Towards the Better Adoption of HCI Methodologies by Technology Startups

Roger Puks, Zahhar Kirillov
Tallinn University, Institute of Informatics, Narva Rd. 25, 10120 Tallinn, Estonia
Estonian Entrepreneurship University of Applied Sciences, Chair of Creative Industries, Suur-Sõjamäe str. 10A, 11415 Tallinn, Estonia

E-mail(s): roger.puks@tlu.ee, zahhar.kirillov@eek.ee

Abstract. *This paper means to set the grounds for figuring out the current situation of the use of Human-Computer Interaction principles and User Experience methodologies by technology startups. To begin with, a speculation is made, why UX design tends to be left out of the development cycle. Based on that, the article discusses different ways to implement UX-design heuristics in the field of agile sofyware development, that require future study. The final aim would be to propose a new framework that makes effective coexistence of agile methodologies and scientific UX-engineering possible.*

Keywords. HCI, agile methodologies, startups.

1. Introduction

Agile methodologies are becoming more popular by each passing day and are widely considered to be rather suitable for different tech-startups, mobile- and even game development. The main reason these methodologies have become so popular resides in the fact that they require less resources to kick-off the project, less time to deliver the initial outcome and are more customer-oriented than other fundamental methodologies. The customer is kept posted during the whole development cycle, he can interact with the team and provide feedback just-in-time, keeping development aligned with his business goals. Be it Scrum, Kanban or Extreme Programming, they all share common values: to satisfy all the business requirements of a stakeholder, while using up as little resources as possible. While classic methodologies, such as the Waterfall Model, allow the customer to see the outcome of the project only in the final stages, agile methodologies on the other hand, can produce their first sensible results by the end of the first week [1].

This, however, places interaction designers in a difficult position. How can one, barely having enough time for product development, find even more time to deal with usability issues? Does this mean that UX should be forgotten about and every member of the development team should just rely on his own knowledge? Or perhaps we should finding solutions on how to include UX-design without requiring too much time and effort would be more appropriate?

2. Is there a space for UX-design?

To begin with - why do agile development teams tend to leave out UX-designers? Usually it all starts with the client. The two top-most priorities for the customer are to limit the *time-to-market* as much as possible, and to adjust the application as the current business situation requires it to be adjusted. That usually means that locking all the software requirements for the development period is very difficult, if not impossible. Flexibility during the whole project is a rather important prerequisite for a decent application-in-development [2]. That often leads some teams to the thought, that user interface design and usability testing use up lots of resources but give too little in return. Why create a bottleneck effect by saying 'yes' to UX, while the result may not even last long?

Claiming to be user-centered would obviously lead one to the conclusion that UX-design has an important role in the whole development process. Unfortunately that is not the case. When one talks about an agile software development team, most of the team usually consists of software developers. In addition to leaving out architects, project managers and business analysts, agile teams tend to sacrifice most that relates to software design and testing as

well, decreasing the quality of the product. Thus, some agile methodologies have one grand downside: they do respect software developers and stakeholders, but not the end-user for whom the product is being developed [3].

Besides all that, user interface design and application programming can vary on the time they require. Although both processes consist of iterations, the length of these stages can differ tremendously. While an agile developer can come up with a sometwhat working version of the application by the end of each day, that's not even enough time for an UX-designer to get an user-validated prototype of an interface [4]. Nevertheless, if a project has a fixed deadline, prolonging that only causes more economic loss for the customer. What's there left to do?

3. Bridging the Gap

Changing the way developers look at UX-designers would probably be a good start. Instead of thinking of them as unnecessary obstacles to a fast deployment cycle, they should be viewed upon as allies. There's always the possibility of educating every single back-end and especially front-end developer in the field of UX design. Each member of the team can fill himself in with the latest domain practices and trends, be aware of technological possibilities and constraints, as well as getting to know some heuristics and how to use them.

In addition to having your development team educate themselves concerning UX basics, one could plan actual UX-testing in parallel with planning each development phase. Good groundwork is essential at this point. While the rest of the team is working hard on planning the next iteration, the UX-designer could find the people he'll be testing the software on.

Now assuming we have a list of participants who have agreed to help with testing. The main purpose of usability testing is to discover the ways in which the proposed interface makes it hard for users to reach their goals. This takes time, as regular usability testing requires at least five to twenty participants [5]. But does that rule also count when dealing with agile methodologies, or would three participants suffice as well? If two of them fail to accomplish the task, the proposed solutions should be improved. Agile testing shouldn't find all possible problems, it should rather find critical usability-errors that need to be fixed. Having no more than three test subjects allows the UX-designer to go through with the scenario in no more than a single day. Our experience shows that testing different short scenarios on different people is usually more productive than testing a single complex scenario on a larger number of people at once. The same goes for re-testing improved user interfaces: a series of tests with early mock-ups works better than one large-scale test of a finished product.

Finally, why not let the developers themselves go through with the testing phase? Letting a developer act as a spectator and later a moderator, for an UX-testing session, can dramatically improve his sense of the product in development. In addition to that, the developers can fix all the newly-emerged deficiencies before the next stand-up meeting and present their findings and achievements to the team. How can this be effectively done is a question of further research.

4. Conclusion

One thing is for certain: agile methodologies and user-centered design have both consolidated their positions in the field of software engineering and that situation – should the absence of better alternatives continue – shall most likely remain the same for years to come. In order for the outcome to be as perfect as humanly possible, both should find a perfect way to collaborate with one-another. There is still no consistent framework, respected both in academic circles as well as by field practitioners, about how a product can benefit from the cooperation of agile developers and UX designers.

Currently the authors are about to conduct a thorough survey among startups to explore their experience in adopting latest UX methodologies in their agile development process. The primary intention is to uncover problems and to propose such a framework but in addition to that, conclusions on whether UX and agile methodologies ultimately belong together, will also be made. The emphasis of this research would be on user experience and its heuristics since focusing on HCI would mean having to comprise a much wider amount of topics, thus possibly making the final paper substantially vague.

5. References

[1] Conboy K, Fitzgerald, B. Toward a conceptual framework of agile methods: a study of agility in different disciplines. ACM New York, 2004. p. 37-44.

[2] Reifer, D.J. How good are agile methods? IEEE Journal on Software 2002, vol. 19, issue 4, p. 16-18.

[3] Sharp H, Keynes M, Biddle R, Gray P, Miller L, Patton J. Agile development: opportunity or fad? Extended Abstracts on Human Factors in Computing Systems. ACM New York, 2006. p. 32-35

[4] Ungar J, White J. Extended Abstracts on Human Factors in Computing Systems. ACM New York, 2008. p. 2167-2178

[5] Bevan N, Barnum C, Cockton G, Nielsen J, Spool J, Wixon D. The "magic number 5": is it enough for web testing? Extended Abstracts on Human Factors in Computing Systems. ACM New York, 2003. p 698-699