Distributed Coding Dojo Randori: An Overview and Research Opportunities

Bernardo Estácio[1], **Josiane Kroll**[1] **Rafael Prikladnicki**[1]

[1]PUCRS - Pontifícia Universidade Católica do Rio Grande do Sul
Computer Science School – Porto Alegre – RS – Brazil

{bernardo.estacio,josiane.kroll }@acad.pucrs.br, rafaelp@pucrs.br

Abstract. *Distributed Software Development (DSD) is widely adopted in the software industry. Its adoption require highly responsive teams that can support a product development in an environment that poses several challenges. Coding Dojo Randori (CDR) is an collaborative software practice for training people that offers the opportunity for developing new skills. Thus, CDR can be applied for different contexts that includes DSD. In this paper, we present an initial discussion about the concept, hereby called Distributed Coding Dojo Randori, by analyzing untapped research opportunities for the area.*

1. Introduction

Distributed Software Development (DSD) is a trend in the software industry. Companies are looking for benefits such as the reduction of high travel costs, the access of a geographically skilled staff as well as the possibility to innovate and share best practices inside the organization [Conchúir et al. 2009]. Despite that, efforts are still need for evaluating the applicability of coding practices in distributed contexts, specially for those dealing with agile development [Jalali and Wohlin 2010].

Coding Dojo Randori (CDR) is a collaborative practice that can be applied for different software development contexts. It is defined as a collaborative practice where a group of developers train and learn about some technology concept (programming language, framework). CDR is particularly known as a technique to train agile concepts as Test-Driven Development (TDD) [Sato et al. 2008].

CDR have been practiced in the software industry over the years [Rooksby et al. 2014], but a little is known about this practice in DSD contexts. In order to better understand the Distributed Coding Dojo Randori (DCDR), we present an initial discussion about this concept and untapped research opportunities for the area.

2. Coding Dojo

A Coding Dojo is defined as a meeting where a group of programmers gets together to learn, practice, and share experiences. There are different formats for Coding Dojo. All of them depends on the rules that participants should follow. The main formats adopted in the software industry are Kata and Randori.

In the Kata format, exercises are practiced in advance and then performed in front of the audience during the meetings [Rooksby et al. 2014]. Instead of showing the final code and tests, the presenters start from scratch, explaining each step and allowing the other participants to ask questions or make suggestions. The main goal here is

making everyone to reproduce the steps and solve the same problem after the meeting [Sato et al. 2008].

The Randori format consists of a group of developers working together to solve a specific task. One participant acts as the driver, another one as the navigator, and the remaining participants are the audience that can also participate.The roles of individuals are changed in rounds. Sato et al. [Sato et al. 2008] recommends that each round should last from 5 to 7 minutes. At the end of a round, the driver moves to the audience, the navigator turns into the driver and someone of the audience starts to act as a navigator [Sato et al. 2008].

Despite Randori format is being widely practiced, in the literature there are few studies that evaluate this technique. Sato et al. [Sato et al. 2008] presents how is the different formats with some lessons learned about the organization of a Coding Dojo session in industry and academy. Da Luz et al.[Da Luz et al. 2013] reports a investigation about the learning of TDD in Randori sessions. Though the perceptions of participants, the Randori session helped them to learn TDD.

3. Distributed Coding Dojo Randori

Since 1990 when the first research report in DSD was published [Prikladnicki et al. 2010], studies have been providing an understanding about DSD challenges. Studies have discussed the importance of training teams for DSD [Damian et al. 2006]. Thus, CDR as seem as an opportunity for DSD. For the purposes of this position paper, we define "Distributed Coding Dojo Randori" as "an collaborative practice based on Coding Dojo Randori concept applied em Distributed Software Development environments". Figure 1 shows the research topic emerged from the union of DSD and CDR.

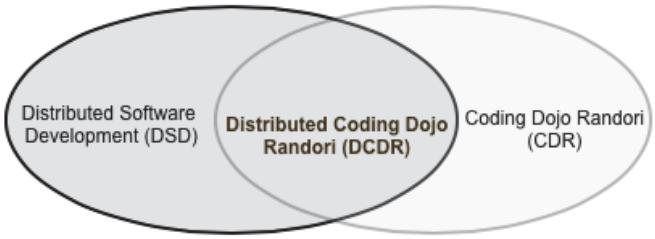

Figure 1. Distributed Coding Dojo Randori research topic.

There are a number of reasons that CDR makes particular sense for DSD. First, for training distributed software teams, in particular newcomers or new hires, to introduce or practice on a particular new concept (programming language, technique). Second, CDR fits to agile software development context, for instance TDD. Finally, DCDR can help in the knowledge transfer of the project among the distributed teams.

Traditional or agile development in DSD involves team members distributed in different places, countries or even continents. Any activity within the software development life cycle (SDLC) can be executed in DSD. Teams geographically dispersed can work in testing, coding, designing or any activity. There are many team settings and examples range from remote sub-teams producing specific modules of a system to teams where different functional roles such as developer or business analysis are executed at dif-

ferent locations [Lane and Agerfalk 2008]. Some examples where CDR could be applied in DSD are described bellow.

Newcomers in Distributed Teams: DCDR can help new members to get introduced in the distributed team, mainly due the close collaboration perspective that practice has. Through the DCDR the sessions would be possible to promote a socialization among the team as well to support the alignment and intregration among the developers.

Agile Software Development: There is a growing interest to evaluate Agile and DSD [Jalali and Wohlin 2010]. In this context, DCDR can be applied through the team especially to train agile techniques such as TDD and Refactoring.

Knowledge Transfer: The different levels of knowledge transfer are challenging for DSD [da Silva et al. 2010]. DCDR can help the team to get in contact together in the project, spreading the knowledge among the developers. DCDR can also be used to solve impediments in the project together, distributing the knowledge between different sites and developers.

4. Research Opportunities for DCDR

There are many research opportunities for DCDR. Here we categorize these opportunities in three large research topics that can be spread in specific topics. Next, we present each of them.

Context of application: CDR can be adopted in academic and industry contexts [Sato et al. 2008]. Thus, global software education brings an opportunity to evaluate the adoption of DCDR in courses to understand how the distributed teams work combined with software tasks to train and learn a specific concept in programming, such as TDD. There are some aspects that should be investigated such as the presence of a tutor, the configuration of the courses, and also how the learning can be assessed.

With the skilled professionals distributed, DCDR can help especially in knowledge transfer between the teams. In addition, DCDR can be applied to train teams in a specific concept, technology, or just to facilitate new distributed team members to get familiar with the project and technology used. There are challenges in adopting DCDR, especially the number of teams members, the task complexity and also the level of experience between the professionals.

Tools to support DCDR: Tools are essential to make DCDR works. There are opportunities to investigate which existing tools are useful to support the practice properly. Also, we need to analyze if there are special requirements or features that need to be develop for them, and how can we develop those.

Tools should support the rules of Randori (driver, navigator, and audience) and provide screen sharing with text and audio channel. In this context, it is important to assess if the infrastructure for synchronous code-collaboration with a group of participants is feasible or not.

DSD settings and aspects: the software development distribution involves several challenges such as coordination, culture, language, and organization of the company in terms global development. These aspects need to be investigate in DCDR to identify how they can impact in the dynamic of the practice.

There is an opportunity to analyze if the rules in the Randori format should be modified or not, for instance the participation of the audience during the session. Also, the coordination of the session through the presence of a organizer and how can we can support it.

5. Final Remarks

In this paper we promote an initial discussion towards the research opportunities in DCDR. We summarize the relevant information about the topic and untapped research opportunities for the area. As next steps, we plan to run empirical studies to contextualize DCDR and analyzed its benefits and limitations of the practice. We are also planning to further identify the main variables that can influence the teams training in distribute settings.

References

Conchúir, E. O., , P. J., Olsson, H. H., and Fitzgerald, B. (2009). Global software development: Where are the benefits? *Commun. ACM*, 52(8):127–131.

Da Luz, R., Serra Seca Neto, A., and Vida Noronha, R. (2013). Teaching TDD, the coding dojo style. In *IEEE 13th International Conference on Advanced Learning Technologies (ICALT), 2013*, pages 371–375.

da Silva, F. Q. B., Costa, C., Franca, A. C. C., and Prikladinicki, R. (2010). Challenges and solutions in distributed software development project management: A systematic literature review. In *2010 5th IEEE International Conference on Global Software Engineering*, pages 87–96.

Damian, D., Hadwin, A., and Al-Ani, B. (2006). Instructional design and assessment strategies for teaching global software development: A framework. In *International Conference on Software Engineering*, pages 685–690, Shanghai, China. IEEE.

Jalali, S. and Wohlin, C. (2010). Agile practices in global software engineering - a systematic map. In *2010 5th IEEE International Conference on Global Software Engineering*, pages 45–54.

Lane, M. T. and Agerfalk, P. J. (2008). On the suitability of particular software development roles to global software development. In *2008 IEEE International Conference on Global Software Engineering*, pages 3–12.

Prikladnicki, R., Audy, J. L. N., and Shull, F. (2010). Patterns in effective distributed software development. *IEEE Softw.*, 27(2):12–15.

Rooksby, J., Hunt, J., and Wang, X. (2014). Agile processes in software engineering and extreme programming. chapter The Theory and Practice of Randori Coding Dojos, pages 251–259. Springer-Verlag, Berlin, Heidelberg.

Sato, D., Corbucci, H., and Bravo, M. (2008). Coding dojo: An environment for learning and sharing agile practices. In *Agile Conference, 2008*, pages 459–464.