# Pair-Programming from a Beginner's Perspective

Irina Tsyganok(✉)

YOOX NET-A-PORTER Group, NAP Commerce,
1, The Village Offices, Ariel Way, London, UK
irina.tsyganok@net-a-porter.com

**Abstract.** This experience report offers a beginner's perspective on pair-programming with experienced developers. It discusses issues faced by juniors and seniors when working together and highlights the importance of emotional maturity in pairs with disparate skill sets. This paper considers personal characteristics of junior and senior developers in identifying their needs from the pairing session and shares tactics used to improve pair-programming experience on individual and team-wide levels.

**Keywords:** Pair-programming · Knowledge-sharing · Collaboration · Culture · XP

## 1 Introduction

I joined YOOX NET-A-PORTER (YNAP) Group as a Technology Graduate in September 2014. During the 12 months of the company's graduate training programme I worked in the roles of a developer in testing, UX researcher, front-end developer and back-end developer, in multiple teams. As a junior, I was paired up with experienced developers to work on each story. It was the company-wide assumption that senior developers were the best candidates to introduce new team members to the technology stack.

Interestingly, most (if any) of my pairs had not practiced pair-programming in their daily work and working with me was, for many, the first exposure to pairing across skill levels. Having no framework to follow, we were largely guided by our instincts in conducting pairing sessions. It is through that experience I realised that social skills and emotional intelligence were powerful influencing factors in the success of pairing relationships.

My inspiration to explore Extreme Programming (XP) came from working with Nat Pryce, combined with support and insights from my manager. Nat introduced me to XProLo - a meet up on XP, which he attended along with other like-minded software engineers. Having become a regular member of the group myself, I have learned different ways of applying XP behaviours in the workplace and gained reassurance in my belief that pair-programming experience if approached appropriately, could benefit our team in many ways.


H. Sharp and T. Hall (Eds.): XP 2016, LNBIP 251, pp. 270–277, 2016.
DOI: 10.1007/978-3-319-33515-5_25

## 2 Getting Started with Pair-Programming

YNAP provided a consistently supportive learning environment across all teams I worked with. My input was always welcome and mistakes were treated as learning opportunities. I had the opportunity to join any team on any project at any time, which gave me complete control over my professional development. This autonomy allowed me to accelerate progress in areas, which I found most interesting and relevant.

But despite the thriving external environment, pair-programming with senior developers was much less of a success. It was not rare for me to feel frustrated, overwhelmed, disengaged and even insecure when pairing. Granted, a lot of these symptoms are a natural human reaction to facing a steep learning curve. However, six months into my role I repeatedly faced similar problems.

## 3 Getting Frustrated with Pair-Programming

> "If I were given one hour to solve the planet, I would spend fifty five minutes defining the problem and only five minutes finding the solution" – Anonymous, often attributed to Albert Einstein

When considering our issues with pair-programming, I found it useful to categorise the challenges we faced into three groups: physical, session management and social.

Physical challenges are concerned with physical comfort. They can take the form of unsuitable equipment or inadequate personal space and can result in poor posture and discomfort. Session-management challenges are interruptions to the session caused by developers without consideration of the schedule of their pairing partner.

For junior/senior pairs, the biggest challenge is frequent unavailability of senior developers. Social challenges are less tangible and are therefore, the hardest to deal with. They are dependent on the personality traits and emotional intelligence of both partners. Physical and session management challenges can be more easily resolved if the social challenges are eliminated first.

### 3.1 The Vicious Cycle of Non-learning

I discovered that most of my senior pairs were reluctant to allow me to experiment with solutions. At the slightest sign of uncertainty, they were very eager to take over the keyboard and demonstrate the solution by coding it themselves. Although instinctive and seemingly efficient, this tactic undermines the purpose of knowledge-sharing in pairing across skill. It is also easily developed into a pattern, which if becomes systematic, leads to the vicious cycle of non-learning, as illustrated in Fig. 1.

The pattern illustrated in Fig. 1 sets traps of short-term convenience at each step. It is more convenient for an experienced developer to type in the code than to watch their junior pair struggle through an imperfect solution. Similarly, watching someone writing code for prolonged periods of time almost always leads to disengagement. Breaking this pattern requires taking a step outside one's comfort zone, and can be difficult to achieve.

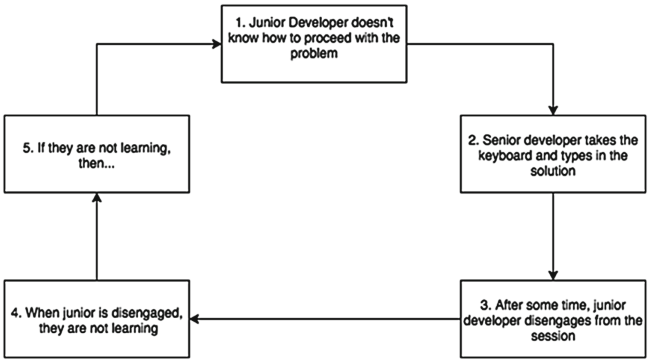


**Fig. 1.** The vicious cycle of non-learning

## 4 Observations

To investigate the relationship between social characteristics and levels of technical expertise in software developers, I drew up high level personas for the novice and the expert in the context of a programming session.

The diagram in Fig. 2 shows that behaviours of these personas oppose one another. For my analysis I chose the two extremes of professional spectre - the very junior and the very senior, as they most accurately reflect my experience at YNAP. However, despite obvious differences, junior and senior developers have one goal in common - they both want to learn in the course of the pairing session.

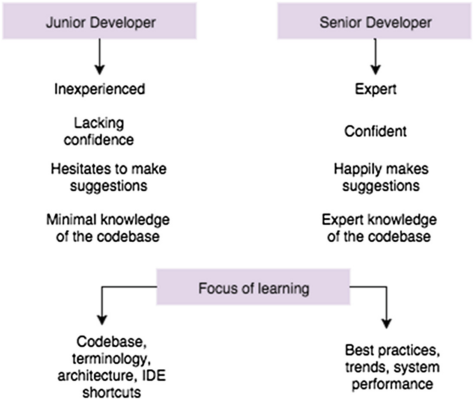


**Fig. 2.** Junior and senior developer personas

### 4.1 The Needs of Each Persona

Considering the disparate learning focus of the expert and the novice, I identified the expectations of each one from the pairing session. To analyse the needs of the novice, all I had to do was reflect on my own experience. And this is what I identified to be the most important:

- opportunity to experiment with solutions
- permission to make mistakes
- constructive feedback
- engagement in the session
- a flowing dialogue
- friendly disposition from my pair

In order to find out what the seniors need from the pairing session, I turned to my technical team at YNAP. I gave a presentation 'Pair-programming From a Beginner's Perspective', in which I shared my observations, concerns and proposed solutions with my team. When I asked my senior colleagues about their needs, we were all surprised to discover that the needs of experienced developers are identical to the needs of the junior.

I repeated the talk and the question at XProLo meetup in front of a very experienced audience, and the results were the same.

Through these discussions I learned that senior developers can also be prone to insecurity and that just like the juniors, they need opportunities to experiment with solutions and desire a flowing dialogue from their pair.

## 5 What We Learned

> "A teacher-student relationship feels very different from two people working together as equals, even if one has significantly more experience." - extremeprogramming.org

Industry perception of pairing across skill often assumes a *teacher/student* relationship. Such teacher-student role division promotes a familiar classroom teaching style, whereby the teacher talks and demonstrates and the student listens.

My experience has shown that pair-programming is most effective when both developers are equally involved and proactive throughout the session. Therefore, I propose to view pairing across skill in the light of the *leader/adopter* pattern. The latter suggests equal participation of both developers in the session, with the experienced developer acting as the leader and the junior, as the adopter. The skill set of the leader should include both technical excellence and emotional intelligence - one or the other alone is insufficient. The skill set of the adopter is incomplete, hence the role.

During my time at YNAP I have had to heavily rely on my social skills to facilitate my own learning. This led me to a conclusion that technical expertise of an individual does not imply emotional maturity; consequently not all senior developers are good leaders. A simple metaphor of a parent teaching their child to ride a bike might help. The parent tasked with teaching is expected to not only be a confident cyclist themselves, but to be also adequately patient and articulated to lead their child through learning experience. From the adopter's perspective, quality of pairing experience is heavily influenced by the emotional expertise of the leader. Continuing the cycling metaphor, the most effective teaching method involves a parent and a child working together as equals. In contrast, a parent demonstrating the technique by cycling around their child in silence, is obviously ineffective. Yet, when it comes to pair-programing, senior developers often choose the leading strategy of cycling around their junior pair.

### 5.1 Can Anyone Be a Leader?

Just like teaching a child to ride a bike, pairing with the junior demands patience, a teaching plan (that can change), time and willingness of the leader to get deeply involved. The latter is the deciding factor, yet in my experience it has not always been considered.

The cycling analogy demonstrates the importance of acknowledgement and consent. A child who wants to ride a bike has no choice but to master cycling first. The parent, on the other hand, has a choice of either teaching their child themselves or asking someone else to do it.

In making this decision, the parent has to analyse whether the amount of time they can spare for teaching, their own physical fitness and emotional skills needed to guide their child throughout learning experience are sufficient at a given point in time. If a parent identifies that for whatever reason they are unable to meet one or more of the suitability criteria, it is probably better to ask a friend or another family member to lead the teaching. To the child on the other hand, quality of teaching is more important than the person providing teaching.

The suitability and unsuitability of the senior are not permanent. Circumstances may change over time to enable the parent to teach their now eagerly racing child new stunts on their bike.

Coming back to pair-programming, in addition to acknowledging the fact that their pair is inexperienced, a senior developer has to asses whether they are willing and able to take on the role of a leader at a given moment in time.

### 5.2 The Social Aspect of Pair-Programming

I have learned that for two developers to work enjoyably and productively together, they need to get on well socially. This applies to all pairs regardless of technical expertise, but the experience may feel less natural in a junior/senior pair.

I observed that the act of pair-programming despite being emotionally intense, can be socially isolating. Work environment places focus on technical expertise and professional status and filters social interactions through a prism of organisational culture.

However, when two developers engage in an informal activity, such as having lunch or a drink after work together, their professional status matters much less and so does the organisational culture. Instead, the focus of their interaction shifts to personal and emotional. Those developers are no longer ‘a junior’ and ‘a senior’, they are just two people having a conversation. I have found that positive effects of informal communication transcend environments. That is, once the two developers get back to work, they find that their communication flows better, which in turn, empowers the pair to overcome challenges imposed by the experience gap, status and other constraints of the work environment.

### 5.3 Pair-Programming – Child's Play?

In the final year of my degree, I organised a Code Club at my local primary school, where I taught a class of seventeen children programming for one year, using a project-based curriculum and encouraging a free-form learning environment. My only two objectives throughout the year were that the children learn to code and have fun. Whether they wanted to work solo, in pairs or in a mob, was entirely up to them.

Throughout that year, I observed most organic transitions of my students' learning through various techniques.

All children started the year in the classroom learning style. They had a computer and a project sheet each and worked solo, raising their hand as they needed my help. As projects became increasingly complex, many children chose to pair up with their friends - in a grown-up XP world we call this 'pair-programming by association'. In their pairs, the children conversed freely, exchanging jokes and clearly enjoying each others' company.

However, as the difficulty of learning the material continued to increase, some children gravitated towards pairing with their more able classmates who were not necessarily their buddies outside of the classroom. Their conversations became more focused on the task, but the dialogue kept flowing at all times. In each pair there was a leader and an adopter, and both remained engaged throughout.

Finally and very importantly, two of my students chose to work solo throughout the whole year. They did not object to helping their peers when prompted, but they clearly performed better and seemed more content having their own space.

All children made amazing progress and many of them have taken their learning further. Reflecting on our experience, it is clear to me that the children loved the social aspect of our club just as much as they loved problem-solving. Another observation I made that year is that not everyone is happy in the role of leader despite displaying excellent technical aptitude. And that is a personal choice everyone should be entitled to. Unless these individuals want to step out of their comfort zone and take a leadership role, forcing them into pairing with a less experienced partner will never lead to a good experience.

The children gracefully demonstrated the significance of emotional intelligence in pair-programming. In comparison, as adults, we seem to have a harder time to effectively apply the practice in the workplace

## 6 Fixing Pair-Programming

"In theory, theory and practice are the same. In practice, they are not." - Anonymous.

During my time at YNAP I learned that communication is the foundation of successful pairing. Therefore, I focused on improving social interactions with my pairs and team-wide by including jokes, casual conversations and team socials into our day.

Specifically to pairing across skill, I gave a talk to my team, in which I offered a beginner's perspective on pairing with experienced developers. The talk drew attention to the challenges, which senior developers did not know existed and initiated interesting discussions in the team. Our improved communication enabled us to make sound team-wide decisions, which included:

- Swapping pairs to promote pairing fluidity
- Agreeing on the WIP limit and adhering to it
- Setting up our desks and pairing stations to provide comfortable working environment
- Giving each other sufficient time apart and synchronising our breaks
- Working on a story from start to finish, in the same pair
- Finding time for small talk in pairs and socialising more as a team

### 6.1 How We Can Improve

Pair-programming culture in our team has significantly improved as a result of the increased awareness and positive changes we made. However, we are still working on the assumption that all our senior developers are equally good at pairing with juniors. Going forward, I would like to adopt a more personalised approach in forming junior/senior pairs, particularly, when new graduates join our team. I would also like for us to pair with other stakeholders, such as designers, testers, product owners and data analysts more often.

## 7 Conclusions and Way Forward

Over the last six months, pair-programming in our team transitioned from being a subconscious training tool to becoming a considered cultural choice for everyone. Working in pairs brought us closer as a team, sometimes as opposing parties of long debates and sometimes as good friends truly collaborating and learning from each other. Even at this early stage of adopting the practice, pair-programming has formed a core part of our work ethos.

Our cultivation of a thriving pairing culture has not been smooth and it is far from over. After many frustrating sessions, it was the acknowledgement of the importance of good communication and emotional maturity in pairs that allowed us to make key positive changes in the way we work together. I am confident that a little more perseverance will take our team to new strengths in applying pair-programming effectively across skill and beyond.

This extract from an email, which Ward Cunningham sent to pdxruby mailing list in 2012 sums up this experience report beautifully: "…Our willingness to work together could be the juice that will push computers forward. We will all have to master pair-programming (not just mentoring) to make this work. It will be awesome".

**Acknowledgements.** I would like to thank Nat Pryce for the inspiration, support and honest feedback. I appreciate the opportunity to work on the report and attend XP2016, presented to me by Claire Lamb and James Wyllie. Thank you to all my senior pairs for providing the material for this paper. Finally, a special thanks to Avraham Poupko for having faith in this project, and for his undeniable support and valuable insights at every stage of its life.