In this article, however, we argue from an epistemological standpoint that participatory design is needed to gradually build up the knowledge required for developing and using a new system. There is a common explanation when an attempt to practice PD fails: "The users and the system developers did not understand each other." The statement is often followed by the recommendation of a specific technique or tool to remedy the situation. However, in our experience there is no foolproof method. System development projects fail in communication even though they use the most promising techniques. In one project horizontal prototypes were used extensively during the time the requirements were defined. A horizontal prototype shows all intended functions, but they are not implemented in detail as required in the final system [16]. The intention was to ensure that the users understood what they accepted. The system had to undergo substantial changes, however, before it could be used [2]. In another case the users were unable to define system requirements at meetings with system developers. The system developers then made an elaborate vertical prototype and expected a response from the users. A vertical prototype offers a selection of functions iraplemented in their intended final form [16]. However, they did not receive any response fi:om the users. How do we account for these apparent paradoxes? It is difficult to find relevant explanations. Most papers and books deal specifically with techniques and tools, not with underlying theories enabling us to discuss the context and the limitations of the techniques and the tools. Comparative surveys of methods [3, 18, 29] are usually thorough on details but lacking in explanatory theory. In this article we suggest an answer to the communication paradoxes in terms of a model of userdeveloper communication. The model is based on theories dealing with system development as well as with communication. The model may help us understand why some approaches sometimes yield fruitful communication, while in other situations the same approaches turn out to be obstacles. The distinctions offered by the model may act as a catalogue-or toolbox--where system developers may find ideas appropriate for specific situations. We use the model to categorize communication methods and description tools in relation to their application area. Thus our model may form the basis of a contingency strategy, as proposed by Davis [11] and Boehm [4]. The model covers communication related to analysis and design (i.e., to defining requirements and creating solutions). It does not cover all userdeveloper communication. It excludes communication related to management and implementation. User-Developer Communication in System Development We want to discuss possibilities and obstacles for successful communication in system development. Therefore we relate the communication processes to their results and to the context in which they take place. Describing the system development process, Clements and Parnas [8] state: "The most useful form of a process description will be in terms of work products." They proceed by describing the documents they would produce during a project's lifetime. We agree with them, although our concept of results is not confined to documents alone. We would also like to include the knowledge developed by the people involved as results. What then are the results of the system development process? The final results are, of course, a system and a completed technical and organizational implementation process. Intermediate results are documents and knowledge obtained by the participants. Regardless of the development model--be it waterfall, spiral, incremental or parallel--these results form the basis of important decisions. These decisions deal with determining the system's level of sophistication, evaluating the usefulness of the system, freezing the requirements, and designing the system's internal structure. Thus the goal of analysis and design activities is to produce documents and knowledge enabling decision-making with regard to the system and its environments. How can we produce these results (i.e., what kind of methods do we need?) That depends on the prerequisites for the development process, especially the limitations of user-developer communication. The following section presents a model of communication in order to answer this question. 78 June 1993/Vol.36, No.4 ¢OMMUN|¢ATION$ OP THE ACM Communication Models

Communication is of course a key issue in collective activities such as system development. People with different backgrounds, education, training, and organizational roles exchange facts, opinions, and visions in order to inform, persuade, and maybe even threaten one another. How is communication possible in such a context. We sketch two communication models relevant to understanding and designing user-developer communication: a traditional model and an alternative model. It is our opinion that many current tools and techniques rely heavily on the first model. Current methods usually support written communication based on formalized languages, prototyping being the major exception. These methods rely on a communication model which can be described by a tube-for-communication metaphor. Communication is perceived as something created at one place (e.g., the developers' office), then carried through "a tube" to the receivers (e.g., the users). The tube could be some kind of written system description. This communication model takes for granted that successful communication is determined by the "sender's" ability to form a rigorous message. How is it, that the same message in the same form can be interpreted so differently by various "receivers"? An alternative communication model focusing on the prerequisites of those involved in a communicative situation enables us to approach this question. When people communicate, the speaker's words may trigger a change of state in the listeners. According to Maturana and Varela [27] "communication depends on not what is transmitted, but on what happens to the person who receives it." The key criteria for successful communication within this model relates to the people involved, rather than to some kind of 'tube' between them. Thus, successful communication depends on the ability to establish situations in which mutual perturbations trigger changes in the state of those involved, which in turn lead to structural congruence (social coupling) among communicating partners. Writing and speaking do not guarantee reading or listening--or, even more important--do not guarantee the establishing of the concepts and models intended by the 'sender'. Communication is created by people who interact. Maturana and Varela state that a person's interaction domain is his or her domain of cognition. This implies that the kinds of activities in which we are involved delimit the kinds of knowledge we are able to develop. It further implies that the tools we apply in these activities delimit the kind of knowledge we are able to develop. The rejection of the tube-for-communication metaphor implies that developers and users must set aside much time for discussions and for joint activities. This is done at the expense of working alone and communicating solely in writing, which current methods primarily support. Techniques such as prototyping, mapping, future workshops, and metaphorical design (see section entitled "Tools and Techniques for Knowledge Development), are alternatives which support the development of social coupling, and thereby successful communication.

**A Model of User-Developer Communication**

We want to be able to address such questions as: "Why did a specific project fail even though it contained many user-related activities?" "Which methods should be applied in specific system development situations? .... How do system developers ensure active user participation?" In order to discuss these questions we have created a model of the communication between users and system developers. The model highlights important factors and relates them to one another. The factors are: the results of the system development process (including intermediate results); the participants' prerequisites, and tools and techniques for system description. The model is based on two distinctions--dealing with three domains of discourse and two levels of knowledge. The three domains of discourse are illustrated in Figure 1. Figure 1 illustrates the idea that design is bridge-building, since something new is created from two separate things. Design is based on two domains of discourse: the users' present work and the technological options. Here technology incorporates not only hardware and software, but also work organization. While this may seem strange, in this context we find it useful and acceptable to group these matters. Various organizational options, as well as several hardware and software options, should be considered and coordinated in order to fit together as well as possible. The result is a third domain of discourse: a new (or changed) computer system and changes in the content and the organization of the users' work. These domains typically reflect the users' and developers' knowledge and understanding prior to entering the system development process. At the outset the users have some knowledge of their present work and of organizational options. The system developers have some knowledge of the technical options with regard to hardware and software. At the outset this is all they need to know. Based on this distinction we state:

**Thesis: The main domains of discourse**

The main domains of discourse in design are:

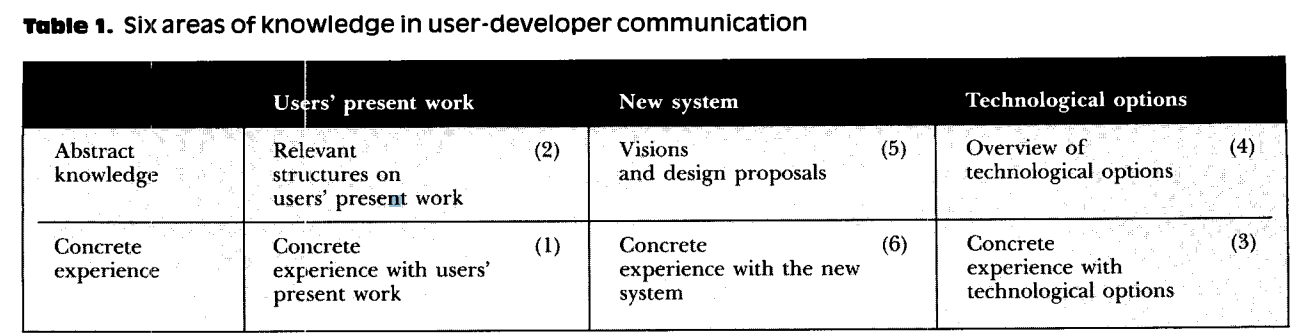
\* users' present work

\* technological options

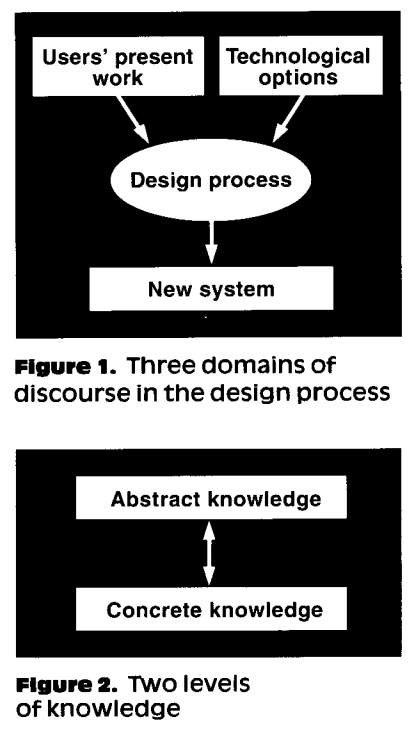
\* new system

*Knowledge of these domains must be developed and integrated in order for the design process to be a success.*

The second distinction is illustrated in Figure 2. It shows we need two levels of knowledge. We need abstract knowledge to get an overview of a domain of discourse and we need concrete experience in order to understand the abstract knowledge.

We combine the two distinctions into the model shown in Table 1. The model describes three main domains of discourse on two levels of abstraction. Altogether, we get six areas of knowledge in user-developer communication. The numbering 1 to 6 in Table 1 does not reflect a time sequence, (i.e., we are not proposing a new waterfall model). The numbering is done for the purpose of convenient reference. Various methods propose different sequencing when dealing with the six areas. Normally we would expect some degree of iteration. However, a discussion of methods is beyond the scope of this article. The six areas of knowledge comprise a classification of system development tools and techniques. The following subsections discuss, each area in more detail. The reader may wish to look ahead at Table 2 to see examples of tools and techniques in each area.

**Concrete Experience with Users' Present Work**

Developers need this area of knowledge [19]. They must have' some feeling for the users' work in order to be able to understand and to produce structured descriptions or representations of this work (area 2). They cannot rely on users' talking about their work, and they cannot rely on a requirement specification. Developers must experience users in action. If developers have no concrete experience with what is going on in the user organization and if they have no idea of the cultural potentials for change, they cannot judge the relevance of a structured description of the work. User representation in the design team does not overrule this statement. The results of dealing with this area of knowledge may come in terms of experiencing differences in working styles, normal and stress situations, exceptions, power relations, and so forth. Results may also be the formation of a common language among users and developers. Relevant Structures on Users' Present Work A relevant structure defines a common and rigorous language in which users and developers can communicate. A structure is a model of the present situation in the user organization. The model is used to identify desired changes and to evaluate consequences of proposed designs. We refer to structures in the plural, as we cannot expect to capture the richness of the users' work in a single structure. Which structures are relevant depends on the situation. Information flow is a structure offered by many methods. It is relevant when we want to automate existing data processing.

A control model is a relevant structure when we want to discuss management information systems. A model showing the variety and interrelationship of tasks carried out by individuals or a group during a typical working day is relevant when we want to discuss requirements for a new communication system.

**Concrete Experience with Technological Options**

If we want users to play an active role in system development we must provide them with technological options. This is done to stimulate their imagination and to enable them to better understand abstract descriptions of technical and organizational solutions. The relevance of activities in this area is of course dependent on the users' present experience. Even if they are daily users of some kind of system, they might not have experienced the variety of existing hardware and software. If we want designers to play an active role in designing the use of technology in organizations (although this is seldom an explicit goal, they often do this anyway) they must have organizational options. This is done to stimulate their organizational thinking and to enable them to understand the users' concrete experiences with, as well as their abstract descriptions of, organizational options. **Overview of Technological Options**

This area of knowledge is the input of technical and organizational ideas into the design process. The system developers must be well informed about possibilities and limitations regarding hardware and software in order to justify their presence in the process. If nobody in the user organization has an overview of organizational options, then this subarea has to be developed during the design process to ensure that the new computer system and the new organization fit together.

**Visions and Design Proposals**

These descriptions are developed throughout a project's lifetime. Here too, it is a question of many structures, as one alone cannot capture the totality of a new computer system and its use. The structures document the actual progress of the project as it approaches the final result, forming the basis for renewed contracts, even if these may be informal. Therefore some of these descriptions must be understandable to the users. Abstract descriptions are normally required as part of a system development project. These may be difficult for the users to understand, but they are necessary to the developers. We stress that in order for users to make decisions and assign priorities, they too need abstract descriptions to provide them with relevant structures of the new computer system, as well as of the organization in which it is to be implemented. These descriptions might very well differ from those needed by the developers.

**Concrete Experience with the New System**

The purpose of this area is to enable the users to understand abstract descriptions of the new system (area 5), and to let them experience how the new system meets their needs. The system developers also need concrete experience with the new system in order to check whether it fulfills the descriptions. In a specific project this area may already be covered through experience with technological options (area 3). This depends first and foremost on how radically the new system transcends current practice.

**Theses Based on the Model**

We now relate the model to the participants' prerequisites and we discuss which areas of knowledge each party must develop in order to facilitate genuine cooperation. The minimal starting point for a design process is actually rather narrow. Therefore it is the system developers responsibility to apply tools and techniques which allow the participants to acquire an understanding of areas in which they have little or no knowledge.

***Thesis: Areas covered by the users.***

*We can usually be sure that users cover area 1: Concrete experience with user work. We can usually expect nothing more.*

Obviously, users may be ignorant of technological options and the future system. However, it is not so obvious that they normally do not possess relevant structures or representations of their own work. The keyword here is "relevant." Traditional structures, such as organization diagrams and descriptions of the formal division of labor are not necessarily relevant. They 