Designers' Role Orientations in User- and System-Centered Design Phases

Jarmo Sarkkinen
`jarsa@rieska.oulu.fi`
HCI & Group Technology Laboratory
University of Oulu, Department of Information Processing Science
P.O. Box 3000, FIN-90401 Oulu, FINLAND

Abstract

The differences between user- and system-centered design practice have been discussed broadly in the literature for a long period. The value of user-centredness has even been taken for granted occasionally, though there is clear evidence implying that broader participation of users should be evaluated relative to the situation in hand. In this paper, both user- and system-centered conducting of a design is studied with respect to how to behave in different roles. The core idea is that user- and system-centredness are needed in different phases of a design. The main problem is the proper role orientation in user-centered design. Change agentry models are used as a framework. As a result of this study, a proper role orientation in both system-centered and, especially, user-centered design is suggested.

Keywords: user-centredness, system-centredness, participatory design, role orientations

BRT Keywords: AA, FA, FB

Introduction

There has been active debate on whether paying attention to more human-centered and context-dependent participatory design yields benefits of high degree. Researchers have discussed participatory design abundantly in the literature. For example, Ehn (1993) discussed participatory design from the Scandinavian viewpoint, referring to a number of Scandinavian research projects. Greenbaum (1993) considered user-centered design as a philosophy, thus being difficult to apply, whereas participatory or cooperative design was seen as less adverse with respect to its applicability. Participatory design, if modified appropriately, can be applied even outside Scandinavia according to the view of Greenbaum (1993). Nevertheless, an understanding of how participatory design relates to *user-centered design* lacks thorough clarity of the differences. Bannon (1991) argues that user-centered design as a concept might as well be understood only as a demand, intended to software engineers, to know users, not as a design approach to promote the active participation of users to design process. Bannon (1991) would use the term *user-involved design* instead of *user-centered design* since it lets one to better grasp, what it is all about, when users are involved in a design process. This paper, however, does not try to contribute to discussion related to concepts. Postulating that both of these terms refer to the same definition is preferred. Focusing attention to the gap between system-centered way of undertaking design and user-centredness is stressed strongly in this study.

The gap between the contrary approaches is real but it needs not to be a factor having negative impacts. System-centered way of developing information systems is emphasized frequently. Transition from current practice to user-centered way of undertaking design is realized frequently by increasing the weight of user involvement to some extent, likely in a requirements gathering phase. In this paper, the weight given to user-centered approach is equal to that of system-centredness. Clausen (1994) illustrates how communication artifacts used by a designer depend on who s/he communicates to. Designer should talk about a design in a different way, depending on the other person - is s/he a programmer, another designer or a prospective user? The corresponding means of communication are, in same order, formal presentations, structured language, and with a user, either observation, histories, prototyping or scenarios. They can be used in communication in the sense that they offer an understanding of how to behave according to a situation. Clausen's remarks are important since they show how the opposite approaches differ from each other with respect to how to communicate.

There are two important factors related to systems development: *complexity* and *uncertainty* (Mathiassen et al. 1995; Dahlbom & Mathiassen 1993). It can be assumed that they both are evident in any systems development project. However, another is likely stressed more than the other, or in some cases they both can be in balance. Mathiassen et al. (1995) stress that in specifying the focus is on dealing with complexity, whereas in prototyping uncertainty is considered. Dahlbom & Mathiassen (1993) discuss construction and evolution or analytical and prototyping approach as they refer to complexity and uncertainty. These dichotomies presented provide a basis for this paper.

The research question of this paper is what kind of role orientation is suited for a designer in user- and system-centered approaches or phases. The aim is to provide designers with consciousness of how to act in different situations since a transition from system-centredness to user-centredness requires support due to the oppositeness of the approaches. Being conscious of a proper role orientation is important since a designer has to change his/her role if there is a need for transition. *Cognitive dissonance* (Festinger 1957; Lindsay & Norman 1977) can be a consequence in a situation in which a software engineer familiar with the principles of system-centered analysis and design is required to adapt to user-centered way of conducting design. The theory of cognitive dissonance implies incongruity between suddenly occurred state of affairs and the strong mental model (Johnson-Laird 1985) of a person.

The framework for the study

More accurate definition of both "user-centredness" and "system-centredness" in a design process (or a phase) must be provided since the main aspiration in this paper is to increase designers' understanding of how design processes characterized in terms of user- and system-centredness differ from each other. There is a need for pointing out the proper background to be referred to in the context in which we emphasize user-centredness. Participatory design (PD) is used as a background since it acknowledges users' central role in a design process. Traditional practice concentrates especially on system-centered modeling of internal structures and dynamics as well as implementation of a system (referring to a coding phase), whereas participatory practices stress users' opportunities to influence.

In the paper a short review of traditional and cooperative approach (Greenbaum and Kyng 1991) will be brought out to explain the biggest discrepancies of user-centredness compared to traditional system-centredness. A couple of PD principles (Greenbaum 1993b) will be introduced to describe PD as broadly as possible, however, without forgetting to depict it in brief.

It is possible to concentrate on the approach as the whole that is either user- or system-centered according to a situation, or alternatively, this whole can be divided into the distinct phases, having either user- or system-centered nature. Each of the methods used in systems development should have its user- and system-centered characteristics. In order to illustrate this point of view, a set of such phases is later identified. They are, however, only thoughts picked up from the literature, recognized to be of great value. Nevertheless, there is no aim to propose any specific characteristics since it would require further studies. In this paper, Contextual Inquiry (Beyer & Holtzblatt 1998), cooperative prototyping (Sanderson 1996; Bødker & Grønbæk 1996) and usability evaluation (Rubin 1994) are recognized to be useful for the purpose of user-centered design.

The role orientations of the change agent models (Markus & Benjamin 1996), intended to touch a work done in an organization's internal IS unit, will be applied to the context in which designers are outsiders to a user organization. To give a definition for the concept of *role orientation,* it is referred to Markus and Benjamin (1996) who offer the following definition: Role orientation includes both attitudes along with beliefs and concrete behavior of a person. Each of the role orientations of the change agent models will be reflected to what is said about concentrating on either users or systems. Consequently, conclusions will be drawn regarding what kind of role orientation is suitable in situations having different natures. The role orientations presented by Markus and Benjamin (1996) were chosen since they cover utmost and opposite role orientations. In addition to the role orientations, some structural conditions (Markus and Benjamin 1996) are applied, so that it could be possible to say how a transition to a desired role orientation can be facilitated. The assumption is that if the required structural conditions take effect, it is easier to strive for the corresponding role orientation.

User- and system-centredness

Even though the participation of users is widely preferred in the PD literature, an inherent contradiction persists between user- and system-centered ways of thinking. This can be observed by investigating the systems design literature. Both the community of PD practices and the system-oriented community too readily ignores the viewpoint of the other. *Hard systems thinking* is a normal way of thinking in natural sciences, as stated by Dahlbom and Mathiassen (1993). It has also intruded upon systems development conducted nowadays. According to *hard systems thinking* the focus is on the internal structures of a system. A system itself is torn apart from its natural context. Moreover, a means of getting rid of complexity, related to the world, is to divide a system into its subsystems and further into its properties. The previous fact poses it is natural that there is no aspiration for broader participation of users. Participation would increase a quantity of complexity. As opposed to *hard systems thinking,* there is *soft systems thinking* (Dahlbom and Mathiassen 1993) covering the opposite area, user-centredness, denoting that people can be seen as constructors building the surrounding world by perceiving their environment.

However, there seems to be an ongoing change in the direction of approving of more user-centered practices and methods. Regardless of the ongoing change in attitudes to involve users more actively (Contextual Design in theory: Beyer and Holtzblatt 1998.

Contextual Design in practice: Cleary 1999), there is still a need for bringing user- and system-centered approaches closer to each other, so that they could be treated easier within the confines of a common method. One possible way of doing this is to increase designers' understanding of how to behave in a role of designer, as involved in a user or system-centered process, and especially, either in a user- or system-centered phase of a systems development process. The assumption is that it is not a good practice to carry out design only according to system-centered *hard systems thinking*. Both kinds of phases must be incorporated in whatever design process conducted by professionals. In this section, emphasis is directed on participatory practices so that it could be possible to say something concrete about user-centredness in practice. During the next subsection this understanding will be deepened and some examples will be offered to illustrate both user- and system-centered systems development phases. System-centered phases do not achieve great emphasis here with respect to their characteristics. Henceforth, system-centredness can be alluded by comparing it to *hard systems thinking*.

Table 1: Traditional vs. Cooperative approach

Traditional approach *focus is on:*	**Cooperative approach** *focus is on:*
problems	situations and breakdowns
information flow	social relationships
tasks	knowledge
describable skills	tacit skills
expert rules	mutual competencies
individuals	group interaction
rule-based procedures	experience-based work

Greenbaum and Kyng (1991) contrast how traditional and participatory or cooperative approaches differ from each other. Consequently, they illustrate the main differences, presented here in table 1 in their original shape. Traditional way of conducting design seems to be strongly characterized by small-scaled and explicitly treatable things, whereas in cooperative approaches emphasis is on broader questions and things that can be considered implicit or tacit. Software engineers who stick to 'guidelines' of a traditional system-centredness valuate separate problems and tasks, formal rules and individuals highly, compared to cooperative approach in which designers valuate situations, relationships and group interactions to be of higher significance. It can be said that there is a prominent gap between the two.

Briefing to the basics of participatory design

Participatory design is a manifestation of what is understood by active participation of prospective users of a system being designed. Participatory design can be studied with respect to its ability to deal with questions related to either individuals in collaboration, individuals relative to product quality or even the whole micro-community (i.e. groups of people in collaboration interorganizationally). Product quality as a concept in the sense presented by Greenbaum (1993b) is intended to cover both the quality of a software, a product (whatever it is) and a broader service provided for customers.

In order to brief readers to participatory design and user-centered thinking, the perspectives presented by Greenbaum (1993b) are introduced. They are a pragmatic, a theoretical and a political perspective, being sufficient in pointing out the most significant themes of participatory design. However, they are too complex as self-explanatory and clarifying concepts. Therefore, they are called a viewpoint of software and services, a viewpoint of employees and a viewpoint of work community. These terms can hardly be misunderstood.

Viewpoints of participatory design

According to *a viewpoint of software and services*, both software engineers, executives and middle managers are able to gain benefits. In order to profit from PD, software engineers and managers have to emphasize participation of prospective users extensively and use appropriate artifacts and techniques capable of facilitating developer-user communication. Software engineers using artifacts and techniques (e.g. cooperative prototyping) are able to result in better software (also users can be assumed to be more gratified to the outcomes) and managers in better overall quality of both products and services. In this paper, it is useful to notice that the concept of *software engineer* has a clear connotation, implying system-oriented attitude, whereas the concept of *designer* poses more user-oriented stance.

From *a viewpoint of employees*, PD techniques are capable of mediating designs to prospective users more effectively. Users conduct in a hands-on manner with the help of these techniques. As a consequence, both future users and software engineers are able to understand better each other. Prototype for example is one of the possible artifacts. Of course, the properties of such an artifact are of high value, which is, however, another question not considered in this paper at all.

The third of the viewpoints, *a viewpoint of work community*, refers to the democratic organization of work community. Everyone has to be able to influence his/her own work arrangements, including the development of needed technology. What was said in the Collective Resource Approach (Ehn & Kyng 1987) about trade unions as active parties in participation is not emphasized. Participatory design is understood here as a design approach and it is seen only from the viewpoint of how design should be carried out in conjunction with users. Tendency in general seems to be in the same direction today as Bansler and Kraft (1994) state. They noticed this approach to be concentrated nowadays on design as such, rather than political questions with respect to establishing democratic work organization. There is no interest here to connect this viewpoint of design to how this prerequisite, a design context, can be achieved. However, as Bansler and Kraft (1994) submitted, we should try to strive for a new policy underlining the current needs, being a problem not included to the scope of this paper.

There is a condition that says there have to be appropriate circumstances to allow users to participate in design process. So far there has not been shared acceptance of whether the Collective Resource Approach in its original and very-cutting sense can bring about change in design situations and facilitate the introduction of participative practices (Kraft & Bansler 1994; Kyng 1994). There are numerous obstacles confronted by the approach, enumerated by Kraft and Bansler (1994). For example lack of resources (knowledge and time) was reported to be the problem along with the problems related to the broader applicability of the approach (Kraft & Bansler 1994). Trade unions clearly serve as a channel to advocate more democratic situations in which users can take part in the development of the organization of their work and in the introduction of tools to be used in this work. Regardless of ignoring this how-to-get-there, democracy is recognized to be an important factor in any systems development process. Moreover, MIS researchers (Hartwick & Barki 1994; McKeen et al. 1994), by conducting empirical studies, have questioned the significance of participation as a sole factor having effects on successfulness of a system. Participation was demonstrated to have a positive impact on user satisfaction (McKeen et al 1994). However, McKeen et al. (1994) also demonstrate that user satisfaction can be achieved without active participation. They showed situational factors to be the mediating variables capable of influencing user satisfaction. Conclusions can be drawn to say that in addition to taking the participatory principles into account, also situational factors should be considered.

First two of the viewpoints describe what kind of approach participatory design is at the end of the 20th century. Nevertheless, the last viewpoint cannot be underestimated since users possess right to influence design decisions regarding themselves. These three viewpoints together served as a basis while the corresponding role orientation was developed for user-oriented designers. Due to the problems identified above, there is a strong rationale for studying role orientations.

Examples of user-centered design

Participatory design is not a method but a set of design principles (or statements) presented in Ehn and Kyng (1987). They can be enumerated briefly as follows: 1) use context should be taken into account, 2) users know their work better than designers, 3) formal presentations are not enough, 4) jobs are not allowed to be impoverished and 5) tools should be used in a hands-on manner. The principles of participatory design can be seen as an umbrella covering different possible ways to contribute to a user-centered design process. Design process is here regarded as a user-centered one if there is at least a user-centered initial phase on which the rest of the phases of a design process are tightly based. However, there can be user-centered phases in the middle of or in the end of the design process. Even though designers can contribute to a design process by means of user-centered tools, methods and practices, the principles of participatory design appear one way or another in a way of how designers act. Generally speaking, participatory design can take shapes of different sort. Contextual Inquiry, cooperative design and usability evaluation are depicted in the following section as possible ways to contribute to a design process in a user-centered way.

Contextual Inquiry is a user-centered phase of Contextual Design method (Beyer & Holtzblatt 1998). The core idea of Contextual Inquiry is to require designers to aim at good contacts in a user organization. This is to say that they should be ready to engage in an active observation of users, wherever users' work takes place. Users are in a focal point of action in Contextual Inquiry since they are allowed immediately to comment on what they are doing in parallel with conducting real work tasks. Designers can in turn set questions to be answered. Hereby, users are provided with a chance to influence the design of computer tools. Simonsen and Kensing (1994) also encourage designers to observe users in real work situations since by doing that it is possible to create a mutual learning situation between the parties and to catch multiple viewpoints of different users.

Cooperative prototyping as a user-centered approach, described for instance by Bødker and Grønbæk (1996), is based on user-centredness. In cooperative prototyping users take part in the evaluation sessions of a future system prototype in circumstances resembling their real work situations. In this approach users evaluate alternatives so that they are able to offer designers proposals for refinements, regarding the design of a future system (Bødker and Grønbæk 1996). Cooperative prototyping is a means of conducting evaluations when there are only tentative proposals envisioned. Bødker and Grønbæk (1996) identified several learning situations wherein prototypes play central role: *future work simulations*, *idea exploration* and *making a current work visible*. Cooperative prototyping lays focus on the work context as well as the functionality of a system. Design proposals can be realized for example as mock-ups, computer-based simulations or storyboard prototypes (Bødker and Grønbæk 1996).

Usability evaluation is also a user-centered way of contributing to a design process. Due to the ambiguousness of this concept, there are several interpretations that clearly need clarification. Normally this term is divided into the two categories being *usability testing* and *usability inspection*. Moreover, it may bewilder a reader that indeed the concept of *usability testing* and *cooperative prototyping* can readily be added one to the other since the both emphasize the central role of a user as a person who evaluates 'systems' being in question. *Usability testing* is defined in terms of observation of a user whereas *usability inspection* is based on evaluation accomplished by designers themselves by means of either introspection or a set of heuristics. Usability testing and cooperative prototyping on the other hand do not differ from each other but in this paper a distinction is suggested. *Cooperative prototyping* is a technique to be utilized in the phases in which only tentative ideas exist to be evaluated whether they meet the requirements of a work situation i.e. whether envisioned design fits the intended use context. Bødker and Grønbæk (1996) state that conversations between user(s) and designer(s) occur frequently in cooperative prototyping sessions. Based on this statement it is clear that cooperative prototyping and usability testing conducted at the lab differ in terms of uni- or bidirectional communication. In usability testing facilitators guide users only prudently. As opposed to cooperative prototyping, usability testing is supposed to be a technique intended to the assessment of relatively frozen and already concretized and tangible design ideas, paying attention to the polishing of functional prototypes. This may be rather artificial distinction but it brings about needed clarity between the confusing concepts.

Presenting role orientations as a part of user-centered and system-centered design

In this chapter the aim is to study the models of change agentry (Markus & Benjamin 1996) in order to construct an insight into what kind of roles designers or software engineers play both in user- and system-centered phase. However, the models were not evaluated despite their possible shortcomings. The models were adopted as they are. They were used as a means of identifying role orientations suited for designers or software engineers in user- and system-centered phases of an anonymous and imaginary systems development method. They were applied loosely since they originally described role orientations of designers or software engineers in an organization's internal IS unit. For example, clients can be seen as users and IS specialists can be seen as designers. Nevertheless, there are no obstacles in order to apply them to the context in which designers belong to an independent service provider organization. Even though the models heavily focused on designers' ability to change, they were only used to describe how designers stance towards users.

Markus and Benjamin (1996) introduced three *change agentry models*. They have been adopted for this study in order to inquire into what kind of roles software engineers play when they are hired to join user- or system-centered design (phase) or in both kind of design phases to conduct requirements gathering, prototyping or systems design. The other supplementary or even underlying question is whether software engineers have an important role as an influencing power or whether users play more important role in underlining what should be decided on with regard to a design target (being either a system internally or its use situations and properties). Presumably this depends on what kind of design phase is in question.

The models of change agentry being used are *the traditional IS model, the facilitator model* and *the advocate model*. To be able to discuss what it means to be a software engineer who adheres either to user- or system-centered design, it is required to study whether user- or system-centredness directs software engineers to some role orientation. On the other hand this denotes equivalent to the question asking whether some of the structural conditions of the change agentry models might exist in user- and system-centered design. Moreover, it is important to study whether software engineers' roles in certain design situations reflect typical characteristics found in the role orientations of the change agent models. To answer the former question, the structural conditions of each model will be compared to the corresponding ones considered important in user- and system-centered design (phase). In order to answer the latter question, the relevance of the role orientation of each model will be compared to the role orientation suggested in user- and system-centredness.

The traditional IS model as a model in design

To study the applicability of *the traditional IS model*, the structural conditions compatible with role orientation and the relevance of the role orientation will be compared to the evident structural characteristics of user- and system-centered design phases and to the role orientation suggested in these phases. The role orientation of this change agentry model has been reformed so that it corresponds to a context wherein the aim is not to study organizational change process but the interaction of two parties, prospective users and designers.

The structural conditions of the traditional IS model

Markus and Benjamin (1996) suggest a set of structural conditions that should take effect before the application. After introducing the conditions they will be considered precisely by comparing to nearly opposite design approaches (design phases) defined in terms of user- and system-centredness. The structural conditions of the model can be listed as follows.

1. Software engineers are the sole providers of IS services.
2. Users are provided with a restricted quantity of different alternatives.
3. There is no external competition in an environment where service providers operate in order to produce services to a user organization.
4. Software engineers are accountable for supplying IS solutions to users within the confines of budget and predefined schedule and they have to work in a centralized organization.
5. The structure of an organization has to be hierarchical and given orders should

move downwards from top management to operational level. The main reason for giving orders to subordinates is to fulfill the business goals.

It is decided to start with considering the existence of five structural conditions as relevant initial conditions in user-centered phases of an imaginary design approach. Designers of a given organization are not the sole providers of IS services since users are able to choose the service provider they rely on. Originally this condition was aimed at taking effect inside an organization. Consequently, it does not fit the environment wherein markets are free. Moreover, in user-centered design designers do not tend to limit a number of different alternative ideas. One possible technique such as cooperative prototyping deliberately requires designers to offer vast quantities of mutually exclusive alternatives to users who are the proper persons to accomplish evaluations. Users can then pick up the best possible ones from this set. Designers or IS providers have to vie for the best favor in the eyes of potential consumers. The best possible service surely yields the best possible loyalty in the long range.

The structure of an IS organization has to be flat one as opposed to the structure of a *bureaucratic organization* (Laudon & Laudon 1996) if a user-centered approach to systems development is approved. It should preferably resemble the structure of the organization of *adhocracy type* (Laudon & Laudon 1996). Some structural outlines can be derived, for example, from *Business Process Reengineering approach* (Hammer & Champy 1993). Process teams or virtual teams (Hammer & Champy 1993) established to live only as long as needed to accomplish a given task are able to better concentrate on users' needs since there is more decision power directed to them. Designers in this kind of an IS organization take actively heed of the opinions of users, instead of obeying the orders of either line managers or operational managers. However, contracts and budgets cannot be totally ignored, even though there is more flexibility than in large bureaucracies. Designers are responsible for fulfilling the contracts and they also have to report on progress upwards in organization's relatively flat hierarchy.

Discussion demonstrates us that a large gap exists between the structural conditions presented in the traditional IS model and the conditions implicitly suggested in user-centered design. Evidently, conclusions can be drawn to say that the presented conditions do not direct designers to the role orientation of the traditional IS model. The different nature of competition has to be taken into account. The first and the third listed conditions are not explicitly suited for the comparison between two. However, the other items can be adopted for the context of external service providers.

Next the focus is on system-centered design phases. Neither the first nor the third condition is considered since they do not fit the situation. Users are provided with a limited number of different choices being equivalent to what is said in the second condition. There is no need for participation since software engineers have technical capabilities required to tailor the best possible solution regarding the internal structures and dynamics of a system. The rest of the listed items, namely, the fourth and the fifth are not so self-explanatory and they require additional elaboration with respect to their relevance as structural conditions in system-centered design phase. Centralization presented in the fourth condition is a characteristic not suited for the situation since the context is not an internal IS unit of an organization. Flexibility is not needed in fulfilling the requirements of schedules and budgets. Designing a system to behave in a certain way is easier than to fulfill the needs of future users. Consequently, schedules and budgets can be predicted easier, as was implied in hard systems thinking. The structure of an organization does not have to be a flat one since the job of software engineers is based on what is required to implement. Someone else is responsible for carrying out requirements gathering. There is a prominent reason for why line managers or operational managers should intervene to technological decision making of software engineers. Superiors control that a team of software engineers designs and constructs a system according to what was required by the future users.

Discussion clearly points out that in system-centered design phase the structural conditions of the traditional IS model are more relevant than in user-centered design phase, even though there are a couple of structural conditions found not to be of significant nature. The most relevant conditions found to be valuable in system-centered phase are 2, 4 and 5. The first and the third conditions are not suited for the selected context. Moreover, the concept of centralization in the fourth condition does not fit the context. However, discussion points out that the structural conditions (2, 4 and 5) direct software engineers strongly to the role orientation of the traditional IS model.

The traditional role orientation

IS specialists familiar with the role orientation of the traditional IS model can be characterized in terms of technical expertise and individuality of performance with no responsibility related to organizational change process. These two qualifications imply that software engineers of this kind ignore use situations. According to the model, IS specialists comply with the orders of operational managers without questioning the acceptability of the orders. Thus they are seen as persons motivated by the goals of the others. IS specialists consider technology to be the power capable of altering organizational structures, working procedures and even organizational culture. However, according to the insight of system-centered software engineers, they have only an obligation to build solutions to given problems without taking any responsibility for the consequences appearing in an organization wherein the technology is used. Software engineers having role orientation of this type are called lonely riders in this paper. (Markus & Benjamin 1996)

User-oriented attitude cannot be based on the above-mentioned traditional role orientation model. Designers cannot consider technology to be the changing power if they acknowledge that users know their own work best. Neither can they rely on formality of technology nor act as persons who do not take part in a mutual learning process in conjunction with user representatives. They cannot be motivated by what the superiors say but they should be present as users participate. As opposed to the traditional role orientation, user-centered designers should not emphasize technological problems but be conscious of use situations and breakdowns occurring. Based on what was said in hard systems thinking, the traditional role orientation seems to be suited for software engineers with system-centered stance.

The facilitator model as a model in design

The facilitator model enables to study user- and system-centredness from the other utmost viewpoint, compared to the traditional IS model. There are lots of similarities compared to the way of how consultants of an external service provider enterprise consult clients. Since this model fits the decentralized environment, the structural conditions are more favorable to use, relative to the ones of the traditional IS model. Though the structural conditions ought to be used in the context of IS unit having departments of a same organization as clients, the interaction between designers and users can be looked at on the same basis. Rather than regarding the conditions of the model as structural characteristics, three fourths of them ought to be seen as qualifications of a role orientation.

The structural conditions of the facilitator model

Markus and Benjamin (1996) brought out four structural conditions in the facilitator model. They can be listed briefly as follows. 1) Software engineers should not use their technical competence. 2) They have to be outsiders to a user organization (this is the only one regarded here as a structural condition). 3) Software engineers as facilitators do not have to be responsible for questions related to business. 4) There cannot be hierarchical authority over users. These conditions are rather self-explanatory, and due to this, they will not be elaborated. However, they are compared to user- and system-centered design phase. Clearly they are reverse to the structural conditions of the traditional IS model. This oppositeness is one of the possible reasons for why it can be difficult to software engineers to embrace user-orientation.

Software engineers possess a vast amount of technical competence, but in user-centered design phases they are not allowed to use it. Designers strive for increasing users' capabilities to make decisions, with respect to their organization and information systems used in their work tasks. The facilitator model considers users as persons capable of selecting and even building technical solutions, whereas in system-centered phase software engineers are in charge of system-oriented modeling and construction of an information system.

Designers cannot be the employees of an organization they are hired to serve in user-centered design. This is required since outsiders as neutral participants intervening to a user organization can observe users objectively. However, being either an outsider or an insider is not a critical question in system-centered design. Actions of a lonely rider are based on what the others have already decided. It is supposed here that requirements gathering should, of course, be a user-centered phase.

Contracts should not be blindly complied with in user-centered design since requirements set in advance hardly correspond with what is achieved as a result of a user-centered and iterative design approach. This implies that user-centredness is best suited for the design conducted in a flat organization wherein there is flexibility to fulfill what is said in contracts. However, there are things such as schedules and budgets that cannot be wholly ignored. Self-directing teams dealing with process-oriented work (Hammer and Champy 1993) can be seen as a good basis for a user-centered design situation. Consequently, designers are responsible for business goals to themselves instead of managers. As opposed to facilitators, software engineers as lonely riders act as persons dependent on decisions made by someone else. The role of them is clearly the opposite compared to that of facilitators.

Designers can't appeal to their authority over clients in user-centered design with respect to defining what the functionality of a system should be. They should always take users' opinions into account, for example, during each iteration time of gathering requirements in cooperative prototyping. Neither can they regard users as inferior to them in any user-centered phase, even though they have technical authority over users in system-centered phases. Having technical competence implies that users do not need to take part, for example, in the modeling phase of a system.

Conclusions can be drawn with respect to the structural conditions. Briefly depicted it can be said that the structural conditions direct designers strongly to the role orientation of the facilitator model in user-centered phases, whereas in system-centered phases software engineers embrace the traditional IS model. The second condition is not a significant one with respect to whether the structural conditions of the facilitator model direct designers or software engineers to the corresponding role orientation in user- and system-centered design phases. The conditions 1, 3 and 4 clearly direct user-oriented designers to the facilitator role orientation, whereas the conditions 1, 3 and 4 differ from the corresponding ones of system-centered design.

The facilitator role orientation

The worldview of the traditional IS model seems to be clearly technology-oriented, whereas the role orientation of the facilitator model is in favor of human-centered viewpoint to design process. Human beings and in this case *users* are in charge of change, neither technology nor designers as change agents. In accordance with this human-centered stance technology can be seen as a tool used by users in the reformed circumstances of an organization. Facilitators as consultants advise users of how to get literally rid of the dependence on designers. To be able to make own decisions, users need a great number of design proposals suggested by designers. The main purpose is to increase users' capabilities to create joint insights of the subjective thoughts of each person. Rather than providing technological solutions, designers as facilitators should inform users of how to build systems and how to make common choices.

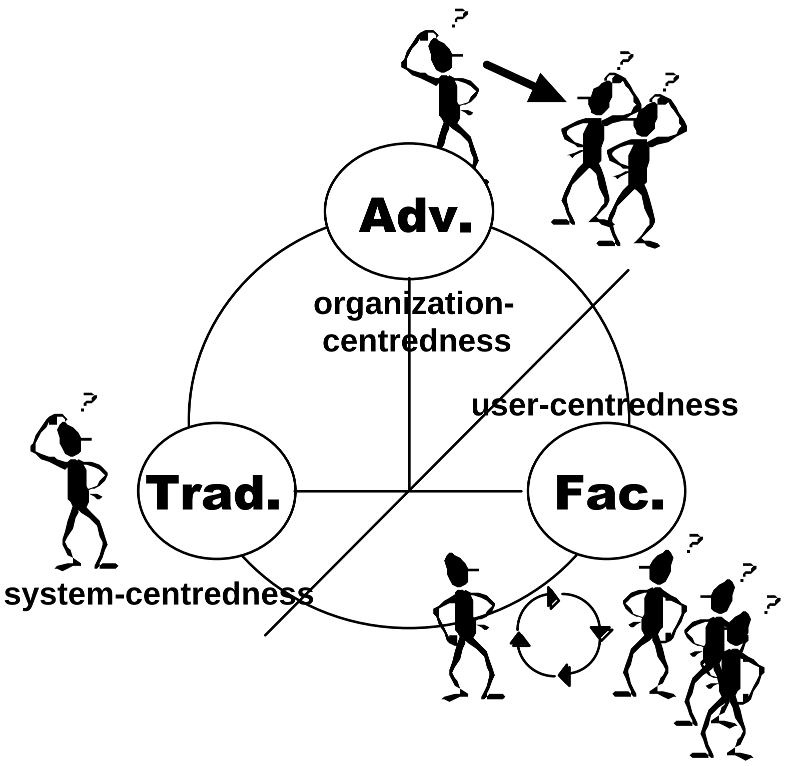
The biggest discrepancy between the role orientation of the facilitator model and the role orientation of user-oriented designers is an attitude on the implementation of a system. Facilitators regard users as implementers, whereas user-centredness does not aim at contributing to the implementation of a system at all. User-centered phases establish a basis to be utilized later during a development process in system-centered design phases. Then the aim is for example to define how the objects of a system collaborate with one another and how messages move between the objects. In cooperative prototyping sessions users are in a focal point of action. Designers are not permitted to recommend any technical solutions, but they have to provide users with appropriate presentations describing how the requirements of users could be incorporated in a future system.

It is suggested here that there should be a shared responsibility of users and designers for the results in user-centered phases. Though decision power is channeled to users in quantity, software engineers reserve a right to model and implement a system. Evidently, software engineers are capable of making technical decisions. Users are vice versa more capable of saying what a good solution is. The core idea is that the construction of a system per se is not a development phase to alter the structures of a work community. Responding to questions related to the re-organization of work tasks is appropriate earlier in a phase that is a user-centered one.

Finally conclusions can be drawn with respect to the role orientation of the facilitator model. The facilitator role orientation is consistent with the role orientation being suited for user-oriented designers. However, there is a big difference with respect to responsibility. User-centredness requires both of the parties, designers and users, involved in the design process of a new system to take a shared responsibility for the results, which is not consistent with what was required in the facilitator model. The facilitator role orientation does not fit the system-centered design in any sense since the facilitator role orientation and the role orientation of lonely riders totally contradict. Each of the presented constituents of the facilitator role orientation is from the opposite edge of the scale compared to the role orientation of lonely riders.

The advocate model as a model in design

The third of the models, *the advocate model*, is used in order to supplement the description of the roles software engineers or designers play, while they are developing a system in both user- and system-centered phase. Even though the advocate model does not share any characteristics with above-mentioned two other models, it is situated in the middle of the traditional IS model and the facilitator model. However, it is located a bit further from the center point of the traditional IS model and the facilitator model. Picture 1 illustrates how the models lie on the opposite edges relative to one another. Symbols in the picture have the following meanings. Interrogative mark symbolizes decision making process and responsibility. Arrows symbolize the direction of proposals. Human being figures have been set in the next order: Software engineers or designers are on the left side, whereas users can be found on the right side of the software engineer.



Picture 1: Three types of 'centredness'

The advocate model as applied to systems development is a means of attaining the goals of an organization. It is thus a way to promote the strategy defined by the strategic management of an organization. The concept of *organization-centredness* can be used to point out the nature of it. Consequently, it does not fit the phases in which the emphasis is on systems (system-centered phases). Information systems are a part of an organization, even a very significant section. However, it is not wrong to divide them into the distinct parts: *information systems* and *organizations infrastructure*. The aim of the distinction is to point out that *systems building* (construction of a system comprising both modeling and coding) should not be a phase having effects on the reengineering of the structure of an organization or the re-form of work tasks of employees. Due to this the model will not be applied to system-centered phases.

The advocate model provides quite a heterogeneous set of structural conditions, as stated by Markus and Benjamin (1996), compared to the traditional model and the facilitator model. The model regards software engineers as superior to users, not from technical point of view, but since they think they know better than users what the proper choice or strategy is. Markus & Benjamin (1996) articulated this point as follows: *"Users don't know what they want, and what they want is not what they need"*.

The structural conditions of the advocate model

Markus and Benjamin (1996) enumerated three structural conditions being somewhat paradoxical compared to the worldview of the model. Therefore these conditions are here suggested to be mutually exclusive. They can be listed as follows.

- Software engineers as advocates do not use authority over their users since they have not been delegated power to control. They are able to dispense valuable resources.
- Alternatively they can be considered line managers with an appropriate quantity of authority, and therefore they are accountable for achieving business returns.
- Also they could be seen as peers compared to operational employees, and according to this view both advocates and future users are the employees of a same organization.

The first condition can be elaborated by saying that dispensed resources comprise funds, equipment, advice and positive regard. Nevertheless, there is a problem with respect to this condition. Software engineers having no authority are not capable of advocating their solutions effectively, which means that software engineers cannot persuade users to accept presented ideas. The second condition needs no additional elaboration. Supremacy makes it possible to advocate proposals efficiently. However, there is a problem found to be an evident part of this condition. Ideas being not one's own are easily resisted, especially if any rationales have not been pointed out. Therefore in conjunction with this condition, the importance of commitment should be considered, as Markus and Benjamin (1996) mention. Regardless of questions related to acceptability, this condition is suggested here to be the best one in aiming at successful applying of the model. The approval of this condition indicates, by no means, that the model per se would be the best possible one. Being an advocate of the third type suffers from a problem regarding power relations of employees. If one of the employees at the operational level of an organization acts as an advocate being superior to the others, lack of legitimacy appears to be strong among the employees. This in turn hinders the concretization of decisions. It can be assumed that the best possible advocates come from an external service provider organization. As was demonstrated, each of the conditions clearly carries a built-in restriction.

The second condition cannot be considered a basis on which user-centered design can be established. Cooperation between users and designers cannot be based on the strong authority of designers. Neither is the third condition a relevant one since in this paper the context requires designers to come from a separate service provider enterprise. The first condition saying it is appropriate to dispense resources is a suitable condition in user-centered design. According to this condition, designers do not possess authority. Designers have to provide users with information, equipment etc. in user-centered design. Nevertheless, the condition does not direct designers to the role orientation of the advocate model, which is due to the nature of the model: Advocates should be able to use power to persuade users. The second condition would have directed advocates to the corresponding role orientation. Nevertheless, it would not be a suitable one to be consistent with user-centredness.

The advocate role orientation

Software engineers are characterized in accordance with the advocate role orientation in terms of qualifications presented next. Advocates are required (presumably by the top management of an organization) to direct users to a desired goal by means of communication, persuasion and the use of exciting future visions, even shocking, manipulating and using the power to some extent. Using the power implies the legitimized authority of software engineers. Advocate regards users as persons who do not know what they need. In some cases, they can ignore users who resist advocated change, and as opposed to this, they can offer rewards to users in order to achieve desired goal supposed to be the best possible one. Advocate thinks that anything goes. If users won't commit to the proposed strategy, advocate will give up. The importance of a mutual commitment should be recognized in case of the application of the advocate model.

Technological artifact as such cannot be considered end but it should have to do with implementing the preferences of the users in question. This is an underlying assumption in user-centered way of thinking. Technological solutions should not define a use situation but vice versa a use situation should determine how the technology should function to fit the situation where it is used. The technological choices implemented in a system by lonely riders should be as hidden as they optimally can be in a use situation. Software engineers should stress to what extent they unintentionally, or even deliberately, manipulate users to favor their proposals while they provide users with the sketches of a design for example in cooperative prototyping. In user-centered phases, users participate in decision-making process regarding questions of the use of a system being developed for their needs. The advocate role orientation supports participation by laying stress on the shared responsibility of participants. Nevertheless, the procedures applied and committed to by advocates are not suited for user-centered design.

Obviously, we can conclude that the traditional IS model as a whole is eligible for the basis of the role orientation of system-oriented designers, whereas user-centered design practices require speciality of different sort to characterize the behavior of a corresponding designer. By blending together qualifications of facilitators and advocates, it is possible to determine an appropriate role orientation for designers favoring the principles of user-centredness. Majority of the specific qualities of facilitators fit the user-oriented designers, apart from an aspiration to get rid of the responsibility for outcomes. The role orientation of user-oriented designers has to be added to with a requirement to share responsibility jointly with users. Shared responsibility originated in the advocate role orientation. It is the only qualification of the model to be applied by designers favoring the principles of user-centredness.

Conclusions

The main contribution of this work was to gather a coherent understanding with respect to how software engineers or designers act according to a situation (either user- or system-centered) by taking situational roles of either lonely riders, facilitators or advocates. User- and system-centredness were found out to be approaches having different models of thinking (soft and hard systems thinking). Moreover, they greatly differ in reference to how software engineers or designers conceive the work, including tools, procedures, attitudes and beliefs. The traditional role orientation is here pointed out to be an eligible role interpretation for system-oriented software engineers. On the contrary, user-oriented designers, however, are deliberately urged to consider facilitator role orientation as a suitable one. Needless to say again, they should not underrate an obvious importance of shared responsibility, while they collaborate tightly with users to catch design ideas of users to be integrated in a system. Prominently, each of the models of role orientation promotes the interest of a different group, either consisting of software engineers, user representatives or managers. Lonely riders, in essence, aim at self-interest, although they, of course, serve also the goals of users, as a consequence of fulfilling their own self-interest. On the contrary, facilitators work for the goals of users, whereas advocates in turn emphasize decisions made by strategic management.

The practical importance of this study culminates in providing an understanding of how designers applying the principles of user-centredness can be supported by offering consciousness with respect to the role orientation seen as a proper one in user-centered design. On the basis of this paper we can assume that an awareness of the role orientation of the facilitator model weakens the negative impacts of problems related to the transition from system-centered to user-centered design. It is likely that in a very small development organization only a few persons take part in each phase of systems development. In the context of these enterprises, provided knowledge can be of great value since a same person should change one's role orientation, if a transition from user-centredness to systems construction or vice versa is undertaken. Since there is a large gap between the traditional IS model and the facilitator model, as depicted in picture 1, software engineers need support. Prior to trying to support designers with a set of tools, as well as proper practices and even a method, there first has to be an appropriate insight into how to act in different situations. This paper focused on the former problem related to the qualifications of designers.

Proposals provided in this paper are suggested to be applicable for how designers could conceive their role in a situation where they follow the principles of user-centredness. Further studies should, however, be conducted in the future to point out it more comprehensively that the facilitator role orientation with certain reservations is the best possible one in guiding designers in user-centered design. This poses a need for carrying out additional literature-based investigations. In addition, there shall be further studies related to the role orientation of designers who carry out distributed participatory design in a geographically distributed environment. Moreover, as a research topic of future studies will be a question such as, whether tool-based support and related practice in a geographically distributed design situation can provide appropriate help.

References

Bannon, L. 1991. From Human Factors to Human Actors: The Role of Psychology and Human-Computer Interaction Studies in System Design. In Greenbaum, J. & Kyng, M. (eds.). Design at Work: Cooperative Design of Computer Systems. Hillsdale (N.J.): Lawrence Erlbaum, pp. 25-44.

Bansler, J. P. & Kraft, P. 1994. Privilege and Invisibility in the New Work Order: A reply to Kyng. Scandinavian Journal of Information Systems, Vol.6 (1994), No.1, pp. 97-106.

Beyer, H. & Holtzblatt, K. 1998. Contextual Design: Defining Customer-Centered Systems. San Francisco (Calif.): Morgan Kaufmann.

Bødker, S. & Grønbæk, K. 1996. Users and designers in mutual activity: An analysis of cooperative activities in system design. In Engeström, Y & Middleton, D. (eds.). Cognition and Communication at Work. Cambridge: Cambridge University Press, pp. 130-158.

Clausen, H. 1994. Designing Computer Systems from a Human Perspective: The Use of Narratives. Scandinavian Journal of Information Systems, Vol.6 (1994), No.2, pp. 43-58.

Cleary, T. 1999. Communicating Customer Information at Cabletron Systems, Inc.. Scandinavian Journal of Information Systems, Vol.VI (1999), No.1, pp. 44-49.

Dahlbom, B. & Mathiassen, L. 1993. Computers In Context - The philosophy and Practice of Systems Design. Cambridge (Mass.): Blackwell Publishers.

Ehn, P. 1993. Scandinavian Design: On Participation and Skill. In Schuler, D. & Namioka, A. (eds.). Participatory Design - Principles and Practices. Hillsdale (N.J.): Lawrence Erlbaum, pp. 41-78.

Ehn, P. & Kyng, M. 1987. The collective Resource Approach to Systems Design. In Bjerknes, G., Ehn, P. & Kyng, M. (eds.). Computers and Democracy. Aldershot: Avebury, pp. 17-58.

Festinger, L. 1957. A theory of cognitive dissonance. New York: Harper.

Greenbaum, J. & Kyng, M. 1991. Introduction: Situated Design. In Greenbaum, J. & Kyng, M. (eds.). Design at Work: Cooperative Design of Computer Systems. Hillsdale (N.J.): Lawrence Erlbaum, pp. 1-24.

Greenbaum, J. 1993. A Design of One's Own: Towards Participatory Design in the United States. In Schuler, D. & Namioka, A. (eds.). Participatory Design - Principles and Practices. Hillsdale (N.J.): Lawrence Erlbaum, pp. 27-38.

Greenbaum, J. 1993b. PD: A personal Statement. Communications of the ACM, Vol.36 (1993), No.4, p. 47.

Hammer, M. & Champy, J. 1993. Reengineering the Corporation - A manifesto for business revolution. London: Nicholas Brealey.

Hartwick, J. & Barki, H. 1994. Explaining the Role of User Participation in Information System Use. Management Science, Vol.40 (1994), No.4, pp. 440-465.

Johnson-Laird, P. N. 1985. Mental Models. In Aitkenhead, A. M. & Slack, J. M. (eds.) Issues in Cognitive Modeling. Hillsdale (N.J.): Lawrence Erlbaum in association with The Open University, pp. 81-99.

Kraft, P. & Bansler, J. P. 1994. The Collective Resource Approach: The Scandinavian Experience. Scandinavian Journal of Information Systems, Vol.6 (1994), No.1, pp. 71-84.

Kyng, M. 1994. Collective Resources Meets Puritanism. Scandinavian Journal of Information Systems, Vol.6 (1994), No.1, pp. 85-96.

Laudon, K. C. & Laudon, J. P. 1996. Management Information Systems - Organization and technology. New York: Macmillan.

Lindsay, P. H. & Norman, D. A. 1977. Human Information Processing - An Introduction to Psychology. Orlando, Florida: Academic Press, Inc.

Markus, M. L. & Benjamin, R. I. 1996. Change Agentry - the Next IS Frontier. MIS Quarterly, December 1996, pp. 385-407.

Mathiassen, L., Seewaldt, T. & Stage, J. 1995. Prototyping and Specifying: Principles and Practices of a Mixed Approach. Scandinavian Journal of Information Systems, Vol.7 (1995), No.1, pp. 55-72.

McKeen, J. D., Guimaraes, T. & Wetherbe, J. C. 1994. The Relationship Between User Participation and User Satisfaction: An Investigation of Four Contingency Factors. MIS Quarterly, Vol.18 (1994), No.4, pp. 427-449.

Rubin, J. 1994. Handbook of Usability Testing. New York : Wiley, cop.

Sanderson, D. 1996. Partial Success and Partial Failure in a Commercial Development Project. In Blomberg, J., Kensing, F. & Dykstra-Erickson, E. (eds.). Proceedings of the Participatory Design Conference PDC'96, 13-15 November 1996, Cambridge, Massachusetts: USA, pp. 81-92.

Simonsen, J. & Kensing, F. 1994. Take Users Seriously, But Take a Deeper Look: Organizational and Technical Effects from Designing with an Ethnographically Inspired Approach. In Trigg, R., Anderson S. I. & Dykstra-Erickson, E. A. (eds.). Proceedings of the Participatory Design Conference PDC'94, 27-28 October 1994, Chapel Hill NC: USA, pp. 47-58.