The Journal of Systems and Software 86 (2013) 1663–1678

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

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

A survey study of critical success factors in agile software projects in former Yugoslavia IT companies

Dragan Stankovic[a,*], Vesna Nikolic[b], Miodrag Djordjevic[c], Dac-Buu Cao[d]

[a] Faculty of Technical Sciences, University of Pristina, Kosovska Mitrovica, Serbia
[b] Faculty of Occupational Safety, University of Nis, Nis, Serbia
[c] Faculty of Sciences and Mathematics, University of Nis, Nis, Serbia
[d] Siemens PLM Software, CA, USA

ARTICLE INFO

Article history:
Received 2 July 2012
Received in revised form 13 February 2013
Accepted 13 February 2013
Available online 21 February 2013

Keywords:
Software development
Agile methods
Critical success factors

ABSTRACT

Determining the factors that have an influence on the success of the software development projects has been the focus of extensive research for more than 30 years. In recent years agile methodology of software development has become the dominant one for all kinds of software development projects. In this paper we present the results of empirical study for determining critical factors that influence the success of agile software projects which we conducted among senior developers and project managers from IT companies located in the former Yugoslavia countries within South Eastern Europe (SEE) region. This study is inspired by the similar study conducted 5 years ago (Chow and Cao, 2008). With this study we were not able to confirm the model developed in the previous study. Moreover it disconfirmed not only part of the factors, but very much questioned the whole scheme. However, we were able to shed additional light regarding agile software development in former Yugoslavia countries from SEE region as a reference region for investigating outsourced projects done in agile way.

1. Introduction

A process of software development has been the focus of interest of many managers, engineers and researchers due to a large percentage of failures in the software industry. Failures range from inability to provide software solution that fits the requirements on time, to providing solutions that are a maintenance nightmare or in the worst case inability to provide any solution at all (abandoned software projects). One of the main problems that make software development so special and which is causing the above mentioned difficulties is that during the project both technology and the business environment change (Williams and Cockburn, 2003). That change is, due to technology advancement nowadays, even more dynamic than at the time Agile movement started its development; it is causing customers to have difficulties not only to state their needs in the beginning of the project but even to have a basic idea of what they need at that time and to form requirements only after a few iterations of the demo product. The process of forming requirements includes changes as well. The result of this situation is the development of a variety of methodologies and practices that embrace changes like SCRUM, Extreme Programming, Lean Software Development, Kanban, Crystal, Feature Driven Development, etc.

Over the years agile methods have proven to overcome many of the problems stated above and have become dominant in the software industry. The agile approach is basically driven by self-organizing teams that have the power to coordinate their work on their own. This increases productivity, enables employees to learn, innovate, and finally makes them happy with what they do (Smite et al., 2010a). In the traditional plan-driven (waterfall) software development processes, work is coordinated by managers and there is a clear separation of roles (Moe et al., 2010). Similarly, in larger organizations there is a tendency of organizing people around component teams – grouping people in the way that they have the influence over the small part of the product thus giving the teams less control and losing the ability of close collaboration. Introducing new features in the organization of that kind requires synchronization of many different component teams (Larman, 2011). In the agile approach, a self-organizing team decides how work is coordinated and has the complete control over development process and introduction of new features. However, many organizations, and especially large organizations, still base their software development around plan driven or component teams. The transition of such teams to agile teams and how to overcome difficulties that occur during that process has

* Corresponding author. Tel.: +381 63 1089310.
E-mail address: dragan.stankovic@pr.ac.rs (D. Stankovic).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.02.027

been the subject of many case studies (Moe et al., 2010; Ogle et al., 2011). All of them agree that this kind of transition is a hard process. That process often has dead ends in means of trying and abandoning various software development practices that do not work for the current team. Sometimes it is filled with team members' frustration, skepticism, denying, but regardless of everything mentioned, achieved benefits ramify the cost of transition.

As Smite et al. (2010a) comprehend there is a growing need for companies to explore global sourcing leading to distributed software projects with geographically, temporally and socio-culturally dispersed teams facing additional challenges when trying to successfully implement agile values and principles. The research in Smite et al. (2010b) has shown that it is possible to successfully apply agile principles in distributed environment although these two can be considered as opposite extremes.

As stated by many authors (Smite et al., 2010b; Hansson et al., 2006) agile development practice has always been ahead of research. Academic research has mostly been involved with trying to understand what is going on and exploring in a scientific way techniques and procedures which were already established and used by the community of software developers and agile practitioners. One such example is work by Hansson et al. (2006) where they investigated the differences between industrial practices and agile development practices in several companies. They have concluded that the actual industrial practices used by companies were dependent on companies' characteristics and projects on which they worked on. Another study (Chow and Cao, 2008) among agile professionals has shown that only 10 out of 48 hypotheses were critical to success of agile projects. Based on the survey from that study in the work presented here we have tried to identify critical success factors in agile software projects on the sample of companies operating in SEE region (region sometimes referred as Western Balkans). Similarly to the authors of previous study we have used statistical methods to evaluate responses we received from our interviewees and verify obtained model. It can be debated whether model driven estimation based on statistics such as the one used in our study is the proper method to use when it comes to software engineering. For example, a study by Johansson (2000) suggests that the use of statistics is perhaps inappropriate in the field of software engineering due to all the difficulties associated with interpreting the results and many existing uncertainty factors. Similarly, Jørgensen and Boehm (2009) suggest that the use of both formal methods and expert judgment might be best. They also conclude that future efforts should be headed toward judgment based methods. Nevertheless, we have decided to use the same method as in previous study mainly to make more accurate comparison with the results of previous study. Since our study was not able to confirm the model developed in the previous study, in conclusions of this paper as part of our future research efforts we announce the use of AHP (Analytic Hierarchy Process) in order to create a better model.

Our survey was conducted among developers from over 20 companies. The general characteristic of our interviewees is that they mostly work in distributed environments meaning that they face additional challenges mentioned in the section above. We were interested in whether these specific environmental factors will influence the conclusions made by Chow and Cao (2008). In an article by Freudenberg and Sharp (2010) which was created as a result of panel discussion where practitioners identified the list of issues related to agile software development they would like to be researched, three of the top 10 items focus on distributed teams. This survey will hopefully show the difference to the previous study which can be attributed to both demographic and distributed teams effect.

2. Background (intro to agile methods)

Our survey showed that the most popular agile methods used among former Yugoslavia agile teams are: XP, Scrum, and Feature Driven Development. We have also received responses in which some hybrid methods were used. Interestingly, there were no reports on use of Lean or Kanban which are becoming increasingly popular in IT industry in recent years (Anderson, 2010; Kniberg and Skarin, 2010). The reason for this could be the delay in implementing the latest practices that exist in developers' communities from the more developed regions (for example in Silicon Valley). The possible cause of that delay could be attributed to the absence of modern trainings lead by qualified and experienced professionals that are available in more developed regions, or the fact that cutting edge projects are less likely to be outsourced to overseas teams. Nevertheless the most popular agile practices are there and were considered in our study.

Many agile methodologies share a lot of practices and have common characteristics and approach to projects, but each is unique in its own strategy of their implementation. In the following sections we will briefly describe each of the popular methods which we identified as being used by surveyed companies.

2.1. Extreme programming

Extreme Programming (XP) is one of several popular agile processes. It was originally described by Kent Beck (one of the authors of Agile manifesto) and has been proven to be very successful at many companies of all different sizes and within various industries. The focus of this approach is on customer satisfaction so it empowers developers to be able to respond to changing customer requirements and to deliver high-quality software quickly and continuously. Working software is delivered to customer typically in intervals of 1–3 weeks. XP improves software projects by embracing communication, simplicity, feedback, respect, and courage. The original XP recipe contained 12 rules: Planning Game, Small Releases, Customer Acceptance Tests, Simple Design, Pair Programming, Test-Driven Development, Refactoring, Continuous Integration, Collective Code Ownership, Coding Standards, Metaphor, and Sustainable Pace.

Like in every agile process these rules are not written in stone and over time certain rules were modified, new rules appeared and for example in Shore and Warden (2008) authors make distinction between two versions of XP described by Beck (1999) and Beck and Andres (2004) and even introduce their own approach to XP which emerged from their practical experience.

2.2. Feature driven development (FDD)

FDD was introduced by Jeff De Luca in 1997 and later as a result of collaboration with Peter Coad first incarnations of FDD appeared (Coad et al., 1999; Palmer and Felsing, 2002). FDD is a model-driven, short-iteration process. It begins with establishing an overall model shape and identification of features. Features are then grouped in work packages. One work package can be finished within a single iteration and it actually represents working software with which customer can play with.

FDD designs the rest of the development process around feature delivery using the following eight practices:

1. Domain object modeling
2. Developing by feature
3. Component/class ownership
4. Feature teams
5. Inspections