The Ecology of Participants in Co-Evolving Socio-Technical Environments

Gerhard Fischer[1], Antonio Piccinno[2], Yunwen Ye[1,3]

[1] Center for LifeLong Learning & Design (L3D), Department of Computer Science, University of Colorado, Boulder, USA,
[2] Dipartimento di Informatica, Università di Bari, Bari, Italy,
[3] SRA Key Technology Lab, Tokyo, Japan,
gerhard@colorado.edu, piccinno@di.uniba.it, yunwen@colorado.edu

Abstract: The traditional notions of developer and user are unable to reflect the fact that many software systems nowadays are developed with the participation of many people of different interests and capabilities. The sharp distinction between users and developers gets blurred. Many researchers have used different concepts such as end-user developer, prosumer, pro-am to describe those new in-between roles. This paper provides a conceptual framework for characterizing varied activities that all people involved in using and developing software systems from a socio-technical perspective. The conceptual framework clarifies the spectrum of different use and development activities by a continuum of participants with different roles. Based on the framework, we analyze how participants change their roles to migrate from users to developers through interactions, and how such interactions co-evolve both the community and software artifacts.

Keywords: Open-source software, ecology of participants, Software Shaping Workshop, end-user development, meta-design

1 Introduction

Users and developers are considered two distinct groups of people: users are those people who own a problem, and developers are those who implement software systems for supporting users to solve problems. Nowadays, with the widespread use of web-based software systems, the sharp distinction between users and developers is quickly disappearing: they are no more considered as two mutually exclusive groups of people. A lot of users are not only using software but also getting involved in designing software. In this way users increasingly take an active role in the development of software tools suited to their needs. This results in a continuum ranging from passive consumer, to meta-designer [1], to developer. It is also the case that the same person is and wants to be a consumer in some situations and in others a designer; therefore "consumer/designer" is not an attribute of a person, but a role assumed in a specific context. Our aim is to study and characterize virtual organization in which richer ecologies of participants, i.e., *professional amateurs* [2], *prosumers* [3], *power users, local developers, and gardeners* [4], and *communities of practice* [5], can develop according to their own needs. A deeper understanding of this ecology, needs to be exploited to create multi-faceted computational environments [6] tailored to the interests, needs and expertise of different stakeholders to support the *migration path* [7] between the different roles.

To face with end-user needs, the challenge is to develop software environments that support end users in performing their activities of interest, but also allow to tailor their software environments to better adapt them to their needs, and even to create or modify software artifacts. The latter are defined as activities of End-User Development (EUD), to which a lot of attentions are currently devoted by various researchers in Europe and all over the world. EUD requires the active participation of end users in the software development process and tasks that are traditionally performed by professional software developers need to be transferred to the users, who need to be specifically supported in performing these tasks.

To allow EUD activities, we have to consider a two-phase process, the first devoted to design the design environment, the second one to design applications using the design environment. These two phases are not clearly distinct, and are executed several times in an interleaved way; because the design environments evolve both as a consequence of the progressive insights the different stakeholders gain into the design process and as a consequence of the comments of end users at work. This two-phase process requires a shift in the design paradigm, which must move from user-centered and participatory design to *meta-design* [8]. Through meta-design, design environments can be created that permit applications to be designed and evolved at the hands of end users in accordance with their own culture, skills and languages.

This paper is organized as follows. Section 2 presents a spectrum of participants in socio-technical environments. Section 3 presents Open Source Systems as an example of socio-technical environments, the ecology of involved participants. Section 4 discusses the role migration in the considered ecology of participants, and Section 5 provides conclusions.

2 A Spectrum of Participants in Socio-Technical Environments

To support EUD with meta-design, it is imperative to break down the sharp boundaries between users and developers. Being a user or a developer is a continuum ranging from passive consumer, to well-informed consumer [9], to end user, to power users [4], to domain designer [10] all the way to meta-designer (a similar role distribution for domain-oriented design environments is defined in [1]). Moreover, the same user is often a consumer in some situations and in others a designer.

A critical challenge is to support a *migration path* [7] between the different mentioned roles: consumers, power-users, and designers are nurtured and educated, not born, and people must be supported to assume these roles. Supporting migration requires to view software systems not only as a technical system but also a *socio-*

technical environment [11] in which the functionality of the software system is shaped by the interaction of all stakeholders that constitute an ecology of participants for the software system. Figure 1 depicts the ecology of participants in a software system from the socio-technical perspective. The *x* axis represents the user expertise in software design and *y* axis represents the technical complexity of participating activities. A zone delineates a participation space. The top-right space is called "Software design space" in which development activities are mainly carried out by professional software developers and meta-designers. The bottom-left space is the "Software consuming space" whose participants are mainly passive consumers or users of software systems and they are not actively involved in the development process of the software. In between, an EUD space exists, in which users, thanks to available techniques made available in their software system are able to modify their software. Rather than being distinct, these three areas usually overlap and their boundaries are blurred.

3 OSSs as Co-Evolving Socio-Technical Environments

EUD and meta-design shares many common features with Open-Source Software (OSS) development practices that actively seeking the participation and contributions of users at different levels. There are abundant lessons in OSS to be discovered and learned for the success of EUD systems, especially in the aspects of understanding what motivates so many people to dedicate their time, skills, and knowledge to OSS systems, and how users of OSS system become developers.

OSS grants not only developers but also all users, who are potential developers, the right to read and change its source code. Developers, users, and user-turned-developers form a *community of practice* [5]. A community of practice is a group of people who are informally bounded by their common interest and practice in a specific domain. Community members interact with each other for knowledge sharing and collaboration in pursuit of solutions to a common class of problems. An OSS project is unlikely to be successful unless there is an accompanied community that provides the platform for developers and users to collaborate with each other.

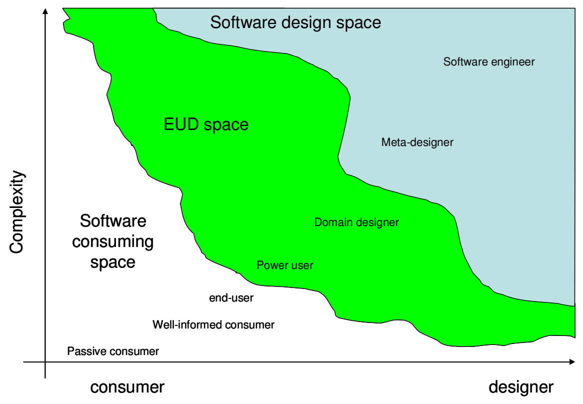

Fig. 1. The ecology of participants in a socio-technical environment

Members of such communities are volunteers whose motivation to participate and contribute is of essential importance to the success of OSS projects. In OSS users are usually developers, professionals or beginners. OSS refers to software systems that are free to use and whose source code is fully accessible to anyone who is interested. Most OSS systems start out with developers who want to solve their own particular problem and make the system available to others for free. It often attracts many users who have a similar problem, and because of the free access of source code, some interested users become co-developers by extending or improving the initial system. Together with the original developer, users and co-developers create a collaborative and evolving OSS community around the system [12]. OSS exploits meta-design techniques to empower their users to be able to develop the system, even if they are not professionals.

3.1 Mapping the Ecology of Participants in OSS

We will use OSS as an example to illustrate the ecology of participants in socio-technical environments. In OSS, the right to access and modify source code itself does not make OSS projects different from most "Closed Source Software" ones. All developers in a project in any software company would have the same access privilege. The fundamental difference is the *role migration* of the people involved in a project. In Closed Source Software projects, developers and users are clearly defined and strictly separated. In OSS projects, there is no clear distinction between developers and users: all users are potential developers. Borrowing terms from programming languages, *developers* and *users* are types, and *persons* involved in a project are data objects, Closed Source Software projects are static, binding languages in which a *person* is bound to the type of *developers* or *users* statically, and OSS projects are dynamic-binding languages in which a *person* is bound to the type of *developer* or *user* dynamically, depending on his or her involvement with the project at a given time.

Most OSS systems are not completely designed in advance. They evolve in response to the needs of users in the OSS community, and the evolution is carried out by contributing (co-)developers of the same community. Although the evolution of an OSS system is not well planned, "giving users of a product access to its source code and the right to create derivative works allows them to help themselves, and encourages *natural product evolution* as well as preplanned product design [13]."

To understand how the "natural product evolution" happens in OSS systems, we have conducted case studies [12] and presented a broader perspective by examining not only the evolution of OSS systems, but also the evolution of the associated OSS communities, as well as the relationship between the two types of evolution. Although an OSS project might have a leader (often the one who initiates the project), the leader neither has a grand plan for the system at the beginning, nor dictates the evolution of the system. It is the whole OSS community that collaboratively drives, as both users and developers, the evolution of the system. Therefore, a full understanding of the evolution of an OSS system cannot be complete without understanding the evolution of the OSS community and its role in driving the evolution of the system.

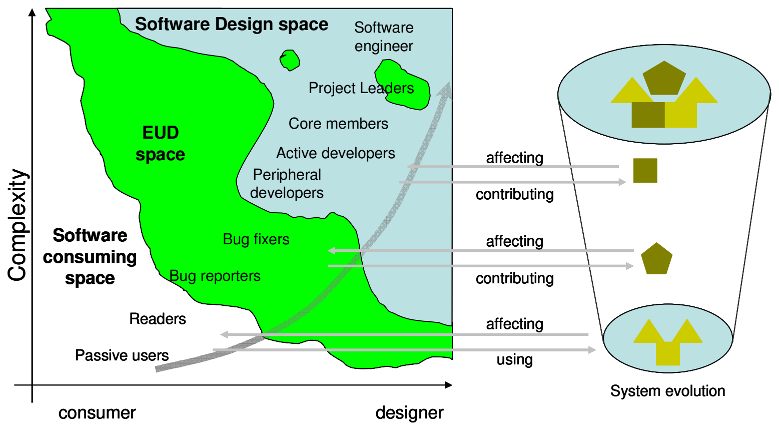

Fig. 2. Role migration in the co-evolution of participants and systems.

Participants of an OSS community assume a role by themselves according to their personal interest in the project, rather than being assigned by someone else; the different roles are the following [12]:

- *Passive User* just uses the system in the same way as most of us use commercially available Closed Source Software. They are attracted to OSS mainly due to its high quality and the potential to be changed when needed.
- *Reader* refers to those active users of the system; they not only use the system, but also try to understand how the system works by reading the source code. Given the high quality of OSS systems, some Readers read the systems to learn programming. Another group of Readers exists who read an OSS system not for the purpose of improving the system per se but for understanding its underlying model and then using the model as a reference model to implement similar systems [14].
- *Bug Reporter* discovers and reports bugs. They assume the same role as testers of the traditional software development model. The existence of many Bug Reporters assures the high quality of OSS, because "given enough eyeballs, all bugs are shallow" [15].
- *Bug Fixer*, that are called to fix bugs that either they discover by themselves or are reported by other members.
- *Peripheral Developer*, that occasionally contributes new functionality or features to the existing system. Their contribution is irregular, and the period of involvement is short and sporadic.
- *Active Developer*, that is the person that regularly contributes new features and fixes bugs; they are one of the major development forces of OSS systems. *Core Member*, that is responsible for guiding and coordinating the development of an OSS project. Core Members are those people who have been involved with the project for a relative long time and have made significant contributions to the development and evolution of the system.
- *Project Leader*, that is the person who has initiated the project and is responsible for the vision and overall direction of the project.

Not all of the eight types of roles exist in all OSS communities, and the percentage of each type varies. Different OSS communities may use different names for the above roles. For example, some communities refer to Core Members as Maintainers. The difference between Bug Fixers and Peripheral Developers is rather small because Peripheral Developers might be mainly engaged in fixing bugs. Mapping those roles into the ecology of participants of Figure 2, we can see that Readers and Passive Users participate in the Software Consuming Space; Bug Fixers and Bug Reporters participate in the EUD Space; and Project Leaders, Core Members, Active Developers and Peripheral Developers participate in the Software Design Space.

3.2 Supporting the Ecology of Participants with SSW Methodology

To support co-evolution of users and systems in socio-technical environments and to allow EUD activities, we have proposed the Software Shaping Workshop (SSW) design methodology [16]. This approach views the development of an interactive system as the results of the interaction among several virtual software environments, each of them is called virtual workshop. Furthermore, when a complex activity has to be performed by a team of people of different cultures, each member of the team performing different tasks, the SSW methodology prescribes the development of a network of environments, each being devoted to the performance of specific tasks by well identified members of the team, while the overall environment has to be customized to the culture and skills of the people who will use it.

Overall, according to the SSW methodology an interactive system to support the work practice in a given application domain is developed as a set of interconnected virtual workshop. There are two types of virtual workshop: *application workshop* is a software environment used by a community of end users to perform their daily tasks in a certain domain, it is properly designed for the specific needs of that community of end users; *system workshop* is a software environment used by a community of experts in the design team to generate and update other workshops. An interactive system is always organized as a network of system and application workshops, always presenting three main levels. *Meta-design level*, in which software engineers use a system workshop to provide the software tools necessary to the development of the overall interactive system, and to participate in the design, maintenance, and validation of application and system workshops. Software engineers produce the initial programs, which generate the virtual workshop to be used and refined at the same or at lower levels, and participate in the maintenance of virtual workshops by modifying them to satisfy specific requests coming from lower levels. *Design level*, in which HCI experts, and domain experts cooperate in design, maintenance, and validation of application workshops through their own system workshops. *Use level*, in which end users (not participating to the development process) belonging to a certain community participate in task achievement using the application workshop devoted to their community. The network is thus organized, as in OSSs, so that it reflects the working organization of users and developers. Both meta-design and design levels include all the system workshops that support the design team in performing the activity of participatory design.

According to the ecology of participants (Figure 2) and to the SSW methodology, we identified a mapping between the network levels involved in the virtual workshops network devoted to participants in OSS with the three main areas in the framework characterizing the ecology of participants in the development process in OSS. At meta-design level there are software environments supporting user in the Software Design Space (see Figure 2); Project Leaders, Core Members, Active Developers and

Peripheral Developers will find here the virtual workshop devoted to them. At design level two system workshops are identified to support Bug Fixers and Bug Reporters activities. At use level Readers and Passive Users participating in the Software Consuming Space will have application workshops to accomplish to their tasks. In each of the three levels, communication paths among virtual workshop belonging to are provided to support the co-operation in the development process.

4 Role Migration in the Ecology of Participants

The ecology of participants (Figure 1) depicts the varied roles that participants assume in using and developing software systems. The software systems are developed and evolved through the intensive interactions among all the participants, and the interaction between users and software systems. At the same time, participants also evolve through the same process and assume bigger roles in shaping the functionality of the software systems. At this aim, they are supported by the Software Shaping Workshop methodology that foresees a virtual workshop for each role in the ecology of participants in the OSS development process. The network of virtual workshops allows them to communicate and collaborate to the system design, implementation, use and evolution by working with a workshop customized to them and using their own languages and notations, so that they are not disoriented and may overcome the gaps existing among them. Figure 2 describes the co-evolution that we have observed in OSS systems. Many participants started as users, and during their interactive use of the software system, some of the participants become interested in reading and making bug reports of the system, migrating into the roles of readers and bug reporters. Some got more involved and continued their migration path into bug fixers and peripheral developers as they gain more knowledge of the system. Some even became active developers and core members by contributing more development the system. As the members migrated into bigger roles, their contributions made the system evolve, and the evolution of the system in turn relied on the active participation and contributions of different levels of participants.

5 Conclusions

In this paper we discussed a conceptual framework to characterize the rich and varied ecology of participants, at various levels, in open, evolvable and living socio-technical environments. Nowadays, the sweeping kinds of end users are increasingly involved in the design and development of the tools they use, thus they need to be supported through techniques that are suitable for them. In particular we explored the ecology of participants in Open-Source Software, by analyzing the various roles of involved end users in the development process belonging to three different spaces (software consuming, EUD and Software design space) and matching them with the three different levels (use, design and meta-design level) required by the Software Shaping Workshop design methodology. Finally we provided some insights about the evolution and the consequent migration of user roles along the migration path.

References

1. Fischer, G., Giaccardi, E.: Meta-Design: A Framework for the Future of End User Development. In: Lieberman, H., Paternò, F., Wulf, V. (eds.): End User Development, vol. 9, pp. 427-457. Springer, Dordrecht, The Netherlands (2006)
2. Leadbeater, C., Miller, P.: Pro-Am Revolution. How enthusiasts are changing our economy and society. Demos, London (2004)
3. Tapscott, D., Williams, A.D.: Wikinomics: How Mass Collaboration Changes Everything. Portofolio, Penguin Group, New York, NY (2006)
4. Nardi, B.A.: A Small Matter of Programming. The MIT Press, Cambridge, MA (1993)
5. Wenger, E.: Communities of Practice — Learning, Meaning, and Identity. Cambridge University Press, Cambridge, UK (1998)
6. Myers, B.A., Ko, A.J., Burnett, M.M.: Invited Research Overview: End-User Programming. Human Factors in Computing Systems, CHI'2006 (Montreal), pp. 75-80. (2006)
7. Burton, R.R., Brown, J.S., Fischer, G.: Analysis of Skiing as a Success Model of Instruction: Manipulating the Learning Environment to Enhance Skill Acquisition. In: Rogoff, B., Lave, J. (eds.): Everyday Cognition: Its Development in Social Context, pp. 139-150. Harvard University Press, Cambridge, MA - London (1984)
8. Sutcliffe, A., Mehandjiev, N.: Introduction. Communications of the ACM 47, 31-32 (2004)
9. Beyond 'Couch Potatoes': From Consumers to Designers and Active Contributors, in FirstMonday (Peer-Reviewed Journal on the Internet) http://firstmonday.org/issues/issue7_12/fischer/
10. Fischer, G.: Domain-Oriented Design Environments. Automated Software Engineering 1, 177-203 (1994)
11. Sutcliffe, A.G.: Requirements Engineering for socio-technical systems. In: Proceedings Fifth IEEE International Symposium on Requirements Engineering, pp. 27-31. IEEE Computer Society Press, Los Alamitos CA, Toronto (2001)
12. Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., Ye, Y.: Evolution Patterns of Open-Source Software Systems and Communities. In: International Workshop on Principles of Software Evolution (IWPSE 2002), pp. 76-85. Orlando, FL (2002)
13. O'Reilly, T.: Lessons from Open-Source Software Development. Communications of the ACM 42, 33-37 (1999)
14. Aoki, A., Hayashi, K., Kishida, K., Nakakoji, K., Nishinaka, Y., Reeves, B., Takashima, A., Yamamoto, Y.: A Case Study of the Evolution of Jun: An Object-Oriented Open-Source 3D Multimedia Library. In: 23rd International Conference on Software Engineering (ICSE'01), pp. 524-533. IEEE Press, Toronto, Canada (2001)
15. Raymond, E.S., Young, B.: The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly, Sebastopol, CA (2001)
16. Costabile, M.F., Fogli, D., Mussio, P., Piccinno, A.: End-User Development: the Software Shaping Workshop Approach. In: Lieberman, H., Paternò, F., Wulf, V. (eds.): End User Development, vol. 9, pp. 183-205. Springer, Dordrecht, The Netherlands (2006)