Stigmergy and Implicit Coordination in Software Development

James Howison
University of Texas at Austin
jhowison@ischool.utexas.edu

Carsten Østerlund
Syracuse University
costerlu@syr.edu

Kevin Crowston
Syracuse University
crowston@syr.edu

Francesco Bolici
Universitá degli Studi di Cassino
francesco.bolici@eco.unicas.it

ABSTRACT
How do distributed, loosely connected software developers coordinate? That is, how do they understand and manage the dependencies between their work and the work of others? We contribute to this important and frequently studied area by developing a theoretical perspective that brings together insights from implicit and stigmergic perspectives on coordination. We illustrate our theoretical contribution with analysis of "lore" on best practices for open, distributed software collaboration.

Author Keywords
Coordination; Stigmergy; Sociomateriality

ACM Classification Keywords
H.5.3 Group and Organization Interfaces: Computer-supported Cooperative Work

INTRODUCTION
How do distributed, loosely connected software developers coordinate, that is, how do they understand and manage the dependencies between their work and the work of others? Prior answers to this important question can be broken into three broad categories of coordination: explicit, implicit and stigmergic.

Explicit coordination mechanisms are those where the work of coordinating (the articulation work) is performed as separate, identifiable work. Most clearly this work involves making specific plans, such as meeting early to decide a work breakdown and standard, documented APIs, and executing these plans. In addition, as empirical studies consistently find, planning alone is insufficient and is augmented by mutual adjustment, that is, coordinated work is also accomplished through discussion: informal sharing and talking as the work unfolds [2].

CSCW'12, February 11–15, 2012, Seattle, Washington, USA.

In many case though, co-workers can be observed to work without needing explicit discussion. The implicit coordination perspective explains these situations by arguing that well-developed shared mental models enables people to determine what needs to be done even in the absent of explicit communication and coordination [5, 11]. In other words, people's background knowledge and mental models allows them to engage in interdependent activities without separate coordination mechanisms or explicit communication.

A third line of work has posed an alternative explanation for coordination with little communication by focusing on how people use the outcome of their shared work as coordination devises. This work draws on the biological process of stimergy, "a class of mechanisms that mediate animal-animal interactions" [6]. As Heylighen writes, "A process is stigmergic if the work ('ergon' in Greek) done by one agent provides a stimulus ('stigma') that entices other agents to continue the job." [7]. Such an approach suggests that the shared material itself can be a coordination mechanism, without recourse to separate coordinative activities, either explicit or implicit. Continuing a line of work in CSCW, Christensen observed stimergic coordination amongst building architectures, arguing that their work is "partly coordinated directly through the material field of work" [3].

STATEMENT OF CURRENT WORK
While the concepts of implicit coordination and stimergy both offer plausible explanations to how people coordination works without explicit coordination mechanisms, they dont take each other's contribution into account. Thus, the literature maintains a dichotomy between a mental and social explanation on one side and a material argument on the other. Such a dichotomy has been called into question by the recent debate on sociomateriality [9, 10]. Our current work strives to bring these perspectives into confluence by asking: How do the socio-cognitive and the material work together in the interpretative process through which participants come to know what to do next? How do they do that in a way that is coordinated with others?

To work is to engage in practice, an ongoing historical process in which peoples doings are caught up and responsive to what others are doing. Taking inspiration from Smith [13]

and Bakhtin [1], we suggest that a work input is rarely completely original; it is always an answer (i.e., a response) to work that precedes it, and is therefore always conditioned by, and in turn qualifies, the prior activities. What the engineer or bricklayer does when facing somebodys work is responsive and partially determined by what has been going on up till now. Every next act picks up on what has been done and projects it forward into the future. However, the ability to read what has been done as a stimuli is also important: a novice and expert may reach very different conclusions from the same prior work.

This perspective seems particularly useful in understanding long-running open software projects whose end-point is discovered rather than pre-determined. In software development, we note that the codebase is part of a practice; it can be useful to think of a code contribution as a turn in a conversation. A contribution addresses earlier contributions but at the same time points towards a response. When a coder posts new code to the repository the code is always conditioned by, and in turn qualifies, the prior activities. It picks up on what has been done and projects it forward into the future. The new code thus works as both a model of work done and more importantly a model for future work. It stands as a question posted in a conversation waiting to be answered.

However, a developer, prompted in a stigmergic way by the appearance of work, actively interprets and adjusts. They do so by not only understanding the current work of others, but by projecting it forward. They do this drawing heavily on their experience and knowledge of past patterns (as work on implicit coordination suggests). We argue that they project forward not only the path of work of others, but also their own likely path and account for the intertwining of those branching paths. This complex branching projection, of course, is subject to bounded rationality, but can be updated and refined as the actual course of others work is revealed.

This unified perspective allows us to highlight three material characteristics of collaborative work systems: transparency, provenance and experimentability. Transparency, of course, is key to both the stigmergic prompting of the attention of others and to the communication of the raw substance to be interpreted (akin to Kellog et al.s idea of "display" and "represent" [8]); making software work appropriately transparent has been a focus of developer-support systems [12]. Provenancemeaning the history and sourceof shared project is a related but extended characteristic: source code control systems not only communicate a new piece of work, they make available the entire current state of the project as well as its history. Provenance systems highlight the work paths of individuals (even if just by sorting patches by committer ids) and thus create very fertile ground for projects; just as the likely future path of an arrow can be judged better from a video than a photograph. The importance of this characteristic was noted in forthcoming work on Github by Dabbish et al. [4]. Experimentability means that the current work can be placed not only in the context of an evolving codebase but allows a reacting participant to experiment with possible next moves before making them available to others, discovering interdependencies and framing their contribution in a way that accounts for their projections of others work paths. As illustrative evidence for our perspective we analyze existing "lore" on appropriate socio-material work practices. "Dont break the build," for instance, mean that contributions ought not to leave the shared work object in a non-functional state. Such contributions would interrupt the process of interpretation and projection: just as a fractured, stuttering video is more difficult to interpret. The sayings "Dont 'go dark'; Avoid codebombs" highlight that contributions ought to be a certain size and ought to be contributed at an appropriate frequency. The concern primarily is to avoid codebombs, large patches that are hard for others to interpret (let alone test), undermining transparency, provenance and experimentability. "Going dark" refers to withdrawing from transparent work for long periods of time, usually resulting in disconnected, larger patches. The resulting recommendations encourage participants to produce contributions that accord with our theorizing, above: they are maximally interpretable and come in a flow of work that aids projections and allows the rapid and frequent refining of those interrelated projections. Finally we analyze a disagreement between those who argue that git, as a distributed SCCS, will promote better practices (more atomic, more provenance, more frequent), and those who argue it will promote worse practices (less provenance due to git rebase command, more "going dark"). The discussion makes the value of understanding the process as socio-material: the technology has its effect only as enacted in shared practices.

BIOGRAPHIES

James Howison is an Assistant Professor in the Information School of the University of Texas at Austin, having earned his Ph.D. in Information Science and Technology from Syracuse University in 2009. He researches the organization of work on information technologies, where he has focused on free and open source software and software work in science.

Carsten Østerlund is an Associate Professor at the School of Information Studies at Syracuse University. He earned his Ph.D. in Management from Massachusetts Institute of Technology in 2003. His research explores the organization, creation, and use of documents in distributed work environments where people's daily practices are characterized by high mobility.

Kevin Crowston joined the School of Information Studies at Syracuse University in 1996. He received his Ph.D. in Information Technologies from the Sloan School of Management, Massachusetts Institute of Technology (MIT) in 1991. His current research focuses on new ways of organizing made possible by the extensive use of information technology.

Francesco Bolici is Assistant Professor in Organization Studies at Universitá degli Studi di Cassino (Italy). He earned his Ph.D. in Management of Information Systems from Luiss Guido Carli. His research investigates the network organizational structures emerging from the use of information technology and the consequent coordination mechanisms adopted by the actors in a digital environment.

REFERENCES

1. Bakhtin, M. The problem of speech genres. *Speech genres and other late essays* (1986), 60–102.
2. Cataldo, M., Herbsleb, J. D., and Carley, K. M. Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement (ESEM '08)* (Kaiserslautern, Germany, 2008).
3. Christensen, L. The logic of practices of stigmergy: representational artifacts in architectural design. In *CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work* (2008), ACM–568.
4. Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. Social coding in github: Transparency and collaboration in an open software repository. In *CSCW* (2012).
5. Espinosa, A., Lerch, J., and Kraut, R. E. Explicit vs. implicit coordination mechanisms and task dependencies: One size does not fit all. In *Team cognition: Process and performance at the inter- and intra-individual level*, E. Salas and S. M. Fiore, Eds. APA, Washington, D.C., 2004.
6. Grassé, P. P. La reconstrution du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportament de termites constructeurs. *Insectes Sociaux*, 6 (1959), 81.
7. Heylighen, F. *Why is Open Access Development so Successful? Stigmergic organization and the economics of information*. 2007.
8. Kellogg, K., Orlikowski, W., and Yates, J. Life in the trading zone: Structuring coordination across boundaries in postbureaucratic organizations. *Organization Science 17*, 1 (2006), 22–44.
9. Leonardi, P., and Barley, S. Materiality and change: Challenges to building better theory about technology and organizing. *Information and Organization 18*, 3 (2008), 159–176.
10. Orlikowski, W. The sociomateriality of organisational life: considering technology in management research. *Cambridge Journal of Economics 34*, 1 (2010), 125–141.
11. Rico, R., Sánchez-Manzanares, M., Gil, F., and Gibson, C. Team implicit coordination processes: a team knowledge-based approach. *The Academy of Management Review 33*, 1 (Jan. 2008), 163–184.
12. Sarma, A., Redmiles, D., and Hoek, A. V. d. Palantír: Early detection of development conflicts arising from parallel code changes. *IEEE Transactions on Software Engineering* (in press).
13. Smith, D. *Institutional ethnography: A sociology for people*. AltaMira Press, 2005.