

**Christine Halverson**  
IBM T.J. Watson Research Center  
Yorktown Heights, NY 10598  
krys@us.ibm.com

### **ABOUT ME AND MY METHODS**

By training I'm a cognitive scientist who uses primarily ethnographic techniques to understand how people and technology make organizations work. In most cases the end goal of one of my studies is to help develop technology that "at least does no harm" in cases where new technology is going into established work practices. However I almost always uncover new practices, whether in an individual's experience as they collaborate with others, or in a more collaborative interaction. Theoretically I look at data through the lens of distributed cognition (Hutchins, 1995) using a mix of methods from cognitive science, anthropology, and sociology.

While I prefer in depth (and thus time consuming) participant observation approaches I am not always able to justify it. Nor is it always the appropriate method. So I tend to be pragmatic about the approaches I use. I also use "quick and dirty" ethnography, as well as structured and semi structured interviews (f2f and im), surveys, and lab experiments. I have also been involved in delivering and supporting code and incorporating feedback into the ongoing development process.

I've worked in a variety of domains including: air traffic control, the auto loan process (car dealers and banks), speech and multimodal technologies, help desks, consultants, and online communities. More recently I have been studying programmers engaged in a range of endeavors. As this is the most pertinent to this workshop I'll focus a bit more on several recent studies. (See also Halverson et al to be presented at CSCW 06).

### **STUDIES OF PROGRAMMERS**

I began looking at programmers in connection with the development of programmer tools to support web application development for multiple mobile devices. The goal of the project--Multiple Device Authoring Toolkit (MDAT)--was to provide the support so that programmers could write a web application at a high level of abstraction based on the MVC model. MDAT would then convert that abstraction to code that would run on a number of programmer specified devices. The programmer would still have to tune and tweek the resulting program for each individual device. However it reduces the number of expert developers you need (one rather than one per device) making the issue of developing an application for 20 different mobile devices a more manageable problem.

Development of the MDAT system was based on two key assumptions. One, that any developer picking up MDAT

was likely to be both experienced in programming a single device and a novice to the MDAT system. Two, developer experience meant there was a common underlying set of notions that all programmers would share and that would serve as the basis for the abstraction.

We tried a number of approaches in this case, collecting data from prospective users and customers, interviewing the development staff and observing novice users of MDAT. This last area provided the most fruitful as we realized how even those who were experienced in web development and mobile devices did not necessarily all have the same background or level of experience. We analyzed videos of two subjects working their way through the online tutorial with MDAT, coding the test examples and writing their first program in MDAT.

What we discovered was that there was a quite divergent understanding of the basic concepts that the developers had considered common knowledge. In particular while all had an understanding of the MVC (model, view, controller) foundation of OO programming each developer had different practical experience with it. For example the STRUTS instantiation of MVC is different from the J2EE instantiation. Because the MDAT system used the STRUTS model of MVC interaction as a foundation programmers without that experience had a hard time understanding aspects of writing in an abstract view, largely because the MVC model they were familiar with was in conflict. (See Banavar et.al. (2004) for more details). While this study did not start out as a cognitive study, its findings ended up focusing on individual cognitive aspects.

Involvement with this project also led to a more social and collaborative study about how a globally distributed development team collaborated. Based on my research groups' earlier work with supporting online interaction (Bradner et. Al, 1999; Erickson and Kellogg, 2003) we deployed a Loop (Erickson et al, 2006) to a software development team distributed in North America, Japan and India. A Loop consists of a set of user-definable places, each of which can contain a conversation (persistent), static text, and people, as well as interface elements for seeing who is present, viewing, navigating and modifying the environment. (See Halverson et al, 2003) for more detail.

While our analyses centered around their novel use of Loops rather than their programming practice per se what is of interest here is our discovery of how the development team appropriated and used features in the Loop to coordinate their collaborative work. In addition we saw their use coincided with the high tempo activities just prior

to a major release build. In this case we analyzed log files, observed their interaction on line and supplemented with interviews.

Most recently I have been involved with a DARPA funded project (competition) to design a peta-flop high performance computer for the US government. Within the High Performance Computing (HPC) Community there is a frustrating mismatch between increasing machine performance and steady or sometime decreasing programmer performance. This sparked a US Government program striving for increased programmer productivity as they are seeking to advance to petascale computing – creating a new HPC where the P stands for productivity. As part of this effort we have done a number of studies. These range from a carefully designed and controlled experiment using students and a combination of automated data capture techniques with careful observations (Danis and Halverson, 2006) to in depth field work.

In order to get a better understanding of what the state of current practice I negotiated access to one of the US National Labs that is part of the National Nuclear Security Agency (NNSA) and over the course of 6 months performed multiple interviews and observation sessions. I focused on 3 teams of different sizes, all working on parallel codes with decades long life spans and hundreds of thousands of lines of code.

Although analysis is still preliminary I expected to hear about and see problems with parallelizing code, particularly on newer machines as they scale up. Instead I observed what might be considered the old chestnuts of programming – issues with code maintenance, modularity and portability as well as classic issues of debugging and memory management. While IBM's intent with this work is to understand how to build tools for programming a peta-scale computer, I am struck by how some issues seem endemic to programming even with experienced programmers. I was able to focus on observations where each programmer was making a small addition to code and spent the majority of their time on single node programming and debugging as they chased down programmer induced errors. From my perspective one of the intriguing things about this is the mismatch between the focus of the development program and actual practice. The program is all about new radical leaps in parallel programming, but the programmers are still struggling with errors and complexity introduced by large scale programs spread across multiple files.

Another interesting aspect is where collaboration comes into existing practice. The stakeholders involved in the HPC community run the gamut from occasional academic programmers who are mostly interested in the science they are using HPC to explore to long-lived installations with experienced developers, often dedicated to a single project team—and everything in between. Some teams are collocated while others are geographically dispersed and still others go it alone, running their parallel program on a

distant machine. Observing this diversity of practitioners has led us to develop a number of observation techniques to capture finely grained user behavior while programming.

## CREATING MASHUPS

I have had no experience creating mashups, but I'm willing to learn, including investing some time before I come, in order to possibly participate in creating a mashup. My programming skills are admittedly out of date, but I have coded in C, LISP and Pascal (as well as a few proprietary languages like ITS and REXX). a description of any experiences you have with creating mashups, whether you are interested in participating in the mashup creation activity, and if so, your programming skills

## REFERENCES

1. Banavar, G., L. Bergman, R. Cardone, V. Chevalier, Y. Gaeremynck, F. Giraud, C. Halverson, S. Hirose, M. Hori, F. Kitayama, G. Kondoh, A. Kundu, K. Ono, A. Schade, D. Soroker, K. Winz (July-Sept 2004) *An Authoring Technology for Multi-Device Web Applications* IEEE Pervasive Computing. 3, 3, pp83-93
2. Bradner, E., Kellogg, W, and Erickson, T. (1999) The Adoption and Use of Babble: A Field Study of Chat in the Workplace. *Proceedings of ECSCW '99*. Kluwer, pp. 139-158.
3. Danis, C. and Halverson C.A. (Feb 2006) *The value derived from the Observational Component in an Integrated Methodology for the study of HPC Programmer Productivity*. Workshop on Productivity and Performance in High-End Computing. 12<sup>th</sup> International Symposium on High-Performance Computer Architecture, Austin TX.
4. Ellis, J.E., Danis, C., Halverson, C.A., and Kellogg, W.A. (April 2006) *Social Visualization in Software Development*. In proceedings of CHI'06. Montreal, Quebec, Canada.
5. Erickson, T. and Kellogg, W. (2003) Knowledge Communities: Online Environments for Supporting Knowledge Management and Its Social Context. *Sharing Expertise: Beyond Knowledge Management*. (eds. M. Ackerman, V. Pipek, V. Wulf). MIT Press, pp. 299-325
6. Erickson, T., Halverson, C. A., Kellogg, W. A., Laff, M. R. Sussman, J., Wolf, T. and Edwards, D. (June 2006) *A Persistent Chat Space for Work Groups: The Design, Evaluation and Deployment of Loops*. The Proc. of DIS '06. Penn State, College Park, PA. ACM Press, 2006.
7. Halverson, C. (March 2006) *Inside large-scale parallel codes: It's not as different as we thought*. Ethnographies of Code. Lancaster, UK.
8. Halverson, C., Ellis, J.E., Danis, C. and Kellogg, W.A. (Nov 2006) *Designing Task Visualizations to support the Coordination of Work in Software Development*. In

proceedings of CSCW'06. Banff, Calgary, Canada. ACM Press 2006.

9. Halverson, C.A., Erickson, T. and Ackerman, M.S. (Nov 2004) *Behind the Help Desk: Evolution of a Knowledge Management System in a Large Organization*. CSCW 2004 Chicago, Nov 6-10, 2004.
10. Halverson, C.A., Erickson, T. and Sussman, J. (Nov 2003) "What Counts as Success: Punctuated Patterns of Use in a Persistent Chat Environment. GROUP 2003, Sannibel Island, FL. pp 180-9