Longstanding Success without Awareness Features: Lessons from a Collaborative Programming Tool

Cristian Bogdan
HCI Department, School of Computer Science and Communication (CSC), Royal Institute of Technology (KTH), 10044 Stockholm,
cristi@csc.kth.se

Abstract. The paper is a reflective account of a successful five-year experience with a Web-based collaborative programming environment for nomadic student programmers. While after implementing the basic functionality there were good opportunities for collecting and providing awareness information within the tool, resources did not permit that. Still, the users appreciated the tool and inferred the needed awareness information or provided for it using other channels like instant messaging. This experience suggests that well-designed basic articulation is much more important than awareness for the success of collaborative tools.

Keywords: awareness, articulation, nomadic, programming, collaborative, learning, contingency, amateur

1 Introduction

Collaboration in programmer groups poses specific CSCW issues (e.g. Grinter 1996) and it becomes even more challenging when programmers are not paid, often novices in a frantic learning process, not co-located and often on the move. This paper describes ParaDe, a collaborative programming tool designed together with the setting members who subsequently used the tool for over five years. The paper then mentions the improvements that are currently being made to the tool and concludes with the lessons drawn from the whole process, especially in regard to learning, support for articulation and awareness. The driving question of the paper refers to the elements and conduct of the design that made ParaDe a successful application in spite of not attaining the initial ambitions. From that the paper draws more general implications for CSCW.

2 Framework and Related Work

The programming tool discussed in this paper is used within a European student organization that is active in 70 locations in Europe and builds all its software “in-house”, developed by the Tech Group, a 25-strong group scattered across the locations, with a large membership turnover (up to 10 new members a year, and up to

10 leaving, with a member staying for an average of three years). Students in our setting are not professional programmers, and many of them are not studying computer science or informatics. *Learning* is thus an important concern when reflecting on the ParaDe experience. The *Community of Practice* approach to learning (Lave and Wenger 1991) takes a situated view, with setting members learning from and with each other while carrying out their activities within the community. Learning takes place mainly in an informal way. The fundament of this learning approach is *legitimate peripheral participation*, whereby members are peripheral in the beginning, yet they participate in the community of practice activities, and learn from this participation. Through that, they slowly move to the core of the community.

We also characterize our setting members as *amateur* programmers, and their group (the Tech Group) as an *amateur community* (Bogdan 2003, Bogdan and Bowers 2007). In the amateur community approach, work is driven by *challenge*, which, for a long-term sustainability of the group and its activity, should be *inexhaustible*, yet *actionable*, i.e. possible to address with the skills at hand, hence the connection to learning. Another important work motivation, in the absence of wages, is the *audience* of peers who review the work and of those who benefit from the work, in this case the thousands of organization members who use the software crafted by the amateur programmers. Audience and challenge in the student setting has similarities with hacker cultures (e.g. Levy 1984) with the difference that the skills involved can be much lower-level for the student newbie and the ‘obsession with coding’ a bit less intense.

A further characterization of the student programming work in question is its *nomadic* character (Bogdan et al. 2006). *Nomadicity* is defined as a work condition resulting from the variability of work location, duration, work session attendance by group members, as well as the variability of technology support for the work at hand. We found certain inflection points in these variabilities to have a more pronounced effect on work, we call them *discontinuities* and emphasize discontinuity *bridging* as an important aspect of computer support for nomadicity-affected work. The existence of a fixed “home base” while carrying out such work provides members with a resource to refer to in case of problems when on the move, and is often employed by professional mobile workers, but is lacking in e.g. student settings.

Like Grinter (1996) we are concerned with the collaborative aspects of the programming activities. Grinter emphasizes the articulation (Schmidt and Bannon 1992) involved in configuration management systems. An important and often overlooked aspect of articulation is the collection activities necessary for putting the workplace together, also known as “the work to make things work” (Bowers, 1994).

3 Setting

We have already introduced the organization and its Tech Group. The IT application developed and maintained by the Tech Group, by using ParaDe as a main tool, will be called here the Intranet and consists mainly of IT support for the activities organized (applications to courses organized, student CVs for a European-wide virtual job fair, etc) as well as member data, international team data, document archives for local and international entities, calendar of international events, etc. The Intranet is continuously growing through the work of the Tech Group, and currently consists of over 2800 Java Server Pages scripts and 200 Java application logic modules, served from two application servers.

The Tech Group members are normally working from their respective university locations, but have 2-3 annual Tech Group meetings and they also use opportunities to meet in the larger organization meeting. Prospective members are recruited during such organization meetings, and are given a short training about Makumba[1] the technology used to put together the web-based Intranet. Each newbie who wants to continue is then assigned a tutor and possibly a small task, which they usually start with during their first Tech Group encounter. The rest of the work and learning go on online until the next meeting.

4 Methods

This paper is a *design rationale* (Carroll and Moran 1996) exercise: the cooperative design and appropriation of ParaDe are reflected upon, with the goal of influencing wider CSCW design for related areas such as collaborative support for programming, for nomadicity affected groups, and for peripheral learning. The beginning of the ParaDe experience is detailed in (Bogdan, 2003). The subsequent success of the system, apparent through its long-term usage as well as through user reactions, has lead to increasing reflection and re-consideration of the process, while appropriating the design to the subsequent user needs, as well as to technology evolutions. The rationale is complemented with an *evaluation questionnaire* ran in 2005, designed to gather user feedback on ParaDe features as well as to collect data on related aspects such as the learning taking place in the setting, the sustainability of the Tech Group, etc. Throughout, as a former member of the setting the author was an increasingly conscious *participant observer*.

5 Design for Parallel Development (ParaDe)

Issues with “the bundle”

ParaDe was designed as a response to the problems experienced with a collection of programming tools called “the bundle”. The student organization has long been proud of tailoring its own information technology. At the early stages of the Tech Group existence (late 1990s) a number of technologies were tried out, including Lotus Notes, which was dropped due to its learning curve and its visual programming approach, which was difficult to use in the distributed setting. Once plain-text Java, Java Server Pages (JSP) and SQL programming were chosen, a *bundle* was built, which was an

[1] www.makumba.org

archive file containing the necessary tools (Java developer kit, MySQL database engine, Ant build tool, Tomcat Java-based WWW engine, CVS version management, and documentation about them) available in Windows and Linux versions. The bundle was supposed to be downloaded or copied from a CD or portable hard-disk (there were few cheap USB sticks at the time, in 2000) to a computer at hand, be it at home, at the university, or while traveling at some foreign university or student dorm. Simple set-up scripts existed, that were setting all necessary environment for the tools to be used, without the need for administrator rights.

The most important issue with the bundle was that problems often occurred during its installation. The notion of *contingency* if often employed in CSCW to refer to such situations that may occur but are not necessarily expected, and graceful negotiation of such contingencies is at the heart of articulation work (Schmidt and Bannon 1992). Addressing the installation contingencies was often hard or impossible for the amateur programmers, especially for beginners but even for skilled ones. What is more, the level of expertise needed for installation seemed to be much higher than the level of expertise needed for programming once installation was done.

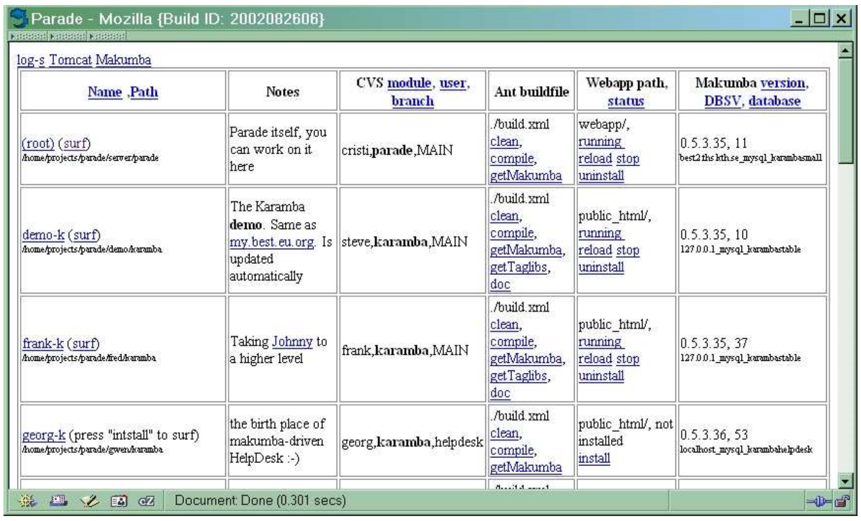
While amateur communities (Bogdan, 2003, Bogdan and Bowers 2007) generally welcome contingencies in their activity domain, contingencies become a problem when they are hardly addressable, or when they are not in the precise domain but in a related one. As such, while the Tech Group welcomes contingencies in regard to programming, contingencies in the area of tool installation are not part of the main goal, and are not as welcome.

Emergence of the ParaDe Home Base

The ParaDe approach addresses the installation contingencies by keeping all the programming tools (plus simple Web-based text editors) already installed and accessible through a website (see Figure 1). Each Tech Group programmer has a personal *ParaDe context* with its own copy of the Intranet source code, runnable against a test database. Within the "context" the user has an interface that likens an Integrated Development Environment (IDE) tailored for the specific purpose of developing Java-based server-side applications. The IDE gathers the most basic functionality of tools like the Ant build tool (just some specific build tasks like compile and clean), CVS version management (only basic update and commit, with access to version history), or Tomcat (only web application deploy and un-deploy). If one needs more sophisticated features, they have to log in to the machine and use them in the command line, outside ParaDe.

The ParaDe welcome page shows an overview of the different contexts, with the versions of the Intranet worked upon, the versions of the libraries used, etc. All contexts are readable and writable by all users, thus supporting learning-in-doing by both being able to look at peer's work and being easily able to let them help.

Nomadicity aspects are supported by this design, by reducing the installation overhead, which would hinder work and collaboration even if one would have the required skills for it. To consider this aspect through the notion of discontinuity developed in (Bogdan et al. 2006), the physical space discontinuity (moving within the city or within Europe) does not bring a technology support discontinuity; ParaDe is the same everywhere and the users find it as they left it last time they had time for Tech Group work. ParaDe does become a virtual "home base" for the group, in the absence of a physical one.


Parade - Mozilla {Build ID: 2002082606}
log-s Tomcat Makumba
Document: Done (0.301 secs)

Name ,Path	Notes	CVS module, user, branch	Ant buildfile	Webapp path, status	Makumba version, DBSV, database
(root) (surf) /home/projects/parade/server/parade	Parade itself, you can work on it here	cristi,**parade**,MAIN	./build.xml clean, compile, getMakumba	webapp/, running reload stop uninstall	0.5.3.35, 11 best2.ths.kth.se_mysql_karambasmall
demo-k (surf) /home/projects/parade/demo/karamba	The Karamba **demo**. Same as my.best.eu.org. Is updated automatically	steve,**karamba**,MAIN	./build.xml clean, compile, getMakumba, getTaglibs, doc	public_html/, running reload stop uninstall	0.5.3.35, 10 127.0.0.1_mysql_karambastable
frank-k (surf) /home/projects/parade/fred/karamba	Taking Johnny to a higher level	frank,**karamba**,MAIN	./build.xml clean, compile, getMakumba, getTaglibs, doc	public_html/, running reload stop uninstall	0.5.3.35, 37 127.0.0.1_mysql_karambastable
georg-k (press "intstall" to surf) /home/projects/parade/gwen/karamba	the birth place of makumba-driven HelpDesk :-)	georg,**karamba**,helpdesk	./build.xml clean, compile, getMakumba	public_html/, not installed install	0.5.3.36, 53 localhost_mysql_karambahelpdesk

Fig. 1. The ParaDe welcome page showing the list of available contexts

ParaDe and Learning

It is important to comment on the basic ParaDe design principles through the theoretical notions of Community of Practice. Having a separate product (Intranet) copy for each programmer (a so-called checkout) is customary in software engineering. It is less customary however to keep and visualize all these copies together, on equal footing, so everybody can look at everybody else's work and learn from it, as well as review it, in the spirit of peripheral participation. Having no access control on the copies leads to easy intervention by peers when in need of help, or in case of ad-hoc collaboration.

As a general learning-related observation, a low learning threshold is favored by just exposing the most essential features of the programming tools, letting the users focus on their primary task (programming). They are spared the details of more professional features of the programming tools they use. A need for such sparing was felt ever since the group started with Lotus Notes and members had great difficulties separating the essential features from the advanced ones, as they were presented in single, long lists in the interface.

Awareness in ParaDe, from early intention to late realization

While the *learning* and *nomadicity* concerns were central in designing ParaDe, the designer saw early the possibilities for providing *awareness* information in the tool.

Awareness being a wide concept (Schmidt 2002), we will restrict here to social awareness (who has been using the tool lately, who is “around” now) and awareness of the work artifacts (what happened with the ParaDe context since last login, what happened in other contexts that may be of interest). As ParaDe has a collection of all personal work contexts in digital format, it is in a favorable position to provide such awareness information by harvesting through the work contexts and finding inter-relevancies.

However, for awareness information to be provided, the basic IDE-like functionality had to work first, so all design and implementation efforts were focused on that. By the time the tool was in a useable state, the interest for implementing the awareness features decreased, as the tool was well usable without them anyway! It is true that social awareness existed somewhat, as ParaDe prints on the log lines the user whose action are logged, but one has to check the server log explicitly to infer that presence information, and users typically only check the log if they have serious problems while developing.

On the other hand, the questionnaire data and observations in the setting revealed that users easily and skillfully addressed the need for social awareness by using instant messaging to check who is around and may be able to help, and combined that with ParaDe usage in real time. Many questionnaire respondents request a specific form of awareness related to the artifacts of work: they would like to know whether there are newer versions of files in the CVS version management system.

Awareness is then an obvious point of improvement for ParaDe today. Since simple social awareness is already arranged for by the users through other means, we are working for putting to use the aforementioned large potential for awareness in regard to artifacts of work. For that, more elaborated awareness mechanisms are considered, such as Aether (Şandor et al. 1997), which combine the interest of the user (known as “focus”) with the request for interest from the other users (called “nimbus”) to infer the “awareness level” that the user will have of a certain artifact. One of the problems with such mechanisms is that they employ heuristics, which, if not understood at least at principle level by the users, are prone to be distrusted. To address that, we are looking into the notion of translucency (Dourish and Button 1998) which supports a sufficient level of system openness towards the user that would allow them to understand the complex underlying computing mechanisms that led to certain system actions.

6 Discussion and Conclusion

We will conclude by reflecting on some realities of collaborative system implementation: in order for a tool to be adopted by users, implementation needs to start with must-have features like sharing and browsing data, while more ‘advanced’ features such as awareness support, notifications, etc, get postponed and sometimes left out due to lack of resources. In the case at hand, ParaDe responds to the nomadic and peripheral learning needs but awareness and impromptu communication (for learning) and collaboration needs were not addressed. Instead, they are skillfully supplied for by the users using instant messaging and log reading. However, generic tools like instant messaging cannot provide work context awareness, hence the ideas for addressing such awareness are still of actuality. Still, providing such high-level awareness features at a high quality requires a lot of resources, while much of the awareness inferred by the users comes at no cost in terms of programming resources.

While we regard awareness as an essential feature of CSCW support for highly nomadic settings, the ParaDe experience shows a case where addressing “the work to make programming work” (cf Bowers 1994) is of higher priority. We are also concerned with the lack of importance such work is given in texts about articulation work (e.g. Schmidt and Bannon 1992) where illustrations of the work considered are concerned with well-advanced work practices and well-established protocols and work routines (albeit affected by contingencies). Our example, and Bowers’ show that early articulation support is of essential importance for the adoption and sustained use of CSCW tools, maybe even more important than favorite CSCW topics such as awareness support (itself a form of support for articulation). Furthermore, as users may be able to supply for awareness through other channels, one needs to invest in sophisticated awareness mechanisms to really add value in complementing the simple user-procured awareness mechanisms, available through generic tools like instant messaging or logs.

References

1. Grinter, R. E. 1996. Supporting articulation work using software configuration management systems. Computer Supported Coop. Work 5, 4 (Dec. 1996), 447-465
2. Lave, J., Wenger, E. (1991): *Situated Learning. Legitimate peripheral participation*, Cambridge University Press3.
3. Bogdan, C., (2003): *IT Design for Amateur Communities*, doctoral thesis, ISBN 91-7283-44467, KTH, Stockholm

4- Bogdan, C, Bowers, J. (2007), Tuning in: Challenging Design for ‘Communities’ Through a Field Study of Radio Amateurs, *3rd International Conference on Communities and Technologies*, Springer, 2007

5. Levy, S. (1984): *Hackers. Heroes of the Computer Revolution*, Dell Publishing, New York
6. Bogdan, C., Chiara Rossitto, Maria Normark, Pedro Jorge "Adler", Kerstin Severinson-Eklundh (2006), On a mission without a home base: conceptualizing nomadicity in student group work. In Hassanaly P., Herrmann T., Kunau G., Zacklad M., *Proceedings of the COOP 2006*, 137, 23-38, IOS Press, The Netherlands, May 2006
7. Schmidt, K. and Liam Bannon. Taking CSCW seriously: Supporting articulation work. Computer Supported Cooperative Work (CSCW): An International Journal, 1(1-2): 7-40, 1992.
8. Bowers, J.M. (1994): The work to make a network work. Studying CSCW in action, in *Proceedings of CSCW 1994*, ACM Press
9. Carroll and Moran (eds) (1996). *Design Rationale. Concepts, Techniques, and Use.* Lawrence Erlbaum Associates
10. Schmidt, K. 2002. The Problem with 'Awareness': Introductory Remarks on 'Awareness in CSCW'. Comput. Supported Coop. Work 11, 3 (Nov. 2002), 285-298.
11. Şandor, O., Bogdan, C. Bowers, J. (1997): AETHER, an awareness engine for CSCW, in *Proceedings of ECSCW 1997,* Kluwer
12. Dourish, P. and Button, G. 1998. On Technomethodology: Foundational Relationships between Ethnomethodology and System Design. Human-Computer Interaction, 13(4)