

Workplace Warriors: Identifying Team Practices of Appropriation in Software Ecosystems

Sebastian Draxler
University of Siegen
Hoelderlinstr. 3
57076 Siegen
+49 (0)271-740-4763

sebastian.draxler@
uni-siegen.de

Adrian Jung
University of Siegen
Hoelderlinstr. 3
57076 Siegen
+49 (0)271-740-4036

adrian.jung@
uni-siegen.de

Alexander Boden
University of Siegen
Hoelderlinstr. 3
57076 Siegen
+49 (0)271-740-4763

alexander.boden@
uni-siegen.de

Gunnar Stevens
University of Siegen
Hoelderlinstr. 3
57076 Siegen
+49 (0)271-740-4036

gunnar.stevens@
uni-siegen.de

ABSTRACT

Since the 1990s, the forms of production, distribution, configuration and appropriation of software have changed fundamentally. Nowadays, software is often embedded in *software ecosystems*, i.e. in complex interrelations between different stakeholders who are connected by a shared technological platform. In our paper, we investigate how small software teams deal with the challenges of appropriating and configuring software in the Eclipse ecosystem for their daily work. We empirically identify three different approaches for dealing with appropriation in software ecosystems which are represented by the “ideal types” *lone warrior*, *centralized organization*, and *collegial collaboration*. Based on a discussion of these strategies and the underlying appropriation practices we found in the field, we suggest theoretical and practical implications for supporting appropriation in software ecosystems.

Categories and Subject Descriptors

J.4.4 [Computer Applications]: Social and behavioral sciences – sociology. H5.3 [Information Interfaces and Presentation] Group and Organization Interfaces –Computer Supported Cooperative Work.

General Terms

Management, Human Factors.

Keywords

Tailorability, Appropriation, Software Engineering, Software Ecosystems.

1. INTRODUCTION

For a long time, research on software usage has been interested in aspects of customizing and tailoring single applications to the needs of end users [5,8,16]. This was well suited in times when the software market was limited and applications had a clear border. However, several trends have changed the ways of software production, distribution and configuration considerably over the last two decades: 1.) the establishment of the Internet as the dominating infrastructure for disseminating and marketing digital goods; 2.) the spread of new business and development

models in the Open Source domain, encouraging users to share software with others; 3.) the establishment of software ecosystems which consist of different manufacturers and hobbyists, creating small-scale components that can be individually assembled by end users. These developments do not only affect the ways how software is developed and maintained, but also how it is appropriated by the users (who are increasingly understood as prosumers instead of mere consumers of software). As the social and collaborative aspects of appropriation in software ecosystems are not well understood, we have conducted a study which investigated how small teams of software developers deal with the necessities of appropriating the software that constitutes their work tools—in this case the Eclipse IDE and its surrounding software ecosystem [3,12].

The paper is organized as follows: after a discussion of the related work and our methodology, we provide an overview on three different approaches of dealing with appropriation as well as the underlying practices and implications that constitute these strategies. Based on our findings, we then identify implications for the theoretic understanding of appropriation as well as for designing supportive tools for these practices.

2. A BRIEF HISTORY OF APPROPRIATION

In the 80s and 90s, the research on the social and socio-technical aspects of software use and development coined the terms *customization*, *adaptability* and *tailorability*. Driver of these discussions was the fact that the existing, monolithic off-the-shelf products did not accord well with the complex, heterogeneous needs of a dynamic market. While tailoring was often considered to be an individual effort, later research highlighted the collaborative aspects of these practices. A major contribution to this shift was the work of Mackay et al [7], who conducted empirical studies within organizations to examine patterns of sharing self-created email filter rules as well as configurations of Unix desktop systems. In a similar vein, Gantt and Nardi [5] investigated into the tailoring practices of CAD users. Under their lens, collaborative efforts became visible. While tailoring has been mainly discussed from a technical perspective, recent research has broadened the understanding of the corresponding practices by analyzing the ways how users fit the technology at hand into both the pre-existing culture and into the local patterns of use and life rhythms [13]. Conceptually labeled as *appropriation*, this line of research highlights the link between the creative re-configuration and the re-interpretation of given features as two dialectically connected forms of end user development [2,11], implying that technical concepts as tailorability (in terms of re-configuration of software) should be

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHASE'11, May 21, 2011, Waikiki, Honolulu, HI, USA.

Copyright 2011 ACM 978-1-4503-0576-1/11/05... \$10.00.

supplemented by social-technical means for supporting appropriation [2,10,11]. Generally speaking, while tailoring focuses on the technical customizations of software artifacts, appropriation also considers the social-technical process of interpreting software in daily work practice. While research in this domain has predominantly focused on the appropriation of single applications so far, we want to understand how software users design their workplaces by assembling heterogeneous resources from global software ecosystems.

3. RESEARCH METHODOLOGY

From 2006 to 2009, we participated in a publicly funded research project to develop tailorable applications for different domains (like monitoring complex technical plants, eLearning based on business simulation and using basic groupware technology). The aim was to study the potential of the Eclipse platform [4] as a core technology for realizing the concept of component based tailorability [9]. Eclipse was chosen as a platform as it was considered being a leading edge technology in respect to a consequent component architecture (“everything is a plugin” [4:83]) and a vital ecosystem (with thousands of free or commercial extensions). We expected these features to empower users to personalize their applications by managing individual plugin portfolios, selected from the Eclipse software ecosystem.

Our research was organized following the strategy of theoretical sampling as suggested by Strauss’ Grounded Theory [6]. We started with an open-ended, qualitative study on Eclipse plug-in adoption practices, using semi-structured interviews and on-site observations. Specifically, we cooperated with five German companies with 10 to over 2000 employees that perform software development (see table 1 for details). In each company, we conducted at least three semi-structured interviews of at least one hour (altogether, we conducted 18 interviews between August 2007 and December 2008). Additionally, we visited two SMEs over a period of 3-5 days for on-site observation.

	Employees	Domain	Empiric data
A	250	Development of Multimedia and Web applications	5 day observation 5 interviews
B	10	Software Development	3 interviews
C	6 + free-lancers	Software Development	3 interviews 3 day observation
D	1200 (90 developers)	Insurance company + inhouse software dev.	3 interviews
E	>2000 (30 maintenance dept.)	Research facility, Inhouse development for. Machine maint.	4 interviews

Table 1: Research Partners

All interviews were audio recorded and transcribed. The participant observations were documented by means of field notes taken during the research. In addition, we also analyzed the socio-technical structure of the Eclipse ecosystem, which determines the possibilities of users for designing their personal workspace. Related work done by other researchers was investigated for the purpose of sensitizing our analytic lens. However, we tried to prevent to subsume our observations under pre-defined concepts taken from literature. Instead we focused on categorizing our data by carefully analyzing the documented practices in minute detail.

In a further step, we supplemented our qualitative analysis by an online survey where we used to assess the spreading of the observed practices [12].

4. A CLASSIFICATION SCHEME OF TEAM PRACTICES

Our research provided us with rich insights into the manifold practices of how practitioners of small software teams appropriate software in the context of the Eclipse ecosystem. In general, we observed that practitioners were in a constant struggle of innovating their working environment and keeping it up-to-date in order to perform their work. At the same time they had to prevent possible breakdowns and incompatibilities resulting from updated and new plug-ins. As common Eclipse configurations consist of a plethora of plugins (usually maintained by 3rd party developers), there was a constant need for dealing with updates of particular plug-ins, also in cases when no new plugins were to be installed. Even when developers decided to stick to one version of Eclipse as a stable working environment, APIs to shared tools like the CVS could change and enforce the update of the related plugins. Furthermore, new tasks did sometimes require new tools and therefore new plug-ins were explored and introduced.

The companies we investigated expressed quite different strategies of dealing with the challenges outlined above. These approaches move along a continuum, whose ends are *autonomy* (leaving the responsibility to configure and maintain Eclipse to the individual developers) and *heteronomy* (managing Eclipse configurations in a centralized manner, thus enforcing a homogeneous working environment for the whole team). Following Max Weber [15], we conveyed and conceptualized these approaches into three different “ideal types”, which represent the underlying strategies of the companies for managing appropriation in practice: the *lone warrior* type, the *centralized organization* type, and the *collegial collaboration* type. In the next sections, we will discuss these types and illustrate their underlying practices as well as their socio-technical implications.

4.1 Lone Warrior

The first type is conceptually constituted by a high autonomy of the team members. It refers to the freedom, but also to the individual responsibility workers have for taking care of their workplaces. There are no guidelines for setting up the individual Eclipse configurations, although the limitations of the platform may force team members to use very similar configurations for certain collaborative functions, e.g. if only one plugin is available for connecting to the version control system. If changes to the workplace configuration need to be introduced on the team level, practitioners have to justify this constriction of the individual autonomy by proving the benefits of the suggested change to each of his colleagues.

Empirically, the lone warrior practice is expressed by the circumstances that collaborative appropriation is not the normal case, but occurs at most accidentally (e.g. in response to breakdowns of the infrastructure). In our study, an example for such an accidental collaboration was caused by the breakdown of a shared source code repository that was caused by an update. In this exceptional case, the incompatibility forced the team to search for a solution that was then adopted by the whole workgroup (in the specific case, a developer worked out that the problem could be fixed by using a different Eclipse plug-in. This information was spread and the plug-in was then used by everyone). Except of such rare situations of collective appropriation of new tools, every

developer decided on his own what plug-ins to install and how they should be configured.

As the lone warrior culture forced team members to search for useful tools themselves and experiment with them, practitioners were often very knowledgeable about their individual workplace. On the other hand, we recognized a lack of group-consciousness regarding tools and existing tool expertise. For example, in one company no one in the group was able to tell us what tools his colleagues were using. From a management perspective, this lack of awareness can have negative effects, like the tendency to develop heterogeneous and possibly conflicting work practices, decreasing the diffusion of innovation among the team members, and finally making it more difficult to find the right person to ask for advice in case something went wrong or in case team members evaluated new tools [1].

4.2 Centralized Organization

This type is constituted by heteronomy that stems from a centralized organization of the Eclipse installation. In this case, the workplace is designed by a designated team member and then distributed to the team. Changes to the common installation have to be discussed with the developer in charge, who is responsible for updating the installation and dealing with the possible incompatibilities.

Empirically, we observed one case of centralized organization as the opposite extreme to the lone warrior type. In this case, the team elected a *tool care taker*. Once a month, his duty was to configure a stable working environment that met the needs of the group and afterwards put it on a shared file system where it was accessible for all team members. He described this work as follows:

“Developer 2: [...] I sit down at home on my Vista partition and an up-to-date Eclipse [configuration]. [...] I have a list of plug-ins which should be included. [...] I update this at home, put it on an USB stick and bring it here. Here, I upload it to a shared drive so that everybody can copy it.”

(Transcript from an interview with the tool care taker).

The decisions about what should be included in the common configuration were made by the whole team during stand-up meetings. Although the decision process was collaborative, the *workplace caretaker* realized the common working environment on his own. He was expected to be interested and quite skilled in finding new tools as well as in being up-to-date. Team members even reported to respect him as a “*primus inter pares*” in this regard. Everyone else within the team relied on his skills to regularly set up an up-to-date, stable environment for the whole team.

The approach of centralized organization did not just create the opportunity to benefit from the maintenance of a common and thus homogeneous workplace configuration, but also created a latent obligation for other developers not to customize their workplaces. As experimenting with new tools for personal purposes was not legitimized by the team, the amount of influence team members could bring into discussions about the configuration management was clearly limited. The reason for this was, that it is difficult to say what the concrete benefits are before one has used a new tool in practice. Hence, the central organization type had the tendency to impede individual initiatives of developers to explore new stuff during daily work. It detached innovation activities from the daily production work. This means that the centralized organization type had to solve the

latent demand to innovate in different ways than the lone warrior case, which allowed easy exploration through the integration with the daily work, which was now mainly in the responsibility of one central team member.

4.3 Collegial Collaboration

As we have outlined in the two previous sections, our observations showed that in practice there were often exceptions from the rules impeded by the different ideal approaches that the companies chose to deal with their workplace configuration. This led to the identification of a third type, which is constituted by the dialectics between autonomy and heteronomy. In this type, responsibilities are not pre-defined, but have to be clarified in a situated manner. As a result, the used working environments were often neither as homogenous as a central organization, nor as heterogeneous as in a lone warrior culture.

Empirically, we found many different practices of sharing configurations and plug-ins in the team in an unplanned and ad-hoc way. These were similar to forms of learning that have been described in the literature as learning in over-the-shoulder situations [14]. Generally, sharing was mainly based on a copy-and-adapt installation practice, whereas one user configured and adapted her own Eclipse configuration individually, and afterwards shared this configuration with his colleague(s) in the project.

“I did configure Eclipse a few times. Or rather we configured it more or less together. For example, [within the project] we use the CheckStyle component and some other plug-ins, whose names I forgot, because in fact my colleague did set up the original configuration. And I eventually copied the whole configuration over to my workstation.”

(Transcript from an interview with a junior developer).

An important type of situation was a kind of initiation rite in which experienced Eclipse users introduced new team members or novice Eclipse users to the Eclipse technology (e.g. if a developer joined the company and/or the team). In such situations, senior developers would often copy their working environments to the machines of the junior developers. One observed effect in such situations was that people worked with similar configurations for a certain time. Later, the working environments would drift apart in a lone warrior manner, as everyone modified the environment for his own purposes.

The result of such a working style was that the selective cooperation fostered homogenization to a certain degree, but did not enforce it in a centralized manner. At the same time, the related ad hoc collaboration did not result in a systematic diffusion of innovations. Instead, tools and experiences were typically shared in an unsystematic manner, which was also a result of very limited support by existing technology. Accordingly, we found several attempts to organize self-made infrastructures to overcome existing shortcomings in technology support.

“Developer: So, if the project requires to install something new and interesting, [...] I send a request using the mailing list: ‘Who knows a good plug-in for that?’

Interviewer: What do you mean by ‘mailing list’? Who receives these emails?

Developer: [...] All employees receive this message and everyone who wants to comment can answer directly. [...] Some time ago, when we really wanted to use [a certain plugin], I had already heard about it on the mailing list. [...] And I directly contacted

the colleague, asking where I could get it and how I could use it." (Transcript from an interview with a developer).

In this case, employees utilized the company's mailing list to reach every colleague and to describe what tools they were currently using in a new project, as well as experiences regarding the usage of tools. This presents a notable example of creating a proxy for over-the-shoulder situations, coping with the challenge that several employees worked at the customer's site full time as project leaders, programmers and technology consultants. In particular, these people described that they felt better connected to their company colleagues because of this mechanism. While they considered themselves as part of the team, they also considered themselves as part of the periphery. Living in the diaspora they were afraid of being cut off from their colleagues at the headquarters, who were perceived as leaders in appropriating domain specific innovations.

5. CONCLUSION

In this paper we studied practices of appropriating software in the Eclipse ecosystem, asking how small teams of software developers manage their working environments. We distinguished between three different approaches of managing the workplace configuration. The first one was labeled the *lone warrior* type. It is defined by the concept of autonomy, while the second, the *centralized organization* type, is defined by the concept of heteronomy. Both types have in common that informal collaboration among team members is not a constitutive element. This distinguishes them from the third type of *collegial collaboration*, which is defined by *autonomy* and *heteronomy* as a dialectical unity.

It has to be noted, that the three types are analytic categories in the sense of Max Weber's "ideal types" [15]. In practice, their borders were blurred as we observed various forms of collegial collaboration in all cases, leading to a fined-grained balance between autonomy and heteronomy. Sharing occurred often rather unplanned in over-the-shoulder situations, in case of breakdowns, in situations where newcomers joined the team or just by accident. In these situations, the sharing of configurations and knowledge did lead to a temporal increase in awareness on what is used and homogeneity of configurations. Between such occasions, configurations usually drifted apart as users added new plug-ins or changed configurations to their needs, which led to an increased heterogeneity and a decreased awareness which could sometimes cause breakdowns in the cooperative work. We also observed attempts of overcoming existing shortcomings by realizing collegial forms of appropriation in distributed work. Examples were setting up mailing lists to share experiences about new tools in a more systematic way, or using the source code repository to also store the tools with the code.

Nowadays, most tools provide tailorability functions. Eclipse as our example can be tailored by adding plug-ins, setting preferences and changing the design of the user interface. However, our research showed a clear lack of tools for supporting the various practices of collaboration with regard to appropriating/tailoring. This is especially true for tailoring artifacts in complex software ecosystems like Eclipse, since modifications are usually delivered by 3rd parties. In such cases conditions as practicality, compatibility, stability can hardly be estimated beforehand, making the selection of tools difficult and risky. Therefore, further research is needed to explore how collegial collaboration can be supported with regard to appropriation. Such approaches should not only take into account

the provisioning models represented by the three ideal types, but also the dialectical unity of autonomy and heteronomy that framed the different practices we found in the field. In doing so, they should support tool awareness as well as practices of over-the-shoulder learning.

6. REFERENCES

1. Boden, A., Draxler, S., and Wulf, V. Aneignungspraktiken von Software-Entwicklern beim Offshoring - Fallstudie eines kleinen deutschen Softwareunternehmens. *Multikonferenz Wirtschaftsinformatik 2010*, Universitätsverlag Göttingen (2010).
2. Dourish, P. The Appropriation of Interactive Technologies: Some Lessons from Placeless Documents. *Computer Supported Cooperative Work (CSCW) 12*, 4 (2003), 465-490.
3. Eclipse Foundation. *Eclipse Community Survey 2010*. Eclipse Foundation, 2010.
4. Gamma, E. and Beck, K. *Contributing to Eclipse: Principles, Patterns, and Plug-Ins*. Addison-Wesley Professional, 2003.
5. Gantt, M. and Nardi, B.A. Gardeners and gurus: patterns of cooperation among CAD users. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (1992), 107-117.
6. Glaser, B. and Strauss, A. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Transaction, 1967.
7. Mackay, W.E. Patterns of sharing customizable software. *Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, ACM (1990), 209-221.
8. MacLean, A., Carter, K., Löfstrand, L., and Moran, T. User-tailorable systems: pressing the issues with buttons. *CHI90 Proceedings ISBN:0-201-50932-6*, (1990), 175-182.
9. Mørch, A.I., Stevens, G., Won, M., Klann, M., Dittrich, Y., and Wulf, V. Component-based technologies for end-user development. *Commun. ACM 47*, 9 (2004), 59-62.
10. Pipek, V. *From tailoring to appropriation support: Negotiating groupware usage*. University of Oulu, Oulu, 2005.
11. Stevens, G. Understanding and Designing Appropriation Infrastructures: Artifacts as boundary objects in the continuous software development. 2009.
12. Stevens, G. and Draxler, S. Appropriation of the Eclipse Ecosystem: Local Integration of Global Network Production. (2010).
13. Stevens, G., Pipek, V., and Wulf, V. Appropriation Infrastructure: Supporting the Design of Usages. In *End-User Development*. 2009, 50-69.
14. Twidale, M. Over the Shoulder Learning: Supporting Brief Informal Learning. *Computer Supported Cooperative Work (CSCW) 14*, 6 (2005), 505-547.
15. Weber, M. *The Methodology Of The Social Sciences*. Free Press, 1949.
16. Wulf, V. "Let's see your Search-Tool!" - On the Collaborative use of Tailored Artifacts. *Proceedings of GROUP '99, ACM-Press*, (1999), 50-60.