Chapter 8
Exclusiveness vs. Inclusiveness in Software Development:
The Triple-Loop-Learning Approach

Edeltraud Hanappi-Egger
WU Vienna University of Economics and Business, Austria

ABSTRACT

The question of digital divide along gender and culture is still an important one and has to be revisited after a long period of scientific discourses. Most of the scientific contributions focus on issues of gender-specific use of computer systems or gender-specific segregation of the field of information and communication technologies. Contrary to those approaches, this chapter deals with how exclusion mechanisms are created during the software development process. It is shown that system designers are strongly biased in terms of gender and culture and unconsciously use their own mental models. Selected results of empirical studies demonstrate how these a-priori assumptions can be made visible, and a new approach of system design, based on the triple-loop-learning concept, will be presented.

INTRODUCTION

From the very first beginning software development was exposed to extensive discussions on its self-understanding. As Hanappi-Egger (2006a) highlights there were several substantial changes in the paradigms of software development and software design. In the 1960ies software engineering – as the term already indicates – was based on a rather expert-driven approach where users played only a minor role. The knowledge of future users was needed during the phases of problem-analysis when the functions had to be specified and in the end of the development process when the system was introduced and the users had to be trained how to use the application software. Since the implemented software systems were not always satisfying it became clear that the engineers do not have accurate knowledge on the application field, but people working in the according areas do. Consequently it became obvious that the future users' knowledge has to be considered during the development processes leading to more user-centered and participative

DOI: 10.4018/978-1-4666-0020-1.ch008

concepts of software design and software development. Hanappi-Egger (2004) even argues that software development is organizational development since the specification of system features in workflows changes organizational structures and processes: The introduction of computer systems has a strong influence on re-negotiating work distribution, responsibilities and even qualification and skills for certain computer supported tasks. Thus there exists a mutual dependency of technology development and organizational settings: On one side the organizational framework determines ways of technology development (e.g. by power hierarchies reproduced in hierarchies of computer rights). On the other side the computerization is strongly changing the organizational environment (by forcing the organization to re-engineer internal processes and structures). This approach – called "social shaping of technology" (see Harding, 1986; Wajcman, 1991) clearly questions the view of software development as an objective "engineering task" but stresses the role of social dynamics and power. Therefore the product – namely the software system itself – was seen rather as being a negotiated construction than as representing an "objective" solution (e.g. see also Haraway, 1995; Cockburn & Ormrod, 1993; Hanappi-Egger, 2004, 2006a). Furthermore as Mackenzie (2005) argues software codes themselves are performative by being an articulation of the diverse "realities" of the involved parties.

Based on the fact that software development takes place in social settings and consequently within specific power settings, it can be stated that software development has to be seen as a social process where different stakeholders are involved determining the decisions made during this process. In particular the system developers with their very specific technical expertise strongly determine the design of a software system. Consequently the software product itself incorporates specific views on how to be used. These specific views eventually create unconsciously hidden access barriers for users. That means that users of different cultural and gendered backgrounds might be excluded by ignoring their specific requirements and expectations.

To overcome this, an open and flexible process-oriented perspective of software development is needed. Thus this chapter will focus on how to prevent gender biases and culture biases in software development by using the triple-loop-learning concept. To highlight the advantage as well as the limitations of this approach, selected results of various research studies done in the field of software development will be presented.

BACKGROUND

The issues of "gender and ICT" do already have a long discourse tradition (for an overview see Wajcman 2010). At the very beginning of feminist critiques on technology development a major topic was how to increase the female participation in the branch to assure that women's voice is heard. Although this is a very important issue, it is often overlooked that women always played an important role e.g. in the history of computer science. Be it Augusta Ada Byron, Lady Lovelace giving 1843 the first "program" to Charles Babagge's analytical machine, or be it the "ENIAC girls" programming the first computational machines (see Gurer, 1995), women contributed strongly and actively to the development of computer systems.

While in the 1950ies and 1960ies programming was considered as clerical task, it gained an increase in status in the 1970ies – and turned out to be a male job (see also Grundy, 1996). Consequently a drop of female students in computer science and engineering was occurring, a phenomenon still existing and still causing heavy public discussions. But it is only one aspect how to motivate women to enter the field of computer science and engineering. Another problem is that the organizational structures often lead to the fact that female experts leave again (see also Panteli et al., 1999; Hanappi-Egger, 2004; Hanappi-Egger