Software development involves users from at least two different but interrelated perspectives. Software systems are, directly or indirectly, used by people to manage and perform various kinds of tasks. The main issue of usability engineering from this point of view is how well the software system fits with the intended tasks. The role of user interfaces is crucial, since they are the parts of the software systems that are directly involved in linking the user and the software together for performing the tasks.

The people who design and maintain software systems are, on the other hand, users of software development tools. The tasks that those tools should support include all aspects of software engineering, such as specification, design, and testing of the user interfaces of the developed software systems. Most software development tools are commercial software systems themselves, designed to serve their users through advanced user interfaces. One of the essential elements in the design of software development tools is that professional software engineering is almost always groupwork, in which different aspects of the development of the same system should be visible to a number of people in a consistent way.

An interesting strategic change that has been a vision of the future for a rather long time, but has actually been going on only for a few years, is that end users may take some of the responsibilities of professional software developers, for example in configuration and design of the user interfaces of the software systems that they operate. This certainly also has some consequences in the software industry that develops software engineering tools — the number of non-professional users of software development tools is rapidly increasing. It has indeed been estimated that the number of occasional, non-professional software developers will be over a hundred million people by the year 2010.

In this session we will address the professional side of usability engineering, focusing especially on tools meant for the development of user interfaces. Yet, we will have a great opportunity to discuss the possibilities for providing the kinds of approaches and techniques described in the four papers to people who are not software professionals.
In the first paper, J. Vanthienen and S. Poelmans present and evaluate a framework for selecting software development tools. The framework is not specific to user interface development tools, which actually helps us in positioning the role of user interfaces in modern software systems.

One class of the evaluation criteria addressed in the paper includes human factors, that is, the usability of software development tools. As pointed out by the authors, the kind of an environment in which the tools are more and more often being used, is distributed and network-based.

This brings us to the second paper by R. Huang, who discusses a computer-aided tool for performing and managing software development in distributed work groups. In addition to the architecture of the tool, the author presents a process model for the software development tasks being supported by the tool. In more general terms, user-oriented development of software systems requires thorough knowledge of the tasks to be assisted by the system. Which kinds of task models are needed and how should they be implemented in tools are interesting and practical questions.

The third paper, by C.I.S. Arias and B.M. Daltrini, develops the theme of distributed software engineering environments further, by specifically addressing the tasks of user interface development. The authors discuss software agents, one of the most rapidly emerging technologies for implementing distributed software systems. This technology together with Internet and other network technologies may very well appear to be crucial for assisting not only professionals but occasional software developers as well.

We will close the session by a paper written by N. Dlodlo and C. Bamford, who bring the problem of distribution even closer to the implementation concerns of user interfaces, be they included in software development tools or other applications. The key concept of the paper is the separation of the functionality of the user interface from the application software. This kind of a concept would make it possible for the users of any kinds of software systems to tailor the usability of the system to their specific needs not only once and for all, but any time and anywhere.