

The Closer the Better: Effects of Developer-User Proximity for Mutual Learning

Sturla Bakke^{1(✉)} and Tone Bratteteig²

¹ Department of Technology, Westerdals - Oslo School of Arts, Communication and Technology, Oslo, Norway

sb@westerdals.no

² Department of Informatics, University of Oslo, Oslo, Norway

tone@ifl.uio.no

Abstract. In this paper we report from a software development project, where much attention was given to the users – so much, in fact, that the developers moved in with them and stayed. Our aim has been to understand the effects of this level of proximity in the cooperation between developers and users. We discuss the impact on continuous knowledge exchange, organisational structure and accountability when the developers move in. How do the participants experience the mutual learning process? Based on the findings, we offer the two suggestions: (1) that the mutual learning necessary for establishing a common understanding of the character of a user-centred software system and its intuitive operation has a greater possibility of succeeding when developers and participating users are located in the immediate vicinity of each other, and (2) the impact on user interface design is visible through early user participation, leading to the sense of user interfaces facilitating an immediate user interaction.

Keywords: Participation · Reciprocal learning · Organisational structure

1 Introduction

“Mutual learning” is one of the fundamental characteristics of participatory design, denoting the ways in which developers and users work to share enough knowledge to understand each other’s perspectives on the future IT system [1]. In this paper, we discuss the character of and prerequisites for mutual learning between users and developers in a software development project in a shipping company. We pay particular attention to the level and character of the physical proximity between users and developers, and analyse the project as it unfolded for patterns in organisational structure and looked at the formal and informal knowledge exchange of the mutual learning process.

We have followed the process of developing a new software system in a shipping company. The company is regarded as a world leading shipping company, and is among the biggest entities in the business of transportation and storing chemicals and bulk liquids. The company owns and operates a large fleet of chemical tankers, both globally and regionally.

A secure software solution for stowage planning of chemical tankers is important for the business as well as for security. A secure system supporting these operations might ultimately prove to be the difference between life and death, since stowing, potential, hazardous liquids on board chemical tankers might be, well – hazardous. Getting it right is vital.

A motivation for creating the new system was that the old one had become slow and outdated. Also, they were about to replace their business system, which would render the stowage system they already had, obsolete. When they decided to build a new system, the question was whether they should change and adapt what they already had (an old, slow and a bit dull system) or buy something new. Searching for an off-the-shelf solution, they realized that there was no off-the-shelf software fitting their requirements. The field of stowage planning on chemical tankers is rather specialized, and instead of tailoring a SAP or ORACLE system, as one might have done when developing large information systems in recent times [2], the company chose to develop the new software from scratch.

The paper is based on fieldwork on the development process in the shipping company HQ. The new system turned out to be immediately useful for both expert and novice users, hence we wanted to understand what was done right in the development of this successful system:

- What did they do well – does the proximity of developers and users explain the success?
- How close together should developers and users co-locate in order to experience positive effects? Same address? Same floor? Sharing the coffee machine?
- Our main research question is: how can we understand the prerequisites for and mechanisms of co-location in a developer/user relationship with regards to organisational structure and mutual learning?

The paper is structured as follows. We start with a literature review of mutual learning and proximity from the participatory design (PD) and computer-supported cooperative work (CSCW) fields, respectively, followed by research method in Sect. 3. We discuss our empirical data with reference to specific themes in four sections: (1) goals and visions, (2) organisational and procedural structure, (3) proximity, and (4) Accountability. Section 5 concludes the paper.

2 Literature

In this paper we lean on the participatory design (PD) discourse that was pioneered in Europe, and especially in Scandinavia, in the 1970s social, political and, subsequently, technological transformation of work conditions [2–10]. The ambition of PD was increased influence and participation by the workers towards the computerisation of the workplace. PD emphasized that workers should have a voice and have a say [10] and a lot of methods has been developed to facilitate cooperation between developers and users in the design of a future system or artefact [1, 4, 7]. Mutual learning is a key aspect in PD: developers should learn from users about their needs and wishes for a new solution, but users should also learn about technical possibilities so that they can participate in generating design ideas. The point is that both developers and users learn

from each other during the process, hence the discussion about the new solution develops as their knowledge develops. This also means that users need to be present all through the process since a static description of needs developed before the process cannot replace the learning users who get new ideas as their knowledge about the possibilities and problems grow [1].

Traditionally, mutual learning happens all through the development process, collapsing the traditional division between analysis and design [1]. Prototyping or concretizing bits of the solution acts as a way to learn more [3]. Analysis of the work practices looks for trouble and problems to be solved, hence contributes to both solving and setting the problem [11]. Using Schön [11–13] we see design as sequences of ‘see-move-see’: the designer ‘sees’: understands and evaluates the situation, s/he makes a ‘move’: selects a way to change it and tries it out, and ‘sees’ the new situation: evaluates if the move seemed productive towards the end (the final system/artefact) [5]. Users and developers both can participate in seeing and making moves, however, users normally participate more in making the choices to select from when making a move, and in the seeing parts (before and after the move) [5]. Mutual learning hence needs to support the ‘see-move-see’ sequences so that all participants get the possibility to participate and collaborate as much as possible.

Bratteteig et al. [1] discuss the preconditions for mutual learning to happen, emphasizing repeated contact over time. The activities scheduled to encourage learning from each other represent a shared experience for the participants and act to build a common ground for design. The learning that takes place ideally enables the participants to understand each other better – and build mutual trust [14]. A rule of thumb is that the participants should understand the logic of a design idea and they should recognize how the argumentation is grounded in the professional logic of the other participants [1, 3]. Throughout the mutual learning process, the participants develop their understanding of the use context and the technical possibilities enabling new ideas to emerge as the understanding deepens. When repeated contact over time with the same people is not possible, the mutual learning process has to be designed differently [1].

Repeated contact over time with the same people presupposes some organizational and physical conditions, e.g., a formal organization of the communication that enables the participants to take the time to collaborate and being located close enough for them to meet regularly. This is, however, not always possible [1], e.g., in cases of large, distributed organizations or in designing for the public.

Proximity is, of course, a topic in CSCW: much of CSCW is about supporting collaboration over distance. Thus, the characteristics of proximity are key to understand and develop such support. The focus in CSCW is mainly on the awareness of co-workers’ activities as a prerequisite for one’s own (part of the) work [15–20], and not on how proximity affects the mutual learning or trust.

3 Case and Research Approach

We have conducted a longitudinal study following the development and implementation phases, ranging from late 2011, until summer 2013. The empirical material consists of observations of the cooperative process, namely the communication

between the development team, project owners and participant super-users, in addition to interviews with members of the developer team, management, and super-users.

During this period, we have followed the development of the system development process from the very beginning, following SCRUM teams, project management, and participant (super users) and regular users during this development period.

3.1 Data Collection

Data collection consists of semi-structured interviews with stakeholders, developers, project management, expert- and super-users:

- [MR] - Management representative. Member of the governing group for the ORCA development process. He was involved in the development of the previous system, as a project manager. He has participated as an advisor, and as a member of the steering committee. Interviewed two times; once, early in the project, and once after the system had been in use for about a year.
- [HD] - Head of the Development team, Programmer and SCRUM master in the project. Interviewed two times; once, early in the project, and once towards the end.
- [PO] – Project owner. Involved in the project from mid-2011. Project manager for the development of the new stowage system, ORCA. Interviewed three times; once, early in the project, once towards the end of the development period, and finally after the system had been in use for about a year.
- [dev] – Developer and co-located member of the SCRUM team. Interviewed once, mid-way through the development process.
- [SU] - Super-User, who acted as a premise provider in terms of which functionality that should be included in the software, and as a participating user representative. He was what we in actor-network theory would have called an Obligatory Passage Point; the one person that all involved participants had to go to with wishes, ideas and suggestions for how the new software should function and be operated. Educated as Captain; 25 years of naval experience, 11 of them as Chief Officer or Captain at Odfjell Tankers. The super-user has worked as an operator and having had the role as super-user at the company headquarters for the last 12 years. Interviewed three times; once, early in the project, once towards the end of the development period, and finally after the system had been in use for about a year.

The interviews lasted about 45–90 min. Observation sessions of regular users lasted for approx. 2–3 h. This involved observing the employee's work tasks. Un-targeted observations have been continuous while the researcher has been present in the field.

3.2 Case Overview

The company decided early in 2011 to develop a new ICT platform in order to provide one common software solution to support Odfjell's commercial shipping activities onshore as well as on board the vessels. The new stowage software, named ORCA (Odfjell Resource Control Application), which is described in this paper, is part of that platform. The new system was meant to support a new workflow, allowing the operator

on land and the shipboard officer to work on the same stowage plan with the same software. This development period lasted from late 2011 to early 2013. From the very beginning, the company vision was to facilitate a high degree of user-centeredness, and the key personnel within the company discussed what would be the most important requirements for the new software they were about to develop, such as support for ‘intuitive’ workflow for skilled users, and allowing operators on land and officers at sea to work on the same stowage plan with the same software.

A main characteristic of the software development process is that the software developers and the users moved in and stayed together from the conceptual model phase until the end of the implementation period almost two years later. Bringing users and developers together is not a new or revolutionizing move within software engineering or development processes related to technical innovation. There are, however, aspects of this kind of co-location, related to the level of the proximity and cooperation that are interesting to discuss. The replaced system was developed in a traditional user-developer setting, very different from the dynamics experienced by the co-location of the developer team in this project.

In the following, we discuss our findings thematically, with emphasis on *goals and visions, organisational and procedural structure, proximity, and accountability*.

4 Findings and Discussion

4.1 Theme 1: Common Goals and Visions

In this project, the primary goal focused on developing what they called «a user’s system», meaning that their main focus was a heightened user-experience. The users were included from the very beginning through both formal and informal communication. A group of people was formally picked to contribute to the development, but the initial brainstorming session had the rather informal character of a mountain leisure trip. Other techniques for collaboration included mail, from officers at sea, and mail correspondence between regular users and the super-user, and face-to-face meetings between super-user and the developer team about functionality and interface issues.

There had been some signs of discontent with the previous system, and it was important for the company that input from the users were taken care of in a different manner than their development project of the previous stowage software. During the last eleven years they have based their daily stowage operations on one specific application: Othello, that, although sophisticated, contained features that were rarely used, and in principle just made it more complex and slow to work with. The character of co-location experienced in the ORCA project is contrasted with the development of the Othello system, in which the software company did not co-locate. The Othello Project met a great deal of resistance, with heated discussions among the group of people involved in the development of the software, and the cargo brokers that were going to use it. During the Othello development period, the management struggled to convince people to accept what they were about to design.

This time, both the management and the users wished to develop a software system that would support the natural task flow of how humans would stow a ship.

[PO:] -The intention with the use of the software is, if you know how to stow a chemical tanker, you should, intuitively, understand how the system works. If you do not know how to stow a chemical tanker, then you don't have the experiential knowledge of using the system. Experienced personnel should not need long training. The system should be intuitive. It's a bit like an experienced operator thinking «If I'm doing THIS, then THAT should happen», and then THAT will happen.

Thus, they had an early focus on what meets the user: the user interface, to such an extent that the very first digital sketches on how the screen elements would actually look like, largely survived the entire development process (see also [5]).

[PO] -Focusing on the user interface, was the very first thing on our list when we started the project. This means that the screen elements in the first mock-up are more or less identical to the way they are now. We have changed some details here and there, and changed a bit of the information that is in there, but the main principle, of making the system as user-friendly as possible, is the same. And I, as project manager, was very concerned that we should end up with the users having a good user experience.

The transfer of vocational experience within the user group that are experienced cargo brokers, to the developers of the software, was important in two regards: 1. to ensure that the correct design choices were made and 2. to ensure that the design choices were grounded within the user group. This was achieved by co-location. The first central decisions were made by a group of hand-picked, particularly skilled, employees and the project management, already in the pre-study period. This decision-making helped establish a common goal that remained unchanged throughout the process with just minor changes [5].

[SU] -Quite early, just to get a bit away, we went to my cottage in the mountains. There, we spent three days, five guys of us. We discussed from 8-9 o'clock in the morning until 3-4 o'clock in the morning the next day. We were so enthusiastic and managed to keep our concentration and focus. Many complex problems got disentangled up there. Many decisions were made during these discussions. Most of them were mainly focused on the user interface. Who is the user? How will the system be used? What should we try to make it look like?»

The new system turned out to be immediately useful for both expert and novice users; hence we wanted to understand what was done right in the development of this successful system. Being what the project owner describes as a «user's system», the intention with the use of the system was that, if you know how to stow a chemical tanker, you should, intuitively, be able to understand how the system works. If you do not know how to stow a chemical tanker, then you don't have the experiential knowledge of using the system. Experienced personnel should not need much training: the vision for the system was to make it intuitive to use. The following quotes from users confirm that they evaluated the system as simple to use:

“ORCA is easier, in a way, because you get all the data from another system, OTIS.¹ That didn't happen in SuperCargo, where you had to type everything manually.”
“It's is very easy to use. Even without having that much computer knowledge.”

¹ OTIS: Odfjell Tankers Information System. The system has, since January 1. 2015, been replaced by IMOS (Integrated Maritime Operations System, by Veson Nautical).

“Hold the mouse there. Drag there. It runs by itself. If you have done it once, you’ll understand the workings.”

“I don’t feel that I spend that much energy. I maintain the flow.”

“It’s a puzzle, and a great fun puzzle, which makes it really fun to work with.”

This is in line with Dreyfus and Dreyfus, who describe intuition as a method of problem-solving that distinguishes between experts and beginners; acknowledging the indistinct nature of the intuitive method, and state that:

«Experienced intuitive [practitioners] do not attempt to understand familiar problems and opportunities using calculative rationality [...] When things are proceeding normally, experts don’t solve problems and don’t make decisions: they do what normally works» [21].

Experienced users themselves can hardly ever give a rational explanation for their behaviour in a particular way [21].

4.2 Theme 2: Organisational and Procedural Structure of Participation

During the first phase, the users were very much involved. However, another aspect of early involvement of users was discovered; too many participants could cause the development process to become messy and unclear. This was resolved by running user surveys and conduct so-called ‘back-on-track’ workshops, employing a greater group of people than would be practical for regular developing work.

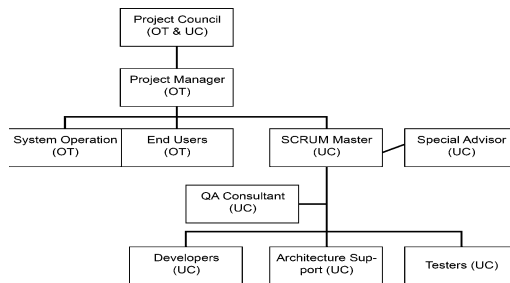


Fig. 1. The project organisation chart.

Figure 1 shows the project’s formal organisation in three main levels, with a project council, consisting of employees from the company (OT) and the developers (UC²) directly connecting with end users and system operation (super-user). We also see the SCRUM Master/Head of the developer team working as an entry point for user participants’ access to the developers. However, since all from the developer team were located in the same room, the communication between the project manager and system operation ([SU]) was informal, continuous and direct.

² Umoe Consulting.

The blend of formal organisational structure and informal knowledge exchange and creation seemed well suited for employing an agile development method. Also, in a development process where such emphasis has been put on the users, we saw that SCRUM as an agile development method was rather well suited for encompassing the informal liaison between developers and users. The feedback that the developers received from the users during a Sprint could take on a rather informal character. This was not the case regarding all information exchange, however. After the iterations, there would be a sprint review, a sprint retrospective, and a sprint planning. Also, the workshops, where the theme would be certain specifics, would, to some extent, have a formal character.

After the system was released and implemented in the organization, and during implementation on the ships, the development process was described as extremely agile. So agile in that fact, that some of the Sprint reviews were considered unnecessary, since everyone, through the continuous presence of the developer team, already knew what was in the release. The developer team got continuous feedback, from the users through the super user, during the process.

By running these surveys and workshops, they engaged a large group of people in a controlled manner. The surveys provided brief clarifications while the workshops would bring out the engagement within the user group. It also provided the users with an outlet where they could present ideas and suggestions for different functions or menus; the way it could possibly look; what they would need on board or at company headquarters, and so on.

During the first phase, the users were very much involved. The early involvement of users engaged many participants and could result in a messy and unclear development process [22], but by running user surveys and conduct back-on-track workshops, employing a greater group of people than would be practical during regular developing work, this problem was addressed.

The formal structure provided an opportunity for the informal structure to evolve, by first getting to know each other in formal settings, like meetings and workshops, and conversely, that the close and informal organization made the formal easier or even superfluous, as e.g. making Sprint reviews unnecessary. The formal influenced the informal and vice versa. A strong focus on mutual learning was salient in the project, knowing that its outcome would be dependent on a shared easy access to the discourse of the field [3, 23] through being *close* together, side by side, meeting each other several times a day by the coffee machine, or for lunch, or simply by being in the same place – mutual learning by walking around – which provided the opportunity to exchange knowledge both formally and informally.

4.3 Theme 3: Proximity for Mutual Learning

Locating developers close to the users was a joint decision. According to the management representative and project leader on the previous system development, user involvement was rather similar to the last time they developed a stowage system. One notable difference between these two projects lies in the actual location of the developers. During the ORCA development period they have been situated on the same floor

(see Fig. 2), which is in contrast to the previous stowage system, Othello, where the developers did not co-locate but were remotely situated.

One of the findings in the study was that communication got a bit less formal. When they encountered a problem, they could pop in and ask whether to try this or that, getting instantaneous, informal, feedback. This development pattern turned out to be so agile with such efficiency that they could, in fact, skip some of the weekly project update meetings.

In an interview with one of the developers about the possible impact of sitting so close to the future users, he underscored the convenience in being able to just walk into the room next door, to the users. With an expressed proximity like this, a developer can receive immediate answers on the questions he might have, without having to wait for any significant amount of time. In an interview with one of the developers, he said:

[dev] -If we had been sitting in a secluded room in the basement, the question is, how many times one had managed to muster enough energy to leave the desk and to go up four floors and down again, with some unfinished business. We would have ended up using the phone anyway, and then there's no point in sitting together, is there?

[researcher] - In what way has the fact that you have been sitting among the users influenced your work?

[dev] -That we have focused on making it easy to use. We were told that there are two different user groups in the company. One of them, the naval personnel, is not so good with computers

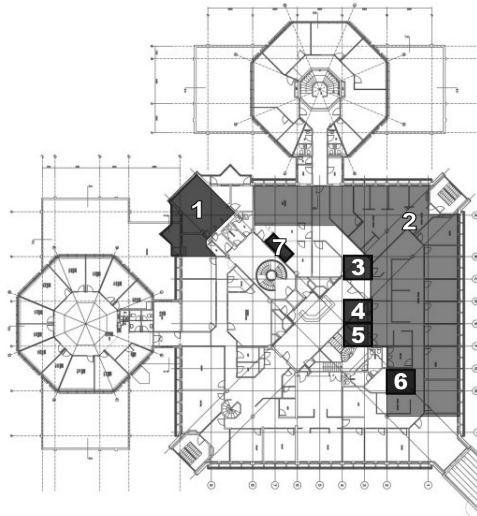


Fig. 2. Proximity: developers and users located on the same floor. The development team (1). The operators (2), project owner, PO (3), management representative, MR (4), super-user, SU (5), expert-user (6), and coffee machine (7).

The required level of proximity in the developer/user relationship proved to be right next door; anything other than the immediate vicinity, i.e. when walking distance exceeding «a few floors» would have led the participants to switch to telephone communication, which would have eliminated the effect of co-location altogether.

The co-location has also made it easier for the *project owner* to work very closely with the developers, and come by often, i.e. several times a day. For the developers this way of co-operating seemed rather unlike previous projects they had been involved in, where they might have to write a list of questions or suggestions and send it to the developers sitting somewhere else and get the answer days later.

[HM]: *-The project owner drops by several times a day. We work very closely together. When we have a question, I don't have to wait for a response. I just talk to him and get an answer immediately."*

[researcher] - In what way do you think it affects your work, that many of the users are working right next door?

[HM]: *-It gets a bit less formal. When they encounter a problem, they can just pop in and ask whether we can try this or that. We get rapid, informal, feedback. Normally we would've held meetings every week, with project updates, but we don't need that here.*

[researcher] - Is all communication informal?

[HM]: *-When we receive feedback from a user during a Sprint, that's pretty informal, but after an iteration, where we have a sprint review, sprint retrospective and sprint planning, I send a report from the sprint review and sprint retrospective to all who were present on the presentation plus copies to other interested parties. We also do workshops, with specific things to be done.*

In this project, tremendous attention has been given to the users. The developers knew nothing about the vocational specifics of the tasks their system had to resolve, which is an interesting aspect of the choice to co-locate. This led to a situation where the developers, during the development process, got to know the users exceptionally well, since they are located in the same place. This gave them the opportunity to identify expertise among the users. This meant to have immediate access to the users' repository of extensive vocational knowledge and experience; their vast knowledge on how to stow a ship but it also meant that they had to relate to strong opinions from the users on how to do their work, and how *not* to do it; on what they want from a task flow supporting software. This sense of being in the immediate vicinity could, potentially, be double-edged. Co-location is all about being available.

According to one of the developers, there would be no point of co-location if they could not be immediately available to each other. However, the developers become equally accessible to the users, and thus vulnerable to being interrupted at work. The company solved this potentially counterproductive aspect of user participation, by channeling user input through the super-user, who became a kind of 'user proxy'. We argue that this level of proximity facilitated the mutual learning necessary for developing a system that supported the work tasks in a manner that was perceived as intuitive by its skilled workers, and that one of the possible explanations of the successful user acceptance and work-flow support was the continuous mutual learning between skilled users, and developers. The users could get a sense of the possible complexity of programming seemingly simple functionality, while the developers got direct and continuous access to the vast knowledge base of the chemical tanker trade, i.e. the importance of tank coating, or the precision of Trim & List.³

³ The boat's position in the water.

4.4 Theme 4: Accountability and Task Distribution

In this project, the task of giving the developers a certain direction were split between the project owner and super-user. In addition to the project owner providing regular requests and input for software functionality, the super user was instrumental in giving the developers a certain direction, determining what vocational specifics that needed be included in the (next) release. In addition, there were three naval officers involved from day one. From them, the developers received some initial comments. They were also involved in the preliminary project, when they ran the initial workshop, discussing user interfaces and which features were adamant to include. The group of naval personnel was actively involved in the pilot phase, but not to the same extent in the implementation phase.

We use the term ‘accountability’ in an ethnomethodological sense [24–26], referring to the fact that by collaborating so closely, both users and developers of the ORCA software became mutually responsible for its clarity and user-friendliness. In an interview, the project manager for the development of the previous system, who is the present management representative, said:

[MR]: -The fact that the developers are sitting here, on this floor, makes the operators aware that the guys in the room next door develop the new stowage planning system. Instead of the rather abstract «somebody out there is working on a program for us», it becomes much more concrete.»

During the development of the ORCA software, the users were aware that the developers were sitting right nextdoor, working on ‘their’ new stowage system. It seems like this awareness made the project more immediate and real, rather than the kind of an abstract notion that “someone out there” is making a new program. Both developers and users expressed that this reciprocal accountability made an impact on the exchange of knowledge that took place during the development process and made it possible for them to not develop a clear requirement specification and rely on communication for clarifying the expectations. The ship management and the personnel from the ships and ‘nautical personnel’ at company headquarters made their expectations clear from the start. The base of user representatives was quite extensive, in its inception it was more extensive in ORCA project than the Othello project. During the Othello process, the company based the development on a requirement specification that was developed by a management consulting firm, not on workshops and dialogue between the involved parties as in the case of ORCA.

This kind of proximity also made the software developing team aware of being regarded as colleagues, and the fact that they could bump into regular users on their way to the coffee machine, or at lunch, contributed to the sense of being accountable for developing choices, and for the contribution in the process of empowering the user with a good tool.

5 Concluding Remarks

In this paper we have reported on a software development project, where much attention was given to the users: the developers moved in and stayed with the users throughout the project. Our aim has been to understand the effects of this level of

proximity on the cooperation between developers and users. We found that the proximity had an impact on the continuous knowledge exchange, the organisational structure and the experience of being accountable in the software development process. Moreover, we think these elements of the software development process influenced the quality of the end-result and, in particular, how it facilitated immediate activity.

Based on the findings, we suggest that the mutual learning necessary for establishing a common understanding of a user-centered software system and its intuitive operation is more likely to succeed if developers and users can build up a common knowledge base over time facilitated by being located in the immediate vicinity of each-other. Our data analysis demonstrates that one impact from this level of proximity in a user-developer relationship is a resulting system that is immediately usable by skilled users and easily learnt by novice users.

References

1. Bratteteig, T., Bødker, K., Dittrich, Y., Mogensen, P., Simonsen, J.: Methods: organizing principles and general guidelines for participatory design projects. In: Simonsen, J., Robertson, T. (eds.) *Routledge International Handbook of Participatory Design*. Routledge, New York (2013)
2. Bansler, J.P., Havn, E.C.: Information systems development with generic systems. In: ECIS, pp. 707–715 (1994)
3. Bjercknes, G., Bratteteig, T.: Florence in wonderland. In: Bjercknes, G., Ehn, P., Kyng, M. (eds.) *Computers and Democracy - a Scandinavian Challenge*. Avebury, Wiltshire (1987)
4. Bjercknes, G., Bratteteig, T.: User Participation and democracy: a discussion of scandinavian research on system development. *Scand. J. Inf. Syst.* *7*(1), 73–98 (1995)
5. Bratteteig, T., Wagner, I.: *Disentangling Participation: Power and Decision-making in Participatory Design*. Springer, Cham (2014)
6. Ehn, P.: Scandinavian design: on participation and skill. In: Schuler, D., Namioka, A. (eds.) *Hillsdale Participatory Design: Principles and Practices*, pp. 41–77. Lawrence Erlbaum Associates, Hillsdale (1993)
7. Greenbaum, J.M., Kyng, M.: *Design at Work: Cooperative Design of Computer Systems*. L. Erlbaum Associates Inc., Hillsdale (1991)
8. Nygaard, K.: The iron and metal project: trade union participation. *Computers dividing man and work - Recent Scandinavian research on planning and computers from a trade union perspective Demos project report 13*, pp. 94–107 (1979)
9. Schuler, D., Namioka, A.: *Participatory Design: Principles and practices*. L. Erlbaum Associates Inc., Hillsdale (1993)
10. Simonsen, J., Robertson, T.: *Routledge International Handbook of Participatory Design*. Routledge, New York (2013)
11. Schön, D.A.: *The Reflective Practitioner: How Professionals Think in Action*. Basic books, New York (1983)
12. Schön, D.A., Wiggins, G.: Kinds of seeing and their functions in designing. *Des. Stud.* *13*(2), 135–156 (1992)
13. Schön, D.A.: Knowing-in-action: the new scholarship requires a new epistemology. *Change: Mag. High. Learn.* *27*(6), 27–34 (1995)

14. Bjerknes, G., Bratteteig, T.: The memoirs of two survivors: or the evaluation of a computer system for cooperative work. In: Proceedings of the 1988 ACM Conference on Computer-Supported Cooperative Work, pp. 167–177. ACM (1988)
15. Bellotti, V., Bly, S.: Walking away from the desktop computer: distributed collaboration and mobility in a product design team. In: Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work, pp. 209–218. ACM (1996)
16. Dourish, P., Bellotti, V.: Awareness and coordination in shared workspaces. In: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work, pp. 107–114. ACM (1992)
17. Heath, C., Svensson, M.S., Hindmarsh, J., Luff, P., Vom Lehn, D.: Configuring awareness. *Comput. Support. Coop. Work (CSCW)* **11**(3–4), 317–347 (2002)
18. Luff, P., Heath, C.: Mobility in collaboration. In: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, pp. 305–314. ACM (1998)
19. Schmidt, K.: The problem with awareness: Introductory remarks on awareness in CSCW. *Comput. Support. Coop. Work (CSCW)* **11**(3–4), 285–298 (2002)
20. Schmidt, K., Bannon, L.: Taking CSCW seriously. *Comput. Support. Coop. Work (CSCW)* **1**(1), 7–40 (1992)
21. Dreyfus, H.L., Dreyfus, S.E.: *Mind Over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*. Free Press, New York (1986)
22. Brooks, F.P.: *The Mythical Man-Month*. Addison-Wesley, Reading (1975)
23. Bødker, S., Ehn, P., Kammersgaard, J., Kyng, M., Sundblad, Y.: A UTOPIAN experience: on design of powerful computer-based tools for skilled graphic workers. In: Bjerknes, G., Ehn, P., Kyng, M. (eds.) *Computers and Democracy: A Scandinavian Challenge*. Avebury, Wiltshire (1987)
24. Garfinkel, H.: *Ethnomethodological Studies of Work*. Routledge, New York (2005)
25. Heath, C., Luff, P.: *Technology in Action*. Cambridge University Press, Cambridge (2000)
26. Suchman, L., Trigg, R., Blomberg, J.: Working artefacts: ethnomethods of the prototype. *Br. J. Sociol.* **53**(2), 163–179 (2002). doi:[10.1080/00071310220133287](https://doi.org/10.1080/00071310220133287)