Switching from waterfall to agile software development

Applying a combination of Distributed Cognition and Activity Theory to analyze the problem

Telle Zeiler

6/1/2011

Introduction

Theory is a philosophical approach to explaining occurrences in the world. Its concerns are epistemological, with ways of knowing why things are as they are and how they came to be that way. The best theories are pragmatic; they do things to help people understand the world better. People construct theories for several reasons: first, to account for things that are happening in the world, and provide an avenue for shared understandings about the way things are in the world. Second, theories can help people to envision possibilities of situations that don't or can't exist, and describe those situations using theory. Third, theories can help people gain control over things; people can use theoretical perspectives to tell a story about a circumstance or situation and the theory provides a shared understanding and vocabulary that allows people to talk about it. In short, theory is a tool which people use to complete difficult tasks. Selecting the appropriate tool to solve a problem or explain a phenomenon in the world makes a huge difference in the success of problem solving and explaining. (Mark Zachry, personal communication, March 30, 2011)

In this paper I combine two theories – distributed cognition (dcog) and activity theory (AT) – to better understand and explain the process my work team went through transitioning from a traditional waterfall software development lifecycle methodology to an agile one. This computer-supported cooperative work (CSCW) activity can be supported and explained by a focused combination of the two theoretical approaches to understand how the cognition of the entire team and its individual members had to adjust to a completely different project approach. Tools to support the requirements gathering process had to be used in different ways; activities surrounding how requirements are gathered also had to shift. First, I will explain each of the theories, how they are alike and where they differ. Next I will describe how I combined the two together, which pieces I took from which theories, and how this combination tells us more about the transition from a waterfall to an agile software development lifecycle (SDLC) than either of

the two theories could do on their own. Finally, I will discuss the implications and uses of this combined perspective for the field of Human-Computer Interaction (HCI).

Foundations of Distributed Cognition and Activity Theory

Distributed Cognition

Distributed cognition theory, (or "dcog" as it is often referred to) grew out of the need to understand information processing and problem solving activities beyond the unit of individual, looking at multiple individuals, teams and their environments. It extends cognitive theory, which focuses on the individual mind, to systems, groups, and human/system interaction. "Analysis of systems using distributed cognition permits the inclusion of all of the significant features in the environment that contribute toward the accomplishment of tasks." (Perry, 2003) Distributed cognition theorists such as Edwin Hutchins challenged cognitive theory's individual mind perspective, arguing that context and culture must also be taken into perspective when looking at cognition. Hutchins performed cognitive ethnography studies in the 1980s on Navy ships and in airline cockpits to examine how groups of individuals operate and navigate these large systems as a cohesive unit. He found that one individual could not run these large complex systems; rather, the cognition required to run a large system such as a Navy ship or an airplane necessitated that the cognition itself be distributed across multiple individuals and tools or artifacts. (Hollan, Hutchins, & Kirsh, 2000)

There are several key tenets of dcog. Cognition is not just in the head but in the world, it must be studied and applied "in the wild" rather than in a lab. It is social, meaning that cognitive processes can be distributed across multiple members of a social group, may involve coordination between internal and external structures and may be distributed across time so that earlier processes can inform later ones. (Hollan, Hutchins, & Kirsh, 2000) It focuses on representational

states (extending the internal representation to an external artifact, or an external artifact to an internal representation) being passed from one "medium" or "node" (such as a human mind) to another (such as a physical artifact, like a computer). There is symmetry in dcog: the human mind is treated as one node in a system of equal nodes making up the larger cognition of a system. (Kaptelinin & Nardi, 2009)

Distributed cognition is embodied; organization of the mind is based on the coordination of both internal and external resources, which can include tools and artifacts, for example the mind together with a calculator can perform complex arithmetic equations to balance a checkbook. DCog theorists also believe that cognition can't be separated from culture. The boundary of cognition goes beyond the individual's skin to include the tools and cultural environment in which an individual is situated. (Hollan, Hutchins, & Kirsh, 2000)

Activity Theory

Activity Theory (sometimes referred to as AT) was developed in Russia in the early 20th century. Even though it was developed before the post-cognitivist theories, it is grouped with them when discussed in HCI circles because it goes much farther than dcog in espousing that cognition is far greater than the individual mind. AT argues that what's important is the purposeful interaction of the subject with an object (can be a physical object or an objective). Subjects do things to acquire objects, but the objects alone do not determine the activity. The relationship between the subject and the object determines the course of the activity. (Kaptelinin & Nardi, 2009) Therefore, activity is irreducible: no properties of the subject and the object exist before and beyond activities. From an activity theory perspective, you are what you do. (Mark Zachry, personal communication, May 11, 2011) Cognition cannot be understood outside of the situated context of an individual.

Activity theory has several beliefs in common with distributed cognition. Both are critical of mind-body dualism and believe that cognition both incorporates mind and body and extends beyond the skin of the individual to include tools, artifacts, environment and culture. Both also believe in the essential role of technology, meaning tools and artifacts, in human life. Cognition cannot be understood at the level of the individual mind because tools and artifacts are always used in cognition, and they must be incorporated into a complete understanding of cognition. (Kaptelinin & Nardi, 2009)

Activity theory goes farther than distributed cognition however. Where dcog is a symmetrical theory, equating the internal mind with external tools and considering both as "nodes" that pass representational states between each other equally, activity theory has an asymmetrical view of humans and tools. According to AT, humans use tools based on their motivations and needs, but the same is not true of tools. It is the purposeful actions of humans in situated contexts that differentiate activity theory from dcog. (Kaptelinin & Nardi, 2009) To use the example of the navigation of Navy ships, AT considers the human's motivation for navigating the ship and all of the pressure and factors that impact the navigator. There may be personal motivations that impact the ship's navigation, for example if a sailor is mad at his fellow sailor, he might not cover for him when asked because he wants this person he dislikes to suffer the consequences of not doing his job. Dcog can't account for individual motivations and pressures. Activity theory focuses entirely on the holistic activity, including both subject and object(ive) and how one influences the other.

Combining the perspectives to address the problem space

The Problem Space

The team of business analysts at our company bridges the gap between business stakeholders and IT, writing business requirements and ensuring they are coded and implemented according to specification. Our team has always performed under a waterfall SDLC model, where the software development proceeds in a prescribed order: first requirements, then design, then development, then testing and finally release to production. In our current project, we were asked to bring a mobile optimized web site for the company to market much more quickly than a waterfall SDLC would allow, thus we were tasked with completing the project using a more iterative agile methodology. This posed some challenges for the business analyst team charged with writing the requirements. Requirements would have to be written collaboratively amongst team members using creative documentation methods in order to move quickly through requirements and iteratively release them to design and development.

How can one account for the cognitive challenges associated with transitioning a co-located work team using a traditional waterfall software development lifecycle methodology to an Agile methodology which is more iterative and less structured? A combination of distributed cognition and activity theory can help put into context and explain the cognitive challenges associated with this shift.

Distributed Cognition – a good starting place

Distributed cognition can help us to understand how cognition is extended and shared between humans and artifacts in an agile SDLC methodology. According to Hollins, Hutchins, Hirsh, there are three kinds of distribution of cognition: cognition distributed across multiple members of a social group, cognition involving coordination between internal and external

structure, and cognition distributed across time. (Hollan, Hutchins, & Kirsh, 2000) We can look at the unique circumstances of an agile project methodology through this lens.

1) **Cognitive processes can be distributed across multiple members of a social group.**

In the case of the agile team, there were two business analysts working on the same set of requirements; we worked collaboratively on the requirements for all of the different functional areas. Each of us had primary ownership over certain areas, but we also knew a lot about each other's functional areas' requirements as well. This was necessary because with an iterative methodology such as agile the work must be distributed so that requirements for the upcoming sprint get completed before the associated sprint planning meeting with IT to discuss, estimate and divvy up the work. The requirements knowledge for the project is thus distributed amongst the team of BAs. In a waterfall project, the work is more siloed with certain BAs holding most of the knowledge about particular features of a system in development.

2) **They may involve coordination between internal and external (material/environmental) structure.**

To document our iterative requirements we chose to use Excel instead of the traditionally used Word. Excel is more modular, and allowed us to easily add additional requirements and sub-requirements, and split functional areas into separate tabs within the document. Again, this distributed the work and the cognition amongst multiple team members, tools and artifacts. Meeting minutes, our modular requirements document, multiple team members and even our open issues list all served to distribute the cognition for this mobile project. In a waterfall project, the requirements are minimally distributed between the BA who wrote them and the Word document containing the BA's writing.

3) **Processes may be distributed across time so that earlier processes can inform later processes.**

For this project, in order to save time we chose to meticulously document every requirements working session with detailed meeting minutes. Stakeholders would approve or correct the meeting minutes, and decisions that were made in meetings were subsequently documented in the Excel requirements document. Things that were decided in meetings one day might be coded the very next day, at the same time as the BA was documenting it in the Excel requirements document. In this way, the requirements gathering process was distributed across time so that what took place and was documented in meetings then informed the feedback that was received from stakeholders which consequently informed the documentation of the requirements in Excel and the subsequent coding of the system. In a waterfall project, the process of requirements gathering is contained to the requirements stage of the project and does not get distributed across other portions of the SDLC that occur earlier or later in the process.

These three types of distributed cognition were all present in our agile requirements gathering process. Dcog can account for the sharing of cognition between humans and tools and artifacts, which is a very helpful basis from which to understand our team's challenges in moving to an agile methodology. However, we can gain deeper insight into this transition by understanding not only the distributed cognition across the "system" of requirements, but also by understanding the motivations of the various individuals involved in the new agile process. Only Activity Theory can help us see this level of cognition.

Taking it further - Activity Theory

In Activity Theory the unit of analysis is the individual rather than the entire system (e.g. ship navigation) that dcog investigates. Now that we understand more about the overall system cognition of an agile team through applying distributed cognition theory, we can extend our

understanding by applying activity theory to understand individual motivations of the subject (team members) toward the object (requirements document) that impact the agile process.

Activity Theory looks for the "creative possibilities of breakdowns, conflicts and contradictions… Activity is not locked down in predictable flows of state from one medium to another in a stability-seeking system. There is potential for movement/change even in what appear to be highly regulated activities." (Kaptelinin & Nardi, 2009) Kaptelinin and Nardi identify three types of creative fissures in activity that can only be explained by individual intervention in a process (unexplainable by the larger, systems-level view of dcog): coordination, cooperation and co-construction. (Kaptelinin & Nardi, 2009) It will be helpful to look at the agile SDLC challenge through this lens to broaden our understanding.

1) **Coordination is defined as the basic form of collaboration.**

People work towards a common goal but carry out individual activities basically independently. (Kaptelinin & Nardi, 2009) The agile process our team implemented provides an excellent example of this. The two BAs on the team both worked towards the common goal of writing and completing the business requirements for each sprint, but each of us had our own cognitive process for completing our assigned requirements. As Kaptelinin and Nardi state: "In coordination we might find creativity in very small actions." This held true for us in that each BA ended up formatting our assigned requirements in a different way: I included screen shots associated with each individual requirement, while my colleague attached screenshots at the bottom of all requirements for that functional area for reference. But in the end, we both coordinated to work towards the common goal of documenting accurate business requirements for IT to implement.

2) **Cooperation is a more advanced form of collaboration.**

Individuals relate their goals to the overall objective of a collective activity. Team members are aware of the actions of other team members and adjust their actions to the actions of others. (Kaptelinin & Nardi, 2009) Cooperation can be illustrated again in the collaborative effort of gathering and documenting requirements for an agile project. Team members had to be more collaborative; because the knowledge and work was distributed amongst all team members, it was critical for our agile team to work closely together to endure that no requirement was missed and that the overall project was a success.

3) **Co-construction: collaborative individuals cooperate to accomplish a pre-specified common object but can collectively redefine the object and activity.** (Kaptelinin & Nardi, 2009)

Co-construction can be illustrated in the mobile agile project in that the BA team members worked together to accurately gather and document the business requirements, but we had to redefine what requirements documentation looked like. We decided collaboratively to restructure the waterfall requirements process by creating a new process that was more flexible and allowed us to quickly gather, document and change requirements on the fly as needed. Our meeting minutes to Excel requirements process, discussed previously as part of the dcog model of processes distributed across time, can be extended with activity theory to understand how our community of practice (the subject) redefined the object (requirements document) and the activity (requirements gathering) from our waterfall mental model.

Activity theory gives us a broader perspective of not only the cognitive system of a team developing requirements in agile project and how tools and humans distribute cognition, but also each team member's individual motivations in developing the requirements and the reciprocal question of how those motivations impact the agile process and how the agile process impacts the team members' motivations.

Conclusion

As a participant in this agile project, applying a combination of distributed cognition and activity theory revealed several things to me about our move from waterfall to agile that I would not have discovered otherwise. First the extension of cognition to tools and artifacts can be applied across both waterfall and agile methodologies, though in our experience agile demands more distributed cognition than waterfall does. Every project at our company relies on written documentation of some kind and multiple team players working together towards a common goal. Cognition is certainly distributed in both types of projects. In agile however, because of the fast pace, the team must rely heavily on creative tools to document requirements in a way that waterfall projects don't – the output of a waterfall requirements phase is always a long Word document. The output of agile documentation differs based on the team working on the project and the tools available to them. This is where activity theory can help us gain a deeper understanding of this iterative SDLC. By looking at coordination, collaboration and co-construction, we can see the individuals' (subjects') impacts on the requirements documentation (object), both as single people choosing how to structure their requirements, and as a team collaborating on how to document requirements quickly and coming up with the Word meeting minutes to Excel requirements solution. A different team of people might have come up with a different solution. These kinds of highly individualized activities can only be explained through activity theory.

This combined perspective of dcog and activity theory can and should be used to examine similar problems in the field of HCI and complex systems. There is a tendency in theory circles to discount the value of one theory in favor of another. I believe that dcog and AT fit quite well together and do not need to be mutually exclusive. Dcog and AT can be used together to analyze

complex problems at different levels. Dcog should be used to analyze the cognition of an overall system and how its parts work together to accomplish a common goal. Then, to understand the idiosyncrasies of the "system" that can't be explained by dcog one can use activity theory to understand the motivations of the people using the artifacts and how that impacts the running of the system. To give a brief illustrative example, consider the Egypt Air 990 crash in 1999 when a large jumbo jet crashed without explanation in the Atlantic Ocean off the coast of Massachusetts. According to the NTSB, this crash could not be explained by system failure (dcog). All systems were running correctly as per the distributed cognition in the cockpit – each role both human and mechanical was doing their job. However, once the captain left the cockpit for a few minutes, a co-pilot took the helm and crashed the jet, repeating nine times "I rely on God." If one examined this crash only using dcog, one would see that a piece of the system failed because a person pointed the jet towards the water. However, one would not understand the human motivation behind the crash; only activity theory explores that. By using both, we can see that there was a failure of the system due to a human's personal motivation.

Works Cited

Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed Cognition: Toward a New Foundation for Human-Computer Interaction Research. *ACM Transactions on Computer-Human Interaction* , 174-196.

Kaptelinin, V., & Nardi, B. A. (2009). *Acting with Technology: Activity Theory and Interaction Design.* Cambridge: The M.I.T. Press.

Perry, M. (2003). Distributed Cognition. In J. M. Carroll, *CI Models, Theories, and Frameworks: Toward a Multidisciplinary Science (Interactive Technologies)* (pp. 193-223). San Francisco: Morgan Kaufmann.