

may be insufficient when it comes to discussing inexpediciencies in the users' present work and requirements for new systems, since they do not necessarily reflect what can be observed in the organization. Relying on such descriptions has often led to solving the wrong problems.

Thesis: Areas covered by the system developers.

We can usually be sure that system developers cover areas 3 and 4: technological options. We can usually expect nothing more.

The ETHICS Approach

Enid Mumford
CHESHIRE, ENGLAND

Effective Technical and Human Implementation of Computer-based Systems—ETHICS—is a technique and also a philosophy that future users of new technical systems should be able to participate in the design process and help create systems that are humanistic and friendly as well as efficient and effective.

ETHICS has three principal objectives:

- to enable future users to play a major role in system design, and to assume responsibility for designing the work structure that surrounds the technology. This involves a learning process and a set of simple diagnostic and socio-technical design tools.
- to ensure that new systems are acceptable to users because they both increase user efficiency and job satisfaction.
- to assist users to become increasingly competent in the management of their own organizational change, so that this becomes a shared activity with the technical specialists and reduces the demands on scarce technical resources.

The methodology is not necessarily aimed at producing a computer-based solution, as the emphasis is on obtaining the right balance between the social and technical aspects of the complete system. ETHICS incorporates the joint philosophies of participation and socio-technical design.

It assists user design groups to create a decision structure that incorporates all interested groups affected by the new system; a process that enables the de-

The first part of this thesis is rather obvious, or designers would have no role in the process. However, designers may also be challenged by the technological options. For example, when new development tools are applied, when simultaneous development of basic software and applications occurs, and when new standard software or hardware is introduced.

With regard to the second part of the thesis, the developers may of course have worked for the organization before, or they may have worked

sign task to be smoothly carried forward from identification of need to change, to successful operation of the new system. It also sets an agenda that enables business efficiency and employee job satisfaction improvements to be considered in parallel and given equal weight.

ETHICS incorporates the following diagnostic and design tools:

- a framework to assist the identification of mission, key tasks, important constraints and factors critical to effective operation,
- a variance analysis tool to assist the identification of systemic and operational problems and problem areas,
- a questionnaire to measure job satisfaction,
- a framework to identify what is likely to change in the internal and external environments, and
- a set of guidelines for individual and group work design.

The approach is currently used in three ways. First, it is used for the task for which it was originally designed—to help future users diagnose their needs and problems, setting nontechnical objectives for the system and restructuring their work situation. In this form it has been used by shop floor and office workers, sales staff and nurses.

ETHICS is also assisting managers to define their information needs prior to introducing a semantic logic-based executive information system. Here, a simplified form of ETHICS, called QUICKETHICS—Quality Information from Considered Knowledge—has been developed.

Finally, the method is employed as a general problem-solving tool to enable groups to systematically analyze needs and problems with a view to improving performance. □

Mumford developed the ETHICS approach.

for a similar organization. In that case they may have concrete experience as well as abstract knowledge about the users' present work. This would make things easier, but this is not something that can generally be taken for granted. Also when it comes to the new system, developers may have prior experience (e.g., from implementing standard systems). However, neglecting the characteristics of the specific organization will prevent the new computer system and the organization from fitting together.

Thesis: Areas of knowledge to be acquired by the users through the development process.

It is the system developers' responsibility to apply tools and techniques allowing users to develop

- * relevant structures on users' present work (area 2),
- * visions and design proposals (area 5),
- * concrete experience with the new system (area 6).

The reasons for this thesis are the following: abstract descriptions of the new system (area 5) and relevant structures on users' present work (area 2) are needed when the users evaluate design proposals. As some part of the user organization must normally make a decision about accepting or rejecting proposals, the users' knowledge of these areas is indispensable.

In order for users to play a creative role in design they need abstract descriptions of their present work (area 2) as well as of the new system (area 5). Concrete experience is also needed, however, in order to understand abstract descriptions. Thus the users need concrete experience with the new system (area 6) before they can understand abstract descriptions of the new system (area 5).

Thesis: Areas of knowledge to be acquired by the system developers through the development process.

It is the developers' responsibility to apply tools and techniques allowing them to develop

- * visions and design proposals (area 5),
- * relevant structures on users' present work (area 2),
- * concrete experience with users' present work (area 1),

* concrete experience with the new system (area 6).

It goes without saying that the developers must understand abstract descriptions of the new system (area 5) since they are major intermediate results. Relevant structures on users' present work (area 2) must be understood in order to identify and evaluate desirable changes.

The developers must have concrete experience with users' present work (area 1) in order to understand and produce descriptions of relevant structures on the users' present work. System developers who have developed this area of knowledge have a better background for communicating with the users, as they are able to refer to and understand references to concrete events in the users' organization.

Finally, the developers must have concrete experience with the new system (area 6) in order to be able to test and evaluate the products of their own work.

We conclude this section with the observation that our theory recognizes that all areas of knowledge must be dealt with in any normal system development process. The toolbox necessary for this work is discussed briefly in the following section.

Tools and Techniques for Knowledge Development

Our model of user-developer communication can be used to define a toolbox for tools and techniques to facilitate this communication. The toolbox presented in Table 2 consists of 6 sections, one for each of the areas of knowledge discussed in the previous section. It accentuates the differences between the purposes of the tools and techniques, even though some fit into more sections.

The use of any tool or technique must be adapted to the particular conditions of each system development process. In particular, the users' prior experience with the tools and techniques must be considered. Some tools and techniques are almost always used successfully by developers and users working together. Others should rather be part of an extended process, in which developers

STEPS—A Methodical Approach to PD

Christiane Floyd
UNIVERSITY OF HAMBURG

Software Technology for Evolutionary Participatory System Design—STEPS—is a methodological framework for developing interactive application systems. It guides developers and users in carrying out their cooperation as well as supports the design of computer-based work. STEPS emphasizes the development process and intertwines development with use.

The method has been tried successfully in participatory development of an information system called the PETS project, which is designed for handling archives of union contracts in Germany. This participatory project resulted in a working system that is still in use today.

STEPS considers the anticipation of use an inherent part of the development. Whenever software systems—as tools or media—are to be fitted into work processes, development consists in unfolding the problem as well as in elaborating a solution. The technical concerns for providing high-quality products are inherently tied up with issues of communication, work, and social processes, which define the very nature of the problem.

Therefore, software development becomes a learning process for both developers and users. In STEPS, *evolution* refers to the emergence of insights into the functionality and the potential use of software. *Participation* refers to the strategy of mutual learning. Those participating in a software development project are creating a product, and at the same time, the development process itself. These two complementary dimensions are reflected by product-oriented and process-oriented activities. STEPS provides guidance to developers and users in both dimensions.

This approach relies on perspective for gaining insight—making perspectives explicit and allowing them to interact. The use perspective held by those who interact with software is distinct from the development perspective held by software developers. Furthermore, they both arise in a variety of views related to functional roles, collective interests, and individual tastes. Multiperspectivity is a fundamental prerequisite for cooperative work which rests on perspective-based

modeling and evaluation when using software. Software requirements are related to the context of user work processes as a whole. They cannot be completely fixed in advance as they evolve because of changes in the organization. Also, the software system itself gives rise to new requirements.

Thus, STEPS is an approach that accommodates various forms of prototyping and portraying system development in cycles of version production, application and revision. A system version consists of software and its defining documents, supplemented by guidelines for the organization of work to be supported. The interplay between each software version and the associated reorganization of user work is anticipated in the cooperative design and evaluated in the revision step. Thus, it supports mutual learning by developers and users by carefully establishing and coordinating processes of cooperation.

STEPS rests on a project model suitable for PD. This model is cyclical, all development steps and products being subject to revision. It combines software production and application, visualizing the tasks of both developers and users. It allows the choice of a situation-specific strategy in the actual project. It relies on a minimum of predefined intermediate products, thus allowing freedom for choosing them as needed. It provides for temporal flexibility in cooperation. It incorporates the dynamic coordination of the ongoing project through establishment and reference lines. STEPS embodies a human-centered notion of quality and creates a platform for discussing criteria on how to design computer-supported work and facilitates the emergence of quality through cooperative design. □

References

- Floyd, C., Reisin, F.-M., Schmidt, G. STEPS to Software Development with Users. C. Ghezzi, J.A. McDermid, Eds.: ESEC '88, Lecture Notes in Computer Science Nr. 387, Springer-Verlag, 1989, pp. 48–64.
- Floyd, C., Mehl, M., Reisin, F.-M., Schmidt, G., Wolf, G. Projekt PETS: Partizipative Entwicklung transparenter Software fuer EDV-gest. tztbe Arbeits-pl Arbeit, Gesundheit und Soziales des Landes Nordrhein-Westfalen, Technical University of Berlin, 1990.
- Floyd, C., Zilghoven, H., Budde, R., Kell-Slawik, R., Eds. Software Development and Reality Construction. Springer Verlag, 1992.