The Theory and Practice of Randori Coding Dojos

John Rooksby[1], Johanna Hunt[2], Xiaofeng Wang[3]
[1] University of Glasgow, UK
[2] Eventyr Ltd, UK
[3] Free University of Bozen-Bolzano, Italy
john.rooksby@glasgow.ac.uk, johanna.hunt@eventyr.co.uk, xiaofeng.wang@unibz.it

Abstract. The coding dojo is a technique for continuous learning and training. Randori is one implementation format. Even though experience and lessons learnt on how coding dojos could be better organized have been reported in agile literature, the theoretical bases behind it have never been investigated. In this paper we propose to use reflective practice as a sense-making device to underpin the investigation and improvement of coding dojo for effective learning. Based on the examination of two dojo sessions we argue that the insights from the reflective practice and related theories can open new and interesting inquiries on coding dojo, and eventually help to better understand the dynamics of coding dojo, and improve the dojo practice accordingly.

Keywords: coding dojo, deliberate practice, reflective practice, reflect-in-action, reflect-on-action, randori, agile methods, learning

1 Introduction

"If I want to learn Judo, I will enroll at the nearest dojo, and show up for one hour every week for the next two years, at the end of which I may opt for a more assiduous course of study to progress in the art. Years of further training might be rewarded with a black belt, which is merely the sign of ascent to a different stage of learning. No master ever stops learning. If I want to learn object programming... my employer will pack me off to a three-day Java course picked from this year's issue of a big training firm's catalog." [1]

No master, as Bossavit and Gaillot say, ever stops learning [1]. Bossavit and Gaillot's concern is with how professional developers can master their trade. In order to support continuous learning, Bossavit and Gaillot appropriated the dojo format from martial arts. They instigated a coding dojo in Paris, and the format has since been replicated around the world ([2], [3], [4]). The coding dojo is by no means the only available approach for professional developers to continuous learning (see [4] for an overview of how one organisation supports this). But the coding dojo format (specifically the "randori" format) is widely practiced and worthy of serious attention.

Although the dojo format has been discussed widely and is advocated in several textbooks (e.g. [5]), few studies of how coding dojos are practiced have been published. Luz et al. [6] have used interviews and questionnaires to find out about people's experience in dojos. They found dojos to be an activity that favours

"*participation and collaboration in an inclusive learning environment*". A survey by Bravo and Goldman [2] found "*as far as participants' perception goes, the coding dojo is a very effective technique for learning agile practices, independently of how much is already known about them*." These studies are of participants' perceptions and enjoyment of dojos. By doing retrospective evaluations (of what people say after the event, rather than during it) these studies have avoided issues of what happens in a dojo. The studies indicate there is value in dojos, but it is not clear where that value is.

We suggest that attention is turned to practice (by which we mean the embodied, situated and practical conduct of the dojo itself). Much has been said about the coding dojo format, but little about practice. We are well aware that coding dojos are rarely rigidly organised, are often run by enthusiasts, and are often oriented more to local needs and contingencies than any rulebook. But rather than dismiss such dojos as falling short of a grand theory, we suggest that examining their practice gives an opportunity to rethink and question the grander ideas. The paper will argue that rather than tighten up conduct in dojos to more readily resemble "deliberate practice"[7], that the theory itself should be rethought. A candidate 'alternative' theory we consider is Schön's "reflective practice" [8]. Schön advocated that learning is done through practicing. He did not see practice as being a means to becoming so proficient that one does not have to think when acting, but as a means of learning to think when acting; becoming a "reflective practitioner". The primary research question that guides our investigation of the coding dojo is: *Does the perspective of reflective practice enable a better understanding of the coding dojo*?

Section two of this paper will further outline the theory of the coding dojo, and explain the papers focus on "randori" dojos. Section three will use examples from two dojo sessions in the UK to consider how randori dojos play out in practice. Section four will discuss the gap between theory and practice, and suggest that rather than dismiss the dojos we observed as "bad practice" they can be better appreciated with reference to an alternative theory of learning. Section five concludes the paper.

2 The Coding Dojo in Theory

The coding dojo, Bossavit and Gaillot [1] explain, has its origins in Thomas' concept of code kata. Kata is a Japanese term meaning (literally) *form*. The term is used in martial arts to refer to choreographed movements that are practiced repeatedly. Dave Thomas brought this idea to coding [9]. Thomas' "code kata" are exercises for developers to repeat over and over, striving for mastery. The purpose of kata, explains Martin [5], is "*to make the perfected movements automatic and instinctive so that they are there when you need them … you repeat the exercise over and over again to train your fingers how to move and react*" (p.90). According to Martin, kata are useful for learning hotkeys and navigation idioms, and techniques such as test driven development and constant integration. Most importantly, explains Martin, they enable you to develop problem-solution pairs.

Inspired by code kata, Bossavit and Gaillot [1] looked further to martial arts training and appropriated the idea of the dojo. Dojo is a Japanese term that means literally *place of the way*. The term is used in martial arts to refer to a place of formal

training. As with code kata, the coding dojo was developed to support the development of skill through repetitive practice. Indeed, the coding dojo can incorporate kata. Unlike code kata however, dojos are fundamentally social and cooperative.

Dojos can take one of several formats. At *kata dojos*, kata exercises are practiced in advance and are then performed in front of an audience. Bache [10] describes of a kata dojo workshop that it will comprise *"perhaps 6 kata performances … [who] will be chosen in advance and work in pairs*." Bache explains that the remaining participants will be there to learn from and critique the pairs. The performances last 10 to 30 minutes, and are followed by feedback. Alternatively, *wasa dojos* are two-person kata where the pair can either work cooperatively or spar with each other. Martin [5] describes wasa dojos where developers take turns, the first writing a unit test and the second implementing code to pass that test. Another format, the *kake dojo* is discussed in [4]. Kake dojos are events at which a code with the same functionality is simultaneously developed in two or more languages. A fourth format, the *randori dojo*, is outlined below. The randori format is, in our experience, the most commonly implemented, and constitutes the focus of this paper.

Randori is a term used in martial arts to describe free-style sparing. Martin [5] describes of the randori coding dojo format: "*With the screen projected on the wall, one person writes a test and sits down. The next person makes the test pass, writes another test and then sits down. … This can be done in sequence around the table, or people can simply line up, as they feel so moved. In either case it is a lot of fun*." Aniche et al. [4] describe a slightly different format in which, instead of writing one thing and then sitting down, the pairs work "*in time-boxed rounds (usually 5-7 minutes)*." At the end of each time-box, the driver moves to the audience, the navigator becomes the driver, and an audience member becomes the navigator. Aniche et al. explain that the problems are usually simple and the goal is not to solve them but to share knowledge, to practice and to learn. Martin [5] states: "*It is remarkable how much you can learn from these sessions. You can gain an immense insight into the way other people solve problems. These insights can only serve to broaden your own approach and improve your skill*."

The dojo format has grown from a recognition that deliberate practice over a sustained period is at the heart of developing and improving expertise. "Practice" is not meant here in the sense that the developer must be a practitioner, but that they literally have to practice. The idea is not for on-the-job learning. Mastery from this point of view is not gained solely through experience, but through discipline. The developer should practice in the way a professional musician, and indeed a practitioner of martial arts, must practice. But how exactly does the randori format fit with this? How can taking turns (not just with people who may not agree with you, but with people with different interests and competencies) constitute deliberate practice? Is working in front of an audience on a shared project something that instils discipline in the pair doing the doing, or something that enables the audience to witness and consider discipline or the lack thereof? Or does the randori dojo depart entirely from the idea of deliberate practice?

3 The Coding Dojo in Practice

We have obtained a set of twelve recordings (specifically screen and voice recordings) made at a randori dojo organised in the UK. One of the authors of this paper was the facilitator at these sessions. The other authors were not in attendance. The facilitator gained advice from Bossavit and Gaillot before running the dojo sessions. The advice she received was largely pragmatic. Some points were about organisation "*always meet at the same place and the same time - it helps to build a community and a ritual*"; "*encourage discussions, but on green bar only, don't discuss about the code when it's in an unstable state*". Other advice was more practical "*don't cram too much into a session*"; "*use kata to introduce a new language*". Other advice referred to cooperative aspects of the dojo: "*keep a collaborative journal*"; "*make the dojo everybody's responsibility*". Finally, the organiser was invited to: *"Trust your process … part of the learning experience is figuring out what is happening. … Part of building the community is about finding out why you want to meet every week or so as a group. Answers will become clearer as time goes by, provided that everyone feels free to talk and listens to others' opinions."*

The recordings have each been transcribed in full. Of our twelve recordings, half contain independent sessions, and half contain tasks carried across sessions. Participation in the sessions varied in number and mix of experience, with a hard core of regular attendees and others who came and went. For the purposes of this paper we selected two sessions for in-depth, qualitative analysis. Our method of analysis has been very coarse and discursive. Essentially we have sought to understand and characterize what happens in a dojo as the first step of analysis.

The two selected sessions both focus on the same task (one continues from the other) and therefore constitute a pair. The sessions are on the task: *"Write a Text Adventure Game in Java. The game must contain: a house; a cat; a blue necktie; a nodding dog ornament; something orange; a lift"*. The task was originally intended to be for one session, but at the end of the first session the participants elected to continue on it during the next session. Both sessions saw 12 pairs work together in time-boxed rounds of about 5 minutes. Each session had about 10 people in attendance. The task itself was somewhat irreverent, but this is not to say the dojo was not serious. A characteristic of this and other dojos was that the software and, moreover, the interesting aspects of the problem for the participants to work on, were not specified by the facilitator but were emergent in the dojo sessions themselves. The fact that the participants elected to continue for a second session on this task attracted us because it strongly indicates that they found something of value in their progress.

The first pair (a male and female) of the first session began by reading out the task. They then talked about where to start. In their words:

"Is it just something like north, south, east, west, up, down kind of thing? Or are we going to go for something completely weird and strange?"

"Do you think we need to think about that already?"

"Uh, I think we need to have a vague idea of which way we're going."

The pair continued to discuss where to start, and in what level of detail. The male, as driver and thus with control of the keyboard, won out. This was not necessarily through strength of argument but by starting to code, creating a public class "game"

within the first minute. The male suggested the best thing to do was implement a hash map to represent locations. The choice was immediately met with questions of clarification from the audience and a discussion of hash maps. Hash maps are data structures that not everyone in the room was familiar or comfortable with, and not something that everyone felt was necessarily the best data structure to use.

For the second round, the female switched to the driver role, and another male joined her as navigator. The second and subsequent rounds largely continued to be concerned with movement and location (and therefore only a subset of the broader issues in implementing the game). Some work is also done on "items" to be found in the game, but by the very end of the second dojo session, the participants built a game in which it was simply possible to move between several arbitrary locations.

The second and subsequent sessions also continued to use hash maps. In fact, the major topic of discussion, deliberation and sometimes argument during both dojo sessions was hash maps: how, when and why should you implement these? For some participants, hashmaps required basic explanation:

"Yes, hashmaps contain keys and values, and so the key gets you the value."

Interestingly, examples are present in the data of people not previously familiar with hashmaps trying to put what is going on into their own words. Sometimes they did so well; sometimes they had to be corrected. The discussions also covered ways of implementing hashing in java:

"There is with hashmap in Java, a rather neat way of doing this … we can get the key set which will contain north, south, east and west, or what have you, and there's a list, we can ask whether it contains a direction we want to go in."

The person who explained this, later realised that "*it returns a set, not a list*". Even he was learning and discovering, not simply instructing. Elsewhere the discussion turned to consideration of whether and how hashmaps were appropriate in comparison to and in conjunction with other forms of data structure:

"Personally, I don't like the use of vectors, not that vectors are necessarily a bad thing, but they're lists which are designed for a much bigger environment, and I think we should stick to the idea of a hashmap where you can look at a set via a string."

We have said very little so far about how the participants' work conformed to or exemplified good practice. Concerns for agile methods, and good practice more broadly, were present in the sessions but were far less pervasive than the discussions of data structures. Unit tests were not spoken of, let alone implemented until the third round, and it was not until round five that a test-driven development strategy was introduced. The pair in sessions five and six included someone who was clearly passionate and skilled in test-driven development, and he was able to show how this could be done in this situation. Subsequent sessions did not continue with test-driven development but this is not to say that the presence of this was ignored or forgotten, indeed it was appreciated and is referred back to in later sessions as "*inspiring*". Regarding unit testing generally, there was an ebb and flow to this. Some pairs would have a push on writing tests and testable code, whereas others would ignore the tests. One pair even decided to comment out all the unit tests. Similar issues can be seen in the data regarding refactoring, some would take this seriously and treat it as an important part of their practice, while others did not. Some took their roles in pair programming seriously, whereas others would not.

Each dojo session ended with a reflective discussion. A key consideration at the end of the first session was whether it was a failure that they had not ended with a functioning game. One person expressed dissatisfaction with having spent some much time considering data structures and writing tests:

"We got tests. Tests on data structures!"

But others defended how the session had progressed. When asked whether the "*actual aim*" of the dojo was "*to get something functioning*" or to get "*really nice crisp code that does a little bit*" the majority consensus was the latter. One participant pointed out:

"It's the journey not the destination."

Clearly, the dojo was not a productive way of producing a game, but was it actually valuable as a "*journey*"? What does it mean that the journey did not adhere to best practices but that there was an ebb and flow to these? Some people heavily oriented to test-driven development, refactoring and so on, others were unfamiliar with these, and yet others were frustrated by them. We do not believe the situation we have observed of this dojo is unique. The authors of this paper have experience of running and participating in dojos other than the ones discussed here, and while there has been a diversity of practices in these, never have we experienced a dojo run strictly to agile principles.

4 Discussion

Considering the dojo in practice reveals there is something of a gulf between theory and practice. At least, there seems to be something of a leap from kata to randori dojos. The former is about repeating a good practice until it is perfected, but the latter seems to rely on group dynamics and seems susceptible to what might be considered poor or inexpert practice. With this in mind, we have been taking interest in the work of Schön ([8], [11]).

Schön was concerned that professional work (not specifically programming, but expert work more broadly) was too often misconstrued as the application of technical knowledge to clearly defined problems. He believed this view devalued the importance of skill and artistry in practice. He argued for "*a new epistemology of practice*" and sought to demystify what constituted skill and artistry. This led him to articulate what he called "*reflection-in-action*". He explained that professionals routinely grapple with "*situations of uncertainty, uniqueness and conflict*" and when doing so, they can be seen to "*reflect-in-action*". Schön was not referring to a need for taking time out to reflect but was interested in how thinking is intertwined with action. Schön was interested, as Moon [12] puts it, "*thinking on your feet*".

One of the professions Schön studied was architecture. He recognised that when designing, architects are rarely confronted with a clear problem. Rather architects articulate problems in conjunction with their solution. When architects work on problem-solution pairs, they go through a "web a moves", exploring options without necessarily dismissing others, articulating parallel alternatives, and backtracking from dead ends to other lines. He describes architects' reasoning processes as thoroughly dependent on talking and on sketching. Designing, he argued, is not a linear problem

solving but is a material, embodied practice in which alternatives are explored and a problem-solution articulated. His description of designing has been influential, and his ideas used to describe the software design.

Schön went on to question professional education, which he thought concentrated too much on "technical knowledge" and too little on "skill" and "artistry". Not only were graduates leaving education with a lack of skill in their chosen profession, but a lack of the ability to even recognise or articulate what skilful practice is. He developed the idea of "*the practicum*", a setting in which students could be coached rather than instructed in a practice. This was not a call for students to be given work experience, but to create a setting that approximates the practice world. Schön said the practicum should be *"a virtual world, relatively free of the pressures, distractions, and risks of the real one, to which, nevertheless, it refers"*. This way, he explained, practice happens in a double sense: the students engage, usually in simulated form, in the practice they wish to learn; but the students also practice "*as one practices the piano*". Schön proposed the practicum should be under the guidance of "*a studio master*" who would from time-to-time do conventional teaching, but would usually function as a coach "*whose main activities are demonstrating, advising, questioning, and criticizing*". Schön pointed out that as a coach, the teacher does not oversee or instruct each students every move, and that students will often be as important to one another as the coach. In the practicum, the students are not just applying techniques they have been taught in lectures or textbooks, but are developing and learning to recognize skill.

The randori dojo, we suggest, resembles Schön's practicum. Both look beyond 'technical knowledge' and to artistry and skill. Both encourage learning in a virtual world away from real world pressures. Both prioritise interaction among the students over formal teaching, and to a degree the randori dojo allows for coaching. Importantly, we suggest, Schön does not propose that the students in practicums should already be proficient in skilled or best practice, but rather the practicum provides a space in which the students can begin to recognize what skill actually is and to develop these. Of course, there are differences between the dojo and the practicum. Schön's focus was on the training of pre-professionals whereas the dojo is open to people already working as developers. The turn-taking structure and collaborative format of the dojo is also different to Schön's practicum where students would work on individual projects. But on the whole we note some strong similarities.

With reference to Schön's ideas we think it is possible to begin to see value in what was happening in the dojo we reported above. The dojo rarely exemplified good practice, but there were occasions of this and several discussions. With reference to Schön we should consider that this might be characteristic of a group among which there may be some who are skilled, and others who can barely recognize what skill is, let alone understand which certain skills might be beneficial. Schön's work enables us to consider that in order to learn, it may be better that we allow the topics to emerge and be worked out during the course of learning. We should perhaps not be worried if we see that the learners are more interested in data structures than (say) test-driven development. We should perhaps not be critical of the dojo participants for making rash decisions or rejecting good practices (such as commenting out all the tests), but rather understand that this is all part of a learning process - steps on a long road to mastery.

The randori dojo, from this perspective, is unlike kata. It should not be about the coach instructing the participants in a good practice to follow, and should not necessarily see the coach intervene when something untoward happens. We might better see the Randori dojo as allowing mastery to emerge through participants' reflective action.

5 Conclusion

In this paper we have explained and characterised Randori dojos. We argue that these are unlike kata (and kata dojos) and suggest that they are best understood not as demonstrations among a group of best practice, but as cooperative learning where good practice has the opportunity to emerge. We have suggested the work by Schön on reflective practice and practicums offers an appropriate theoretical perspective to consider randori dojos. Our work neither proves nor disproves the value of dojos, and in fact shows that we are some way off being able to evaluate the effectiveness of dojos – we are still deliberating what criteria to judge effectiveness by. The literature on education and learning is expansive (not limited to Schön). It seems opening the door to this raises many more questions than it settles.

References

1. Bossavit, L., Gaillot, E.: The Coder ' s Dojo - A Different Way to Teach and Learn Programming. XP2005. pp. 290–291 (2005).
2. Bravo, M., Goldman, A.: Reinforcing the Learning of Agile Practices Using Coding Dojos. XP2010. pp. 379–380 (2010).
3. Sato, D.T., Corbucci, H., Bravo, M.V.: Coding Dojo: An Environment for Learning and Sharing Agile Practices. Agile 2008 Conference. pp. 459–464. IEEE (2008).
4. Aniche, M.F., de Azevedo Silveira, G.: Increasing Learning in an Agile Environment: Lessons Learned in an Agile Team. 2011 Agil. Conf. 289–295 (2011).
5. Martin, R.: The Clean Coder: A Code of Conduct for Professional Programmers. (2011).
6. Luz, R. da, Neto, A., Noronha, R.: Teaching TDD, the Coding Dojo Style. ICALT'13. pp. 371-375. (2013).
7. Ericsson, K.A., Krampe, R.T., Tesch-romer, C., Ashworth, C., Carey, G., Grassia, J., Hastie, R., Heizmann, S., Kellogg, R., Levin, R., Lewis, C., Oliver, W., Poison, P., Rehder, R., Schlesinger, K., Schneider, V.: The Role of Deliberate Practice in the Acquisition of Expert Performance. 100, 363–406 (1993).
8. Schön, D.A.: The Reflective Practitioner: How Professionals Think In Action. Basic Books (1984).
9. Thomas, D.: Code kata: How to become a better developer, codekata.pragprog.com.
10. Bache, E.: Test Driven Development : Performing Art. XP2009. pp. 217–218 (2009).
11. Schön, D.A.: Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions. Jossey-Bass (1990).
12. Moon, J.A.: Reflection in Learning & Professional Development: Theory & Practice. (1999).