, 2006). By contrast, agile software development approaches deploy autonomous self-organizing teams (Moe, Dingsøyr, and Dybå, 2009). Agile teams set and accomplish their own goals through participation among team members, have autonomy in the way they perform tasks, and repeatedly reorganize by allocating work among themselves based on changing circumstances (Nerur, Mahapatra, and Mangalraj, 2005).

It would therefore be interesting to examine autonomy and interdependence in plan-driven versus agile software development teams and how they affect team cohesion. To accomplish the aforementioned

E-mail address: akakar@alasu.edu

https://doi.org/10.1016/j.ijhcs.2017.11.001
Received 13 July 2016; Received in revised form 17 October 2017; Accepted 7 November 2017
Available online 11 November 2017
1071-5819/

goals, this study first identifies the relevant constructs by gleaning concepts from a multi-disciplinary review of literature and then develops a theoretical model of relationship between the constructs. The model is then tested with team members of industrial software development projects. The findings are analyzed and their implications for the practice of software development in particular and work groups in general are discussed.

2. Literature review

2.1. Autonomy and interdependence in the era of scientific management

Work design research began with the economic perspectives on the division of labor and task specialization (Babbage 1835; Smith 1776). Adam Smith (1776) suggested division of labor by breaking down complex jobs into simpler jobs as a way of enhancing performance. Expanding on these ideas Babbage (1835) pointed out the added advantages of job simplification such as requirement of less skilled and hence cheaper labor. Specialization and division of labor creates interdependencies within work groups or departments (Saavedra, Earley, and Van Dyne, 1993; Thompson, 1967; van de Ven, Delbecq, and Koenig, 1976).

The concepts of Charles Babbage and Adam Smith influenced the methods of software development during the early stages of its evolution. Methods such as the waterfall method (Royce, 1970) and its variants encouraged division of labor leading to specialized roles of business analysts, system architects, programmers, and testers (Melnik and Maurer, 2006). These plan-driven methods were also influenced by the concepts of Taylor (1911, 1947) who introduced Scientific Management with the aim of controlling every work activity, from the simplest to the most complicated. He applied to workers the ideas Whitney (see Mirsky and Nevins, 1952) earlier used for making interchangeable parts.

Taylor analyzed tasks into their minutest details and arrived at a standardized process; the one best way to do the job (Kanigel, 1997), just as Eli Whitney analyzed a musket into its smallest parts and made a machine to manufacture each part (Mirsky and Nevins, 1952). Together the ideas of Whitney, Taylor and Ford (of moving assembly line) ushered in the era of mass production. As applied to software development, these concepts led to the development of factory like concepts. Bemer of General Electric (Bemer, 1969) was among its earliest proponents. He suggested that General Electric adopt standardized tools to reduce variability in programmer productivity and keep a database of historical records for management control. Mellroy of AT&T (Mellroy, 1968) emphasized systematic reusability of code for enhancing productivity.

By the late 1960s, the term ‘software factory’ was in popular use and became associated with computer-aided tools, management-control systems, modularization, and reusability (Cusumano, 1989). Taylorist approaches such as the waterfall model (Royce, 1970) and its variants promoted upfront requirements gathering, systems design, and linear sequential development phases. These concepts were implemented through detailed planning, defined processes, coding standards, inspections and reviews, productivity metrics, and statistical quality control. Efficiency of software development processes were measured through the use of control charts. Process models such as Capability Maturity Model (CMM) gained popularity for defining and improving software development processes (Huh, 2001). Overall, these developments had an adverse impact on autonomy of team members.

2.2. Autonomy and interdependence and the human relations movement

In the domain of manufacturing, while mass production resulted in an improvement in the standard of living of society, it had deleterious psychological consequences for the workers. Repetitive jobs were found to be boring, tiring, dissatisfying and potentially damaging to mental health (Fraser, 1947; Walker and Guest, 1952). These costs of division of labor and task specialization diverted the focus of researchers to human issues at work. Studies were conducted to investigate whether employee satisfaction and motivation could be enhanced by improving working conditions (Mayo, 1933, 1945; Roethlisberger & Dickson, 1939). Some researchers proposed that enriched job characteristics such as enlarged rather than narrow tasks improve employee motivation and satisfaction (Herzberg, 1966; Herzberg, Mausner, and Snyderman, 1967; Turner and Lawrence, 1965). Hackman and Oldham (1976) suggested that when employees have freedom to schedule their work and decide on procedures it increases the motivating potential of work.

This transition in focus from process to people, and from division of labor and rigid task interdependencies to task autonomy and integration, was also seen in the evolution of software development methods with the introduction of the Agile manifesto in 2001. Agile development proponents questioned the assumption that change and uncertainty can be controlled through a high degree of advanced planning and rigid processes (Nerur, Mahapatra, and Mangalraj, 2005). Software developers realized that while the Tayloristic plan-driven methods do work well in stable conditions, under uncertain conditions, managers planning, assigning and controlling tasks of software developers may not work. Group members should be able to deal with disruptive events as and when they arise. Agile methods therefore emphasize team and employee autonomy in organizing and performing work.

However, despite these trends of increased employee autonomy and integration of tasks into meaningful work, the importance of interdependence did not decline. With tasks becoming more socially embedded than at any other time in the past, work design researchers recognize that work is inextricably intertwined with interactions among team members and interpersonal relationships (Grant and Parker, 2009). Therefore work in agile teams is not defined by industrial engineers and assigned by supervisors but by self-organizing teams through mutual adjustment amongst team members. Therefore, in today’s context of uncertain business environments and rapidly evolving customer requirements, it is reasonable to assume that both employee autonomy and interdependence are important considerations in successful team design.

2.3. Impact of autonomy and interdependence on team cohesion

But how do employee autonomy and interdependence impact team cohesion? Cohesiveness is the degree to which team members like each other, identify themselves positively with the team and want to remain its members (Hackman and Morris, 1975; Shaw, 1981). Two meta-analyses (Evans and Dion, 1991; Mullen and Cooper, 1994) have reported a positive relationship between cohesiveness and performance. However, team cohesion is not a panacea for all ills. Cohesive teams may be more susceptible to group think and may not generate the most creative solution to problems due to increased conformity and conservatism in problem-solving approaches (Janis, 1972, 1982; McAvoy and Butler, 2009). High team cohesiveness can also lead to ineffectual decision making such as groupthink and the Alibene paradox (Janis, 1972, 1982; McAvoy and Butler, 2009). Abilene Paradox is a decision taken by a group which no individual decision maker would have taken (Harvey, 1974). In groupthink socio-psychological factors prevent dissension and the individual accepts the view of the group as correct (Janis, 1972, 1982; Manz and Sims, 1987).

Yet, cohesion in teams is critical for keeping the team members aligned with a common purpose and goals Ramesh, Cao, Mohan, and Xu, 2006). Team cohesion is a one of the six key facets of Team Work Quality (Hoegl and Gemuenden, 2001). Without a sense of togetherness and belonging no meaningful collaboration is possible in groups. Further team cohesion promotes sharing of tacit knowledge amongst team members. For example technicians are known to learn more about repairing copiers by “hanging around swapping stories than from company manuals” (Fortune, 1991; Madhavan and Grover, 1998).

Hardy, Eys, and Carron (1995) noted that team cohesion provides numerous psychosocial and work benefits outcomes of teams. Cohesive teams demonstrate increased collective efficacy (Paskevich, Brawley,