Information and Software Technology 52 (2010) 779–791

Contents lists available at ScienceDirect

Information and Software Technology

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

Process models in the practice of distributed software development: A systematic review of the literature

Rafael Prikladnicki *, Jorge Luis Nicolas Audy

Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS, Computer Science School, 6681 Ipiranga Avenue, Building 32, 90619-900 Porto Alegre, Brazil

ARTICLE INFO

Article history:
Received 23 March 2009
Received in revised form 1 February 2010
Accepted 15 March 2010
Available online 28 March 2010

Keywords:
Distributed software development
Global software engineering
Offshoring
Process models
Process improvement

ABSTRACT

Context: Distributed Software Development (DSD) has recently become an active research area. Although considerable research effort has been made in this area, as yet, no agreement has been reached as to an appropriate process model for DSD.
Purpose: This paper is intended to identify and synthesize papers that describe process models for distributed software development in the context of overseas outsourcing, i.e. "offshoring".
Method: We used a systematic review methodology to search seven digital libraries and one topic-specific conference.
Results: We found 27 primary studies describing stage-related DSD process models. Only five of such studies looked into outsourcing to a subsidiary company (i.e. "internal offshoring"). Nineteen primary studies addressed the need for DSD process models. Eight primary studies and three literature surveys described stage-based DSD process models, but only three of such models were empirically evaluated.
Conclusion: We need more research aimed at internal offshoring. Furthermore, proposed models need to be empirically validated.

Contents

* Corresponding author.
E-mail addresses: rafaelp@pucrs.br (R. Prikladnicki), audy@pucrs.br (J.L.N. Audy).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.03.009

1. Introduction

As part of the globalization efforts currently pervading society, software project team members have become geographically distributed [3,8]. That is a characteristic of Distributed Software Development (DSD). When the distance becomes global, with team members distributed around the world, this characterizes Global Software Development (GSD). The many factors that contributed to DSD or GSD are well documented in the literature [11,13,35,57]. Engineers, managers, and entrepreneurs are facing many challenges on technical, social, political and cultural levels. This change is having a considerable impact on the way products are conceived, designed, tested, and delivered to customers [32]. Thus, the organizational structure and development processes required to support DSD are different from those used in collocated environments [18]. According to Herbsleb and Moitra [32], DSD has different effects on many levels: strategic issues (decision on developing a distributed project); cultural issues; technical issues (technological infrastructure and technical knowledge); and knowledge management issues.

In this context, DSD is a growing field within the Software Engineering (SE) domain. Many companies are distributing their software development facilities, looking for competitive advantages in terms of cost, quality, and skilled professionals [53]. According to Carmel and Tjia [11], the DSD phenomenon started in the early 90s, but it was only recognized as a powerful competitive strategy in the last ten years. Whether local (onshoring) or global (offshoring), within the same company (insourcing) or as a third-party relationship (outsourcing), organizations are facing several important challenges from a SE perspective [43]. After observing practices in the industry, it makes sense to try to understand how these practices have evolved over time, and whether there are process models based on these practices which can be used by organizations that are starting to adopt DSD [48].

Most of the existent literature on DSD stage-based process models tackles strategic aspects of the phenomenon, such as establishing distributed software development centers [12,34], project allocation decisions [25], and client–vendor relationship [45], e.g., from a business perspective. At the same time, however, there are several reasons to consider DSD process models from a technical perspective [43,52,59]. However no agreement has been reached concerning proper process models for DSD.

For this reason, the goal of this systematic review is to identify papers that describe either process models or the need for process models. Our contribution is to identify and categorize studies addressing or developing DSD process models, differentiating between models based on external vendor organizations and those based on wholly-owned subsidiaries.

This paper is organized as follows: in Section 2, we present the concepts involved in the identification of process models (also called stage, capability and maturity models). In Section 3, we set forth the taxonomy used in this systematic review, while in Section 4 we present the method. In Section 5, we set out the results, while in Section 6 we discuss the findings and future directions in this area.

2. Process models

Process models encompass a set of practices or a set of standard steps (or stages) that were successfully followed in the past by individuals, project teams, or organizations, and were documented as a successful practices capable of adoption by other peers. Carmel [10] defines stage models as powerful frameworks in understanding a phenomenon, given that they capture evolution and growth, and also reflect learning curves and diffusion. Carmel [10] argues that such models are useful for both research and practice: practitioners can use such models to understand where they are, where the competition is, and what they can do to evolve. On the other hand, researchers can not only identify and propose the models, but also use them to better understand the behaviors behind a given phenomenon. Such process models can also be defined as maturity and capabilities models. Chrissis et al. [15] define capability as the predictability of the process and its outcomes, or the range of expected results that can be achieved by following a process. On the other hand, the authors define maturity as the growth in the process capability, a well-defined evolutionary path toward achieving a mature process, where each maturity level provides a layer in the foundation for continuous process improvement. Achieving each level of a maturity framework means an increase in the process capability.

But despite the usefulness of such models, they have always been an easy target for criticism, as stated by Carmel [10]. Some criticism includes: the models are heuristically developed, they are usually not validated, they are incomplete, and they assume a linear evolution through each stage. While such criticism is valid, the author also states that, in the end, the collective understanding of a phenomenon would be poorer if these models were not identified. In addition, the author also argues that these models are more useful at the early stages of the phenomenon. Once the phenomenon is mature, the interest in such models is not so evident.

The use of process models or stage models is not something new in Computer Science. They are also very common and can be found in the Social Sciences, where Tuckman proposed a well-known model [64]. The author developed a model to describe the stages (or sequences) of group development. In Computer Science, within the Information Systems domain, one of the first stage models was proposed by Nolan [46], with the purpose of analyzing the evolution of managing the computer resource. In SE, it is possible to find the influence of Nolan's thoughts on the development of models such as the SW-CMM and CMMI [15], among many others. During the development of his work, Nolan [46] also said that stage theories have proved to be useful to develop knowledge in several fields during their formative periods, which is exactly the case of Distributed Software Development.

In DSD, after a couple of years understanding specific problems faced by organizations [11,18,19], both academia and industry realized that it might also be useful to understand which models, by documenting successful practices and processes, can be derived from this past experience [31,52,57,59]. That is exactly what we aim to accomplish with this systematic review.

3. Taxonomy used in this systematic review

The terminology used for Distributed Software Development is not standardized. In this paper we are concerned solely with situations where software development is moved to another country, which is sometimes called Global Software Development. We used a taxonomy based on previous studies [11,39,49,55,56,63] to define the way how distributed software development was organized:

- Offshore outsourcing is used when software development is moved to an external third party in another country.
- Internal offshoring is used when software development is moved to a division of a specific company established in another country.
- Offshoring is used as a generic term when the relationship of the overseas company with the client's company is unknown.

The company that requires the software is referred to as "client". In the context of internal offshoring the client's company