Universality of Egoless Behavior of Software Engineering Students

Pradeep Waychal, Western Michigan University, Kalamazoo, MI, USA
Luiz Fernando Capretz, Western University, London, Canada

ABSTRACT

Software organizations have relied on process and technology initiatives to compete in a highly globalized world. Unfortunately, that has led to little or no success. The authors propose that the organizations start working on people initiatives, such as inspiring egoless behavior among software developers. This paper proposes a multi-stage approach to encourage egoless behavior and discusses the universality of the egoless behavior by studying cohorts from three different countries, i.e., Japan, India, and Canada. The three stages in the approach are self-assessment, peer validation, and action plan development. The instrument to assess egoless behavior is based on Lamont Adams’ “Ten commandments of egoless programming” – seven of the commandments are general, whereas three are related to coding behavior. The authors have found that students display relatively poorer egoless behavior in coding related than general commandments. The authors found traces of universality in the egoless behavior among the three cohorts.

KEYWORDS

Comparison of Software Engineering Students from Different Cultures, Egoless Programming, Human Factors in Software Engineering, Software Engineering, Software Psychology

INTRODUCTION

Software engineering concentrates much less on people than process and technology dimensions (Broman, Sandahl, & Baker, 2012), (Dagenais, Ossher, Bellamy, Robillard, & De Vries, 2010). Glass, et al. (2002) have studied 369 papers in six leading journals and discovered that software engineering research is fundamentally about technical and computing issues and seldom about behavioral issues. Recently, Lenberg et al. (2015) noticed increased attention to the human aspects of software, but found the increase to be not sufficient. In industry, a discussion about the people dimension appears to be limited to training people for new processes and technologies (Zowghi & Namuliani, 2002). Since this process-technology centric approach has not accrued any perceptible gains in productivity (Brynjolfsson, 1993), we argue to explore the people dimension deeply and earnestly; even though it is new to software engineering researchers, and depends on many factors such as social and corporate ecosystems.

The paper attempts to study an important sliver of the people dimension, egoless programming, which was initially established in Weinberg’s book, ‘The Psychology of Computer Programming’ (Weinberg, 1971). Our study introduces a multi-stage approach to develop egoless programmers. We are using contemporary terms such as egoless engineering and development and general terms such as egoless behavior to mean the same thing: egoless programming. Our multi-stage approach consists of developing an instrument to assess “egoless behavior” by individuals, validating the self-
assessment with peer assessments, and formulating group and individual action plans. In this paper, we concentrate on the first stage of developing an assessment tool to gauge egoless behavior, and explore universality of the egoless behavior among software engineering students. Towards that, we have chosen three cohorts from three culturally different countries: India, Japan, and Canada. Essentially, the paper contributes to knowledge of the people dimension in software development by presenting and analyzing self-assessment of egoless behavior of students from three different countries.

The next section discusses the problem of productivity in software organizations. It is followed by the research design of our experiment. We then analyze the results and end with concluding remarks.

BACKGROUND

Many software engineering stalwarts have emphasized the criticality of the people dimension in software engineering. Dijkstra (1979) proclaimed that programming (software engineering) has to be considered as a human activity. Weinberg (1971) clearly stated that human personality is more important than human intelligence in software. Cockburn (1999) has emphasized importance of the people dimension by stating that the fundamental characteristics of “people” have a first-order effect on software development and must become a first-order research agenda item in software engineering. Potts (1993) has claimed that “all the real problems in software engineering are people problems.” Many studies have asserted criticality of teamwork in organizations (Bendifallah & Scacchi, 1989; Boehm, 1981; Mahnic, 2012; Scacchi, 1995) Therefore, the people dimension appears to be of critical importance.

It is important to note that software engineers function in groups and a greater understanding of groups from a human science perspective may help in improving group and organizational performance. This would require delving into the human sciences such as sociology, anthropology, organizational behavior, and psychology. However, most of the empirical software development research is performed on individual programming activities (Curtis et al., 1986; Curtis, 1987). Curtis and Walz (1990) asserted that software development must be studied at several behavioral levels as indicated in their layered behavioral model. The model emphasized the factors that affect not only cognitive, but also social and organizational processes of software development. At the individual level, only cognitive and motivational processes matter, but at the team level, social processes play a critical role. In that context, Curtis (1987) described five psychological paradigms in the realm of software development. One of the paradigms covers group dynamics, which includes team structure. Curtis has discussed two structures – centralized or chief programmer and decentralized or egoless.

Weinberg (1971) proposed an egoless structure where no central authority is invested in any specific team member. Individuals based on their relatively unique skills pick up tasks. The model, therefore, mandates a free flow of information and public ownership of all artifacts. All share the final work-product and all decisions are team decisions. In essence, the structure requires a high egoless behavior of all the team members.

Many more researchers espoused the cause of egoless behavior. Hewitt and Waltz (2005) pointed out that the information system development projects require knowledge from disparate and different domains that is spread over various team members and stakeholders. They, therefore, suggested shared leadership – on the lines of the egoless programming model – to foster knowledge sharing. Faraj and Sambamurthy (2006) used two types of leaderships – directive and empowering – the latter coming close to egoless programming model. They found that empowering leadership has an important impact on team performance, especially in case of high task uncertainty or team expertise projects. Clarke, et al. (2014) proposed “in-flow peer review” – i.e., peer review done while an assignment is in progress – and underlined the importance of egoless behavior in the review process. Lewis and Smith (2008) concluded that the problem solving styles influence conflicts and performance of software engineering teams. Cockburn (1999) observed that projects progress well, when people “just talk together” and added that good project teams have to keep the person-to-person communication
The Better Part of Valor: Meeting Current Human Needs Online without Compromising Future Selves
www.igi-global.com/chapter/better-part-valor/64943?camid=4v1a