Erran Carmel, Randall D. Whitaker, and Joey F. George

PD AND JOINT APPLICATION DESIGN: A TRANSATLANTIC COMPARISON

Europeans and Americans have often "talked past" each other about aspects of organizational life. One such aspect is *user involvement* in systems development.

The methodologies of Participatory Design (PD) and Joint Application Design (JAD) have established themselves in Scandinavia and North America as influential thrusts in software development, yet there is virtually no cross-fertilization. PD and JAD are simultaneously similar, complementary, and contradictory. Consequently, a careful analysis and comparison would benefit those who teach and work in information systems development.

Klein and Hirschheim [19] refer to such differences as *Information Systems methodological pluralism.* Pluralism offers the double-edged sword of offering choices, but leaving the *practicing* designer/systems analyst in a state of confusion. Like Klein and Hirschheim, we do not believe there is one correct methodology. We present this discussion and comparison of PD and JAD (which some have classified, erroneously, as "polarized views") in order to help researchers with cross-fertilization and to help practitioners understand their choices.

JAD and PD are well-known methodologies for operationalizing *user involvement* and *user participation.* Both JAD and PD focus on facilitated interactions between users and designers wherein dynamic group techniques are employed for eliciting and refining ideas. They differ in points of user participation, participant identity, participant selection, technical staff and facilitator participation, team orientation, structure, and development speed. They also differ in their goals—JAD is intended to accelerate the design of information systems and promote comprehensive, high-quality results, while PD seeks to accentuate the social context of the workplace and promote workers' control over their work and their lives.

User Involvement

The Information Systems (IS) community subscribes to the notion that the success of a system is proportional to the degree to which the "users" of that system are "involved" in its design and development. This is seen as axiomatic in the IS community and has become one of the six myths that systems developers use as a guide to design [15]. There is little empirical proof of this truism [16], but the notion of user involvement intrigues IS researchers, since it addresses a confluence of complex social factors [2].

An unambiguous definition of "user" is impossible. The North American reader understands "user" to mean any non-IS/nontechnical individual in the organization who is affected by the system—this includes managers. The Scandinavian reader understands "user" to mean any operational worker who is affected by the system—this does *not* include managers. We shall return to this distinction later.

The primary difference between various user involvement methodologies and techniques lies in the degree to which the users participate in (and therefore influence and are satisfied with) the emerging system design. This range of variation is well illustrated by Mumford [23], who delineates a continuum of involvement covering three main styles:

- *Consultative design* leaves decision-making power to the IS staff. Users are simply sources of information with little to no influence or control. This is descriptive of the one-on-one interviewing approaches still in common use, in which users are involved at discrete points in the System Development Life Cycle (SDLC) via sign-off meetings, managerial reviews, steering committees, and user liaisons.

- *Representative design* involves selected user representatives in the actual design formulation and decision making. JAD falls into this category.

- *Consensus design* assigns responsibility for the system development process to the users. The users are continually involved throughout the design process. We place PD in this category (with some qualification: for PD 'compromise' is more descriptive than 'consensus,' but the thrust of this category is appropriate).

This continuum is illustrated in Figure 1. Note that these three approaches are differentiated with respect to the relative engagement and influence of users. As such, there is no strict mapping between these three categories and other well-known design and development methodologies (socio-technical design, soft systems methodology, critical systems thinking, and separately, prototyping).

This article focuses on the practical implementation of the methodologies in question—hence our interest is in how to operationalize the abstract notion of user involvement, that is, which methods and techniques to use to get the users involved. Thus, our focus is on the

methodological implementation factors. Since JAD is not as well known as PD, we shall devote a relatively larger share of our background review to its introduction.

JAD

JAD has become, perhaps, the most common user-involvement methodology in North America for two reasons: First, IS organizations realized that a methodology with a high degree of user involvement would lead to better systems, and they found that solution in JAD; second—by and large—it is perceived as working well. The essence of getting the users involved in the JAD methodology is the structured meeting (the session). The JAD user meeting becomes the event around which all other system development activities revolve. The methodology is *participatory* in that the users are queried more, and hence *involved* more, than users typically were before the advent of JAD. The innovation in JAD, as it has developed today, is that the user meeting is structured, disciplined, and is a foundation of the SDLC. JAD is said to lead to increased quality, reduced costs, and life-cycle time reduction.

JAD originated at IBM in the late 1970s[1] and began receiving industry attention several years later [8, 24]. The interest in JAD has remained exclusively in industry, where, by our estimates, there have now been tens of thousands of meetings labeled JAD (or one of its close cousins that have appeared in the marketplace[2]). JAD is diffused in the community through manuals [14], books [1, 26], and continued exposure in the trade press (e.g., [6]). As JAD has matured it has become part of industry's "new thinking" about systems development methodologies, along with CASE (Computer-Aided Software Engineering), rapid prototyping and others. JAD is one of Yourdon's "eleven silver bullets" [27] and is the fundamental methodological basis for Martin's Rapid Application Development [22].

We suspect the reason for absence of academic interest is that JAD developed and flourished completely outside the academic world. The theoretical basis of JAD is minimal. The JAD meeting methodology has been influenced by the group dynamics discipline and the study of group work and meetings. This makes JAD's contribution one of behavioral underpinnings supporting a technical goal. Indeed, the focus of reported gains, as seen in Table 1, are those of technical progress. The methodologies used for most of the findings in the data are not available and cannot be verified. Some seem to be *post-hoc* estimates by method advocates.

JAD Techniques

There is no one structure or definition for JAD. Over the years JAD has evolved to become a framework for "how to run a meeting" (Note the "typical" JAD room shown in Figure 2). Users attend the meeting to define or design an information system. JAD can be viewed as both a technique and a methodology. It is a technique because it is a structure for conducting a design meeting with user participants. It is a methodology because when introduced into the SDLC, JAD meetings form the core around which all activities revolve.

The JAD methodology emphasizes structure and agenda. This is evident in the JAD literature that reads somewhat like cookbooks [1, 14, 26]. Everything is explained in great detail: "to do" lists are included, as are masters of useful forms. There are four necessary building blocks for a well-run JAD meeting:

1. *Facilitation.* A designated leader (or leaders) manages the meeting. Some JAD practitioners consider the meeting leader to be key to process success, even more so than the act of gathering the users in one place, the essence of JAD.
2. *Agenda setting/structure.* The meeting must have a plan of action.
3. *Documentation.* One or more designated scribes carefully document everything in the meeting. Various lists are rigorously maintained.
4. *Group Dynamics.* Group dynamics techniques are used for inspiring creativity (e.g., brainstorming), resolving disagreements (e.g., airing facts, documenting them as "issues," taking notes), and handling speaking protocols (e.g., enforcing "only one conversation at a time").

The conduct of the JAD meeting changes at different points in the SDLC. JAD meetings early in the SDLC deal with higher-level issues: defining objectives, decomposing the domain into smaller functions, defining boundaries and scope. In these meetings, participants begin to compile a list of assumptions, constraints and open issues; to target specific people and organizations for tasks; and construct timelines. Lists and other text are often maintained on wall charts, so that, by the end of the meeting, the walls are covered with flip-chart paper. Some facilitators encourage the users to roam around the room and fill in the wall charts while the more classical approach allows only the facilitator writing privileges. Once JAD meetings get into the latter phase—the design phase—the users are asked to provide ever-increasing detail. At this stage meetings are often longer in duration, perhaps 3 to 5 days, compared to 1 to 2 days in earlier stages.

The roles that participants play are revealing of the JAD methodology. For example, the roles in the "classic" IBM JAD methodology defined here are fairly strict:

Users. Users are people who will use the system or are affected by it. The 'users', in typical North American IS parlance, include operational users (end users) as well as their managers. The users most knowledgeable about the use of the system should be present at the meetings(s).

[1]More on the history of JAD: JAD was conceived by Chuck Morris and Tony Crawford of IBM in 1977. The JAD approach was loosely derived from another IBM methodology—BSP (Business Systems Planning). The first JAD meetings were held at IBM's Raleigh, N.C. offices in design of a distribution system called Distribution Center Operations Workshop. This project used the same basic JAD concepts still used today: user participant meetings, magnetic visual displays, and careful documentation of the meeting. JAD was adapted by IBM Canada and further refined, later migrating back across the border to the U.S. in the early 1980s.

[2]Generic names for JAD include: Joint Application Development, Joint Application Design, Joint Application Requirements, Joint Requirements Planning, Interactive JAD, Interactive design, Group design, Accelerated design, Team analysis, Facilitated team techniques.

Executive Sponsor. The (user) sponsor defines the overall project purpose and direction, but is usually not present for the entire meeting, if at all.

Facilitator/Meeting Leader. A neutral facilitator leads the meeting, carefully controlling all discussions, guiding and interrupting where necessary. The facilitator (a member of neither the IS team nor the user group) is specifically trained to lead such meetings (many firms provide training specifically for JAD facilitators). The facilitator should have training in group dynamics (or an instinctive flair) in systems development methodologies. The meeting leader is responsible for all activities: the agenda, the discussion, and documentation of the meeting results.

Scribe. The scribe captures the proceedings of the meeting: charts, flows, lists and definitions. The "group memory" of the meeting is the scribe's responsibility.

IS Project Team. The IS staff includes analysts, project managers, database personnel, and technical experts.

The use of creative visual aids is broadly recognized as helpful for users, many of whom are computer novices, in visualizing the software. For example, one JAD consulting firm offers a $400 suitcase of custom-designed, magnetic, color-coded presentation symbols for use on a whiteboard.

The JAD methodology has evolved over the years, as illustrated in Table 2, with perhaps the greatest controversy among JAD practitioners being computer support at JAD meetings. Today, some parts of some JAD meetings are conducted using CASE tools: graphic tools for depicting data-flow diagrams, Entity-Relationship diagrams, state transitions and other diagramming techniques, and screen painters. Another technology gaining adherents in the JAD community is groupware [4]. In contrast to those seeking technological solutions, some JAD practitioners try to minimize the technology that is brought into the JAD meeting room in order to keep the meetings simple and nonthreatening [6].

Table 1. Reported benefits of JAD

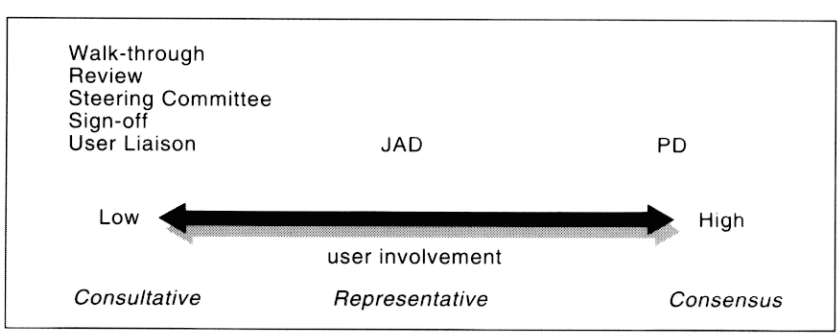
Times Savings:
—Repair effort per defect is only 10% in JAD phase as compared with system test phase [17]
—30–40% in design and 20–30% in implementation
—15% cycle reduction [14]
—8 hrs/Function Point for traditional method vs. 2.5 hrs/Function Point for JAD [8]
—4 to 6 weeks time saved in a project at Western-Southern Life [26]

Cost:
—Cost avoidance of $500,000 in a project at Texas Instruments [26]

Completeness:
—JAD removes 50% of the defects of the requirements phase and 25% in design phase (percentages are not cumulative) [17]
—25% increase in number of Function Points in a project at CNA Insurance [14]

Subjective Evaluation:
—99% of users would do it again at Texas Instruments [14]
—94% of users said they had a better understanding of the system at American Airlines [14]
—100% of the users said the system would be at least "good" at American Airlines [14]

Figure 1. The user involvement spectrum



PD

PD [11, 13]—often termed the "Scandinavian approach" to systems development—advocates a much stronger form of user involvement than that of JAD, one in which workers actively engage in designing the computer systems they will eventually employ. PD, as we know it today, represents a "second generation" of thinking aimed at developing a design methodology based on the principles outlined in the "first generation" of trade union inquiries into the effect of information systems on the workplace [25]. This lineage is apparent in the tenets used by Czyzewski, et al. [7] to characterize PD:

1. Workers should be given better tools instead of having their work or their skills automated.
2. Users are best qualified to " . . . determine how to improve their work and their work life."
3. Users' perceptions and feelings about technology are as important as technical specifications or performance indices.
4. Information technology can only be appropriately addressed within the context of the workplace.

PD is still in its infancy in North America, receiving attention primarily in academic circles (e.g., a now biyearly PD conference). It remains an open question to what degree principles engendered in the Scandinavian socio-political context can be employed in North America. Greenbaum [12] lists four American perceptions that interfere with PD acceptance in North America: 1) PD is too idealistic; 2) PD is biased toward workers; 3) PD lacks method or model; and 4) PD designers need to rely strictly on experience. These last two issues may be addressed without

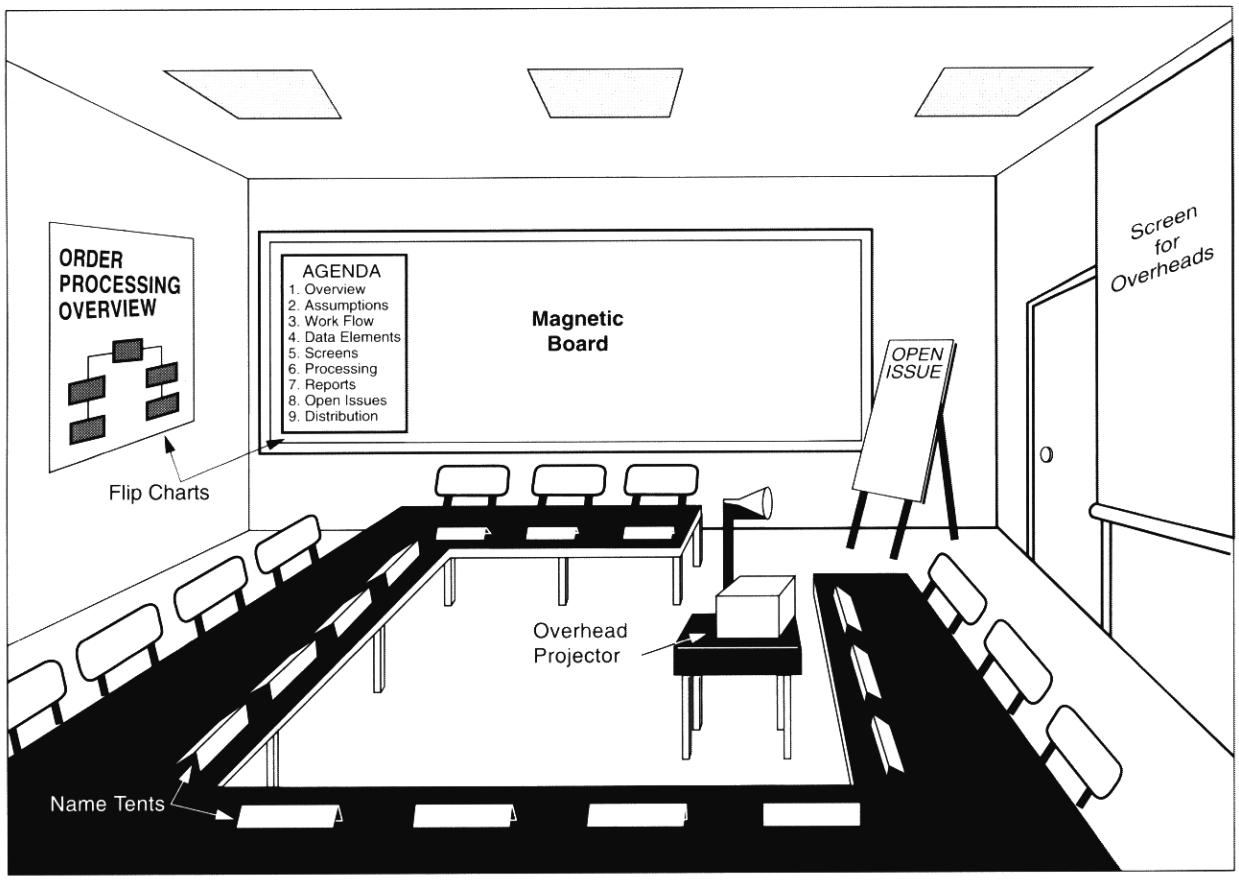


Figure 2. The "typical" JAD room (from [26])

recourse to ideological differences and our review of PD will focus on them.

PD Techniques

There seems to be a reluctance to specify and enforce fixed "techniques" in the PD community. This is presumably tied to PD proponents' opposition to engineering scientism. PD efforts in recent years have led to a repertoire of flexible practices and general guidelines. To the extent that PD projects have been documented to date, there is little evidence that a "standard" set or ordering of practices has been decided.

Two themes govern practical implementation of PD principles [11]. The first theme is *mutual reciprocal learning*, in which users and designers teach one another about (respectively) work practices and technical possibilities through "joint experiences" [5, 20]. The "first generation" PD trade union efforts had aimed, among other things, at familiarizing workers with computers and increasing their skills through training. In mutual learning, this focus is extended to encompass designers' familiarization with their clients' work settings and activities.

The second theme is *design by doing*, in which interactive experimentation, modeling, and testing support "hands-on design" and "learning by doing." PD practitioners have proved very innovative in engaging users in creative design through diverse hands-on practices.

To date, both themes have been pursued largely with "low-tech" tools—those with which the users are already familiar, and which they can easily employ themselves. Blackboards, index cards, and Post-It Notes affixed to a wall are common documentation devices during the modeling phase of a project. Later, prototyping is commonly done with cardboard props and (more recently) HyperCard prototypes. The flexibility inherent in such PD practices challenges the flexibility of current CASE tools, and work is under way toward developing the sorts of software support needed to rapidly incorporate the results of mutual learning exercises [20].

Selected PD techniques are described in the following subsections. These are divided (arbitrarily, we admit) according to: 1) modeling/specification formulation or 2) iterative evaluation of prototypes for the envisioned system. Unless noted otherwise, this sample is drawn from the articles in [13]. (A shorter treatment of PD techniques is found in [18].)

Modeling I: Visualizing the Current Workplace

Historical aspects focuses attention on the historical background to users' work activities to facilitate them in discussing their individual skills, knowledge, and judgement. *Immersion* of designers and facilitators in the target workplace (e.g., via hands-

Table 2. The Generations of JAD

	"Classic" IBM JAD approach	Current JAD directions
Participants in the meetings	Users only	Users and designers
Meeting memory	Scribe/word processing	Design analyst/CASE
Applications	Transaction-oriented	Transaction-oriented, Decision Support Systems and Executive Information Systems
Scope	Applications level only	Applications level, enterprise modeling, functional testing, engineering approach

Table 3. Comparison of JAD and PD

Point of comparison	JAD	PD
Criteria for validation	Quantitative: economic optima, performance indices, time savings	Qualitative: democracy, mutual learning, mutual education, conflict resolution
Background/theory	Group dynamics, software engineering	Labor relations, group learning
Goal	Improved system	Improved workplace
Roots	—Industry —USA, Canada	—Government, unions, academe —Scandinavia
Current practice	Consultancy for profit	Consultancy on principle
Themes	Teamwork, accelerated design, completeness	Democracy of the workplace, social context, industrial democracy, empowerment, humanization
Focal activity	The meeting: —delimited by time —set agenda	Group processes: —satisfaction delimited —agenda negotiable
Techniques' emphasis	Structure	Creativity
Perspective on users	—Both operational workers and managers are considered "users" —User selection based on competence criteria —Users are viewed as one source of knowledge	—"Users" are the operational/users. Managers are grouped separately —User participation *is* mandatory —Users are viewed as primary source of knowledge

on apprenticeship) allows direct experience of the activities the information system must support. For example, in one library automation project designers worked briefly as librarians. *Games* (structured activities and interactions) have long served as means for articulating workplace practices, and they are good devices for winning initial user commitment to the participatory process. One group of PD designers developed a game they called Carpentryopoly.

Modeling II: Visualizing the Possible Workplace

Future workshops are a means for focusing designers and users on visions of a future workplace [10]. This process typically comprises three steps: (1) critique (drawing out current problems); (2) fantasy (creating what-if scenarios); and (3) implementation (determining the resources needed for change through user action and to-do lists). *Metaphor-based design* generates metaphors for the current work situation and projects them into the future as a sort of conceptual prototyping process. *Site visits* allow users to see alternatives in action. For example, users who are graphics designers are taken to visit various printing facilities to see a variety of computer-support systems. This is a simple but powerful way of getting users to understand the range of possibilities (sort of immersion-in-reverse). Other techniques have included storyboarding [21] (known from advertising and cartooning); video productions; brainstorming; improvisational theatre and role-playing; and various types of graphic illustration.

Prototyping: Presentation and Evaluation of Concrete Options

Cooperative prototyping involves the users more than the traditional modes of prototyping, in that they actually work with a prototype and experience it. When a breakdown

occurs, users and designers actively discuss the reason for the breakdown. Prototyping also supports mutual learning by promoting cooperative communication. *Props and mock-ups* of available materials (e.g., cardboard) are frequently used, inspired by the industrial design notion that the artifact is much more tangible than the idea.

Comparison of JAD and PD

It is first worthwhile noting the similarities between PD and JAD. Both methodologies stress a high degree of user involvement as imperative to good design of information systems. Both represent new thinking on the traditional forms of user involvement. Both involve the users in workshops that, to various degrees, encourage creativity and new thinking. Practitioners in both JAD and PD often employ simple, low-level documentation and visualization methods in their workshops. Each acknowledge the central goals of the other—JAD proponents speaking of increased worker empowerment and involvement, PD proponents citing benefits of higher quality systems.

The similarities are not just in the approach but in the contexts. Both PD and JAD face considerable obstacles to implementation. There is a reluctance on the part of both IS professionals and executives to increase user involvement or to experiment with new methods and techniques. Once either PD or JAD is accepted, there are numerous local problems in successful implementation: managerial resistance, user conservatism, lackluster workshops and poor facilitators. Getting user participation is always a test of perseverance; the managers are too busy, the workers are not given approval to spend much time away from their jobs. Finally, as well-intentioned as both methodologies may be, the users themselves can be uncooperative and unmotivated.

Table 3 illustrates some contrasts between the two approaches. The software engineering approach that effectively serves as the basis for development in North America is based on fixed requirements, communication through documentation, and rules of work enforced through methods—functional foci which are deemphasized or dismissed in the PD literature. Conversely, the PD thrusts of mutual learning, joint experiences, and workplace democratization—what might be termed "social" foci—do not receive explicit mention in JAD. With reference to the methodologies' histories, we might say that JAD represents a movement toward more collaborative practices to enhance the viability of given technical goals. In contrast, PD represents a movement toward more technical practices to enhance the viability of given social goals.

JAD represents a movement toward more collaborative practices to enhance the viability of given technical goals. PD represents a movement toward more technical practices to enhance the viability of given social goals.

Several points of departure in the two techniques are discussed in the following paragraphs.

At what point(s) do the users participate?

In theory, both PD and JAD support the entire SDLC. Meetings and workshops with the users can be conducted at all points with great frequency. In practice, however, participation points are not apparent. We have found that JAD meetings are most often used in the requirements stage of the SDLC. The requirements stage is that in which the JAD meeting benefits are considered greatest. However, JAD practitioners stress that multiple JAD meetings need to be run throughout the SDLC—at many points along the timeline. Indeed, JAD meetings also take place at other points along the SDLC: for IS Planning (in which we include Enterprise and Business modeling); for the design stage; to help select software packages; for system test planning. In an iterative approach (e.g., prototyping), the JAD meetings are convened at multiple times as newer versions are reviewed.

PD, like JAD, stresses continuous involvement of the users. In one of its forms, cooperative prototyping, this would indeed be a continuous form of development. However, the PD literature does not position itself compared to the SDLC and some experiences indicate that PD participation points were intermittent (once every week or two in [5]). PD does not lend itself to the "IS Planning" stage, which as typically defined has a strong managerial/executive flavor.

Who are the users and how are they selected?

In theory, PD and JAD support user involvement of all parties affected by the information system. In practice, however, each approach excludes a certain set of affected parties. Generally, the JAD approach has two rules for selecting user representatives (the term "user" does not indicate rank or position, but simply organizational affiliation). First, all areas of relevant expertise should be represented, minimizing potential for an issue being irresolvable owing to insufficient authority or expertise on hand. Second, JAD user participants should be those the organization can least spare from day-to-day operations. As the expression goes: "If you can't afford to lose the person for three days, then that's the person we want." In short, selected users participate by contributing breadth and depth of expertise to the team. In practice, we have found many JAD meetings involve low- and middle-level managers—the employees presumably empowered with decision-making authority over a project in a North American context. When the

JAD facilitator has enough influence the managers and supervisors are augmented by nonmanagerial, operational user representatives.

PD focuses on lower-level, operational users—often excluding management from the process [5]. In the Scandinavian context, empowerment (for project decisions) extends to the operational staff due to codetermination agreements. As such, presence of decision-making authority does not distinguish the two methodologies. PD practitioners presume that operational users are the most qualified authorities on improving their workplaces [7]. Bödker, et al. [3] suggest workshops be composed of people from similar levels to limit any imbalance in power, but sometimes workshops of mixed levels (i.e., with management involved) are unavoidable.

Are the IS technical staff involved?

The "classic" IBM JAD approach suggests the IS technical staff not directly participate in the meeting, so as not to intimidate users or shoot down good ideas. At most, some IS personnel can be allowed to sit in on JAD meetings as silent observers. Many JAD practitioners now emphasize cooperation between IS staff and users as members of an ongoing *team*, involved through JAD meetings in a continuous dialogue [22]. As for PD, the technical presence is limited to designers acting as both facilitators and technical advisors—which leads to the next point of difference.

Facilitators and their roles

While facilitators are of pivotal importance in both PD and JAD, their roles are subtly but significantly different. The JAD facilitator tightly controls the meetings and dictates their pace. PD does not use the term "facilitator" but, rather, the term "designer." The dual role of designers in PD as both facilitators and technical advisors contrasts with JAD, where these functions generally remain distinct and specialized. PD designers typically try to (1) collaborate as peers rather than controllers (e.g., [10]) and (2) promote maximum independent activity by user-participants. Yet, Bödker et al. [3] suggest (in a paradox they acknowledge) that PD workshop rules be strictly enforced, claiming that strict usage of novel communications breaks traditional patterns and allows time for more people to speak and interact.

The team and its interaction

The PD workshop and the JAD meeting both foster a sense of cohesion among the group of workers, users, facilitators, designers, and technical staff; yet the goals of collaboration are differently defined. JAD practitioners emphasize cooperation in the form of a "team." Current North American team-oriented approaches call for self-managed teams in which the individuals in the team are interdependent, interdisciplinary, and hold authority and responsibility for accomplishing a given goal. From the PD (socio-political) perspective, Ehn [9] argues that the "American" concept of team is a poor compromise that takes from workers without giving them anything in return. On the other hand, PD strongly promotes a mutual learning process between members of the group: designers and workers. As the design progresses, both workers and designers are transformed by learning from one another.

Structure

While JAD is a very structured approach, in which manuals and guides are reminiscent of cookbooks, PD does not insist on invariant structure. PD practitioners criticize the rationalistic approach of systems design with its roots in scientific objectivism and the central notion of analysis through decomposition. They specifically avoid presenting any "step-by-step" PD approach, urging designers to improvise and focus on the process aspects of designing. More specifically, PD does not structure the entire time span commonly covered by JAD. The PD techniques are practices defining sessions, not entire project phases. The most highly developed activity plans in PD are those leading to modeling the current workplace and visualizing a future alternative—only a fraction of the total SDLC.

Speed of Development

JAD proponents typically claim that the design and implementation phases are shortened and that maintenance is reduced (Table 1), although we have spoken to some JAD practitioners who concede that JAD increases overall design time. The PD community has not consistently discussed timeframes for their practices, which are typically defined with regard to achieving stepwise goals irrespective of deadlines. Bödker et al. [3] concede the PD approach probably lengthens the design phase.

Conclusions

This article does not attempt to bridge the plentiful ideological differences between PD and JAD. Instead, we conclude by highlighting areas in which each of the two methodologies can benefit from the other (mutual learning in PD parlance). We begin with two areas in the JAD approach that can benefit from PD principles (participant selection and creativity), then discuss one area in which the PD approach can benefit from JAD (structure).

The User Participants. Whether or not one adopts PD's workplace democratization ideal, we have observed numerous North American JAD meetings in which operational employees are overlooked as participants. This results in a meeting room filled with middle managers and supervisors unable to specify details of day-to-day operations (e.g., what 17 fields are needed to fill out form A345). This organizational failure stems in part from an unjustified lack of confidence that "front-line" workers can meaningfully contribute to the design process. If PD benefits could be more clearly demonstrated, this would provide a basis for opening up JAD to increased worker participation.

Creativity. JAD practitioners utilize many creative techniques and paraphernalia in the design process, from magnetic displays that can be moved around a whiteboard, to prototypes of various kinds. However, all too often, these are minimal, and in practice there are many JAD practitioners that utilize the old methods of long documentation, tedious text, and excessive reliance on flow-chart-

ing techniques. It is perhaps difficult for many practitioners to be creative in a JAD workshop, just as many teachers lack the flair to be creative in the classroom. PD practitioners tend to display a flair for creativity that many people in the systems development field simply do not exhibit. Such creativity is not unique to PD, but can be found in diverse sources that emphasize "good design." This suggests JAD's creative potential can be enhanced through facilitator creativity training.

Structure. The JAD approach emphasizes structure, while the PD approach devotes almost no guidelines to structure. This partially stems from the different set of underlying values that drives the two methodologies. Nevertheless, structure has merits; as noted in [13], structure can actually enhance creativity when introduced properly. Introduction of a PD structure summarized in a cookbook format which (to continue the analogy) suggests a dozen ways to cook chicken, would present an important step forward. A PD cookbook would preserve the contextual flexibility that PD practitioners consider important, while at the same time serving to democratize the PD movement by pushing it further into the hands of the average designer/systems analyst in industry.

In closing, we have attempted a comparative examination of two leading user involvement methodologies: PD and JAD. Although there exist contextual differences in their origins and implementation, strong correspondences exist between them. The similarities we have noted suggest a basis for future mutual development, while contrasts suggest points of mutual learning.

Acknowledgments

We thank Jonathan Grudin and Diane Lockwood for their thoughtful comments.

References

1. August, J.H. *Joint Application Design: The Group Session Approach to System Design.* Yourdon Press, Englewood Cliffs, N.J., 1991.
2. Barki, H. and Hartwick, J. Rethinking the concept of user involvement. *Manage. Inf. Syst. Q. 13,* 1 (1989), 52–63.
3. Bödker, S., Greenbaum, J. and Kyng M. Setting the stage for design as action. In *Design at Work: Cooperative Design of Computer Systems,* Greenbaum, J. and Kyng, M., Eds., Lawrence Erlbaum, Hillsdale, N.J., 1991, pp. 139–155.
4. Carmel, E., George, J.F. and Nunamaker, J.F. Supporting joint application development with electronic meeting systems: A field study. *Thirteenth International Conference on Information Systems* (Dec. 1992), pp. 223–232.
5. Clement, A. Griffiths, M. and van

PD: A Personal Statement

Joan Greenbaum
CITY UNIVERSITY OF NEW YORK

Three different perspectives for the need for PD approaches—pragmatic, theoretical and political—represent major discussions that often get in the way of how people talk to one another. We believe this summary could help people identify which arguments they are using, so they can communicate more directly with each other. Our argument, organized around these three perspectives, is rooted in Scandinavian experiences, but is, we believe, adaptable cross-culturally.

• *A pragmatic perspective.* One would obviously argue for getting the job done better. For example, it is generally acknowledged that approximately 60- to 80% of all problems can be traced to poor or inadequate requirement specifications. Obviously, computer systems need to better suit people's working practices. Since those who do the work know how it is done, we need to involve the designers of the systems with day-to-day work experience early in the project, when the basic design choices are made.

For systems developers, PD techniques could mean fostering an environment in which people can express their ideas, for instance by using techniques like a *Future Workshop*—a technique that helps people generate ideas about the future use of technology at their workplace. Applying techniques such as participatory prototyping, offers an up-front way to reduce errors otherwise not found until the final system is put into use.

Participation in design projects offers pragmatic possibilities for both systems developers and management. For systems developers, PD offers an opportunity to build systems that work better. For management, PD offers a way to increase product and service quality.

• *A theoretical perspective.* There are many theoretical arguments supporting the need for PD. Here we develop one from a philosophical perspective. Wittgenstein, for example, argues that "If a lion could speak we wouldn't be able to understand it." Since human beings and lions do not live the same lives we are not able to understand each other.

Since systems developers and people at workplaces do not experience the same things, this limits how well they can understand each other's experiences. One way of getting around this dilemma is to apply a PD approach to prototyping which emphasizes providing people with hands-on experience in a work-like setting.

Turning to the philosophy of Heidegger we may observe that "involved acting—not detached reflection—is our fundamental way of being." For design this implies that the best way for people to relate to a prototype is by use in a, perhaps simulated, work situation.

• *A political perspective.* Political discussions reflect people's beliefs. Coming from a Scandinavian tradition, we believe that in a democracy people have the right to influence their own work place, including the use of computer technology. As systems developers we have the obligation to provide people with the opportunity to influence their own lives. We believe it is our professional responsibility not only to build systems that are cost-effective but that also improve the quality of work life.

Involving people early in project organization, before fundamental design decisions are made, is sound from a political perspective. Applying techniques like Future Workshops and participatory prototyping are ways of designing to meet the needs of the people who are eventually going to use the systems.

From all three of the preceding perspectives, PD is relevant outside Scandinavia. The pragmatic look at things and the theoretical reflections are largely independent of cultural conditions. The political discussions clearly differ among countries, but we could argue that PD supports workplace democracy and that it is time for this argument to be heard.

den Besselaar, P. Participatory design projects: A retrospective look. *Commun. ACM 36,* 6 (June 1993).
6. Crawford, T. People considerations for a more successful JAD. *Am. Program. 4* (Jan. 1991).
7. Czyzewski, P., Johnson, J. and Roberts, E. Introduction: Purpose of PDC-90. In *Proceedings of Conference on Participatory Design,* PDC'90 (Seattle, 1990) Palo Alto, Calif., Computer Professionals for Social Responsibility.
8. *EDP Analyzer.* Developing high quality systems faster. Vol *24,* 6 (June 1986), 1
9. Ehn, P. Speech at *Conference on Participatory Design, PDC'90* (Seattle, Wash., 1990). Author's notes.
10. Ehn, P., Mölleryd, B. and Sjögren, D. Playing in reality: A paradigm case. *Scandinavian J. Inf. Syst. 2* (1990), 101–120.
11. Floyd, C., Mehl, W., Reisen, F., Schmidt, G. and Wolf, G. Out of Scandinavia: Alternative approaches to software design and system development. *Human-Comput. Inter. 4* (1989), 253–350.
12. Greenbaum, J. Panel presentation, *Conference on Participatory Design*—PDC'90, (Seattle, Wash., 1990). Author's notes.
13. Greenbaum, J. and Kyng, M., Eds. *Design at work: Cooperative design of computer Systems.* Lawrence Erlbaum Hillsdale, N.J., 1991.
14. Guide *Joint application design,* Guide International Inc., GPP-147. Chicago, IL, 1986.
15. Hirschheim, R.A. and Newman, M. Symbolism and information systems development: Myth, metaphor and magic. *Inf. Syst. Res. 2,* 1 (Mar. 1991), 29–62.
16. Ives, B. and Olson, M. User involvement and MIS success: A review of research. *Manage. Sci. 30* (1984), 586–603.
17. Jones, C. *Applied Software Measurement.* McGraw Hill, New York, 1991.
18. Kensing, F. and Munk-Madsen, A. Participatory design: Structure in a toolbox. In *Proceedings of the Participatory Design Conference, PDC'92.* (Cambridge, Mass., Nov. 1992).
19. Klein, H.K. and Hirschheim, R.A. Social change and the future of information systems development. In *Critical Issues in Information Systems Research,* R.J. Boland and R.A. Hirschheim Eds. Wiley, New York, 1987.
20. Kyng, M. Designing for cooperation: cooperating in design. *Commun. ACM, 34,* 12 (Dec. 91), 65–73.
21. Madsen, K.H. and Aiken P. Some experiences with cooperative interactive storyboard prototyping. *Commun. ACM* (June 1993).
22. Martin, J. *Rapid Application Development.* MacMillan, New York, 1991.
23. Mumford, E. Participative systems design: A structure and method. *Syst. Object. Solutions 1,* 1 (1981), 5–19.
24. Rush, G. A fast way to define system requirements. *Computerworld* (Oct. 7, 1985), 11–12.
25. Whitaker, R., Essler, U. and Öestberg, O. Participatory business modeling. Luleå Technical University (Sweden), TULEA Report 1991: 31.
26. Wood, J. and Silver, D. *Joint Application Design: How to Design Quality Systems in 40% Less Time.* Wiley, New York, 1989.
27. Yourdon E. *The Decline and Fall of the American Programmer.* Yourdon Press, Englewood Cliffs, N.J., 1992.

CR Categories and Subject Descriptors: C.4 [**Computer Systems Organization**]: Performance of Systems; D.2.2 [**Software**]: Software Engineering—*Tools and Techniques*; K.6.1 [**Management of Computing and Information Systems**]: Project and People Management

General Terms: Design

Additional Key Words and Phrases: Case study, cooperative design, coordination, CSCW, evaluation, user involvement

About the Authors:

ERRAN CARMEL is assistant professor at the Kogod College of Business Administration at The American University in Washington, D.C. Current research interests include development practices in software startups, design methodologies, and group support systems. **Author's Present Address:** Department of Management, Kogod College of Business Administration, The American University, Washington, D.C. 20016; email: ecarmel@american.edu

RANDALL D. WHITAKER is a postdoctoral researcher at Umeå University. Current research interests include enactive cognitive science and interactional models for collaboration. **Author's Present Address:** ADB, Umeå Universitet, 901 87 Umeå Sweden; email: rwhit@cs.umu.se

JOEY F. GEORGE is an assistant professor of management information systems in the MIS department at the University of Arizona. Current research interests include use of information systems in the workplace, including computer-based monitoring, desktop computing, and group support systems. **Author's Present Address:** Department of MIS, College of Business and Public Administration, University of Arizona, Tucson, AZ 85721; email: george@bpa.arizona.edu