Makumba: the Role of the Technology for the Sustainability of Amateur Programming Practice and Community

The Technological Religion of a Student Tribe

Cristian Bogdan
School of Computer Science and Communication
Royal Institute of Technology
10044 Stockholm, Sweden
cristi@csc.kth.se

Rudolf Mayer
Institute of Software Technology and Interactive Systems
Vienna University of Technology
1040 Vienna, Austria
mayer@ifs.tuwien.ac.at

ABSTRACT

We address the issue of sustainability of practice, which we regard as crucial for the sustainability of the community at large. In the absence of material reward, sustaining a specialized activity such as programming is not trivial especially when members move often in and out of the community. Our case is the group of voluntary, amateur student programmers from a European-wide student organization. We present this setting as an Amateur Community and as a Community of Practice, and show how such framing helps in understanding sustainability of practice. Although being totally voluntary and managing a large intranet, the group has been thriving for six years. To explain such high practice sustainability, we examine the role of the technology framework used by the group during this time. We then propose a more general framework for understanding practice sustainability in the context of amateur communities of practice.

Categories and Subject Descriptors

K.4.0 [**Computing Milieux**]: Computers and SocietyGeneral

General Terms

Design, Human Factors, Languages

Keywords

Sustainability of Practice, Case studies, Amateur Community, Community of Practice

1. INTRODUCTION

The Intranet of an European-wide organization had just over 600 dynamic pages at its launch in 2002. The Intranet grew steadily in both size and functionality and at the end of 2008 it has almost 2000 dynamic pages. Throughout, except for a few glitches, the system was highly available and fast. Today the Intranet has around 2000 users from among the organization members, and its public part can be accessed by 4000-5000 organization customers at a time.

C&T'09, June 25–27, 2009, University Park, Pennsylvania, USA.
ACM 978-1-60558-601-4/09/06.

Such figures would not be surprising for a professional international organization, which hires or outsources professional programmers and system administrators. The figures are however unusual for a voluntary student organization (BEST, www.best.eu.org), whose IT crew (referred to as the Tech Committee) is made of amateur voluntary student programmers who often do not study Computer Science or related subjects, or are at the early stages of their education. Since 2002, the Tech Committee had over 10 active members at any given time, and the Intranet was extended with more and more subsystems even if about one third of the Tech Committee membership was renewed each year. Reflection on this *sustainability* is the subject matter of this paper. We further assert that the design of the technology used to build the Intranet, called Makumba (www.makumba.org), is a determining factor in the Tech Committee sustainability. Prior to Makumba introduction, the student organization had disparate IT systems, and the Tech Committee was formed in 1997 to maintain them but had much lower membership counts and was highly dependent on a few people.

In this paper we aim to characterize *sustainability of practice and community* based on this positive experience. We also aim to discuss the role of technology in achieving such sustainability, and to draw design implications that would help to devise technologies that encourage sustainability.

While we recognize the importance of *environmental sustainability* in IT and interaction design [1], we focus on a topic that is only abstractly related, yet we believe important in community and technology research. It is e.g. common knowledge that most virtual communities are "either young or dead" [16], and various cookbooks for making the community thrive were produced [8]. Even for non-virtual communities such as our student organization and its Tech Committee, it is sometimes difficult to sustain a specialized activity within the community, and this issue was raised under the name of sustainability in the field of Participatory Design (PD), e.g. as a basic principle of a PD method [10] or as a call for "self-sustaining [PD] processes within work settings" [5]. While managerial aspects such as leadership and incentives for work are certainly important in sustaining an activity, we also advocate a focus on the methods and tools used in such specialized activities, as them being easy to learn or entertaining to use should lead to better sustainability of the respective practice. As immediately

apparent liabilities to practice sustainability one can exemplify lack of material rewards, or frequent changes in community membership, both of which affect voluntary student organizations. A related useful notion is *self-sustainability*, whereby a specialized activity such as PD [5] or programming is started by professional intervention in the setting, yet it is able to continue using only community resources (human and otherwise), after the specialized professionals leave the setting. In our case both types of activity were initiated in the student organization by the first author as a Human-Compute Interaction researcher and computer engineer, who left the setting in 2003. The programming activity and its sustainability are addressed in this paper.

In what follows, we will present the student organization, its Intranet, and will frame the Tech Committee as a Community of Practice [12, 21] and as an Amateur Community [2, 4]. We will then introduce the topic of practice sustainability, highly related to learning and essential for community sustainability. We then present the design of the Makumba technology that powers the Intranet, then we characterize the sustainable evolution of the Intranet and the Tech Committee, as it comes out from examining the source code repositories and Tech Committee membership lists. We then present the results of a survey that we used to elicit data on Makumba learning aspects in general and sustainability especially. We finally discuss our results with a focus on learning, sustainability and design implications for technology.

2. SETTING

BEST– Board of European Students of Technology, the student organization running the Intranet of our interest, was grounded in 1988 and is currently present in 79 mainly technical universities across 30 European countries. The organization grew from being present in 60 universities in 2003, and has at the moment about 2,500 members. Their aim is to organize *complementary education* in the form of courses and competitions for all students of the member universities (i.e. not just for its members), to assist them with *career support*, and to encourage the educational involvement of students by organizing symposia on engineering education aspects.

BEST maintains contact with and raises funds from the European Union bodies and from a number of company partners. Internally, the organization has two statutory meetings per year and several less formal, smaller meetings in between. While most members worked on international topics only during such meetings, and for the rest they worked in their local organization chapters, since 1997 the organization was able to sustain "committees" on several topics: marketing, fund-raising, training and internal education, information technologies (the Tech Committee), and complementary education program management. The committees met in the international meetings but kept on working on their topic also in between meetings, employing e-mail and instant messaging systems, as well as dedicated tools as part of the Intranet.

The Intranet is supporting the activities of the student organization. It was assembled as an integrated system in 2002 from several systems. The *first* such system was an event application system for complementary education courses. The system registered student applications to the course events, and let the organization manage their acceptance (to one event out of maximum three applied for) and participation. The system had been re-built yearly since around 1993 to different levels of completeness, including several years when it failed to work and the members had to resort to manual management. The application system was based on e-mail for communication initially, and a Web-based system was devised in 1995, built using the C programming language. The system finally stabilized' as a Java software which was re-used for several annual editions of the complementary education program starting in 1997. This system was helping the management of the most important student organization activity and was heavily dependent on the first author until its Makumba re-implementation was completed in 2002, and still runs today.

The *second* system was an internal document archive and member profile management system known as the "Private Area". The system started as a set of manually maintained Web pages, storing documents produced and voted upon in international meetings, and it was automated using Lotus Notes in 1998, when internal event management was added to facilitate member application to and participation in statutory meetings and other internal events. At the time, a need for a common technology for the IT systems of the organization was recognized and Lotus Notes was intended to be that technology. However its shortcomings for the Tech Committee context were recognized and the Makumba technology was designed in response.

Third a "virtual jobfair" was built as part of the 'career support' organization mission, allowing companies to post job adverts, and students to post their profiles. This system was among the first coordinated implementation efforts of the Tech Committee, as different from the previous spontaneous implementation endeavors by individual members. Several technologies were tried out, starting in 1997 with Lotus Notes, continuing with Java Servlet technology before the system was finally launched in 2000, using an incipient Makumba version.

Besides heavily extending and integrating the above subsystems, several Intranet features were added since its inception: a Wiki, email archives for BEST's over 500 mailing lists, a training database, a survey-engine, a career-newletter, and a unique sign-in system that allowed students to share their accounts between the Intranet, the course application system and the "virtual jobfair". Further, a number of team support tools were implemented, like project and task management for both international committees and local chapters, company relations management for both the fund-raising committee and the local chapters, and several specific tools for the committees, often in the form of statistical analysis of student account data, subscription to the newsletters, etc. Another major new tool is an online voting system, intended to make the statutory meetings less crowded.

The Tech Committee is in charge of developing and maintaining the Intranet, as well as supporting it with activities such as helpdesk. The committee also coordinates activities of interaction design for further new areas of the Intranet. Administration of the Intranet, as well as of communication systems such as e-mail are also Tech Committee responsibilities. There are two formal membership levels ("trainee" and "active member") and there is a formal coordinator, elected each year, who is also a member of the BEST's overall coordination bodies. In its early (less sustainable) days, the

Tech Committee consisted of 1-5 members who were mainly responsible for the individual systems and had difficulties backing each other up when they did not have time for voluntary commitment. This is similar to the present day situation of IT bodies from other student organizations in Europe and Canada who have approached the Tech Committee for know-how and technology transfer. Tech Committee membership levels increased after 2002, and new members are usually attracted in international meetings with a three-hour Makumba training, including exercises in making dynamic pages that browse Intranet data in various ways, improving existing Intranet pages, followed by assignment of more complex Intranet-related tasks.

3. METHODS

We regard sustainability as a long-term matter that cannot be investigated with a short focused study. Throughout our contact with and involvement in the Tech Committee we were concerned with sustainability and at times we considered theoretical frameworks and recipes for how to achieve it, thus sustainability was an ever-present research issue for us, but also a practical issue in the community life. Both authors were active in the Tech Committee at different times (1997-2003 and 2004-2008 respectively), and *action research* was a conscious investigation approach for the first author, who also designed the Makumba technology together with Tech Committee members in a conscious act of *cooperative design* [9]. This paper is the occasion for a *reflection exercise* [17], on the part of the second author, who was not a founder and followed a learning path of the kind described in this paper. Sustainability being a long-term issue, we expect other writings about it to be reflective accounts.

We are highly aware that our first-hand involvement with the Tech Committee is a potential hinder for us producing an objective' account of its sustainability and on other matters of interest, and this is probably not uncommon in community-related research endeavors. We therefore complemented our personal experiences with both non-elicited and elicited data, as described below. Furthermore, our involvement having taken place *at different times* has resulted in a fruitful confrontation process that allowed us to depart from our personal opinions and arrive at more reliable and valid results.

For assessing quantitatively and qualitatively the sustainability of the Tech Committee and its practice, we regard as non-elicited data the *code repository* that the community has maintained, which allowed us to assess and reflect upon the progress of the Tech Committee work. The repository uses the Concurrent Versioning Systems (CVS) technology, which allows us to examine the incremental changes and additions that were made to the code, their time, and their authors. We are thus able to reconstitute the Intranet code as it was at any moment in time since 2003. A number of code analysis tools are also available for investigating CVS repositories, and we found them useful for our inquiry.

We have also elicited data from the Tech Committee members, for purposes related to the Makumba technology. In 2002 the first author evaluated the design of Makumba with a questionnaire that had 12 respondents. In 2005 the first author also ran a questionnaire with 32 respondents to asses the status of Tech Committee work and technology use, and thus to indirectly look at its sustainability. Finally, both authors designed a questionnaire at the end of 2008 where 30 out of the 45 past (since 2003) and present members who were approached reflected on their activity in the Tech Committee over their whole 1-4 year-long membership period. This questionnaire provides this paper with both quantitative and qualitative data.

During our membership we had access to the e-mail traffic and other communication of the Tech Committee. A further form of non-elicited data is constituted by Tech Committee *membership lists* at different times during the committee activity since 2002. Such lists can be made by examining member profiles in the Intranet. Number of members, as well as their level of activity are useful indicators in assessing sustainability. Therefore the membership lists were complemented with levels of member activity as elicited from committee leaders and self-assessed by members themselves in questionnaires. Personal acquaintance with many of the members and knowledge of their skill evolution has come to complement this further. Some members continued on an IT career after graduation, and this was yet another indicator of their IT skills.

To be able to better comment on our data, we will go through some conceptualizations of learning, community, practice, amateur and voluntary work, and sustainability. We will also describe in more detail the design of Makumba.

4. COMMUNITY OF PRACTICE

Patterns of learning in the Tech Committee, as well as in the other committees and in the student organization at large, are well described by the Community of Practice social theory of learning [12, 21]. This learning perspective captures the social, informal and everyday aspects of learning, and emphasizes the community aspect of learning, i.e. members learn as part of and the ways of belonging to a community. This is referred to as "learning the ropes" or more formally "learning in doing". As community members learn, they evolve from peripheral participants to 'full' community members, i.e. full participants. At first, they want, try, then pretend to be members, then they identify with the community and finally they become expert members. In effect this is a description of the social relations between newcomers and old timers, and of the individual acquiring of knowledge and skill. The process of becoming a full participant "configures" learning in doing.

This theory captures accurately the kinds of learning processes taking place in voluntary settings, where there is little incentive for attending elaborate formal education or resources for providing it. Instead, learning takes place while doing and participating, informally. Practice is not all informal however, it is slowly reified [21] to canonical forms such as books of rules and regulations, training and other formal learning arrangements and material, etc.

The Community of Practice perspective was extracted from field studies in five settings ranging from midwives to tailors and non-drinking alcoholics [12], but it of course applies well beyond, e.g. in insurance claim sorting [21]. Many programming-related activities and their 'learning/membership paths' covered by the community members are well-described by the theory, for example MUD community members becoming magicians (MUD programmers) [15, 16], open source contributors becoming "committers" i.e. earning the rights to commit new code to the open source project repository. It is worth noting that in both cases a formal, canonical member role has been reified out of existing practice, in a

similar distinction to the trainee–full member reified in the Tech Committee.

5. AMATEUR COMMUNITY

A further conceptualization that encompasses work in voluntary communities such as the Tech Committee, or other student groups, is the Amateur Community [2, 4]. The perspective was extracted from studies of Amateur Radio (ham) [4] and has been tentatively applied also to open source and other programming related communities [2].

The term 'amateur' is often used in a pejorative sense in everyday speech to denote 'novice', 'unprofessional', 'bad approach to work' or 'bad quality of work' . However, upon close examination of people who talk of themselves as being "amateur", authors have encountered that the skills of e.g. amateur mycologists [7], actors, baseball players and archaeologists [18] range from novice-level to an expertise that rivals their professional counterparts. Sciences like astronomy still depend on the work of amateurs for their progress. Amateurs carry out their activities primarily out of their love (French "aimer") for the activities themselves, as different from paid (or otherwise rewarded) work activities such as jobs. An ethnographic account of various amateur settings [18] describes amateurs as being socially situated "on the margin between work and leisure"; amateurs often describe their activity as work, and they rarely pursue it alone. Relevant social categories for amateurs are the *professionals* of the activity that they pursue, which they draw influence from and sometimes exercise influence towards, and the *public* towards which they relate in doing their work an benefit from it. This is referred to the Professional-Amateur-Public system. In relation to the professionals, amateurs can be professionals of their trade at the same time, or they may aspire to become professionals, in which case they are referred to as *pre-professional*, or they have been professionals of the domain and presently continue it as amateurs (*post-professionals*).

Within the Amateur Community perspective, amateur work is mainly driven by *challenge*, which generally originates in *contingencies* that need to be negotiated during the activity [4]. Unlike professional work where contingency is often not welcome, and expensive to address (yet capacity to address it is regarded as competence), in amateur work contingencies of a specific kind are almost requisite. The challenge, and its constitutive contingencies are thus requisite but must also be *addressable*, i.e. there should be enough skill within the amateur community to address the challenge. At the individual level, balancing a challenge with skill is regarded as essential for the psychology of the optimal experience [6]. A healthy *contingency space* will include contingencies addressable by experts or old-timers as well as by novices or newcomers.

Besides being addressable with various levels of skill, a contingency space should ideally be *inexhaustible*. For the example of radio amateurs, the weather is likely to always provide radio-related contingencies. On the contrary, an amateur community whose members aim to travel to all points of integer geographical latitude and longitude[1] have a remarkable, yet exhaustible contingency space. For a further example (also illustrating the diversity of amateur settings), amateurs who aim to teach re-populated bird species to migrate again by leading their flight with light aircraft[2] have a virtually inexhaustible contingency space, as increasing the number of re-populated birds, possibly in new places, or focusing on further bird species are always valuable challenges.

A further Amateur Community work driver is the *audience* [2]. Challenge addressing, shows of mastery, are rarely kept for oneself; peers are normally expected to benefit from contingencies being negotiated, or members of the community *public* will benefit. A notable form of community contribution is *pioneering of new contingency spaces*, i.e try out and pursue new kinds of challenges, different from but related to what the amateur community used to do before. Worth noting here is that challenge is *socially constructed*, i.e. a new contingency space may be adopted by all or some community members, or it can be totally rejected as something that is not worth pursuing by the community.

5.1 Amateur software development

It is useful to characterize amateur software development such as hackers [13] or open source [11] as amateur communities because Tech Committee is a less professional (pre-professional) correspondent of such communities. In such communities, code challenges are hardly exhaustible, as programming a machine leads to a rich set of contingencies, bringing along lots of 'trial and error'.

It is also easy to open new contingency spaces partly due to the immateriality of the working artifact, new projects, or project modules can always be started, or the existing ones can be re-organized ("refactored" in the professional jargon). For a newcomer, the complex architecture of large software projects adds to the challenge, and their skills of mastering the programming language will not be enough, they will also have to learn the way the project is set up.

Code is shared typically via plain text messages, and carefully examined before being committed to the code repository, which suggests another amateur community feature, related to the notion of audience, the *peer review* of challenge addressing. Plain text is regarded as a very suitable medium for such distributed development and review efforts [22]. Another important part of the community audience is its *public*, which can be a crucial motivation factor: for very generic projects it may be the world at large (e.g. in the case of an open source operating system kernel, an office suite), or very specialized professionals and amateurs in the software development field (e.g. for an open source compiler suite).

6. SUSTAINABILITY OF PRACTICE IN AMATEUR COMMUNITIES

Based on the theoretical considerations made on Amateur Communities and Communities of Practice, we can now consider in more detail the issue of sustainability. First, practice being crucial to the community, sustainability of a community appears to be inextricably linked to the sustainability of its practice: a community will thrive if its practice can thrive. For that, in an amateur community, its challenges should not be easily exhaustible, yet they should also be addressable by newcomers. Furthermore, there should, at all times, exist interesting challenges for both newcomers and old-timers. In short, the features of "healthy contingency spaces" are essential ingredients for sustainability of

[1] www.confluence.org

[2] www.operationmigration.org

practice in Amateur communities. But most of all, besides challenges to address, and skills to match them, there should exist enough members who take these challenges. A healthy, sustainable amateur community, will have enough members at all the milestones on "learning paths", i.e. at all the important learning thresholds on these paths.

7. MAKUMBA, THE TECH COMMITTEE TOOL

Makumba was designed with influences from Lotus Notes, but without the visual programming approach that was found to be troublesome in the distributed Tech Committee setting; instead, all code is in plain text (cf. [22]). As another difference from Lotus Notes, Makumba was designed to offer as few features as possible to its user (i.e. a programmer) and this minimalsitic design was thought to facilitate learning.

Makumba organizes a system in three parts. The *Makumba Data Definition* (MDD) describes data structures and relations between them. MDDs are simple lists of data fields, with name and type; an example can be seen in Listing 1 for the data type "Student". The *JSP level* is technically a tag library for the Java Server Pages dynamic web page technology, which allows displaying and changing the data stored in a database and described in MDDs. Finally the *business logic* (BL) describes, in the Java programming language, "business rules" that restrict the changing of data, and provide authentication and authorization mechanisms. The JSP level, illustrated in Listing 2, provides for a combination of HTML and a subset of SQL (Structured Query Language). The example shows a list of students and, for each student, their completed studies. The data for this list is described in the "Student" MDD shown in Listing 1. Notably the SQL data combinations (so-called joins) and data selections (so-called projections) are generated automatically from notations like "s.education" and "s.surname" respectively. Therefore working with a Makumba JSP would typically not require as much knowledge as using SQL by itself.

```
name = char[50]
birthdate = date
hobbies = text

education = set
education->name = char[50]
education->university = ptr University
```

Listing 1: Makumba Data Definition "Student"

```
<mak:list from="Student s">
  Name: <mak:value expr="s.name"/> <br/>
  Born on: <mak:value expr="s.birthdate"/><br/>
  Completed studies:
  <mak:list from="s.education e"
            where="e.graduationDate < now()">
    <mak:value expr="e.name"/>
    (<mak:value expr="e.university.name"/>),
  </mak:list>
  <br/>
</mak:list>
```

Listing 2: Example of viewing data with the Makumba JSP tag library

Year	MDD	JSP	BL	# files	LOC	CVS
2002	42	676	80	801	78479	N/A
2003	52	961	116	1132	104805	1143
2004	64	1208	140	1415	127873	702
2005	76	1354	190	1628	151801	1324
2006	99	1719	229	2062	175315	1632
2007	111	2135	287	2559	219456	2391
2008	114	1860	289	2304	196867	1898

Table 1: Size of the Intranet

Through this design, the assumption was that it will be easy for students to read MDDs, and based on that, it will be possible for them to combine HTML and a subset of SQL (itself based on natural language) which they might know prior to start working with Makumba or might learn "on the job". Later on, they would become interested in writing new business rules, and might learn procedural Java for that, maybe using previous knowledge of another procedural programming language. This then would be the "learning path" assumed for a Makumba practitioner.

Besides the design of the Makumba framework itself, a web-based tool was devised where potential members can experiment with modifying existing code, work with new code, and look at each other's code, using example Intranet data and the necessary programming tools already set up [3]. Such 'sandboxes' were found to be important in cooperative design endeavors [19].

Initially Makumba was designed and developed under another, more neutral name ("Metadata"). When a more specific name was sought, designers remembered of a term that was well-known to many members of the BEST student organization: the Makumba party. The name was widely known but not the actual meaning because of an established ritual whereby members who had attended a Makumba party usually described it in very little detail, so the only way other members can find more about such parties was to attend them. Makumba-initiated BEST members described the parties as follows:

> "You enter a room, the lights are turned off and then somebody tells the Makumba joke."

The rest could only be learned by joining a party (and will not be described here either ...). Ironically, the main Makumba designer had never attended a Makumba party, but knowing about its existence and its fame within BEST was enough to decide on a name that would be connected to the BEST culture. As the word "macumba"[3] refers to a tribal religion, the aim was to achieve a technological religion for the BEST 'student tribe'.

8. SUSTAINABILITY OF THE TECH COMMITTEE

Let us first illustrate the Tech Committee activity in what we perceive as its sustainable period. Table 1 shows the evolution of the size of the Intranet since its launch seven years ago, detailed for the different technological levels MDD, JSP and BL. Additionally, the total number of code files

[3]Using the Makumba spelling was a matter of Internet domain name availability

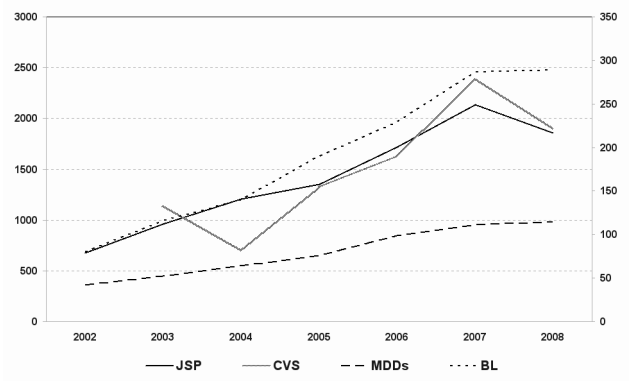


Figure 1: Size of the Intranet

is shown, along with the total lines of code (LOC). Figure 1 shows the evolution of the code figures graphically. It is worth noting that there was a rather high amount of data definition (MDD) files defined in the beginning of the Intranet, representing data used in the first three sub-systems. Even though more data definitions have been added over time, a certain saturation level has been reached in the last years, with only minor additions. It is not unexpected that the data definitions of an organization (its "domain model") change seldom after an initial development time. As different from that, the amount of Business Logic (BL) files has continuously increased, with a leveling in the last year. This leveling, as well as a JSP file number decrease, are due to the Tech Committee finding the time to clean up the code without adding functionality, but simplifying it in the process, thereby allowing future members to understand it easier, and making it possible to implement more powerful features over the new, simplified, code structure. Finding the time and resources for such cleanup is itself a sign of 'establishment', i.e. sustainability.

Table 2 illustrates the number of active members in the Tech Committee since 2003, differentiated on the different levels of Makumba, and on their activity in newcomer ("peripheral") and old-timer ("full") members. This data was extracted on the one hand from the latest questionnaire filled in by the members, and then refined and amended by some of the committee coordinators (for the year 2003, only the total number of active full and peripheral members could be estimated). It is interesting to note that while the overall number of active members increased over the years and stabilized in the last years, the (relative) number of members working on data definitions (MDD) was rather decreasing; this might be explained with the arrival at data maturity level as described above, and thus less work and challenges to take for the Tech Committee members, while plenty of challenge is left at other levels. An important sign of sustainability as we characterized it earlier is the sufficient number of members at both novice (peripheral) and expert (full) members active at all the Makumba usage levels at all times. As Table 2 shows, the "learning paths" are well populated with community members.

8.1 Survey

A survey was conducted among current and former members of the Tech Committee in the end of 2008, and filled in by 30 members. The time of Tech Committee membership ranges from their activity starting around 2001, to fresh members that just joined in the second half of 2008. Also the educational background is diverse – 13 members study computer science, computer engineering or informatics; the other respondents follow various other curricula, ranging from mechanical engineering to physics and biomedical engineering. Also, the membership duration in the committee varies, from 6 members who were active for around four years, to members who were (or currently are) active for one to two years. Most members had no knowledge of Makumba before joining.

Year	MDD		JSP		BL		Active	
	Per	Full	Per	Full	Per	Full	Per	Full
2003							4	5
2004	3	4	7	3	3	3	6	5
2005	5	6	8	11	7	3	8	13
2006	7	8	9	12	9	3	10	12
2007	7	7	14	9	10	5	12	11
2008	6	5	13	7	6	7	12	8

Table 2: Members in the Tech Committee

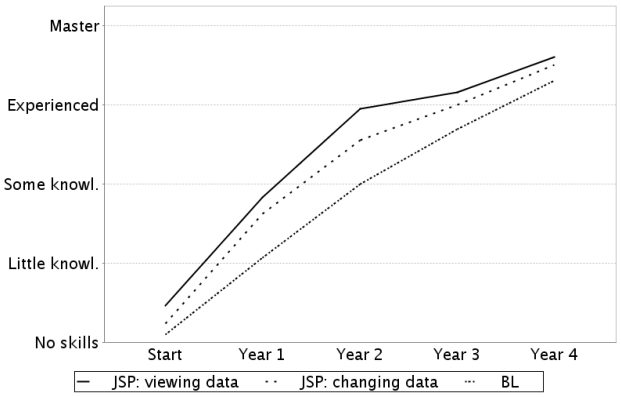
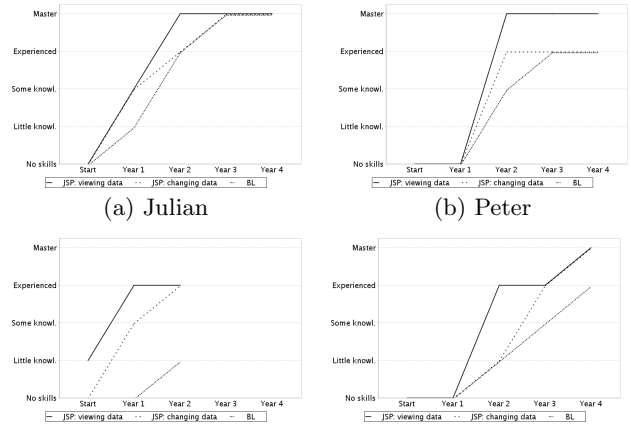


Figure 2: Learning curve for different aspects of Makumba

The questionaire consisted of 53 questions. The main focus was on gathering an understanding of the learning aspect of Makumba in the context of the Tech Committee, thus the participants were asked to answer several questions on their self-assessed skill level on database design, SQL, general programming, Java, and three aspects of Makumba: JSP pages to view data, JSP pages to change (create, edit, and delete) data, and business logics (BL). The questions were asked repeatedly for several different points in time, namely when joining the committee, and after the first, second, third and fourth year of membership, and allowed answers from "no skills" via "little knowledge", "some experience" and "experienced" to "master". To balance the self-assessments and to facilitate recall, the participants were further asked to estimate their contribution to the committee during the specific year, and to list the projects and technologies that they worked with in that year. Also to improve the quality of the self assessment, members were informed that the committee coordinators from their time of activity will review their answers.

Figure 2 shows the averaged skill levels of all the members at different points in time, separated for three Makumba us-

Figure 3: Skill development of selected survey respondents



(a) Julian (b) Peter

(c) Kim (d) Sam

age aspects: JSP for viewing data, JSP for modifying data, and BL. It can be observed that apparently, learning to view and display data in JSP level is faster than handling forms to create, edit, and delete data, while Business Logics are the most difficult concept, especially in the first two years of membership. Most members who stay in the committee for a longer period, however, master all levels of Makumba almost equally well, but that is not a case of challenge exhaustion as they can (and many do) continue working on the internals of Makumba itself as a next step. While MDD editing skills are less and less actual, we can consider that currently the learning path milestones in the Tech Committee are: JSP viewing data, JSP changing data and BL. To further illustrate the skill evolutions, we have selected the charts of four individual members in Figure 3. This also shows that, although the dynamics vary a lot, the JSP data-changing skills almost always lag behind the JSP data view skills, and BL skill is acquired last. Overall, the per-year self-assessment results suggest that the learning path designed into Makumba has achieved its goal, allowing members to have an easy start by visualizing data in JSP (which also requires MDD understanding), then starting to change data and finally learning how to write business rules in Java.

Two open questions concluded the questionnaire. We will comment them briefly here, and elaborate in the discussion. The first question aimed at helping to assess how the members experience learning technologies in the Tech Committee. Several respondents emphasized the *simplicity* and conciseness of Makumba code, therefore easiness to start working with, in various ways: as an intuitive link between code and result, or as a praise to simplicity perceived as being able to express things in only one way, with a direct reference to people coming into and leaving the Tech Committee, or as a technology that provides for rapid achievement, thereby encouraging further involvement. It is important for the Tech Committee in the international meetings to 'catch' people fast, by allowing them to realize that they can do something, that they can contribute. If this would not happen fast, the prospective new member might lose interest and maybe try to join another international team where contribution is more facile.

> "I would say that the code is very close to the result ('what you code is what you get'), or at least, very logically connected."
>
> "One big advantage is the syntax, which makes it straight forward to understand other people's work. With Makumba, it doesn't happen so often that 2 people write totally different pieces of code for the same feature, and this is very useful for BEST (where people change very often)."
>
> "anybody is able to get a working page after 10 minutes of theory and 5 minutes of practice. That's provide motivation to do more difficult features."

A respondent emphasizes that "there is life" beyond the first simple steps, i.e. there are more complex things to be done, and Makumba accommodates for these as well. Several respondents compare Makumba with other Web development frameworks in this regard. More than one respondent referred explicitly to "non-IT people", thereby emphasizing the suitability for amateur settings.

> "[...] allowing to reach very quickly decent results with its basic features (allowing new joiners not to get tired of the learning effort and to contribute quickly) while also providing more powerful stuff for advanced users (allowing them to still find new challenges)."
>
> "All the other frameworks and technologies I know (Struts, PHP + templating engine, Zope) require a higher degree of computer technologies knowledge to achieve similar results. This might be changing with new frameworks like Ruby on Rails in which the model definition is done in a similar way as in Makumba, but still I think Ruby on Rails' learning curve is steeper than Makumba's"
>
> "Makumba being specifically designed for web applications, it is both extremely simple to learn for the basics - easier than PHP, especially for non-programmers - mainly through the easy interactions with databases; yet can be as powerful as any other language"
>
> "But the main advantage is that everyone *can* learn, without too much of a technological background. I think that for [Tech Committee], or for any organization who wants to involve non-IT people in the development of their applications, the use of Makumba can make things much easier."

Respondents refer to their learning primarily from other members, sometimes even without explicit training and without even mastering the basics of the entry-level skill (HTML and SQL). Also learning by reading a mailing list is mentioned. Other informants report having asked questions on the Tech Committee mailing list and getting a multitude of good answers, also usable in the professional life of the then-amateur.

> "I didn't attend any organised Makumba training, but learned the basics during events, in the first year. It was quite easy, although I didn't even know HTML or SQL when joining [the Tech

Committee]. On the first 1-2 projects, I was directly helped by [the Tech Committee]'s experienced members. Afterwards, I started following also the emails and learned a lot from other people's problems/experiences."

"The [Tech Committee] community is helpful, it is very easy to learn new technologies. I think makumba gives a very good starting point for learning web programming (separation of concerns, data model design). Everything that I learned in [Tech Committee] has proved to be true also in the professional web development world."

Learning by reading examples (existing code) is emphasized by most respondents. Knowing how the Intranet is organized helps to find relevant code:

"Learning is easy thanks to a lot of examples: for most of the tasks I could find another place where a similar feature had been implemented. Having a good knowledge about BEST IT systems helps more than having strong programming skills. Other [Tech Committee] members also very often help and give advice when a problem or mistake occurs."

A problem with learning Makumba is raised: since it is little known outside the student organization, there is no chance that some joining members already come in the Tech Comittee with the skill:

"The main disadvantage is that everyone must learn it, since it is quite different from other web technologies."

The last question asked the respondents for a comparison of Makumba with other technologies and frameworks aimed for web development. Many of the former or present members of the Tech Committee, currently working in IT jobs, were able to provide comparisons from a qualified professional perspective, with references to related technologies. Notably a number of members make architectural remarks, referring to some technical imperfections of Makumba, which does not provide a good separation between data ("model"), its visualizations ("view") and the data change mechanisms ("controller"), within a widely accepted professional paradigm, called Model-View-Controller. However most respondents regard this as an asset in the end, leading to easier programmer access to the data and higher maintainability.

"Mak is significantly more suitable for [the Tech Committee] than anything else I've seen, despite the horrible inflexibility of the J2EE platform. Most other frameworks/setups insist on a very clear separation between the model and the view, and in most situations, this makes sense. However, by moving certain typically controller-world operations, such as database queries, to the view **in a simple and accessible way**, pages become much more maintainable without very strong documentation discipline, because they somewhat self-document."

"atm. I can't think of frameworks that do mix view and model, i.e. view and data query. they all go through a DAO layer, which would break the 'what you code is what you get'. other frameworks also do have a lot of configuration overhead and rely on Java, which make it difficult for beginners to grasp. e.g. Struts needs to have several files changed in order to get a form to work, which is not so straightforward and slows down the 'try and error' process."

"Technology-wise, I love Makumba as how it's designed – the whole way the view/controller separation is less sharp than in other frameworks, how this is made very accessible and maintainable using the Makumba tag library, and how in the end this makes a large intranet maintainable by a community of people of variable skills."

9. DISCUSSION

We should first mention that we cannot deny the merits of the non-technological aspects of the Tech Committee work and organization that contributed to its sustainability: the managerial skills of its leaders and experienced members, or the well-organized mentoring of the newcomers, are all aspects that surely help sustaining the Tech Committee practice. However, our data (including explicit statements from members of different generations) shows that Makumba's learning-oriented design and the meanings associated with Makumba in the student community play a crucial role in the Tech Committee sustainability.

Our data shows that designing a programming technology that allows member activity at various levels of skill, thereby prescribing a learning path for the members of the amateur community of practice, allows for sustainability of the amateur programming practice. In the case in point, we saw that the 'learning departments' of the setting, corresponding to the different programming languages used, were well populated at all times, in all our annual counts, even if members left the setting every year while others joined. We can recommend as a design implication to design 'skill-modular' technologies, with 'modules' that require different levels of skill, with the first such module having a low learning threshold (HTML and simplified SQL-queries in the Makumba case), in order to provide for challenge addressability, and the next modules providing new challenge levels (e.g. BL in the Makumba case, with the procedural Java skill to be learned), thereby helping towards challenge inexhaustibility [4]. Such learning modules with new challenges also provide for progressing from the periphery to fuller membership, thus molding themselves well on Community of Practice social learning. The technology can then, as we believe Makumba does, be part of the *configuration* process emphasized by Lave and Wenger [12].

Especially for the entry-level skill modules (the JSP level of Makumba), *code legibility* is an important design feature for an amateur programming technology, in such a context of learning in doing, from peers, or from the artifacts they produced. Most of our respondents mention looking at a working example when performing a task, and this is not unexpected in a Community of Practice. A major part of the Makumba JSP level simplicity and legibility is inherited from the corresponding feature of SQL queries, as legibility and intuitiveness were among its design principles, since it is based on natural language. Members who do not know SQL simply have to rely on their English to find their way around

initially. This is further helped by Makumba JSP using a simplified form of SQL queries. Using plain text code rather than visual programming further helps the sharing and understanding of existing examples [22]. We will also emphasize here the importance of an *amateur programming experimentation* (cf. [4] for experimentation with amateur radio equipment). Few programmers work on the actual 'officially running' system, instead they work on a copy of it running with some example data. The lower the skill available, the more important the space for experimentation becomes, as a space for making initial mistakes, trial and error, and low-risk attempts to find solutions. It is difficult to imagine what Makumba programming would have been without the Tech Committee members' sandbox [3], i.e. without a possibility for them to easily experiment, make mistakes, and to repair them by looking at the work of their peers, or by asking questions to peers.

We were however not satisfied when discussing our data that this theoretical 'skill modularity' picture captures the Tech Committee sustainability entirely.The first author, who proposed it, has never actually been active in the sustainable period of the Tech Committee, and has coded little of the the Makumba Intranet. It was the second author, along with former and present Tech Committee leaders during its sustainable days, who pointed out several other aspects that lead to sustainability, some of which were confirmed by questionnaire respondents. First, BEST student organization members regard Makumba as being something that *belongs* to them but also something that they give out to the world, therefore it is a way for them to relate to a *public* [18], or a wide *audience* [2], thereby increasing amateur motivation. Entering the Tech Committee gives one a privilege to work with a technology that is unique, and designed by previous Tech Committee generations. Entering the Tech Comittee also means being part of the team who had in 1995 an automated online application system at a time when many companies barely had a static website. While such an early achievement is not directly related to Makumba, it emphasizes the Tech Committee long-lasting tradition in the area of dynamic Web applications, Makumba's application domain, and is supplemented by other similar *war stories* [14] that have a role in learning, but also in creating a sense of belonging. Furthermore, entering the Tech Committee often means attending a Makumba party, one of these mysterious events that other people talked about but never wanted to give details. This reminds us of an *initiation rite* [20] and plays a role in attracting 'new blood' to the community. In retrospect, choosing the Makumba name in favor of other, more technology-oriented but less student-community-oriented names was an inspired act. This teaches us that, while skill modularity has its merits, it can always use complementation from cultural meanings ascribed to the technology within the community: traditions, war stories, rituals, world-unique specificities. An implication then is to try to link the technology with community meaning. If this was possible to achieve (albeit somewhat accidentally) for such an intricate thing as a web development framework by a simple act of baptizing, it could work for many other technologies.

The relative uniqueness of Makumba to the student organization has an interesting consequence: since it is not a well-known technology, there are very few Tech Committee members who already know the technology when they join the committee, so they have to learn it when they join, at the same time as they come in contact with the traditions and rites associated with it. As there are not many resources dedicated to Makumba on the Internet, they have to resort to their fellow Tech Committee members, thus being 'forced into socialization', and thus fueling peer learning.

Reflection on Makumba as a technology for *pre-professionals* [18] leads us to two apparently contradictory considerations. On the one hand, Makumba imposes a professional rule, preventing the amateurs from mixing business rules with data views. The two are programmed in very different languages, requiring different skills, and different skills levels, making it difficult for the amateurs to even attempt to mix them. Many technologies in use by amateurs (like PHP) do not enforce this separation, leading to lack of *code scaleability* of application development: applications can be started fast, like with Makumba, but once they grow large, problems start to occur due to not enforcing this major principle, affecting in the end sustainability. On the other hand, another professional rule, the separation of data view from data access is violated by Makumba, as emphasized by several former members, currently professionals. Ironically, the two rules are part of the same professional design pattern, the Model-View-Controller. *Selecting relevant professional principles* is thus yet another design implication that we draw for amateur technology, and as the example shows, it can lead to fine-grained decisions. In other words, even if the professional counterpart is a potential role-model to be followed by the amateurs [18], not all its aspects will be useful in the amateur programming setting. We can draw a simple consequence of this implication: many professional technology designer use "more is better" as a principle in designing a feature set. In an amateur community, like it happened for the Tech Committee with Lotus Notes, members trying to employ such a technology would have a hard time choosing the feature they need. The small number of Makumba data types as well as the small amount of keywords in languages used (HTML, SQL-query) come to suggest that "more is better" is not a good professional principle to select, as it doesn't fit the amateur learning situation, where members need to get involved (and get some sense of achievement) fast.

A further feature that we can recommend for amateur programming technologies, related to code legibility, is to provide for *easy code navigation*. As emphasized by some respondents, Makumba has fewer types of files than professional technologies in the same application domain, therefore requiring less navigation between files when performing a task. Although this sometimes breaks the professional principle of separation of concerns (like data access concern and data view concern, as illustrated above) it helps the novices to make easier progress, and achieve better orientation when contributing to a large project like the Intranet. Furthermore, this source file compactness, especially at the JSP level, leads to little interdependency between the JSP source files (i.e. changing one such file is not prone to affect the working of many others), which helps when several of amateur programmers work on different parts of the Intranet, as they are not prone to affect each other's work, and thus the likely mistakes that a novice makes will not affect his or her peers. Such *team scaleability* is ideal for amateur programming sustainability.

10. CONCLUSIONS

We have described the sustainability of an amateur programming group, and examined it as an Amateur Community and as a Community of Practice. Based on this framework and on data collected over several years, we proposed several technology design implications for achieving sustainable practice in communities. We suggest that technologies should be learning-oriented, more precisely they should be organized around skill modules. We further suggest that technologies should be associated with community traditions, rituals and specificities. Finally, we suggest that a careful selection must be done from among the principles used within professional counterpart of the amateur community. Such a selection should also be made with learning in mind: professional principles can be broken at the expense of easier learning, yet other professional principles must be enforced to ensure long-term sustainability.

11. ACKNOWLEDGMENTS

Thanks to the students, members and associates of the Tech Committee who have worked hard to overcome Makumba imperfections and to make the Intranet what it is today, while also taking time to answer our surveys. Thanks are also due to all the Makumba contributors. Yngve Sundblad and John Bowers have supervised this work with good advice during the crucial Makumba design phases.

12. REFERENCES

[1] E. Blevis. Sustainable interaction design: invention & disposal, renewal & reuse. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 503–512, San Jose, California, USA, 2007. ACM.

[2] C. Bogdan. *IT Design for Amateur Communities*. PhD thesis, Royal Institute of Technology (KTH), Stockholm, 2003. `http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-3470`.

[3] C. Bogdan. Longstanding success without awareness support: lessons from a collaborative programming tool. In *Proceedings of the International Conference on the Design of Cooperative Systems (COOP 2008)*, Institut d'Etudes Politiques, Aix-en-Provence, France, 2008.

[4] C. Bogdan and J. Bowers. Tuning in: Challenging design for communities through a field study of radio amateurs. In *Proceedings of the Third Communities and Technologies Conference*, pages 439–461, Michigan State University, MI, USA, June 2007. Springer.

[5] A. Clement and P. V. den Besselaar. A retrospective look at pd projects. *Communications of the ACM Special issue on Participatory Design*, 36(4):29–37, 1993.

[6] M. Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. Harper and Row, New York, 1990.

[7] G. A. Fine. *Morel Tales. The Culture of Mushrooming*. Harvard University Press, 1998.

[8] M. Goodwin. Nine principles for making virtual communities work. *Wired 2.06*, pages 72–73, June 1994.

[9] J. Greenbaum and M. Kyng. *Design at work: cooperative design of computer systems*. Lawrence Erlbaum, Hillsdale, NJ, USA, April 1991.

[10] F. Kensing, J. Simonsen, and K. Bödker. Participatory design at a radio station. *Computer Supported Cooperative Work*, 7(3-4):243–271, 1998.

[11] P. Kollock. The economies of online cooperation. In M. Smith and P. Kollock, editors, *Communities in Cyberspace*. Routledge, London, 1999.

[12] J. Lave and E. Wenger. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, September 1991.

[13] S. Levy. *Hackers: Heroes of the Computer Revolution*. Penguin Books, New York, 1994.

[14] J. E. Orr. *Talking about machines: An Ethnography of a Modern Job*. Cornell University Press, 1996.

[15] D. Pargman. *Code begets community. On social and technical aspects of managing a virtual community*. PhD thesis, Linköping University, 2000.

[16] D. Pargman. Virtual community management as socialization and learning. In *Proceedings of the Second Communities and Technologies Conference*, pages 95–110, Milano, Italy, 2005.

[17] D. A. Schon. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, 1983.

[18] R. A. Stebbins. *Amateurs. On the Margin Between Work and Leisure*. Sage Publications, 1979.

[19] R. H. Trigg. From sandbox to "fundbox": Weaving participatory design into the fabric of a busy non-profit. In *Proceedings of the Conference on Participatory Design*, pages 174–183, Palo Alto, CA, USA, 2000. Computer Professionals for Social Responsibility, New York, USA.

[20] A. van Gennep. *The rites of passage*. Routledge & Kegan Paul, London, 1909/1960.

[21] E. Wenger. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge University Press, 1998.

[22] Y. Yamauchi, M. Yokozawa, T. Shinohara, and T. Ishida. Collaboration with lean media: how open-source software succeeds. In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 329–338, Philadelphia, PA, USA, 2000. ACM.