Understanding the Software Process as a Social Process

Ian Sommerville and Tom Rodden
Computing Department, Lancaster University, LANCASTER LA1 4YR, UK.
E-mail: is@comp.lancs.ac.uk

Abstract. This position paper summarises a project which is concerned with understanding the software process from a sociological viewpoint. We believe that understanding the social factors inherent in the software process will allow us to derive new requirements for software tools for process support. One such tool, (the Designer’s Notepad) intended to support informal design activities is briefly described.

The vast majority of research on the software process has been undertaken from a technical or managerial viewpoint. As a consequence, the emphasis on this work has been on specifying and controlling the process on the assumption that a controlled process is likely to lead to more efficient software production and higher quality software. This notion derives from the ideas of Total Quality Management and their adoption by the Japanese in their post-war industrial strategy and, since the 1970’s, in their concept of Software Factories (Cusamono, 1991).

Humphrey (1989), for example, classes processes into 5 levels of maturity which are distinguished, in essence by the degree of discipline applied to that process. He argues that only a controlled process can be successfully automated. However, various surveys have discovered that only a minority of organisations currently use a controlled software process.

In order to understand the software process, there has been much work on software process modelling and representation of the software process using some computer-processable notation. When these process models are embedded in an environment, the environment can act in a pro-active way, monitoring user actions and suggesting tools to be used and sources and destinations for objects which are being processed. However, although there has been at least 10 years on research into process modelling, there have only been a very limited number of systems developed and the commercial utility has still to be demonstrated. In short, an awful lot of research money has been spent in this area without much return.

In many respects, work on software process modelling and automated software process support is similar to work carried out in the office automation domain where there have been several attempts to describe and formalise office procedures and to embed these descriptions in office systems. These systems have met with some opposition from office workers and have not been particularly successful. Suchman (1983) showed that they were based on a flawed premise that is, that workers actually follow laid-down procedures. In fact, for a variety of reasons, people adapt these procedures to local circumstances and resources and this adaptation is dynamic and responsive to change. Circumvention of the rules is the norm rather than the exception and the different kinds of circumvention are so diverse that they cannot readily be articulated.

We support the notion that an effective process and process support is essential if we are to improve software productivity and software quality. However, we believe that the current, technical approach is inherently limited in that it tends to equate process support
with process control. Process control relies on understanding broad-grain process activities, supporting these with a limited number of tools and monitoring the process as it progresses. We argue, however, that process support relies on a much deeper understanding of the software process as a social process which is dominated by complex and subtle interactions between the members of the software development team. We also argue that software engineers are not well-equipped to understand the nature of these interactions because they inevitably tend to focus on the technical activities which are being carried out rather than the social interactions which are taking place.

In our current work, we are working in conjunction with a team of sociologists who are studying software engineering as a social activity with a view to deriving a model of the software process from a social rather than a technical perspective. In particular, we are using a technique known as ethnography to understand the nature of the software design process and how designers work together to make design decisions. The objective of this work is to develop a set of requirements for CASE tool support where the tool, unlike current CASE tools, will support the activity of creative design rather than the documentation and checking of software designs.

An ethnographic study of the software process involves a social scientists 'living with' the design team for an extended period of time and observing how they work and the interactions which take place. As a dispassionate observer who has only a limited understanding of the technical activities which are taking place, the ethnographer is in a position to discover subtleties and complexities of the software process and its surrounding activities which are not immediately apparent to either the software engineers themselves or an engineer undertaking a more conventional requirements analysis process.

The specific domain we are studying is the design of hardware and software control systems for photocopying machines. This is a particularly useful domain because the design process involves both hardware and software engineers and, different photocopier models require different hardware/software trade-offs. Some of the specific questions we are addressing in our study are concerned with:

- **Product factors.** What is the practical role of the system specification? What tracking mechanisms are appropriate for particular classes of design? How does design change when the product is principally hardware or principally software?

- **Process factors.** How do standards help/hinder the collaborative design process? What is the role of reviews (formal and informal) and how does error discovery feedback into the design process? How are designers influenced by review comments? How does the language (mathematics, diagrams, natural language, etc.) influence the design process?

- **Team factors.** How does the status (perceived and actual) of individual team members affect the process? How does the group dynamics differ from the organisational setup?

- **Organisational factors.** How does the organisational culture affect the process? How is expertise captured and identified in an organisation?

From a software engineering standpoint, the objective of our work is to produce better CASE tools which provide support for design process activities. We will investigate the notion of including an explicit process model in these tools but we may discover that the actual process complexity is such that no effective modelling is possible. Our initial work in this area has thus been concerned with producing a very flexible software support system which does not impose any pre-ordained process on the software designer.