Participatory Design: Structure in the Toolbox

Finn Kensing

Computer Science Department
Roskilde University
DK-4000 Roskilde, Denmark
+45 4675 7711
kensing@dat.ruc.dk

Andreas Munk-Madsen

Metodica
Nyvej 19
DK-1851 Frederiksberg C
Denmark
+45 3131 6432

Abstract

In system development users and developers need to understand several domains: users' present work, technological options and the system being developed. These domains must be understood both at a concrete level and at an abstract, structured level. No single method achieves this. Therefore we must use various methods. This paper explains why a specific method may work in some situations and not in others. The paper concludes by classifying a number of well-known methods according to their applicability.

Keywords

Participatory design, co-operation, knowledge and methods.

1. Introduction

"The users and the system developers did not understand each other". This is frequently used to explain problems in practical system development. 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 fool-proof method. System development projects fail in communication even though they use the most promising techniques.

In one project horizontal prototypes were used extensively while the requirements were defined. The intention was to ensure that the users understood what they accepted. However, the system had to undergo substantial changes before it could be used [1].

In another case the users were unable to define system requirements at the meetings with the system developers. The system developers then made an elaborate vertical prototype and expected a response from the users. However, they did not get any response.

How do we account for these apparent paradoxes? It is hard 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.

In *PDC'92: Proceedings of the Participatory Design Conference.* M.J. Muller, S. Kuhn, and J.A. Meskill (Eds.). Cambridge MA US, 6-7 November 1992. Computer Professionals for Social Responsibility, P.O. Box 717, Palo Alto CA 94302-0717 US, cpsr@csli.stanford.edu.

Comparative surveys of methods [3,18,29] are usually thorough on details but lacking in explanatory theory.

In this paper we suggest an answer to the communication paradoxes in terms of a model of user-developer communication. The model is based on theories dealing with system development as well as with communication.

The model may help us to 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 the communication related to analysis and design, i.e. to defining requirements and creating solutions. It does not cover all user-developer communication. It excludes the communication related to management and implementation.

In section 2 we define and delimit our subject. We state *the purpose and the result* of the communication in relation to other system development activities. In section 3 we discuss *prerequisites of human communication* in general. In section 4 we define a model of user-developer communication in terms of *concepts* which are useful when discussing the subject. In section 5 we state our *thesis*, that in general we can only assume a rather low-level starting point for user-developer communication. The consequence is that in general system developers need a rather large toolbox. The organization and contents of this *toolbox* are illustrated in section 6. Section 7 concludes by discussing the use of our model.

2. 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." And they proceed by describing the documents they would produce during a projects life time. 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. We will now present a model of communication in order to answer this question.

3. Communication Models

Communication is of course a key issue in collective activities like 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 each other. How is communication possible in such a context?

We sketch two communication models which are relevant to understanding and designing user-developer communication: a traditional model and an alternative model. In our opinion 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 through "a tube" it is carried 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 where 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 that same person's domain of cognition. This implies that the kinds of activities we are involved in delimit the kind 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 mapping, future workshops, and metaphorical design (see section 6), are alternatives which support the development of social coupling, and thereby successful communication.

4. 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 each other. 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.

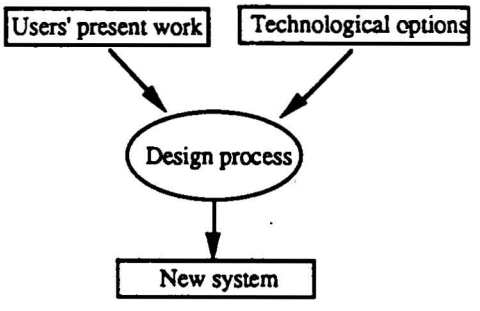

Figure 1. Three domains of discourse in the design process.

The figure illustrates the idea that design is bridge-building in the sense that something new is created based on two existing 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. This may seem strange but 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 reflect both the users' and the developers' typical prerequisites. At the outset the users have some knowledge of their present work and of organizational options. The system developers have some knowledge of the technological options with regard to hardware and software. At the outset this is all they need to know.

Based on this distinction we state:

Thesis 4.1 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 expresses the fact that 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.

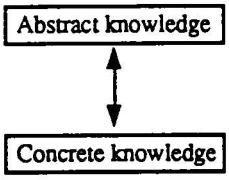

Figure 2. Two levels of knowledge

We combine the two distinctions into the model in figure 3. The model describes three main domains of discourse on two levels of abstraction. Altogether we get six areas of knowledge in user-developer communication (Figure 3). The numbering 1 to 6 in figure 3 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 referencing.

Various methods propose different sequencing when dealing with the six areas. Normally we would expect some degree of iteration. However, this discussion is beyond the scope of this paper.

4.1 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, nor can they 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 relevancy 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, power relations, etc.

Results may also be the formation of a common language among users and developers.

4.2 Relevant structures on users' present work

A relevant structure defines a common and rigorous language among users and developers, in which they 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 automatize 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.

4.3 Concrete experience with technological options

If we want users to play an active role in system development we must expose them to technological options. This is done to stimulate their technological fantasy and to enable them to understand abstract descriptions of technical and organizational solutions.

The relevancy of activities in this area is of course dependent on the users' present experience. Though they may be 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 (and even though this is seldom an explicit goal, they often do this anyway) they must be exposed to organizational options. This is

	Users' present work	New system	Technological options
Abstract knowledge	Relevant (2) structures on users' present work	Visions (5) and design proposals	Overview of (4) technological options
Concrete experience	Concrete (1) experience with users' present work	Concrete (6) experience with the new system	Concrete (3) experience with technological options

Figure 3. Six areas of knowledge in user-developer communication.

done to stimulate their organizational fantasy and to enable them to understand the users' concrete experiences with, as well as their abstract descriptions of, organizational options.

4.4 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 sub-area has to be developed during the design process to ensure that the new computer system and the new organization fit together.

4.5 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, and they form the basis for renewed contracts, even if these may be informal. Therefore some of them must be understandable to the users.

Abstract descriptions are normally required as part of a system development project. These may be hard for the users to understand, but they are necessary to the developers. We stress that in order for users to make decisions and 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.

4.6 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 fulfils 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.

5. 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 they have little or no knowledge of.

Thesis 5.1 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 labour are not necessarily relevant. They may be insufficient when it comes to discussing inexpediencies 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 5.2 Areas covered by the system developers.
We can usually be sure that system developers cover area 3 and 4: technological options. We can usually expect nothing more.

The first part of this thesis is rather obvious, or else designers would have no role in the process. However, sometimes designers are also challenged by the technological options. E.g. 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 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

Tools and techniques	Areas of knowledge 1	2	3	4	5	6
Observations [23, 30]	1					
Interviewing users	1					
Self registration [17]	1					
Developers doing users' work	1					
Videorecording [23, 30]	1					
Mock-ups [14, 15]	1					6
Think aloud experiments [23]	1					6
Drawing rich pictures [7]	1	2				
Conceptual modelling [7]		2				
Culture analysis [5]	1	2				
Object-oriented analysis [9]		2			5	
Object-oriented design [10]					5	
Event lists [28]		2			5	
Entity-relationship diagrams [28]		2			5	
Wall graphs		2			5	
Mapping [25]		2			5	
Future workshop [21, 22, 24]		2			5	
Metaphorical design [22, 26]		2			5	
Data flow diagrams [12]		2			5	
Language analysis [23, 30, 31]		2			5	
Card games [13]	1					6
Prototyping [2, 6, 16, 20]			3			6
Visits to other installations			3	4		
Literature study				4		
Study standard software			3	4		
Forum theater						6

Figure 4. Tools and techniques for knowledge development.

specific organization will prevent the new computer system and the organization from fitting together.

Thesis 5.3 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' preset 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 a 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).

However, concrete experience is also needed 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 5.4 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 by observing that our theory entails that all areas of knowledge must be dealt with in any normal system development process. The next section will discuss the toolbox we need for this work.

6. 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 figure 3 consists of 6 sections, one for each of the areas of knowledge discussed above. It accentuates the differences between the purposes of the tools and techniques, even though some fit into more sections.

A presentation of the tools and techniques chosen to illustrate the use of the toolbox is beyond the scope of this paper. The interested reader may find additional information in the references indicated in figure 4.

7. Conclusion

We find the model listing areas of knowledge in figure 3 useful for a classification of tools and techniques. A classification which developers may find helpful when planning a project.

We also find the theses in section 5 useful in explaining why projects run into trouble. This may be related to power games in the user organization or to other factors which are most often out of the developers' control. However based on our own research (Andersen et al., 1990), we claim that far too often problems in real life projects are caused by developers using inadequate tools and techniques.

We can now explain apparent paradoxes such as: "Horizontal prototypes are insufficient" (Grønbæk 1988) and "Prototypes do not substitute analysis" (Andersen 1987). A horizontal prototype does not really give users an experience with the future system. It is more like an abstract system description: the menu hierarchy implemented on edp-hardware. Thus inexperienced users will not obtain sufficient understanding of the system's functions. Vertical prototypes used successfully might solve the problem. On the other hand prototyping diverts the attention from such questions as: Do we need a new computer system? To answer this question knowledge area 2 in figure 3 must be dealt with. Analysis techniques must also be used.

Figure 4 not only indicates the areas of knowledge, in which the various tools and techniques are adequate but at the same time also highlights the areas in which they are inadequate. Conclusions concerning the more established tools and techniques such as dataflow diagrams are interesting. One of many conclusions we may draw from figure 4 is that all traditional system development methods deal only with areas 2 and 5, resulting in abstract descriptions. Thus, by themselves they are insufficient as guidelines for the entire system development process. They must be supplemented by techniques giving concrete experiences of user work and computer technology.

8. References

[1] Andersen, N.E. et al.: Professional Systems Development. Prentice-Hall, 1990.

[2] Andersen, N.E.: Brug af prototyper (Using Prototypes), Datacentralen, 1987.

[3] Blank et al.: Software Engineering: Methods and Techniques. Wiley-Interscience. 1983.

[4] Boehm, B.: A Spiral Model of Software Development and Enhancement. Computer, May 1988.

[5] Bødker, K. and J.S. Pedersen: Workplace Cultures - Looking at Artifacts, Symbols, and Practice. In [19].

[6] Bødker, S. and K. Grønbæk: Design in Action - From Prototyping by Demonstration to Cooperative Prototyping. In [19].

[7] Checkland, P: Systems Thinking, Systems Practice. John Wiley, 1981.

[8] Clements, P.C. and D.L. Parnas: A Rational Design Process: How and Why to Fake It. In Proceedings of the International Joint Conference on Theory and Practice of Software Development. Springer Verlag, 1985.

[9] Coad, P. and E. Yourdon: Object-Oriented Analysis. Prentice Hall, 1991.

[10] Coad, P. and E. Yourdon: Object-Oriented Design. Prentice Hall, 1991.

[11] Davis. G.B.: Strategies for Information Requirements Determination. IBM Systems Journal, vol. 21, p. 4-30, 1982.

[12] DeMarco, T.: Structured Analysis and Systems Specification. Yourdon Press, 1978.

[13] Ehn, P. and D. Sjögren: From System Descriptions to Scripts for Action. In [19].

[14] Ehn, P. and M. Kyng: Cardboard Computers - Mocking-it-up or Hands-on the Future. In [19].

[15] Ehn. P.: Work-Oriented Design of Computer Artifacts. Arbetslivscentrum, 1988.

[16] Floyd, C.: A Systematic Look at Prototyping. In R. Budde et al (eds): Approaches to Prototyping. Springer Verlag, 1984.

[17] Foged, J et al.: Håndbog om Klubarbejde, edb-projekter og nye arbejdsformer. (in Danish). TIK-TAK projektet, Århus University, 1987.

[18] Freeman P. and A.I. Wasserman: Software Development Methodologies and Ada. DoD, 1982.

[19] Greenbaum, J. and M. Kyng (eds): Design at Work: Cooperative Design of Computer Systems. Lawrence Erlbaum, 1991.

[20] Grønbæk, K.: Rapid Prototyping with Fourth Generation Systems -An Empirical Study. DAIMI PB-270, Århus University, 1988.

[21] Junk, R. and N. Müllert: Future Workshops - How to Create Desirable Futures. Institute for Social Invention, London, 1987.

[22] Kensing, F. and K.H. Madsen: Generating Visions - Future Workshops and Metaphorical Design. In [19].

[23] Kensing, F. and T. Winograd: The Language/Action Approach to Design of Computer Support for Cooperative Work - A Preliminary Study in Work Mapping. In Stamper, R.K. et al. (eds): Collaborative Work, Social Communications and Information Systems. Proceedings of the IFIP TC8 Working Conference. North-Holland, 1991.

[24] Kensing, F.: Generation of Visions in Systems Development. In P. Docherty et al. (eds): Systems Design for Human Development and Productivity. Proceedings of the IFIP TC 9/WG 9.1 Working Conference. North-Holland, 1987.

[25] Lanzara, G.F. and L. Mathiassen: Mapping Situations within a System Development Project. Information Management, 8 (1).

[26] Madsen. K.H.: Breakthrough by Breakdown - Metaphors and Structured Domains. DAIMI PB- 243 Århus University, 1988.

[27] Maturana. H.R. and F.J. Varela: The Tree of Knowledge - The Biological Roots of Human Understanding. New Science Library, 1987.

[28] McMenamin, S.M. and J.F. Palmer: Essential Systems Analysis. Yourdon Press, 1984.

[29] Olle, T.W. et al.: Information Systems Methodologies. Addison-Wesley, 1988.

[30] Suchman, L.A. and H.T. Trigg: Understanding Practice -Video as a Medium for Reflection and Design. In [19]

[31] Winograd, T. and F. Flores: Understanding Computers and Cognition - A New Foundation for Design. Ablex, 1986.