2016 9th International Workshop on Cooperative and Human Aspects of Software Engineering

On the Randori Training Dynamics

Bernardo Estácio
Pontífica Universidade
Católica do Rio Grande do Sul
Faculdade de Informática
Porto Alegre, Brazil
bernardo.estacio@pucrs.br

Franz Zieris
Freie Universität Berlin
Institut für Informatik
Berlin, Germany
zieris@inf.fu-berlin.de

Lutz Prechelt
Freie Universität Berlin
Institut für Informatik
Berlin, Germany
prechelt@inf.fu-berlin.de

Rafael Prikladnicki
Pontífica Universidade
Católica do Rio Grande do Sul
Faculdade de Informática
Porto Alegre, Brazil
rafaelp@pucrs.br

ABSTRACT

Background: Coding Dojo Randori is a collaborative practice of joint training (with discussion). *Objective:* Evaluate Randori training behaviors. *Method:* Qualitative data analysis of recordings of Randori sessions. *Results:* (1) The training may involve different levels of collaboration, from a task level to a concept level. (2) Randori can help novices via the interaction with more experienced developers. *Conclusion:* Suitable behavior and interactions of the developers in a Randori session can help to create an environment with valuable discussion on a specific software topic.

CCS Concepts

•**General and reference** → **Empirical studies;** •**Software and its engineering** → **Programming teams;** *Agile software development;*

Keywords

Randori, Training, Agile Software Development

1. INTRODUCTION

In a Coding Dojo session, a group of developers collaborates to train and learn about some technology concept (programming language, framework) or an agile practice, e.g. Test-Driven Development [7]. There are several variants of Coding Dojo [7] and one of them is called Randori: Free-style joint practice. Rooksby et al. [6] say *"The coding dojo is by no means the only available approach for professional developers to continuous learning [...], But the coding dojo format (specifically the Randori format) is widely practiced and worthy of serious attention"*. We agree. This paper will analyze the Randori format empirically.

There are few empirical studies that explore Randori but the initial results from survey studies are promising. Heinonen et al. [4] and Da Luz et al. [5] present positive results for learning agile practices such as Test-Driven Development (TDD) [1]. Those studies show that Randori is basically effective, but we still need to understand how to do it well.

Our research uses qualitative methods to analyze Randori session recordings. With those rich data, we analyzed the collaborative learning behavior or training dynamic, motivation between the developers and also communication patterns. We decided to pick the training dynamic of Randori as the topic of this study: How do participants interact in order to practice a specific concept or technology (in our case TDD[1]). We conjecture that suitable interaction is key to make a Randori valuable.

The effectiveness of a Randori session is not measured by the amount of software tasks finished [7]. Rather, it is related to the discussions, in particular those aligned with the goal, such as understanding and learning the TDD practice. Our observations involve a few interventions based on what we observed in previous sessions and feedback from participants: We varied the number of proposed tasks and the number of participants. A long-term goal of our research is to provide a set of variables and strategies that can support the training performance of Randori sessions.

We ask the following research question: What mechanisms influence the training dynamic during a Randori session and what is their impact on training effectiveness?

Section 2 will describe some related work. Section 3 will show the research method, presenting the settings of our empirical study. Sections 4.1 to 4.6 shortly present results: collaboration levels, roles, master intervention, newcomer participation, audience participation, and participants' perceptions. Section 5 discusses limitations of our work and Section 6 concludes.

2. RELATED WORK

Coding Dojo, in the Kata or the Randori format [4], is

CHASE'16, May 16 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4155-4/16/05...$15.00
DOI: http://dx.doi.org/10.1145/2897586.2897603

[1]We will assume the reader is familiar with at least the basic idea of TDD: Always write and run a test before you implement functionality and do so in very fine-grained iterations.

often described in the context of learning or training certain agile practices, in particular TDD (Test-Driven Development) and refactoring [5, 7]. In a Randori session, a group of participants practices in the following manner [5]: (i) one participant acts as the driver (writing the code), (ii) another one as the navigator (or "observer", supporting the driver), and (iii) the remaining participants act as the audience that can participate in discussions.

The positions (driver and navigator) of individuals are changed in rounds. Sato et al. [7] recommend that each round should last from 5 to 7 minutes. At the end of a round, the driver joins the audience, the navigator becomes the new driver and someone from the audience becomes the new navigator [7]. Every participant acts at least once as driver and once as navigator.

There are only few studies in the literature empirically assessing Randori. As a first attempt, Sato et al. [7] present the dynamics of the practice and some lessons learned about the organization of the session. More recently, Da Luz et al. [5] reported a Randori experiment that aimed at investigating the learning of TDD. Their subjects perceived the session as helping them to learn TDD.

Heinonen et al. [4] conducted Randori sessions when teaching agile methodologies as part of an undergraduate software engineering course, with similar findings. In addition, most of their students reported perceiving the sessions as a relaxing and non-competitive environment.

Rooksby et al. [6] report a lack of studies of Randori learning effectiveness and analyze two sessions from the perspective of reflective practice [9] – a learning theory that states that the subjects learn by their actions during the practice. They conclude that this theory offers a good way to understand the cooperative learning in Randori.

In previous studies, we evaluated Randori combined with Pair Programming [2, 3], analyzing mainly the perceptions of the subjects. To the best of our knowledge, the study reported in this article addresses the gap of evaluating Randori training perspective empirically beyond survey mode. We are now interested to analyze sessions recordings qualitatively, akin to Schenk et al.'s [8] and Zieris et al.'s pair programming [11] work.

3. METHOD

We base our analysis on three recordings of Randori sessions with students from Freie Universität Berlin. The participants' goal in the sessions was to practice Test-Driven Development (TDD). The participants volunteered to participate in the sessions and all of them agreed to be recorded for research purposes. For the benefit of the first author, the participants talked in English although none of them is a native English speaker.

3.1 Subjects and Sessions Context

All participants are students from the Bachelor and Master courses in Computer Science at Freie Universität Berlin. We used a specific room and time frame for the sessions. The schedule of the sessions involved a short presentation of Randori by the first author (ca. 10 minutes), solving the tasks (intended to be 75 minutes long), and a retrospective about the session (intended to be 15 minutes).

The First Session (**S1**, 1:50 hours) was executed with three developers (three male), two master and one bachelor. The members knew each other well. All three had previous some experience with TDD in open source projects, but they did not consider themselves experienced with the practice. We used three tasks during the session: FizzBuzz, Fibonacci, and Game of Life (in this order, which is one of increasing difficulty). Those tasks are classic Katas.[2]

The Second Session (**S2**, 1:45 hours) also had three developers, again one master and two bachelor (one female and two male). Again, the members knew each other well. One member did not have any previous experience with TDD, and the others considered their knowledge level low. This time, we used only two tasks: Roman Numerals[3] and Mars Rover.[4]

The Third Session (**S3**, 2:10 hours) had six developers (one female and five male), five bachelor and one master. The members did not know each other well. Three members did not have previous experience with TDD, the others had "low" knowledge. We used two tasks: FizzBuzz and Flatland.[5] In relation to FizzBuzz, we took also an improved version of the problem since one of participants had also attended S1.

We used Camtasia Studio to record the screen along with a web cam view from the corner of the room (to capture all participants). We used two audio channels: a dictaphone for the audience and the integrated microphone of the web cam for driver and navigator. The first author observed the live session passively.

3.2 Data Analysis Method

Our data analysis is based on some elements of Grounded Theory Methodology (GTM, [10]): open coding [10, Section II.5], constant comparison [10, Section II.1], and theoretical sensitivity [10, Section I.3].

3.3 Notation

Our results are primarily concepts, for which we adopt a specific notation: We typeset their names in small caps and discriminate levels of elaborateness as follows. "V" (SOME CONCEPT^V), for "vague", represents informal concepts that appeal to intuition and for which there is hardly more description than their name. "S" (SOME CONCEPT^S) marks semi-complete concepts for which a concrete definition is available but where we expect that definition to be incomplete and/or unstable (from the point of view of more detailed further research on the topic). We have not yet reached the third level, fully circumscribed GTM concepts, for any of the concepts described in the present article.

4. RESULTS

4.1 Collaboration Levels

The training dynamic of a Randori session involves different collaboration levels. The desirable CONCEPT-LEVEL^S occurs when the collaboration (discussion and questions) between the participants concerns the main goal of the session, in our context the practicing of TDD. TASK-LEVEL^S interaction concerns the programming task as such. PROGRAMMING-LEVEL^S interaction, the least helpful mode, occurs when

[2]http://codingdojo.org/cgi-bin/index.pl?KataCatalogue
[3]http://codingdojo.org/cgi-bin/index.pl?KataRomanNumerals
[4]https://marsroverexercise.codeplex.com/
[5]https://icpcarchive.ecs.baylor.edu/external/25/2550.pdf

participates dive down into basics, typically programming language questions.

4.2 Roles

We found there are more roles taken by participants in a Randori session than the known ones of driver, navigator, and audience [4, 7]. We describe two we have already understood well enough.

4.2.1 Master

A MASTER^S acts like a tutor for the others. In our sessions, the most experienced participant naturally and implicitly assumed this role. The MASTER^S was be most active in particular during CONCEPT-LEVEL^S collaboration, reminding the other participants of the fundamentals of TDD. In TASK-LEVEL^S phases, s/he would help solve blocker questions.

4.2.2 Newcomer

A NEWCOMER^S has the dual role. Where the MASTER^S supplies knowledge, the NEWCOMER^S will ask for it, again particularly intensively during CONCEPT-LEVEL^S collaboration. Not everybody inexperienced with TDD acted in the NEWCOMER^S role.

In our sessions, all NEWCOMER^{S}s were inexperienced, but NEWCOMER^{S}s with some prior skill are conceivable. The NEWCOMER^S appears to be valuable to increase learning intensity in a Randori, and is most obvious and active when acting as driver. In our sessions, the NEWCOMER^{S}s remained mostly silent as navigators.

4.3 Master Intervention

A MASTER^S will not only supply knowledge upon request, s/he will also actively intervene in the session.

The following scene occurred when the MASTER^S saw that driver and navigator were creating a test case that became larger and larger:

Master: *So, TDD would be to implement [a test] before the implementation. For every test or no?*

Navigator: *Yes, You should do one single assert for one single case that you want and then write the code.*

Driver: *Or we can make multiple (asserts) for huge cases.*

Master: *Hum, er, but the idea of TDD is take one small case, test, implement it, and then take the next.*

At a different time, driver and navigator were both inexperienced and did not have a good idea where to start:

Navigator: *I think that we can start to test.*

Driver: *We can test already many numbers like 0, 2, 4, 9, 33 and then compare.*

Master: *First, write the test for number 0 for instance and then you will see the next approach.*

Driver/Newcomer: *What do you mean?*

Navigator: *He said that first we need to implement one test only and then we will see how we can develop for the others numbers.*

Driver: *OK.*

4.4 Newcomer Participation

Obviously, a NEWCOMER^S can contribute through a NEWCOMER PARTICIPATION^S, mainly at the CONCEPT-LEVEL^S.

Driver/Newcomer: *So, what type of assert should I use here?*

Navigator: *AssertEquals*

Driver/Newcomer: *Er, OK, but how we will test it if we do not have anything implemented and no output?*

Navigator: *Don't worry, let's write the test first for 3 and then we will write an if in the code to pass it later, OK?*

The Newcomer's first question was at the PROGRAMMING-LEVEL^S; she commenced with a CONCEPT-LEVEL^S question.

4.5 Audience Participation

In the large group (**S3**), the audience did not participate a lot in the development process; we mostly observed some PARALLEL CONVERSATION^V and also some DISTRACTION^V with unrelated topics.

In the small groups (**S1**, **S2**), we observed a nearly continuous AUDIENCE PARTICIPATION^V from the single member of the audience. However, the PARALLEL CONVERSATION^V o'ccasionally proved helpful for driver and navigator, especially at the PROGRAMMING-LEVEL^S:

Audience participant 1: *Hmm, but it is a little bit strange because we instantiated s1 as 1 and afterwards we did not get 1. Just strange for me!*

Audience participant 2: *I can't understand [the logic], maybe we can change the name of the variables.*

Driver: *Indeed, it is not so fine.*

4.6 Perception of Subjects

During the retrospective at the end of each session, we discussed positive and negative points and improvements. From the three sessions, the main positive points suggested were: collaboration (different ideas for a solution), fun (a different way to train), group commitment (the group with the same purpose), and learning (especially for a first contact of TDD).

As negative points, the **S1** participants pointed out the strict times for round-switching were not helpful (which we then relaxed for S2 and S3 with good success). They also found a group of three a bit small (which we increased in S3, after failing to find more than three members for S2). In **S2**, they found the task too difficult. The **S3** participants commented about too-slow decision-making (e.g. for the strategy to solve a task) in their large six-person group.

As another improvement, groups **S1** and **S3** suggested to provide a pre-created project with an empty method and a first test case. We will consider that for the further sessions.

5. LIMITATIONS

GTM results aim at adequately explaining a set of observations, but never claim complete generalizability, let alone quantification. If the GTM practices are used correctly, the resulting conceptualizations will always be correct, but they can be unhelpful; this our readers can judge themselves. The list of phenomena we report is likely incomplete, in particular with respect to the behavior of participants with much higher general programming skills.

6. CONCLUSION AND FURTHER WORK

Our study has two important findings, which shed some light on Randori aspects that were not covered before in the literature. First, our observations suggest that one should discriminate three different levels of discussion going on in a Randori session: Concept-level discussion about the practice to be trained, task-level discussion about the task used for the training, and programming-level discussion about basic knowledge needed to complete the task. Concept-level discussion is where the learning happens explicitly and what is presumably the most valuable part of a session.

Second, there are two sets of behaviors that presumably increase the value of a Randori session: actively supplying knowledge when it appears to be lacking in driver or navigator ("master" behavior) and actively asking for such knowledge when identifying gaps in one's own ("newcomer" behavior).

As next steps, we will keep using GTM to understand the training dynamic in Randori, along with small interventions to broaden the set of observations and to improve the training performance (GTM theoretical sampling).

We will refine our understanding of what specific newcomer and master behaviors or behaviors of others help the session and whether further roles should be recognized. Alongside this, we will accumulate more evidence about useful group sizes, gender effects, group compositions, and other session arrangements.

7. ACKNOWLEDGMENTS

The authors acknowledge financial support in this research from CNPq (process 206752/2014-8/SWE) and the research agreement between PUCRS and ThoughtWorks.

8. REFERENCES

[1] K. Beck. *Test Driven Development: By Example.* Addison-Wesley Professional, 2002.

[2] B. Estacio, R. Oliveira, S. Marczak, M. Kalinowski, A. Garcia, R. Prikladnicki, and C. Lucena. Evaluating collaborative practices in acquiring programming skills: Findings of a controlled experiment. In *Proc. 2015 29th Brazilian Symposium on Software Engineering*, SBES '15, pages 150–159, Washington, DC, USA, 2015. IEEE Computer Society.

[3] B. Estacio, N. Valentim, L. Rivero, T. Conte, and R. Prikladnicki. Evaluating the use of pair programming and coding dojo in teaching mockups development: An empirical study. In *Proc. 2015 48th Hawaii Int'l. Conf. on System Sciences*, HICSS '15, pages 5084–5093, Washington, DC, USA, 2015. IEEE Computer Society.

[4] K. Heinonen, K. Hirvikoski, M. Luukkainen, and A. Vihavainen. Learning agile software engineering practices using coding dojo. In *Proc. 14th Annual ACM SIGITE Conf. on Information Technology Education*, SIGITE '13, pages 97–102, New York, NY, USA, 2013. ACM.

[5] R. B. d. Luz, A. G. S. S. Neto, and R. V. Noronha. Teaching TDD, the Coding Dojo Style. In *Proc. 2013 IEEE 13th Int'l. Conf. on Advanced Learning Technologies*, ICALT '13, pages 371–375, 2013.

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

[7] D. Sato, H. Corbucci, and M. Bravo. Coding dojo: An environment for learning and sharing agile practices. In *Proc. Agile 2008*, AGILE '08, pages 459–464, 2008.

[8] J. Schenk, L. Prechelt, and S. Salinger. Distributed-pair programming can work well and is not just distributed pair-programming. In *Companion Proc. 36th Int'l. Conf. on Software Engineering*, ICSE Companion 2014, pages 74–83, New York, NY, USA, 2014. ACM.

[9] D. A. Schön. *Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions.* Jossey-Bass, 1990.

[10] A. Strauss and J. Corbin. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques.* SAGE, London, 1990.

[11] F. Zieris and L. Prechelt. On knowledge transfer skill in pair programming. In *Proc. 8th ACM/IEEE Int'l. Symposium on Empirical Software Engineering and Measurement*, ESEM '14, pages 11:1–11:10, New York, NY, USA, 2014. ACM.