Available online at www.sciencedirect.com
ScienceDirect
Information and Software Technology 49 (2007) 682–693
INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

Interpretation, interaction and reality construction in software engineering: An explanatory model

Kari Rönkkö *

Blekinge Institute of Technology, School of Engineering, Box 520, SE-373 25 Ronneby, Sweden

Available online 13 February 2007

Abstract

The incorporation of social issues in software engineering is limited. Still, during the last 20 years the social element inherent in software development has been addressed in a number of publications that identified a lack of common concepts, models, and theories for discussing software development from this point of view. It has been suggested that we need to take interpretative and constructive views more seriously if we are to incorporate the social element in software engineering. Up till now we have lacked papers presenting 'simple' models explaining why. This article presents a model that helps us better to understand interpretation, interaction and reality construction from a natural language perspective. The concepts and categories following with the model provide a new frame of reference useful in software engineering research, teaching, and methods development.

Keywords: Social; Interpretation; Interaction; Indexicality; Communication; Software development; Natural language; Methods development; Management; Software engineering practice

1. Introduction

There is a growing acceptance that so-called 'soft' issues are a valuable component in software engineering. The social, behavioral and cultural dimension is recognized as part of the international scientific agenda in the field, exemplified by [11,24,1,7]. This is a difficult area in which to work as the incorporation of behavioral and people issues in software engineering research is limited, and it is not yet clear what the relevant literature is. This is exacerbated by the fact that the field is cross disciplinary in nature, meaning that useful prior work may appear outside the software engineering community's usual channels. One indication of the status of the social challenge in our community was given by Fuggetta in his software engineering process road map celebrating the turn of the millennium [15]. He stated that the existing isolation must end, and asked for the results of process research performed in and by other communities. This article contributes to the social agenda by presenting a model that explains from a natural language point of view, how interpretation takes place, and discusses the consequences of this in relation to interaction and reality construction in software engineering practice.

The presented model *the documentary method of interpretation* (explained in detail in Section 2) is originally from sociology and provides an explanation of how people understand each other and also are able to investigate their daily world. It focuses on how the language of daily life is built up through indexical expressions. Indexicality here refers to the fact that words only take on a complete meaning in the context of their production. Words have an indexical relation to the circumstances in which they are uttered. This phenomenon becomes especially visible in expressions such as 'you', 'I', and 'that', as they undeniably draw their meaning from the interplay of people and the objects of interest under concrete conditions within a specific setting. The documentary method of interpretation demonstrates that actually all words are indexical, i.e., not only the obvious ones from linguistics that are exemplified above. The model also demonstrates how words get their

* Tel.: +46 457 38 58 45; fax: +46 457 271 25.
E-mail address: Kari.Ronkko@bth.se

0950-5849/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2007.02.014

meaning from the two dimensions of indexicality: the situated context and what is known.

One recent software engineering study that might serve as example of indexicality is Smolander's study of architectures in software organizations. In this study he identified ambiguous meanings of the concept architecture: *architecture emerges as a plastic concept including diverging and simultaneous connotations for different stakeholders* [29]. Four different usages of architecture were found: *architecture as blueprint, architecture as literature, architecture as language*, and *architecture as decision* [29]. Those different usages can be explained through applying the model that is demonstrated in this paper, the documentary method of interpretation.

In the context of software engineering, the documentary method of interpretation has served as an underlying assumption in the present author's PhD thesis [26]. In this thesis the model heavily influenced the understanding of the social element in the studied software development practices, the applied research methodology, and the industrial methods development cooperation. In the Computer Supported Cooperative Work community the power of the model is very much present in studies where ethnomethodologically informed ethnography has been applied, but it is seldom discussed explicitly (examples of such software engineering studies are [6,28,27]). In the continuation of this article the documentary method of interpretation will be referred to as DMI.

An era starting in the mid 1970s and extending to the late 1980s saw the focus of discourses on professional work shift from technology itself towards its use [20, p. 6]. During these years, several now 'classic' papers were published (although rarely in the mainstream SE literature). Some are experience-based, and more explicitly include the relation to the social context to which software engineering methods are applied (e.g., [19,23,4,25,12]). All of these papers are still influential and can be found referenced in recent publications. The present author found that many of the papers are adequate in providing software engineering with a context of contingencies in which the need for a model explaining the social elements interpretation, interaction and reality construction can be understood, also from a historical point of view.

For the purposes of this article, the model will be discussed in relation to three of the above enumerated papers, although they do not explicitly direct their discourses to the social elements as discussed in the present article. From a product oriented view, Brooks' paper *No silver bullet – essence and accident of software engineering* [4] provides powerful insights into the nature of the material that software engineers must manage in software development projects, i.e., the software. In relation to the model demonstrated in this paper, these insights help to explain why indexicality of words and texts is so hard to achieve and communicate in software development projects. Floyd's paper *Outline of a Paradigm Change in Software Engineering* [12] requested that we move from a product oriented paradigm to a process oriented paradigm. From her interpretative and constructive view it is claimed that software practitioners do not analyze, apply or refer to requirements, methods, and implementations; instead they unavoidably construct their understanding of them (I have borrowed the concept 'reality construction' from her work [14]). In relation to DMI as presented in this paper, these statements are reminiscent of the epistemology that follows from it. DMI explains how Floyd's claims are played out in practice by the practitioners. Up till now we have lacked papers presenting 'simple' models explaining why the interpretative and constructive view is needed within our discipline. The concepts and categories included in DMI provide new information that is useful in management, in the design of software methods, processes and process improvements. Floyd's paper also reminds us of the historically slow development of the social element in relation to software engineering. Finally Naur's paper *Programming as Theory Building* [23] made it painfully clear to us that exemplary resources in the form of material and available support are not enough when modifying others' programs. In fact, if Floyd's claims had been taken seriously by the software developers in Naur's study, and if the same developers had access to an explanatory model like DMI, their difficulties could have been both anticipated and prevented.

The principal result of this article is the presentation of a model that explains how social accomplishments take place; a model that is repeatable and true for different situations. The model provides software engineering with a shared conceptual apparatus explaining how interpretations take place and which dimensions of information are necessary in the process of interpreting. By relating the explanatory model to the three classic papers and to own study, a better understanding of interaction and reality construction in the context of software engineering is furthermore achieved. Altogether, this provides software engineering research, teaching, methods and process development, and professional practice with a new frame of reference. In the following section DMI is presented and exemplified. Thereafter the three classic papers are presented in Section 3. In the following section DMI is discussed in relation to these papers. Here Brooks' four product oriented categories, found in Section 3.1, are converted to three new socially oriented categories in order to fulfill demands from the interpretative paradigm found in Section 3.2, that Floyd requests. Through this act, we move a step away from the product oriented and explanatory perspective, towards a social and action oriented perspective. The contents of the suggested categories are in no way final; more work is needed here. Finally, in the conclusions, improvements to software engineering are suggested, based on the knowledge that DMI provides.

2. The documentary method of interpretation

The *documentary method of interpretation* was borrowed from Mannheim, who originally reserved it for scientific