CHAPTER 11

Dark Agile: Perceiving People As Assets, Not Humans

Pernille Bjørn, University of Copenhagen, Denmark

Revisiting the Agile Manifesto

The agile principles for software engineering were developed as a reaction against structuring software engineering processes in strict stepwise and sequential ways. The idea that it was possible to create a clearly predefined scope prior to the actual software engineering activities was questioned—and the agile methodology was an attempt to rephrase the basic nature of software engineering. The agile understanding of software engineering is that the fundamental nature of software means that we cannot predetermine scope, goals, and objectives up front. Instead, goals, scope, and objectives are transformed throughout the software development process. This setup requires participants (developers and clients) to balance and negotiate resources and priorities, and this is what drives agile development. Agile development is not one thing but can instead be seen as a set of principles that guide the organization of work and can be implemented in different ways. The main principles provided by the agile manifesto (http://agilemanifesto.org) are as follows:

- Individuals and interaction over processes and tools
- Working software over comprehensive documentation

C. Sadowski and T. Zimmermann (eds.), *Rethinking Productivity in Software Engineering*,
https://doi.org/10.1007/978-1-4842-4221-6_11

- Customer collaboration over contract negotiation
- Responding to change over following a plan

These agile principles are based upon the main idea of providing the power over software engineering to the people—the software team. Instead of letting software developers be controlled from the outside, the software teams are to be empowered to find and prioritize their own work. The software team is to be a self-organized team, and the client or customer is to be part of the team supporting the prioritizing of tasks based upon available resources. When we, in computer science departments at Danish universities, teach computer science students about software engineering, we talk about the benefits of agile development and the problems with the waterfall model. We explain how the waterfall model does not take into account the iterative and creative process of developing software. Furthermore, if you visit any kind of Danish IT company and talk to the developers and ask them about methods, they will tell you how the waterfall model does not work and how agile methodologies provide better quality within an appropriate time frame. Agile is seen as a positive perspective on software engineering in Denmark.

However, the story about agile is quite different when we change perspective from Scandinavia and turn to India.

Agile in Global Outsourcing Setups

Based upon a long-term research project called Next-Generation Tools and Processes for Global Software Development (NexGSD; `nexsgsd.org`), we have studied how global software development takes place in different places around the world. Concretely, we went to observe and interview software developers in the Philippines about their experiences working with software developers in Denmark [4, 5, 7], and we also went to India, more concretely Bangalore, Mumbai, and Chennai, to observe and interview software developers about their experiences collaborating with software teams and vendors located in Northern Europe and the United States [6, 8, 11, 12]. Throughout all these empirical studies, we began to notice the consequences of implementing agile principles such as scrum methodologies in global outsourcing setups. We witnessed a transformation in the way global software development was organized between 2011, when we started the project, until 2014, where all the organizations we studied went from waterfall models toward agile models [1, 2].

So, what does this mean? Let's take a closer look at the experience of agile development seen from a software developer working out of India in one of our empirical case studies between Bangalore, India, and Phoenix, United States [3].

Global software development can at a high level be organized as outsourcing or off-shoring. Outsourcing is when you move work from one internal location toward an external partner, who then does the work for you. Differently, global off-shoring is when work is moved to a different location, but still within the same company—like IBM USA working with IBM India. In our empirical cases, we are looking at global out-sourcing, which means that work is moved from either the United States or Denmark to a different geographical location and a different organizational setting.

In outsourcing setups, it is important to note that the power remains with the client. This mean the client chooses which company is doing the work, and deciding to move work to other outsourcing vendors (still in the same region of the world) is always an option. In one of our cases, the U.S. client put together a global agile team comprised of experts from different IT vendor companies in India and then one representative from the client was the project owner. This meant that the team members, even being in the same team, were simultaneously in competition. The client was able to exchange specific members with new people if particular individuals were not performing well accordantly to the client. This multivendor setup created a high-performance team, which despite being geographically distributed was highly productive. The global agile setup raised the competition among the team members, and from a productivity perspective, this was a huge success. But how did the agile principles—concretely manifested in the scrum methodology—impact the global outsourcing team?

Tracking Work to Increase Productivity

One of the main processes in scrum is that members of the team specify what they are currently working on, directly linked to specific numbers of hours. How many hours specific tasks might take is up to the team members, who negotiate the resources required during planning. In this way, each team member is tasked with assignments to be accomplished and finished within detailed time frames. In India, the workday of software developers is ten hours. In all software projects, some hours will be spent on other activities than directly on the project. Therefore, the hours that are tracked are eight hours a day. This means that each day, each team member is committing to produce software tasks resembling the work of eight hours. Thus, regardless of what

might happen, each team member must produce the task assignment. Even if their child gets sick and they need to leave the office, they cannot. They have to stay on task and complete the task as planned or else their client might move the task to a competing IT-vendor company (still in India). Interestingly, the software developers working in Bangalore explained to us how they prefer waterfall over agile. Waterfall had less time pressure since they had a specific target—and longer deadlines, which made it possible to pick up a sick child if needed, rather than being constantly pushed by short deadlines.

Daily Stand-Up Meeting to Monitor Productivity

Besides agile allowing clients to constantly track the productivity of each individual team member, global agile also forced team members to participate in daily stand-up meetings. While the stand-up meeting alone was not problematic, the time of day for the meeting was. Because of the time difference between the East Coast in the United States and India, the time for stand-up meetings were set to late evening (10 p.m.) Indian time. This was regardless of the day of the week—so all days including Friday, there were stand-up meetings in the evening. This meant that team members involved in global agile outsourcing were forced to work out of sync locally to accommodate global work. Working out of sync locally is problematic in terms of family life or social events, especially in situations where the software developers had their families in villages far away. Several developers we spoke with moved to the electronic city of Bangalore during the week and then traveled back on the weekends. The stand-up meetings made it difficult to travel home Friday evening. Furthermore, the tenure of the projects changed from being four- or five-month-long projects to being more than a year. This provided constant pressure on the software developers; there was no time for breaks or vacations. The high level of productivity for the extended time led to a stressful environment.

Stressful Work Environment

Over the three years we conducted interviews, it became apparent that, while the global agile team had high productivity and was the preferred IT vendor for the customer, the software developers working in the global agile setting felt “more pressure, more time pressure, stress” and the experience of agile methodology was that it “is very stressful, at the tester level.” It is important to note that while it can be expected that people in higher

positions working in global projects be available at odd times and work many hours, the people working under pressure in this situation were the developers and testers working in low-level positions. The way global agile was implemented meant that the customer pressured the team on speed constantly—so even though agile principles stipulate that the ideal sprint size is two to three weeks, the customer pushed it down to one week. Analyzing, designing, implementing, and testing workable deliveries within five days of work is hard, especially for the testers. As a delivery manager explained to us: "Yes, for the techies, or for the technical department, it is a very stressful, stressful methodology I would say because the expectation is too high from the customer's side."

Cost of Productivity

There is no doubt that the IT vendor we studied was highly productive in terms of speed and quality, delivered good quality work on time, and was the customers' preferred IT vendor, even in the competitive multivendor setup. As the preferred IT vendor, they gained more tasks, especially in situations where other vendors were not able to deliver. Now the question is, what was the cost of this high productivity?

Financially, global agile is more expensive than waterfall methods for the customer: when talking with the IT vendor, it was clear that they were able to produce the same kind of products much cheaper under the waterfall methodology. The argument for global agile as a way to save costs, which are often a fundamental problem in global software development [10], was not on the agenda. When we asked the IT vendor why they were using agile principles in the first place, they explained that it was a request from the customers: the customers wanted the vendor to use scrum. Let's take a step back and reflect on this request from the customers. When you, as a company, are hired to deliver a service or a product, negotiations about the price, timeline, and collaboration are to be expected. Clients direct requests for how the vendor is to use specific methods are less obvious. So, why did the client request this? Despite it being a more expensive methodology for the client, they gained direct access to highly qualified people, who all had proportionally high salaries (though the IT vendor then had difficulty including and training new people to work on the projects).

What about the human costs of this high productivity? What happens to people when agile goes global? If we return to the principles in the agile manifesto, we find that the principles of "working software over comprehensive documentation," "customer collaboration over contract negotiation," and "responding to change over following a plan"

are all very pertinent in the global agile outsourcing setting as well. In our case, there was close collaboration with the customer, the scope and objectives were a moveable target, and there was a constant focus on working software deliveries. However, if we look at the first principle of "individuals and interaction over processes and tools," we see a shift. The processes and tools created to structure the agile delivery were used to micromanage the software developers' work in all the small details. We can view the global agile principles in our case as an algorithmic machine, with specific input and output features. The input measures are the numbers, the hours, and the deliverables deadlines, which are then used to push people to maximize their efforts. Given the tools and processes of agile, the remote client is able to monitor and control every little aspect of the work done by the software developers. Sure, global agile is very productive. If the only criteria for success is high-quality work done fast, global agile is attractive.

Nevertheless, there is a dark side to global agile, since in the case of scrum comes tools and processes that can be used to micromanage software developers. Focusing only on productivity, we risk losing sight of individuals and the "mushy stuff" that is at the core of the agile ideals. According to Jim Highsmith for the Agile Alliance, "At the core, I believe agile methodologists are really about the 'mushy' stuff about delivering good products to customers by operating in an environment that does more than talk about 'people as our most important asset' but actually 'acts' as if people were the most important and lose the word 'asset'" (http://agilemanifesto.org/history.html).

I that we must consider the conditions for work created by the constant focus on productivity introduced and controlled by agile tools and processes. This risk of the "global agile algorithmic machine" is that it turns people into assets, resources, and numbers—and we lose sight of individual developers. While waterfall methodologies have been criticized for heavily regulating work and introducing micromanagement, our empirical observations point to how the global agile methodology can also be used for micromanagement and strong regulation of software developers.

Global agile provides good conditions for high productivity in software engineering but also these risks:

- Perceiving people as assets, not human beings
- Creating stressful work environments in continuous work cycles
- Supporting clients in micromanagement from afar
- Making developers and testers work out of sync with their local time zones

What we risk losing is the focus on the software developers and the self-organization and empowerment that are supposed to be introduced with agile methodologies. Software engineering organized by global agile methodologies in highly competitive multivendor settings risks resembling the assembly line in factory work. Is this really what we want the future of software engineering to look like?

Open Questions for Productivity in Software Engineering

I am not arguing that global agile is problematic per se. Clearly, in all the NexGSD empirical studies, closely coupled collaboration was essential to get that collaboration to function across sites, and the agile principles enable and stipulate closely coupled collaboration. However, I am arguing that "being a software developer involved in global outsourcing" means different things depending on where you physically are located in the world. Software developers at low-level positions working in Bangalore, India, have different conditions for work than software developers working in Ballerup, Denmark [9]. This means that they will experience the implementation of global agile in different ways. Software engineers located in Denmark have a privileged position in the global setup. For software engineers located in India, the way global agile techniques, tools, and processes shapes work do not provide the same conditions for self-organization and empowerment. Moreover, it means that when we are designing software tools and processes to support global work, we should take into consideration the different conditions and not just focus on productivity. Fast delivery and high-quality code should not be our main measurements; instead, we should start to develop measurements that are more nuanced and take into consideration work conditions. We must think about how artifacts such as "burndown charts" reflect only partial aspects of productivity [10], and we should ask, what is not represented in such artifacts? What are artifacts and tools neglecting to make visible? Finally, we need to consider how to ensure that we do not lose our human values when we think about how we design tools and processes and create good work conditions for all, no matter where in the world they are placed. People work more and more in the global setting; and as life and work starts to blend due to us bringing home our laptops and continuing checking e-mail in the evenings and on weekends, we need to prepare long-term strategies for dealing with the pressure of productivity—even for low-level software developers and testers working in India.

When software developers complain that they have to attend a meeting at 10 p.m. and are not able to leave work to pick up sick children, they are not complaining about agile development per se. Instead, they are complaining about the lack of power and decision-making within the organizational setup. Agile development works well for software developers in Scandinavia, Northern Europe, and United States because the software teams are powerful and privileged. When clients demand agile development from software developers elsewhere, those developers are not empowered. Instead, the power to choose and organize their work is taken away from them. The following are important questions we must ask:

- What kind of productivity and values do we want software engineering to reflect?
- How do we ensure that these values are manifested in our productivity measurements shaping software engineering processes and tools?
- How can we design software engineering practices and technologies to support productivity without losing human values?

Key Ideas

The following are the key ideas from this chapter:

- Global agile software development has several risks: perceiving people as assets, not humans; creating a stressful work environment; micromanagement; and making engineers work out of sync with local time zones.
- Productivity measurement should be about more than speed and quality.

Acknowledgments

This chapter is based upon the academic research paper co-authored by Pernille Bjørn, Anne-Marie Søderberg, and S. Krishna titled “Translocality in Global Software Development: The Dark Side of Global Agile“ and published in the journal of Human-Computer Interaction [3]. Further, the work referred to is part of several subprojects

in the NexGSD research project (nexgsd.org), which was financially supported by the National Council for Strategic Research, Ministry of Science, Innovation, and Higher Education in Denmark.

References

[1] Bjørn, P. (2016). "New fundamentals for CSCW research: From distance to politics." Interactions (ACM SIGCHI) 23(3): 50–53.

[2] Bjørn, P., M. Esbensen, R. E. Jensen and S. Matthiesen (2014). "Does distance still matter? Revisiting the CSCW fundamentals on distributed collaboration." ACM Transaction Computer Human Interaction (ToChi) 21(5): 1–27.

[3] Bjørn, P., A.-M. Søderberg and S. Krishna (2017). "Translocality in Global Software Development: The Dark Side of Global Agile." Human-Computer Interaction 10.1080/07370024.2017.1398092.

[4] Christensen, L. and P. Bjørn (2014). Documentscape: Intertextuallity, sequentiality and autonomy at work. ACM CHI Conference on Human Factors in Computing Systems Toronto, ON, Canada, ACM.

[5] Christensen, L. R., R. E. Jensen and P. Bjørn (2014). Relation work in collocated and distributed collaboration. COOP: 11th International Conference on Design of Cooperative Systems. Nice, France, Springer.

[6] Esbensen, M. and P. Bjørn (2014). Routine and standardization in Global software development. GROUP. Sanible Island, Florida, USA, ACM.

[7] Jensen, R. E. and B. Nardi (2014). The rhetoric of culture as an act of closure in cross- national software development department. European Conference of Information System (ECIS). Tel Aviv, AIS.

[8] Matthiesen, S. and P. Bjørn (2015). Why replacing legacy systems is so hard in global software development: An information infrastructure perspective. CSCW. Vancouver, Canada, ACM.

[9] Matthiesen, S. and P. Bjørn (2016). Let's look outside the office: Analytical lens unpacking collaborative relationships in global work. COOP2016. Trento, Italy, Springer.

[10] Matthiesen, S. and P. Bjørn (2017). "When distribution of tasks and skills are fundamentally problematic: A failure story from global software outsourcing." PACM on Human-Computer Interaction: Online first 2018 ACM Conference on Computer-supported Cooperative Woek and Social Computing 1(2, Article 74): 16.

[11] Matthiesen, S., P. Bjørn and L. M. Petersen (2014). "Figure Out How to Code with the Hands of Others": Recognizing Cultural Blind Spots in Global Software Development. Computer Supported Cooperative Work (CSCW). Baltimore, USA, ACM.

[12] Søderberg, A.-M., S. Krishna and P. Bjørn (2013). "Global Software Development: Commitment, Trust and Cultural Sensitivity in Strategic Partnerships." Journal of International Management 19(4): 347–361.