The structuring of freedom in agile development

Completed Research Paper

Erica Wagner
Ahlbrandt Professor of Management
School of Business Administration
Portland State University
elwagner@pdx.edu

Sue Newell
Professor, Information and Process Management
Bentley University, MA
Warwick Business School, Warwick University, UK
snewell@bentley.edu

Neil Ramiller
Professor of Management
School of Business Administration
Portland State University
neilr@pdx.edu

ABSTRACT

Management literature on agile software development has largely focused on the relationship between agile methods and behavioral control, particularly the role that control plays in fostering flexibility. Although the focus on control has helpfully illuminated the constraints necessary in agile projects, it has provided only a half-finished portrait of the importance of structure in this approach to software development. This paper deconstructs the customary oppositions between structure and flexibility, control and freedom, by examining the socially-governed and technologically-mediated practices that produce and sustain agility in an organization that is an exemplary practitioner of the agile approach. We find that an imbrication of social convention and material agency provides the kind of structure that nurtures those markers of 'freedom' – flexibility, creativity, improvisation, and adaptability – for which agile is touted.

Keywords

Agile development, materiality, sociomateriality, qualitative.

INTRODUCTION

In recent years systems development has shifted away from structured, sequential and time-bound strategies for creating software (often referred to as the Waterfall method) toward iterative approaches that focus on frequent turn-around of software code and design features. This shift in practice has been attributed to the inability of sequentially planned approaches to 'handle rapidly changing environments' (Highsmith 2002), a condition that increasingly characterizes modern organizations. Among approaches falling into the category of iterative development are methods commonly referred to as 'agile', which focus on collaborative development where requirements and solutions evolve over time. The methods are predicated on multi-participant teams that rapidly develop and deploy tested code. The team is able to quickly 'pivot' the code itself, along with the direction of development, when change is necessary. Such teams commonly include the customer for whom software is being developed. The customer sees the work on a frequent (even daily) basis, yielding a short feedback cycle that readily accommodates evolving priorities, as both the customer and developers learn more about the application. This keeps the cost of change low by preventing unnecessary work. Also, it allows for new insights to be gained from the work already completed, and for those insights to be incorporated into the new code. In short, while the developers do not make promises about deadlines and functionality, they do spend time intensively with the customer in matching software capabilities to the real-world requirements.

Scholarship on the process of agile development has to date emphasized the way that control measures shape the approach. This research has helpfully illuminated the constraints necessary in agile projects, but it has provided a half-finished portrait of the importance of structure in creating the desired qualities of flexibility, creativity, and adaptability. This paper attempts a more complete analysis by deconstructing the customary oppositions between structure and flexibility, control and freedom. We examine the socially-governed and technologically-mediated practices that produce and sustain agility in an organization

that is an exemplary practitioner of the agile approach. We find that an imbrication of social convention and material agency provides the kind of structure that nurtures the 'freedom' for which agile is touted.

LITERATURE TO DATE

Although agile methods are of recent advent, their adoption is increasingly prevalent (Laanti, Outi and Abrahamsson, 2011) and a Forrester Research (2010) report states that agile is now the primary software development approach used within organizations. On the other hand, IS scholarship on agile methods remains sparse (see the review by Dyba and Dingsoyr, 2008), provoking calls for further research (Cao and Ramesh, 2007), especially rigorous studies that seek to develop and extend theory about the management of agile projects (Agerfalk, Fitzgerald and Slaughter, 2009).

To this end, Conboy (2009) has created a theory that links agile software development to agility in other fields. Others have developed a theoretical model of coordination and shown how dynamic agile software development projects organize (Strode, Huff, Hope, and Link, 2012). Sarker & Sarker (2009) analyze agility in information systems development by presenting a multi-faceted concept that includes resource agility, process agility and linkage agility. They later assess the relative importance of these facets in the success of a distributed systems development project, and show that people, technology, work transition, temporal and methodological factors support various aspects of agility (Sarker et al., 2009). Meanwhile, empirical studies have begun to lay a foundation for understanding the practice of agile development in use, including consideration of the culture and community of agile development (Sharp and Robinson, 2004), the effectiveness of alternative modes of control (Maruping, Venkatesh and Argarwal, 2009), and the relationship of control to flexibility (Harris, Collins, and Hevner, 2009). Meanwhile, other studies have focused on the tools and techniques that are instrumental for agile project work, and have documented organizations' adaptation of methods to fit specific circumstances (Fruhling & de Vreede, 2006; Fitzgerald, Hartnett, & Conboy, 2006).

In short, theoretical treatments have so far unpacked the concept of agility and identified in broad terms how human and non-human components of a project support agility. Simultaneously, empirical studies have begun to identify specific cultural elements, control regimes, tools, and techniques that can contribute to the operationalization of an agile program. However, a major task that still remains is to explain more fully how *in practice* the material and social facets of a development environment can provide the 'format and furniture' (Pollock and D'Adderio, 2012) that foster the freedom and creativity for which agile is known.

CONCEPTUAL FRAMEWORK

Our discipline's historical preoccupation with control is natural, considering how in so many IS projects control has been lost, yielding outcomes like spiraling costs, overrun time lines, and scope creep. On the other hand, agile development, given its values and ambitions, invites us to take a fresh look at the conditions of software development. Tracing the freedom and creativity attributed to agile methods calls for attention to the *affordances* of agile practice as well as the constraints. Looking at agile development from a practice perspective (Schatzki, Knorr Cetina and von Savigny, 2000) leads us to consider how material and social arrangements work in concert to create *the power* for practitioners to act. In adopting a practice lens we are focused on how individuals never act alone but are rather 'propped up and aided' by all kinds of material artifacts as well as by social norms and conventions (Pollock and D'Adderio, 2012). Indeed, the very substance and meaning of individual action is relational (Orlikowski, 2007), and depends on structured interactions with space, materials, tools, and people to give it shape and direction. Both the individual's power to act and the range of options are therefore immanent in an environment comprised of sociomaterial affordances.

The aim of our research, accordingly, is to examine how the social and material features of agile software teams afford, in practice, the kinds of freedom and creativity that are sought through the agile methodology. (We intend to sidestep, here, the current debate about the ontology of sociomateriality (Leonardi and Barley, 2010) but will, for practical discursive reasons, speak for the most part separately about social conventions and material affordances). In examining this question, we need to begin with the kinds of flexibility agile methods are seeking. For this we turn to *The Manifesto for Agile Software Development*, a seminal document that was written in 2001 by seventeen software developers gathered together to define less rigid and sequential approaches to development (www.agilemanifesto.org). The Manifesto identifies a set of four values: *individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; responding to change over following a plan.* The point of these values is that they favor freedom and creative accomplishment over evaluative measures and documentation. Our aim, then, is to examine how social conventions and material affordances allow each of these values to be enacted in the agile software development projects that we have studied.

METHODOLOGY

Our inquiry aspired to develop an understanding of agile software development as a constellation of socially governed and technologically mediated practices aimed at producing the specific kinds of flexibility entailed in the Manifesto's values. As such, our field research was designed to access the material and social agencies that enable the work practices of agile developers. Empirical observations were based on an on-the-ground exploratory study of one company renowned for agile software development. This ethnographic approach to understanding work practice is recognized as an effective approach to investigating research issues in context (Schultze 2000).

Research Site

Pivotal Labs (PL), founded in 1989, is known as an exemplar of agile-based software development. It boasts sophisticated technical consulting experience in consumer web and mobile application development and enterprise automation. Customers have included Twitter, Groupon, Best Buy, Salesforce.com, and Citysearch. PL also created and maintains a web-based agile project management software called Pivotal Tracker. Headquartered in San Francisco, PL has regional offices in New York, Boulder, and Singapore. Recently PL was purchased by EMC Corporation to help position PL for larger, enterprise-wide agile projects. The case organization was chosen for several reasons. First, it is recognized by the software development industry as a leader in agile development. Second, the organization is exclusively an 'agile shop' and has the luxury because of its reputation of only working with customers who understand what this means in terms of development. Third, the organization was receptive to the research and provided access to its customers, employees, and executives at multiple locations.

Data Collection and Analysis

Direct observation and interviewing occurred in May and September 2012. The first author made two visits of four days each to PL offices to observe particular work practices. Direct observation formed the bulk of this field research, including observation of multiple pairs of programmers working together and tours of multiple offices. Video was taken and notes were made on the use of artifacts (whiteboards, note cards, computer screens, and so on); entrances, exits, and movement in relation to work-spaces; and non-verbal cues. Semi-structured interviews were also conducted with employees and customers. All of the interviews were tape recorded and transcribed.

After the fieldwork was complete, a systematic and careful reading of the transcripts facilitated analysis of the interrelationships between everyday practices, organizational structures, and agility outcomes. We considered data reflecting behavioral norms governing how, when, and where to use particular technologies. We identified evidence of programmers and customer representatives reproducing agile practices on an on-going basis, and reflected on the role in this accomplishment of technology and other material artifacts. In our analysis we framed this role using the vocabulary of *affordances and constraints* (Gibson, 1979; Hutchby, 2001), where we identified properties of things and the boundaries of their use (Hutchby, 2001, p. 9), as well as how people used them under particular conditions of the local context (Pollock and D'Adderio, 2012).

CASE FINDINGS

In reporting our findings, we begin by identifying the different social conventions and material affordances – the *format and furniture* (Pollock and D'Adderio, 2012) – that support the work. Then we link these conventions and affordances to the central values of agile development. Given space limitations, we describe just one area of activity here – the *pair programming* process. Programming in pairs, although not adopted in all agile regimes, is central in PL's agile practice and serves to illustrate the manner in which the apparent constraints of structure generate the freedom that empowers an agile strategy.

Social Conventions

The practice of pair programming involves two developers working as a team on coding. One 'drives' by doing the actual coding and testing and the other watches and talks. They trade command of the keyboard back and forth for 7-7.5 hours of the workday. Despite the appearance that this involves doubling of resources for each programming task, pair programming helps to provide focus, foster learning, and ensure high quality code:

> *"Arguing on solutions, we do that always. That's healthy. We need to do it. Because if we don't do it, then we're just going to blindly accept someone's solution without even questioning like whether it is a good solution or not. You don't do a great product doing that." [Pivot#1, Boulder]*

Pivots (as PL employees are known) talk about how pair programming increases the number of people on the team who need to be run over by a bus in order for the project to fail or incur a serious cost or delay. Pairs typically rotate on a daily basis, which removes constraints (and hence builds freedom) at the project level both because the individuals with knowledge of the code are more numerous and because team members develop a holistic view of both the code and the project goals:

> *"… On our current project we started with a new technology that only two people in the project had had experience with out of six developers…The two people worked together for a little while on it, built a base and now I think everybody is on the exact same plane with working with this tool and technology because you're pairing every day and you have to share knowledge. … Because we rotate pairs and because we're constantly talking through problems and solving them together, even if someone is like the expert at this technology that we adopted in the beginning, by the end, there's no expert." [Pivot#2, Boulder]*

Many Pivots spoke about the intensity of coding in a pair for 95% of the work-day and expressed the exhaustion they felt the first few weeks after being hired. They also expressed the benefits of that cultural norm:

> "So another important part of Agile is try to keep it to ideally a 40-hour week schedule and not – I mean every project goes in times where you have to do a little bit more but…most of the time we come in whenever our stand-up time is and…nine hours later, we leave…We also kind of advocate an hour out for lunch and a lot of that is not just so we're here longer, it's just that when you're pairing, it's a little bit more intense. So that break to kind of *clear out the thought* really helps the quality of the afternoons." [Pivot#3, Denver]

Another informant, reflecting on that intensity, curiously likened the team dynamics to being under military discipline:

> *It's better to be part of a team. And this goes back to like the military and why a lot of veterans can't adjust to normal life anymore after being part of a tightly knit team in Afghanistan, in Iraq…That's part of what people get at Pivotal, is like they become part of a pack and those packs form very quickly through pairing and some of the other kind of communication pieces that we have in place, where the feeling of being in a pack and having success and facing and meeting challenges as a team, getting through them and then succeeding, that's actually much better – that's more rewarding than that feeling of 'I'm an individual with specialized skills and people need me'. It's more powerful than that. [Pivot#4, Denver]*

The practice of pair programming also helps the customers learn their own code base. Many times the customer representative, who is on-site 40 hours a week (which is a contract requirement), is an IT professional who pairs with the Pivots and is seen as just another member of the team. In addition, the "product owner" is employed by the customer and has the ultimate authority and responsibility for the project's direction.

Material Affordances

The PL offices are open plan and quite modern. One is likely to see Pivots organized on the office floor in small groups staring intently at computer monitors and talking. Pivots do not have their own machines and instead select from any open machine on the development floor. Project teams locate in an open space large enough to accommodate all members and to facilitate frequent rotation of programming pairs and regular communication across team members. During the day you will see team members get up for short breaks. One or two will be wheeling around on scooters as a way to move through the office, or taking a few minutes for a game of Pivot Pong. Having scooters and ping pong tables allows developers to blow off steam and then return to coding with fresh eyes. So, too, does the fully stocked kitchen. Pivots are given everything they need in-house to support their work environment and thereby remove impediments to productivity. For their own part, Pivots see these material affordances as supporting their work.

The pair-programming stations include one 27-24" flat iMac and 2 keyboards. These desktops all have the same configuration and are reimaged after each project ends. There are no laptops allowed on the floor. There is no email at the pairing stations; email kiosks are available around the floor, but are elevated off the ground, so requiring standing. When Pivots need to make a private call or have a meeting, they do so away from the main workspace in a separate room.

When a pair sits down to begin programming, their first step is to open Pivotal Tracker, a project management tool developed in house. It is within this application that the *stories* that identify customer requirements are housed. The stories are ranked in priority order by the customer and Pivots, and each story is 'pointed' from 1 to 3 where the number of points corresponds to the amount of time it is expected to take the team to create the corresponding code. The programming pairs will view the

story 'backlog' and select a story to work on by clicking 'start.' After a week or so, the team has a reliable 'velocity' showing how many stories they are averaging a week. The velocity is a computer-based algorithm that allows Pivots to educate customers on "the Pivotal way":

> "Tracker points out that this is our expectation of the next iteration. So it really forces the Product Owner to make decisions about what's most important to the business. It's not an exact science but if the Product Owner sees a three point story in a current iteration and says 'well, that's nice but we have this big conference in two weeks. So I want to move up this other story that does our marketing pages and I can know the effect and I can constantly look at the projection and I can make decisions on what's the focus of this 40-hour week and how does that predict the future', instead of saying 'oh, we have to get all of this done' " [Pivot#4, Denver]

And so the Pivots' and customers' ability to influence the timelines that are produced by Pivotal Tracker imparts a sense of freedom through the very control that it provides. Moreover, the visibility that the tool provides over the work gives a feeling of empowerment, because the tasks completed are validated as being relevant and important work:

> *"So keeping track [and having] a simple mechanism - some number that goes up when you do more work or you do better work - the number goes up. The velocity represents a lot of these things that make us feel proud, feel accomplished, feel good. That element makes teams focus – it makes them know what to focus on because they have the shared to-do list. Then the whole velocity thing and that kind of accept/reject feedback mechanism, beyond just like eliminating miscommunication like 'oh, I meant the button should be on the left.' There's that, but there's also like there's some* ***validation****, like 'my story got accepted! I'm happy.' People react well to that e-mail that they get, 'your story's been accepted'. Just a simple thing but like it makes you want to do more. So it empowers and it also encourages people to do focused work and focused on that which is most important." [Pivot#3, Denver]*

That visibility is built into the testing of the code that pairs produce. Agile development follows a test driven development (TDD) process. Test cases are developed based on the stories, the narrative versions of what needs to be coded. For a given piece of code, the Pivots will write an automated test that takes into account all possible inputs, errors, and outputs. Each time a test is created and run the developers also do regression testing to ensure that the current test doesn't introduce a problem into the pre-existing code. Pairs then write code that will pass the test – and here is where the visibility comes in – and this is indicated by the code turning the green color of a cucumber in the test environment.

This approach works for all levels of testing and enables the customer to view the feature and provide necessary feedback before the code is officially checked into the code repository. This occurs through a specific technology-reliant process:

> "I push my changes up to GitHub, click Finish on my story in Pivotal Tracker, wait for the build to go green, deploy to staging. Then, click Deliver on all the stories that have just been deployed. Other stories were most likely ready to be delivered as well, so I'll check that they were actually included in the deployment, and then Deliver them in Tracker." [Pivot#5, Boulder]

It is then time for the customer to go into Tracker and accept or reject stories awaiting their review on the staging server. In aggregate, the visibility is a matter of public record: The status of the build is clearly visible throughout the office with large "build monitors" mounted from the ceiling.

DISCUSSION: STRUCTURING FREEDOM IN AGILE DEVELOPMENT

Table 1 identifies some of the social conventions and material affordances that we have identified in the PL environment. This collection represents a super-set of those that relate to pair-programming (and therefore anticipates an expanded successor to this conference paper). Notwithstanding the casual style of the work environment, the scooters and ping pong, or the role that pair choice plays in the selection and conduct of the programming work, the tabled elements point to the high degree of sociomaterial structuring in producing and reproducing the PL way of doing work. The method relies for its effectiveness on everyone adhering to social practices and using artifacts in very specific ways. Physical-spatial design and norms both reinforce behavior. Distractions such as email and personal software applications are designed out of the pair-programming stations. Phone calls are made away from the programming area. Even time itself is tightly bounded, with office and team "stand-ups" to start the day's work, pairing to help enforce indefinitely a constant pace (see Principle #2, above), and norms that proscribe no work on weekends, evenings, or beyond forty hours per week. Meanwhile, one's work is subject to public illumination through devices (e.g., cucumber code and ceiling monitors) that constitute a kind of collective panopticon (Foucault, 1995).

From a traditional project control perspective, these social and material arrangements could be interpreted as limiting the individual developer's freedom, for example by disallowing flexible working schedules based on personal temporal rhythms (the late night coder, the crunch time sprinter). And yet what might otherwise appear as constraint, our informants widely experience as liberating. We might suppose that there is some requisite balance of structure vs. freedom that provides enough of the latter to make the former palatable to those who work under this regime. On the contrary, however, we will propose that *balance* is the wrong way to think about it.

What is experienced by individual programmers as freedom, and what helps instantiate in agile process, more broadly, the adaptability that molds the code to client needs, is made from the very structure that control-based perspectives commonly misread as constraint. In this sense, the programmer, the task, and the sociomaterial work environment co-constitute one another on an on-going basis. As we suggested earlier, the substance and meaning of individual action is fundamentally relational, and if materials, tools, other people, and the very architecture of space and time give affordance to the agile programmer, then the programmer herself gives affordance in turn to this structure. In this way, the sociomaterial work environment does not so much constrain practice but rather enables actors to get things done in ways that meet the needs of clients. Power is thus a productive rather than an inhibitive force.

FUTURE RESEARCH

In considering possible future directions for research, there is an obvious opportunity, within the current case, to extend our analysis into other domains of activity, such as project management. Also, the imbrications of the social and material need to be explored more fully, noting especially how social conventions (such as proscriptions on email access) are influential in making material affordances consequential in practice.

Our research is subject to the customary limits that go with the study of a single case. It would be interesting to compare, then, how different social conventions and material affordances might be brought to bear in constituting agile practices that conform equally well to the Manifesto's values.

We note also that the elements of any agile environment are not there by accident. Instead, social conventions and material affordances are subject to design, careful selection, and evolutionary modification. This fact points toward research into the genesis of these arrangements and the role of management in bringing them about.

Finally, the co-constitution we have remarked upon should not be regarded as unproblematic. Human and non-human participants both can decline enrolment in the sociomaterial network that makes up an agile development program. Much as we ought to explore design, then, we should also study the response of the designees.

"Agile Values…	**Social Conventions (SC) and Material Affordances (MA)**
Individuals and interactions over processes and tools	• Email Kiosk stations (MA) • 40 hour work week (SC) • Velocity calculation in Pivotal Tracker (MA) • Open plan office space • Kitchen stocked with food, catered meals (MA) • "Standup" 5 minute meetings and associated white board (SC & MA)
Working software over comprehensive documentation	• Test driven development and Cucumber software (MA) • Build monitors (MA) • Story creation and ranking, accept/reject (MA) • Ranking in inception with chocolate (MA)
Customer collaboration over contract negotiation	• Co-location of customer with Pivots at Pivotal Labs office (SC) • Iteration Planning Meeting (IPM); Retrospective meetings (SC) • Inception meeting – white boards, note cards, collaborative ranking (SC & MA)
Responding to change over following a plan	• Retrospective meetings (SC) • Re-ranking of stories, movement from Icebox, backlog (MC)

Table 1: Agile Manifesto Values and Their Enactment in PL

REFERENCES

1. Agerfalk, P. J., Fitzgerald, B. and Slaughter, S. A. (2009) Flexible and distributed information systems development: state of the art and research challenges, *Information Systems Research*, 20, 3, 317-328.
2. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001) Manifesto for agile software development, retrieved 17 February, 2003, from http://www.agilemanifesto.org.
3. Cao, L. and Ramesh, B. (2008) Agile requirements engineering practices: An empirical study, *Software, IEEE*, 25, 1, 60-67.
4. Conboy, K. (2009). Agility from first principles: reconstructing the concept of agility in information systems development, *Information Systems Research*, 20, 3, 329-354.
5. Conboy, K., Fitzgerald, B. and Golden, W. (2005) Agility in information systems development: A three-tiered framework, in R. Baskerville, L. Mathiassen, J. Pries-Heje & J. DeGross (Eds.), *Business agility and information technology diffusion*, IFIP TC8 WG 8.6 International Working Conference May 8-11, 2005, Atlanta, Georgia, USA, Springer, New York, 36-49.
6. Dybå, T. and Dingsøyr, T. (2008) Empirical studies of agile software development: A systematic review, *Information and Software Technology*, 50, 9-10, 833–859.
7. Fitzgerald, B., Hartnett, G. and Conboy, K. (2006) Customising agile methods to software practices at Intel Shannon, *European Journal of Information Systems*, 15, 200-213.
8. Foucault, M. (1995) *Discipline and Punish: The Birth of the Prison*, 2nd edition, Vintage Books, New York.
9. Fruhling, A. and de Vreede, G.-J. (2006) Field experiences with eXtreme programming: Developing an emergency response system, *Journal of Management Information Systems*, 22, 4, 39-68.
10. Harris, M. L., Collins, R. W. and Hevner, A. R. (2009) Control of flexible software development under uncertainty, *Information Systems Research*, 20, 3, 400-419.
11. Highsmith, J. (2002) *Agile Software Development Ecosystems*, Addison-Wesley, Boston.
12. Hutchby, I. (2001) Technologies, Texts and Affordances, *Sociology*, 35, 2, 441-456.
13. Laanti, M., Outi, S. and Abrahamsson, P. (2011) Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions of agile transformation, *Information and Software Technology*, 53, 3, 276-286.
14. Maruping, L. M., Venkatesh, V. and Agarwal, R. (2009) A control theory perspective on agile methodology use and changing user requirements, *Information Systems Research*, 20, 3, 277-399.
15. Pollock, N. and D'Adderio, L. (2012) Give Me a Two-by-two Matrix and I Will Create the Market: Rankings, Graphic Visualisations, and Sociomateriality, *Accounting, Organizations and Society*, 37, 8, 565-584.
16. Sarker, S., Munson, C. L., Sarker, S. and Chakraborty, S. (2009) Assessing the relative contribution of the facets of agility to distributed systems development success: An analytic hierarchy process approach, *European Journal of Information Systems*, 18, 1, 285-299.
17. Sarker, S. and Sarker, S. (2009) Exploring agility in distributed information systems development teams: An interpretive study in and offshoring context, *Information Systems Research*, 20, 3, 440-461.
18. Schatzki, T. R., K. Knorr Cetina and E. von Savigny, Eds. (2000) *The Practice Turn in Contemporary Theory.* Routledge, London and New York.
19. Shultze, U. (2000) A Confessional Account of an Ethnography about Knowledge Work, *MIS Quarterly*, 24, 1 3-41. Published by: Management Information Systems Research Center, University of Minnesota Stable URL: http://www.jstor.org/stable/3250978. Accessed: 27/09/2012 23:25
20. Sharp, H. and Robinson, H. (2004) An ethnographic study of XP practice, Empirical Software Engineering, 9, 4, 353-375.
21. Strode, D. E., Huff, S. Hope, B., and Link, S. (2012) Coordination in co-located agile software development projects, *The Journal of Systems and Software*, 85, 1222-1238.