Iterative development in the field

by S. L. Greene
L. Jones
P. Matchen
J. C. Thomas

In this paper, we describe Iterative Development in the Field (IDF), a User-Centered Design approach for developing interactive applications. This approach is characterized by repeated evaluation and redesign cycles that are carried out throughout the product life cycle, from initial discovery and gathering of requirements to beyond deployment in the field. The evaluation is based on the use of interactive prototypes and is performed by actual users in the field. We describe how IDF has evolved over the past 13 years through the experience gained from four major projects and offer a set of guidelines for successful IDF that we illustrate with examples from our experience. We discuss limitations in the applicability of IDF and conclude with some comments regarding the future of IDF.

A main goal of User-Centered Design (UCD) is to make systems useful and usable for their end users. The process, as its name implies, places the user in a critical position for both determining system requirements and ensuring they are met.[1–8] As a general approach UCD has been able to demonstrate some very positive results. In a compilation of various studies, Landauer reports gains to be achieved by employing some form of UCD in the system development process.[9] Based on examination of about 15 studies of different systems reported in the literature, he concludes there are significant improvements in work efficiency for the user that can be attributed directly to UCD. The studies cited differ greatly with respect to UCD methods employed, the tasks examined, and the level of completeness of the system being evaluated (e.g., laboratory prototype versus system in use). These factors, coupled with the difficulty of being able to clearly define a "design cycle" within which to assess the efficiency and usability gains of UCD, resulted in great variability in the gains reported across studies. Nonetheless, Landauer concludes that the studies indicate UCD has great potential for improving systems. The benefits include not only improvements in user efficiency, but also others, such as reduced training costs, reduced user errors, reduced maintenance costs, and increased customer satisfaction. Other studies also document the benefits of increased productivity, higher quality results, avoidance of costly, late-cycle changes, and reduced cost of application support and training.[9–13] The field has even matured sufficiently to have standards describing best practices (ISO 13407; ISO TR 18529).[4,5] Cost-benefit analyses can be used to demonstrate projected savings when UCD work is done.[14]

Through our experience developing interactive systems in a variety of domains, we have developed and refined a UCD-based approach that we call Iterative Development in the Field (IDF). It includes four phases: discovery, proof of concept, pilot, and deployment, and each phase involves working with system users and other stakeholders directly in the field.[15] In the *discovery* phase, team programmers ac-

0018-8670/03/$5.00 © 2003 IBM

company usability experts on customer visits to gain a first-hand understanding of the customer environment, problems, and opportunities. This understanding shapes the initial set of system requirements. In the *proof of concept* phase, a small, strategically selected subset of the proposed system's functions and an early version of the user interface are implemented. This "proto-application" is placed in the field and user evaluations are begun (we use the term "proto-application" to emphasize that, unlike a prototype, the code developed is improved over time and becomes the product code). The proto-application code base is furthered developed in the *pilot* phase, when functionality is added, modified, and prioritized based on continuing field-based stakeholder inquiries and evaluations. During the pilot phase, there are increases in the scope, depth, and robustness of the system, the number of field sites where it is installed, and the number of persons using the system. Finally, in *deployment*, the system is hardened, and the team transfers it to the stakeholders responsible for operating and maintaining it.

Over the years, many different UCD techniques and methods have been developed for gathering system requirements and for evaluating designs in order to assess their ability to meet those requirements.[9,11–12,16–19] Although our approach uses many of these well-established UCD techniques, the defining characteristics of IDF can be described as follows.

Design evaluations are carried out ***in the field****, with* ***users of the future system*** *engaged in* ***meaningful behaviors****.* ***Rapid iterations*** *of design and evaluation are ongoing from early design through implementation. The evaluation steps are performed using an interactive* ***proto-application*** *that remains in the field during development and evolves into the final system.*

The rest of the paper is structured as follows. In the next section, we describe IDF in more detail and discuss its potential benefits and risks. In the following section, we describe the four interactive systems we developed using IDF. Then, we provide a set of guidelines for successful application of IDF, and we illustrate these guidelines with examples from our experience. Finally, we discuss limitations in the applicability of IDF and conclude with some comments regarding the future of IDF.

IDF up close

Each of the critical features of IDF builds upon established approaches and best practices in UCD and related fields of social science. We briefly examine these relationships before taking a closer look at the risks and benefits of IDF.

The importance of examining activities and artifacts situated in the context in which they naturally occur (i.e., in the field) is central to work in ethnography, anthropology and sociology,[20] as well as in the design of systems.[21] Participant observation and interviewing are key activities in such research, as they are in IDF. Sperschneider and Bagger highlight the importance of utilizing several different techniques in the field and emphasize that one needs to apply these techniques in a flexible manner in order to be responsive to what is learned.[22] Both the use of multiple techniques and their flexible application in the field are also important characteristics of the IDF approach.

Smith and Dunckley discuss the need for evaluation to be situated, i.e., in the field, and for the data gathered to be used to improve the evolving system.[19] They discuss "developer-user contextual evaluation" (DUCE) sessions, in which users interact with, and think aloud as they use an interactive prototype to carry out task scenarios related to their work practices. General exploratory questions, prepared in advance, are asked at appropriate points of the user interactions, and sessions are videotaped for subsequent analysis. Analysis takes place in what they call "team evidence analysis" sessions, in which the information gathered from DUCE is discussed by a team of developers and worked into potential redesigns.

Although our approach involves similar activities, we emphasize that it is key to employ a host of techniques in the field rather than to attempt to define a single best method. The specific technique used at any point in time is determined based on the user's behavior. For example, we may begin by letting the user explore the system in an unguided way, but if, through his or her interactions or verbalizations, we detect an issue warranting further exploration, we might ask the user to engage in a directed task.

Smith and Dunckley report on the positive effect of providing developers with actual user comments rather than their interpretation. In IDF we take that one step further and provide developers with first-hand observational experience. We find that having the implementors in the field alongside UCD experts is crucial in helping the entire development team gain an understanding of user issues.

A UCD approach that is closely related to IDF's emphasis on work in the field and involvement of users in the design process is Participatory Design (PD). Much of the work in this area has been focused on ways to bring the user onto the design team. Perhaps the most comprehensive treatment of various PD methods is given by Muller, Haslwanter, and Dayton.[23] A major difference between IDF and most PD methods is that IDF focuses on real work in real contexts—even in the earliest stages of design—while many PD methods involve users in various "what-if" types of activities such as theater, games, metaphor, or storytelling. Although such techniques arguably allow people to "think outside the box," they may also introduce various distortions, resulting in a departure from a realistic picture of the context of use. It is well known that people's perceptions and memories of what they do—or what they imagine they would do—is often at odds with their actual behavior. By focusing instead on user behavior during realistic tasks, IDF produces a well-grounded design of the system.

The catalog of participatory design techniques reported in Muller, Haslwanter, and Dayton[23] reveals that most PD methods concentrate on the early phases of software development; e.g., methods that focus on problem identification, requirements, analysis, and high-level design. In contrast, IDF involves stakeholder input and feedback throughout development. Although there are some PD methods that do attempt to address usability throughout the development effort, they consist of a fixed series of steps. IDF, on the other hand, not only advocates flexibility in the design and implementation of the system, but also in the way the method is applied. Rather than follow a predefined sequence of steps, IDF relies on a flexible response to each specific situation.

The evaluation process of IDF is pervasive and central to all activities as it is in the "star" life-cycle model of development.[24] Most UCD practices advocate evaluating design with the use of some kind of prototype.[4,25–27] What varies most is the fidelity of the prototype and how the evaluations are done. Hall discusses the merits of different degrees of fidelity in prototyping and their appropriateness depending on goals and stage of development; he recommends the fidelity be increased as the project matures.[13] Higher-fidelity interactive prototypes are typically more costly in the short run than low-fidelity prototypes, but they have been instrumental in the detection of more usability problems.[26,28]

In IDF, the prototype is more accurately viewed as a "seed" of functionality that is steadily evolved into the final system.[29] This seed takes the form of a high-fidelity prototype, or proto-application. In order to support the evolving nature of this proto-application, IDF uses a tool base that enables low-risk, rapid iterations and incremental development. In this way, one can obtain the advantage of more realistic system evaluation while avoiding the cost of throwaway prototypes. Nielsen talks of horizontal and vertical prototypes—the former covers many features of a system, but lacks depth of functionality, whereas the latter provides more functionality but only for a few selected features.[30] In IDF, we try to build something that initially captures a sufficient number of features concurrently with enough functional depth to begin exploring the feasibility of the design concept; hence the term "proof of concept" as a phase in our development approach.

Another important way in which our approach to prototyping in IDF differs from others is that whenever possible we *leave the proto-application in the field*, even when we are not there to observe its use. Because the proto-application is a functioning system, albeit short of full functionality, we are able to instrument it with monitoring capabilities that allow us to learn about its usage and operating performance, and to respond quickly to any serious system problems.

Benefits of IDF. Aside from the previously mentioned benefits that are associated with the UCD approach, IDF can provide some additional benefits that we describe next.

Focused development. IDF provides a strong orienting focus throughout the development process. Much that is critical to the success of a system can only be detected in the field. Like other UCD methods that gather information from the field in the early stages of design, IDF helps identify many of these issues early in the process. However, IDF goes further in requiring developers to test the effects of design decisions throughout the development cycle, to track how these effects may change over time, and to make appropriate adjustments. It is our experience that this approach offers the most direct, efficient, and accurate means to keep both the usability and usefulness of the application focused on real problems and real stakeholder values.

Deeper and more sustained stakeholder involvement. IDF mandates frequent and repeated episodes of stakeholder participation. Such persistent interaction strengthens stakeholder involvement at many levels and encourages a broad sense of ownership. As work progresses, stakeholders can see their suggestions taking form in the growing system and gain a deeper appreciation for the issues involved in system design and implementation. This in turn motivates them to continue their cooperation with the development team. In addition, by word of mouth the evolving system gains visibility, and the resulting awareness within the organization helps recruit new participants when needed. Enthusiasm expressed among colleagues is often more potent and influential than accolades coming from the management or the developers. Such stakeholder enthusiasm can become an important factor in a smooth system rollout.

The IDF proto-application can reduce costs by also serving as a demonstration or training system.

Early validation of problem formulation. IDF can uncover the need for critical and comprehensive changes in problem formulation while at the same time the empirical data to support such changes are being collected. We have found that working with stakeholders in the field during the discovery stage can result in significant changes in the formulation of the original problem statement. This in turn may change what is actually built. Involving stakeholders in the project and gathering data from actual use lead to the early validation of problem formulation.

Reduced costs for marketing and support. IDF can support and coordinate marketing activities. The IDF proto-application, which serves as the foundation for system development, also plays an important role as a demonstration system for the marketing teams and other stakeholders.[27,31] A single evolving system that fills both roles ensures a degree of efficiency and resource conservation. There is no danger that the development of the actual and demonstration systems will diverge, and hence no redundant effort is needed to support separate systems.

Risks of IDF. Some of the risks often associated with the UCD approach include increased cost, difficulties in coping with time and scheduling constraints, and unavailability of needed UCD skills.[12,19] IDF may add to or exacerbate some of these risks. For instance, UCD project managers typically grapple with creating schedules that accommodate iteration and stakeholder feedback. With IDF, this challenge continues through the lifetime of the project. Other risks associated with IDF are described in the following subsections.

Managing relationships with a large numbers of stakeholders. Because IDF stresses interaction with a large number of stakeholder groups over a protracted period of time, the availability or relevance of specific stakeholders may change. There are many reasons that the availability of stakeholders may fluctuate. Some of these are non-system issues, growing instead from the everyday complexities of working with organizations and individuals. In other cases, the relevance of particular stakeholders may change with the phase the project is in. Nonetheless, it is wise to keep all parties informed and aware of the project during periods of non-involvement, even though this adds to the communication workload.

Unrealistic expectations. The presence of an early working system in the field can inadvertently foster unrealistic expectations among stakeholders. Most developers are aware that no matter how incomplete an interactive computer-based prototype actually is, it presents a remarkably compelling illusion of "finish," even to experienced audiences. This illusion can cause the audience to underestimate (or overestimate) what is yet to be done and the resources required for it.

Impact on the business. Relying on an evolving system to complete real work—of paramount value in collecting accurate data—can present risks that fall predominantly on the business-side stakeholders. For instance, users may find it difficult to balance their new roles as testers and co-designers with maintaining an acceptable level of business performance. In addition, there is the threat that system problems could harm business productivity.

Difficulty of drawing conclusions from field data. For the developers themselves, there is the risk of drawing incorrect conclusions from observations in com-

plex field situations. The very richness and realistic nature of field settings reduces the ability to control and manipulate variables. Typically, in the field, it is not possible to simplify the test situation to examine the causal impact of individual variables as one does in the laboratory; hence, inferences about causal relations are made more difficult.

How to manage risks and reap the benefits. There is clear evidence that User-Centered Design techniques, when practiced correctly, do indeed improve the usability of resulting systems. Why then is UCD not more ubiquitous? Some answers are obvious, e.g., limited budgets, time constraints, and lack of skills. Others, such as the risks mentioned above, are perhaps more subtle, but they also factor into why these practices are not more widespread.

We submit that IDF is a form of User-Centered Design that can minimize several of the forces working against the general practice of UCD. By coming out as soon as possible with a high-fidelity prototype that *remains* in the field and is updated frequently, one can create a "living laboratory." Keeping this proto-application in the field, even without usability experts present, provides many advantages. It enables the gathering of information from stakeholders who use the system on their own. Stakeholders may also provide the development team with feedback based on their observations of others using the installed proto-application. In addition to providing developers with useful feedback, this also strengthens stakeholder buy-in and provides an easy mechanism to maintain awareness of the evolving system. Overhead costs related to transportation, shipping, and system installation for testing are reduced. Connecting the proto-application to the development laboratory makes it possible to monitor the usage and respond to potential problems. Updates, such as bug fixes or design changes, can be installed remotely when usage is low (e.g., overnight) and the system made ready for re-evaluation.

Four systems

Below we describe four systems developed over the last decade using the IDF approach.[32]

EXPO'92 Guest Services System. EXPO'92 Guest Services System (or EXPO, for short) was an information and services system for the 1992 Universal Exposition, a world's fair, which took place in Seville, Spain.[33] Starting in 1989, IBM Spain and IBM Research collaborated on designing this touch-screen system for visitors to the planned 1992 fair. The goal was to showcase IBM technology by creating an engaging and informative system for a broad, international audience. EXPO'92 proved to be a unique living laboratory for the IDF approach, and the result was an award-winning multimedia public access system.[34] For the duration of the fair (April 20th to October 12th, 1992), over 15 million people used the guest services system. Fiber-optic cables linked 33 kiosks to 20 pavilion restaurants and to a kiosk control center. Each kiosk consisted of seven touch-screen guest stations, 231 stations in all. The system functions included an interactive "electronic walk" to explore country pavilions and navigate through the EXPO'92 site, voice, text, and image messaging, finger painting, the daily news, and restaurant reservations. While some functions, such as the electronic walk and reading the daily news, could be accessed without logging on, more personalized functions—such as messaging, painting, and restaurant reservations—required the visitors to log on with their EXPO entrance tickets so that their data could be stored and retrieved. The logons were essentially anonymous in that no personal identifying information was requested or collected from the visitor. In a typical week, we recorded 231 000 visitors who "logged on," as well as the creation of 42 000 multimedia messages, 126 000 video images, and over 200 000 finger paintings.

Six researchers and an equal number of IBM Spain technical staff comprised the core of the project team, and they received a great deal of assistance from others at IBM, EXPO'92, and third-party companies. The scope of the project included all phases of planning and design (hardware and software, kiosk enclosures, signage, operation processes, etc.), programming and engineering, content creation, operation, maintenance, and management. We developed the system in the field over a period of three years, including the six months of the fair itself. During the fair, system content (news, lottery results, etc.) was refreshed daily, functionality was added to support requests from EXPO'92 (e.g., Pavilion of the Week, EXPO'92 lottery), and portions of the user interface were significantly revised a number of times.

The EXPO'92 Guest Services System demonstrated the enabling power of user interface technology in walk-up-and-use contexts. It also laid the intellectual, methodological, and technical foundation for a number of projects at IBM Research, including the other three projects discussed in this paper.

Touch Illinois. During a time of high unemployment, starting in 1993 and through 1994, the Illinois Department of Employment Security (IDES) engaged IBM to help improve the performance of their core responsibility, returning unemployed citizens to the work force. A main task performed in the course of this service was establishing a person's eligibility for unemployment payments. This task consumed the major portion of the staff's time. In order to determine eligibility, IDES staff was required to transfer information from filled-in paper forms into a mainframe legacy system.

To address this problem, we developed Touch Illinois; a touch-screen government benefits system that enabled citizens to enter unemployment claims directly into the state's computer systems. By eliminating the duplicate data-entry step, the new system allowed the staff to spend more time helping the unemployed find jobs. As a team of seven IBM researchers and three IBM sales and services representatives, we installed the application on 25 sit-down kiosks for the Arlington Heights office. These systems were connected via a LAN to twelve staff stations and via a server to legacy systems.

We began the iterative development by installing the system at the Illinois State Fair and testing the user interface, and then following up with weekly updates and evaluations in the Arlington Heights office. Throughout the project, we always had one or more team members working on site. The evolving systems were always in operation in the field in order to obtain continuous stakeholder feedback. IDES office staff participated in the project on a daily basis by evaluating the system and by offering many useful ideas, while the "users" were citizens filing for unemployment benefits and searching for new jobs.

Pilot measurements indicated that citizens who used the touch-screen systems filled out their claim forms 33% faster on average than those using paper forms (an average of 45 minutes reduced to 30 minutes). When the citizen completed entering his or her data, the kiosk alerted the staff stations. This enabled staff to work with citizens as they became ready for the interview phase, supplanting the first-come-first-served "deli-ticket" method previously used. The average time the staff spent with a citizen was reduced from 27 minutes to 10 minutes.

Following the 1994 elections, the State of Illinois changed administrations and funding for Touch Illinois was discontinued. The project, however, was the foundation for a similar system, Touch Colorado, and also served in the development of a set of "assisted entry" interface techniques that were later exploited in the AutoLoan eXchange (described next).

AutoLoan eXchange. AutoLoan eXchange (or ALX, for short), an Internet-based retail loan system, was developed by the IBM Research Division in cooperation with a major financial institution between 1995 and 1997.[35] ALX, which provided on-the-spot retail loan approvals and contracts for customers at automotive dealerships, was connected to a variety of financial institutions and credit bureaus. Application data were sent electronically to dealer-selected credit sources; resulting loan decisions were returned in minutes. The initial ALX pilot involved six dealers connected to a single financial institution over the Internet; an extended field study that lasted a year included 80 dealers, four major financial institutions, and all three major credit-reporting bureaus.

ALX was being considered as an IBM service offering, and the team quickly grew from a group of three researchers and two account (marketing) representatives to a total of over 50 people, including researchers, programmers, a technical writer, testers, account representatives, executives, support personnel, and administrators. As the ALX project grew and became a significant financial undertaking for the stakeholders, it raised the stakes in the challenge of making IDF work. Updates and evaluations were carried out in the field every month during the proof of concept and pilot phases, whereas during the extended field tests, the proto-applications at dealers and financial institutions were updated approximately every three months. The longer interval between updates was necessary for formal testing, documentation updates, training of help desk personnel, marketing considerations, and so forth. These activities roughly followed the Spiral Development software engineering method as described by Boehm.[36]

In the pilot phase, 70 percent of loan applications got responses in less than three minutes (compare this with the then-prevalent turn-around times of 45 minutes to several days, even for people with good credit ratings). Although ALX was a technical success that gained the support of many of the participating dealers, a business decision was made not to pursue ALX as a service offering. Nonetheless, ALX enabled the participants to realistically investigate a number of technical and business opportunities in

the emerging Internet environment and the consumers' responses to the new marketing tools.

Explore Modern Art. In 1999, the Museum of Modern Art (MoMA) in New York City engaged the IBM Research Division to explore new ways to increase museum visitors' access to its collections. Like most museums, MoMA displays at any one time only a small fraction of their numerous collections. Between 1999 and 2002, the IBM team, in collaboration with the curators of the Photography Department, designed and implemented Explore Modern Art, a public access system to help visitors gain an understanding of art and art history while being exposed to the museum's numerous collections. Rather than presenting written or audio lectures, the system represented a tool for discretionary learning, inviting visitors to discover concepts in art and art history themselves through comparing and contrasting, magnifying, and classifying works of art along various conceptual dimensions. Ultimately, the system included works from the Photography, Painting and Sculpture, Prints and Illustrated Books, Drawings, and Architecture and Design departments.[37]

A stalwart executive to champion the cause of the IDF project is essential for its success.

Four flat-panel touch-screen systems, connected to a remotely monitored server, were first installed on MoMA's site in Manhattan; when the Manhattan site closed for a major expansion, the systems were moved to the museum's temporary site in the borough of Queens. Explore Modern Art was designed and implemented by four IBM researchers and three people from MoMA, although the size and makeup of the team evolved over the duration of the project. The MoMA staff regularly participated in observing the users during evaluations of the proto-application. Iterative development in the field was conducted in all design and implementation stages and on all aspects of the system. New versions of the proto-application were installed (often remotely, from the IBM laboratory) and tested as often as once per month. As with the other systems described here, the evolving proto-application remained in use in the field throughout development.

Explore Modern Art generated enthusiastic responses from visitors as documented in the visitor-comment books placed next to the user stations, and by observations and interviews conducted on the site. Informal evaluations provided evidence that visitors learned complex concepts by interacting with the system. Think-aloud protocols revealed that users engaged in hypothesis formulation, testing, and refinement as they went through the classification exercises. The system appealed to visitors of all ages, nationalities, and art backgrounds. The system was adopted by MoMA's education department for further operation and development. At IBM, the "learn-while-doing" approaches developed with MoMA are being refined and further developed in other IBM projects currently underway.

Guidelines for successful IDF

We draw upon our experiences developing the four interactive systems briefly described above to offer lessons illustrating how to carry out some of the critical activities of IDF in ways that maximize the benefits and minimize the risks of IDF. The lessons, formulated as guidelines, are for the most part a subset of recommended practices of UCD—a subset representing the defining characteristics of IDF. By illustrating the ways in which these practices were applied in the development of the four systems, we attempt to elucidate how one may successfully engage in IDF.

These guidelines are highly synergistic, and the efficacy of any one of them is tied in a number of ways to the others. Each of them independently—and perhaps more important, through their interactions with each other—contribute to many, if not all, of the benefits previously described.

Find a strong champion. Finding a stalwart executive champion for the project is essential for the success of the application development effort and vital for the team that uses IDF. The champion must open (and keep open) the cross-departmental channels through which developers gain access to various stakeholders and thus obtain, in a timely fashion, the data that drive the IDF process. The champion's reach must extend throughout the corporation and beyond, to its customers, suppliers, and business partners. This latter requirement suggests seeking the champion in the line of business rather than in the information technology (IT) department.

In the MoMA project, the participation of the chief curator of the Photography Department was principal to the project's success. His ability to promote the project's goals and the IDF methods unlocked for us doors that would otherwise have remained shut. Most important, he agreed that the optimal interface and functionality could not be developed without direct and significant involvement of museum visitors. Although among the museum staff many were not comfortable with an unfinished, imperfect system out on the floor of the museum, the champion was able to help us secure prominent locations to place and test the evolving system. Following the gathering of data from the field, he was instrumental in disseminating and explaining the significance of the data to his peers and others, thus maintaining awareness and support across the museum.

The champion also focused on cultivating support from a group of important, and initially skeptical, stakeholders. He took advantage of the availability of the evolving system on the museum site and encouraged curators from all other departments to contribute to the system's content as well as to the direction of its development. By personally establishing a precedent, he made it easier to engage other stakeholders in usability testing. These types of champion-led activities contributed to increasing stakeholder buy-in and involvement at many levels.

When, during the discovery phase, we began to understand the motivating goals of both the individuals and the institution, we were led to rethink the museum's original request for a visitor-accessible database. Reasoning that the museum at large was strongly concerned with educating in addition to exhibiting, we returned with a proposal for a system that would explore the boundaries of self-directed learning. Our champion provided the cultural and political guidance and behind-the-scenes support that assured a positive reception by MoMA.

The MoMA project champion's activities exemplify how to lessen potential risks related to managing interactions with stakeholders. He provided invaluable guidance in dealing with large numbers of stakeholders by helping us make judicious use of everyone's time. He helped us to not only identify the stakeholder groups and their representatives, but also to understand at what point in the process their involvement would be most valuable. He helped fashion group-oriented approaches to engage and sustain the interest of large numbers of stakeholders (e.g., special meetings and presentations, e-mail updates, blanket invitations to events, and dissemination of schedules for team meetings and usability testing activities). He helped manage expectations by explaining the nature of UCD and IDF and recounting the history of the system's growth.

Identify stakeholders and forge relationships early; scout for new ones as sands shift. In order for IDF to deliver on its promise of improved focus, the team must have access to the most relevant stakeholders at each point of the development process. At the start of a project, it may not be clear which stakeholders will prove to be the most important for carrying out the iterative development process. Yet, if the appropriate stakeholder groups and individuals are not identified early, development can slow or even stop altogether as the needed stakeholders are sought out and introduced to the project. To avoid this problem, we cast our net widely; whenever there is doubt, we err on the side of inclusiveness. In this way, we increase the chance that we will have an existing relationship with the right stakeholder when needed, and we keep open the possibility of encountering unanticipated, valuable points of view. As the project progresses through the subsequent phases, the team continues to be alert to the discovery of new stakeholder groups and representatives.

Maintaining a broad stakeholder base and network of relationships also provides the development team the flexibility to overcome obstacles in the accessibility of specific stakeholders. Workloads are one factor influencing patterns of stakeholder availability. It may happen that a stakeholder of only peripheral relevance in the early stages of the project assumes greater significance later. Stakeholders may leave their positions for other jobs or duties, while in the extreme, entire groups or departments might be eliminated, redeployed, or merged. Continual interaction with stakeholders provides the development team with early awareness of imminent changes that may affect the project, thus enabling the team to be proactive.

During the discovery phase in the MoMA project, we spent approximately one month on location, visiting the curatorial staff as well as the staff of other departments. We roamed the galleries, observing and interacting with museum employees (e.g., guards) and visitors. Activities such as these laid the foundation for a robust set of stakeholder relationships. We maintained these contacts as the project pro-

gressed and added new ones, continually evaluating when and how to optimally draw stakeholders into the IDF process.

As it turned out, MoMA was embarking upon a massive construction project, which began to affect our project very soon after we first placed the proof-of-concept system in the field. Had we not made the early contacts with the Exhibitions Department—the group responsible for putting up and tearing down walls, as well as creating display areas for technology exhibits—we would not have been able to secure coveted spots in the midst of high visitor traffic, as entire wings of the museum disappeared and galleries were rearranged and rebuilt. In addition, our relationship with the IT department enabled us to get the appropriate wiring, phone lines, and server closets as our testing locations shifted.

Stakeholders who are neglected or left out of the project can have a significant negative impact on an otherwise well-planned system development project. Touch Illinois, a project that gravely suffered from *not* identifying and involving all stakeholders, clearly demonstrates this danger. Although the project had a powerful champion in the person of the state commissioner and the team did an effective job of bringing the staff and management of the pilot office into the project, the development team did not recognize the second line of management on the commissioner's staff as relevant stakeholders. Well into the project, the commissioner began a run for state office and resigned in order to conduct her campaign. One of her staff was appointed acting commissioner and shortly thereafter made the decision not to continue support for Touch Illinois. Having brought the commissioner's staff into the project sooner might not have made a difference in the fate of the project. However, it was evident the new administration did not share the same sense of ownership in the project. In this case, we failed to fully carry out the process of creating a broad stakeholder base, and as a result, the project suffered.

What follow are some of the strategies we developed for identifying all the relevant stakeholders and hence minimizing the negative impacts that could result from overlooking stakeholders.[38]

Explore the formal organization. In this strategy, we identify and approach one or more people who know the entire formal organization. Alternately, using a tool that displays the entire formal organization, we explore organizational entities whose functions may be relevant to our project. Thus, one of the first artifacts we obtain is the organization chart. Not only does this help to identify potential stakeholders (both in their roles in the organization and as individuals), it also reveals much about the structure of the organization and the relationships that exist within. The MoMA organization chart, for example, revealed that the curatorial departments operated in a highly autonomous manner, connecting in the hierarchy only at the director level.

Explore the informal organization by networking. Here, we begin by approaching people we know and asking them to identify potential stakeholders or to identify colleagues who may be able to help do so. This mode of inquiry can reveal the informal relationships that are not delineated on any organization chart, such as who has good working relationships with whom, or where the conflicting agendas may lie. For instance, we used MoMA's organization chart as a jumping-off point for an informal networking approach. Informal networking revealed that the support departments (education, publishing, registrar, customer services) operated as integration points, forming the threads of a virtual matrix that connected the curatorial departments horizontally.

Draw on similar past experience. We find it worthwhile to exchange stories of past projects in which some stakeholders were overlooked, and to consider how we could have done a better job of identifying those stakeholders. We may reflect on applying that knowledge to the current circumstances, with the goal of recognizing analogous situations. This exercise can also be carried out in discussions with stakeholders—in this case, we may tell stories about our experiences. For example, we related the story of our failure to identify important stakeholders in the Touch Illinois project to our MoMA champion, hence underlining the importance of our having access to, and buy-in from, all groups of stakeholders.

Stage an event in the town square. In order to attract and involve all relevant stakeholders, we may place a working prototype in some public place visited by stakeholders. We may also stage a real or virtual event, such as a party, a fair, or a Web site. The initial MoMA installation is an example of this strategy. The proof of concept system was placed in Café/ETC, a multi-use location that everyone in the museum—visitors and staff alike—had reason to visit. Similarly, we took advantage of the traffic at pre-opening events at EXPO'92 (for EXPO), and at the Illinois State Fair (for Touch Illinois, as shown in Figure 1).

Figure 1 Visitors to the Illinois State Fair using the Touch Illinois system to obtain an estimate of their Social Security benefits

While none of the methods described here can ensure that all relevant stakeholders will be discovered, we believe that, taken together, such methods increase the chances of identifying the broadest range of stakeholders.

Manage expectations. IDF relies heavily on stakeholders interacting with a high-fidelity proto-application. When placed in the field, such a system may create unwarranted expectations. First, people may overestimate the maturity of the system. This can result in stakeholder frustration and loss of confidence in the design team. Second, even if the stakeholders are satisfied with the prototype, due to the speed with which the user interface "shell" has been developed, they may significantly underestimate the time required to evolve a fully functioning, robust, and well-tested system.[27] In order to prevent or correct such misunderstandings, it is important to set the appropriate context for the particular stakeholder audience. We can do this explicitly, of course, through discussions and presentations. We can also do it implicitly, by designing the prototype in a way that communicates its incompleteness without hurting its usefulness. In the MoMA project we used both approaches.

We held monthly design and planning meetings with MoMA's curatorial staff. We discussed the floor prototype and the test results, we previewed functions and changes to be installed in the next monthly iteration, and we discussed ideas for future iterations. It was common for new curators and others to "drop into" these meetings, and we soon realized that much time was spent educating new stakeholders on the nature and implications of the IDF method. We found that we could make these discussions shorter and more effective by providing a "man behind the curtain" demonstration. When a suggestion was made for a change to the prototype, one of our programmers would make the change on the spot. Not only did this communicate the power and nature of IDF directly and succinctly, but it also helped everyone understand which types of changes were easy and which were more involved. (This also highlights the advantage of having implementors, not just UCD experts, out in the field.)

We took a somewhat different approach to managing expectations in the case of EXPO. Very early in the project, we needed to demonstrate a version of the "electronic walk" function. Because one of the goals of the demonstration was to inspire the EXPO'92 participants to create content for the system consisting of their own interactive stories, we did not want them to perceive the demonstration content too literally and hence limit their expression (e.g., creativity, scope of ideas) or interest. To emphasize the

evolving nature of the function, we used rough hand-drawn illustrations and photographs of architectural models as well as fabricated stories about an "Anyland" pavilion in the instructional material we prepared for the participants. We also included these in our evolving proto-application, which we showed to visiting country representatives more than a year in advance of the opening of EXPO'92. This stand-in content demonstrated the potential of the interactive storytelling format, while highlighting work still needed to complete the system.

Craft realistic field conditions for proto-application development. When planning and implementing the conditions in which we develop the proto-application, our goal is to approach as closely as possible the actual environment in which the system will be deployed. In the MoMA project, for example, this meant getting the system into the galleries. Similarly, it was the dealer's showroom for ALX, the government office for Touch Illinois, and the site of EXPO'92 for EXPO.

It may be necessary—especially in the earliest stages of development—to come up with a "next best" solution. For instance, the actual field site may not yet exist, or it may be unavailable for technical or other reasons. In such cases, it is important to craft conditions to be as realistic as possible in terms of the target audience and the significant aspects of the working environment. Every project presents unique challenges and opportunities. We cannot hope to address such variety here; instead, we share strategies we have employed and offer some guidelines to consider when field situations are designed.

Understand and respond creatively to stakeholder issues. Stakeholders may hold a variety of concerns about placing an evolving system on an actual business site. To the developers, some of these concerns may seem to be a matter of misperception or misapprehension, while others are unambiguously concrete or pragmatic. Nonetheless, it is essential to respond with consideration and creativity in each case.

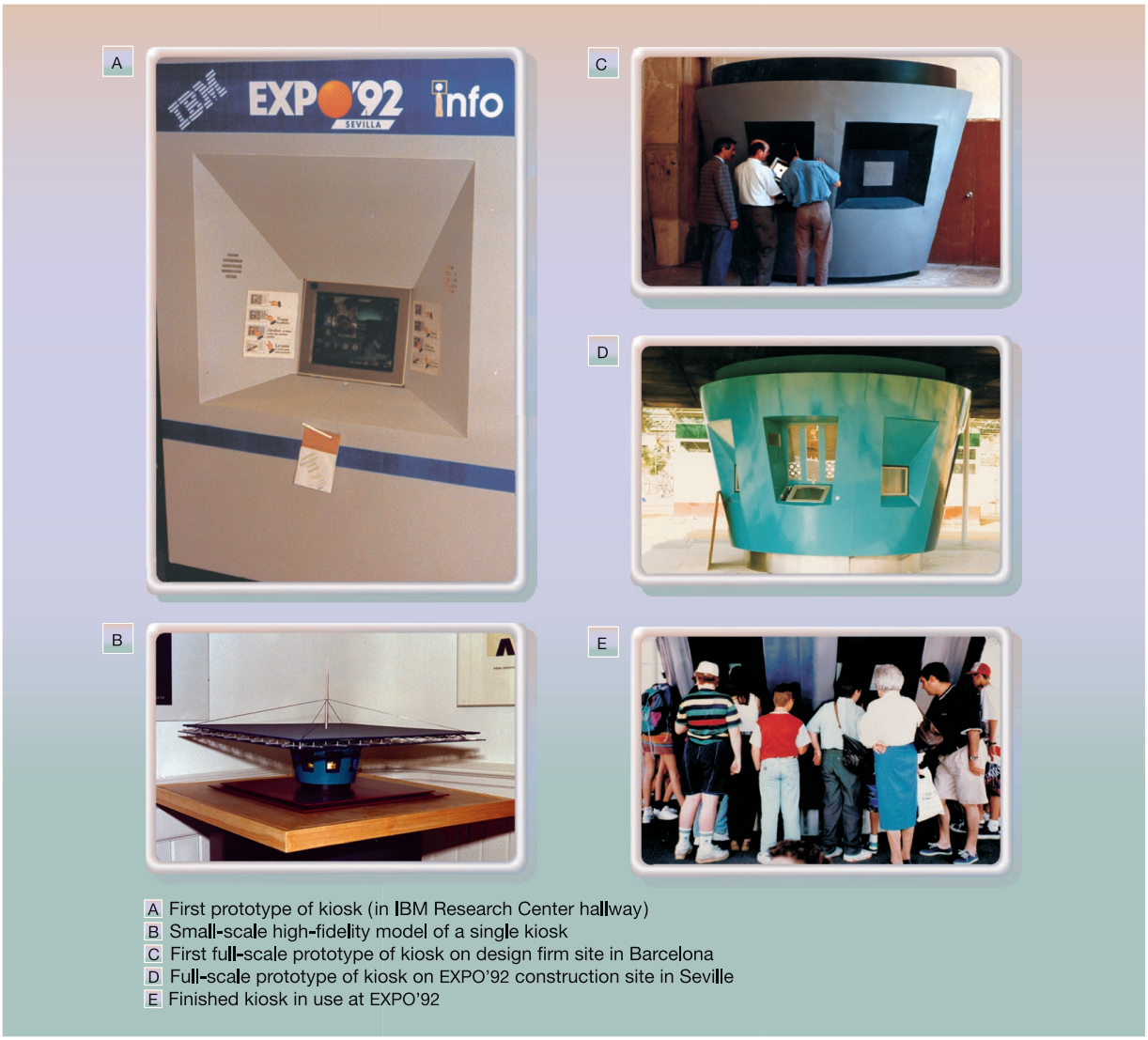
Recall the stakeholders at MoMA who held strong concerns about the implications of placing a "crude" proto-application in the museum. Their concerns had to do with issues such as aesthetics, institutional reputation, and historical statement. To address those issues, we appealed to a pre-existing concept in the discourse of art, labeling the system as a "work in progress," thus creating a context for field-testing with which they were comfortable.

More commonly, stakeholders share a fear that field-testing will disrupt business, or interfere with business relationships. The field site, testing methods, and regime can and should be designed to avoid or reduce these risks to acceptable levels. It is especially productive to obtain the assistance of the stakeholders in developing options for resolving such issues. In the ALX project, for instance, we did not schedule interactions with auto dealers during the last week of a month because that was when they typically worked overtime to meet sales quotas. In addition, before we installed a proto-application system at a dealer's site, we established a series of customized contingency and disaster-recovery plans. One specific plan entailed the creation of software that managed the Internet connections of the ALX systems. This assured that no transactions were lost or delayed due to problems with connectivity. We also provided an easy means to revert to the original paper-based process at any point in the transaction. This minimized some very real risks that otherwise might have inhibited dealer participation.

Start small. It can be very helpful to start field iterations "small" (limited in scope) and grow them over time. This can get the team into the field rapidly, while containing problems that may arise. In Touch Illinois, for instance, the first field iterations were conducted at the Illinois State Fair, while subsequent iterations followed in the Arlington Heights office. Similarly, in ALX, field iterations began with a single pilot dealer and grew by one dealer at a time as new features became available for testing. In this way, working with seven local dealers, we brought the system to a state of sufficient readiness to begin the larger, national field test. It can also be useful to arrange a number of simultaneously active field sites. This should be considered when there are different field or market conditions that need to be covered, when the development team is geographically dispersed, or when there is reason to believe that a particular site may become unavailable at some time in the future. For instance, the dealers for the ALX field test were chosen in part for their geographical diversity.

Different venues for different purposes. The functionality and user interface of the EXPO system were iteratively designed, beginning two years before the opening of EXPO'92. As the site of the fair did not exist at the time, we began testing the evolving system in a hallway of the IBM Thomas J. Watson Research Center in Hawthorne, New York (Figure 2A). As soon as we could, we installed it in the visitor cen-

Figure 2 Views of the evolving design for the kiosk at EXPO‘92 in Seville, Spain



A First prototype of kiosk (in IBM Research Center hallway)
B Small-scale high-fidelity model of a single kiosk
C First full-scale prototype of kiosk on design firm site in Barcelona
D Full-scale prototype of kiosk on EXPO’92 construction site in Seville
E Finished kiosk in use at EXPO’92

ter on the periphery of the EXPO’92 construction site in Seville, Spain (Figure 2D), then in a prototype kiosk structure on the site proper. We continued testing and improving the system while it was in everyday use during the fair.

We learned critical things from the users in the field that would have been missed had we tested the system solely in the laboratory. As an example, we wanted to explore what was at the time, a new touch-screen interaction technique, “touch, slide, lift.” First, users would touch a finger anywhere on the screen, which caused the active areas to be highlighted; next, they would slide their finger into a highlighted area to get preview information, and they would then lift their finger from the screen to make a choice. We created a rectangular-shaped sign (visible on both sides of the screen depicted in Figure 2A) that had instructional graphics that included the words “Touch,” “Slide,” and “Lift” in both English and Spanish. When the touch-screen interaction technique was tested at the IBM location, no problems

arose. When it was tested at the visitor center at EXPO'92, however, we were surprised to observe one visitor trying the touch-slide-and-lift routine *on the sign* instead of the screen, showing bewilderment when nothing happened.

We used the IDF approach not only on the EXPO application, but also on the physical kiosk that housed the systems. Early in the project, we built a prototype kiosk "station" at our laboratory in Hawthorne, New York, and placed it in a high-traffic public area in order to obtain feedback on the design features, such as the angle of the touch screen, the location of peripherals (such as card readers), and the wording and location of instructions (Figure 2A). We made visits to the EXPO'92 construction site in Seville, Spain, and held extensive discussions with those responsible for the overall "look and feel" of the buildings at the fair. This led to our working directly with a design firm in order to create a structure that would meet our unique requirements and would conform to the standards for the buildings at the fair. The design firm (based in Barcelona) created a model of a kiosk that would hold seven user stations and two servers (Figure 2B). Our experience from previous projects led us to believe that many critical issues would surface as the small-scale model took a more realistic form, and we were confident such issues could only be resolved in a full-scale prototype before beginning the manufacture of the actual kiosks. We requested that the firm build a full-scale prototype of a kiosk in their shop in Barcelona (Figure 2C), quickly followed by a complete, full-scale kiosk on the EXPO'92 construction site (Figure 2D). Although the fabrication of the initial full-scale model in the Barcelona shop was a necessary first step, certain important issues—such as the impact of the Mediterranean sun on the computer screens—could only be realistically addressed on the EXPO'92 site in Seville.

After the full-scale prototype was installed on the EXPO'92 site, we began evaluating it. We mounted touch screens at different heights and angles, installed lighting to determine what was required to ensure that the video capture function could work effectively in both bright daylight and dark night conditions, and determined optimal locations for card readers so they could be used easily by children, adults, and disabled people (Figure 2D). This high-fidelity prototype kiosk was located near a stop for the tram (streetcar) tour of the construction site. Its high-traffic location enabled us to get early feedback from visitors, as well as construction workers, whom we invited to try the system. This prototype was critical to our ability to solve a number of significant "show-stopper" problems. Figure 2E shows the final version of the kiosk in use at the fair.

Offer a reward for participation. We have generally found that participating in iterative development and testing is its own reward in IDF because the users interacting with the system carry out tasks that are meaningful to them. At times, however, something more concrete may be called for to motivate participation. This is the case when users do not expect to use the system again (e.g., a foreign tourist visiting a museum), or when a significant amount of time and effort is required of them. MoMA visitors, for example, who elected to spend 20 minutes with us (often taking it from the limited time they had available to spend at the museum), were given small items of MoMA merchandise as a parting thank-you. On the other hand, in ALX, the dealers and banks had made business decisions to participate based on a variety of motivations, such as to obtain a first-mover advantage and to influence the system in a way that maximized the benefits to them. In this case, the rewards were inherent in their participation.

The initial tests of Touch Illinois at the Illinois State Fair offered perhaps one of the greater challenges in attracting test subjects. The fair offered the development team access to a large number of residents from a variety of backgrounds, yet one could not expect that many people would "apply" for unemployment benefits at the fair. Realizing that the basic interaction techniques (assisted data entry of names, addresses, phone numbers, and financial information) could be tested without requiring fairgoers to actually file for unemployment, we came up with a creative solution. The citizens were offered the opportunity to receive, by mail, an estimate of their Social Security benefits. They entered their personal data using Touch Illinois, as illustrated in Figure 1. The required signature was obtained through "finger painting" on the touch screen. An exact copy of the Social Security Administration's form, with the citizen's information in place, was then printed on plain paper and forwarded to the government office.

Use the appropriate method for gathering data. As discussed earlier, the fields of HCI and UCD offer a rich collection of methods. IDF requires an emphasis on those methods that help move the development forward at each iteration cycle. When working outside of the laboratory, perhaps at multiple

sites, the method's portability becomes a vital consideration. The speed with which a method can be applied is also of great importance, especially early in the development cycle, when the team is focusing on issues of overall functional design and general usability. Often, at this point, the precision of a method or its ability to generate publishable statistics is not as important as it may become later. Lightweight, informal, and qualitative methods have proven valuable in quickly and reliably clarifying the direction of system development. Interviews, observations, and think-aloud protocols that do not require videotaping and time-consuming data analysis are examples of techniques that fit these requirements. These methods are employed in succession over a short period. The convergence of different kinds of evidence (from these various methods), pointing to the same conclusions, is an important way to limit errors in interpretation and causal attribution.

Early in the development cycle, lightweight methods for gathering data are valuable for validating the direction of system development.

Immediately after installing the MoMA proto-application in the museum, we began user observations and evaluations. Typically, we began by observing visitors from a distance, then approached and requested permission to watch while individuals used the system. Finally, we conducted brief interviews. From the distant observations, we learned how people approached the kiosks and how the system was perceived in its physical and social context. Sitting next to the visitors, we were able to note which parts of the system they used and how they went about using them. From visitors who were comfortable thinking aloud, we gained insight into their actions. For instance, a repeated series of actions may have indicated the visitor was confused; in other cases, the visitor had simply noticed something interesting that he or she wanted to investigate. In the interviews, we combined preformulated questions with questions stemming from specific actions we had just observed. Flexibility in the choice of method enabled us to follow up on unanticipated behavior on the spot. We checked results of observational techniques against the system usage logs (and the visitor-comment books) in order to understand how representative and accurate our observations were, and conversely, to help us in future interpretations of the logs.

As specific issues arise in the field requiring a more rigorous approach, we may design a field experiment with greater controls or set up an experiment in the laboratory (making certain to evaluate subsequent design changes in the field). For example, we observed in the MoMA project that users chose to compare only the artists that were at the top (the visible part) of a long list of artists. Few users, freely engaging with the system, scrolled through the list to find other artists. To determine why this was happening, we asked them "How might you compare the works of Adams with the works of Weston?" Faced with this task, the majority of users selected Adams, then looked around for Weston (not visible on the screen) and said, "I don't know. He's not listed." They failed to detect a half-inch-wide scroll bar on the left side of the list. Even when encouraged to look around the whole screen, many failed at the task. After being shown the scroll bar, the most frequent response was, "I didn't see it. I'm used to a scroll bar on the right."

On each update of the proto-application in the field, it quickly becomes clear whether the change was successful. If the change does not yield the anticipated results, it is straightforward to reconsider the issue in the next iteration cycle using the newly gathered data. If a particular change is expected to entail risk, its introduction to the field can be managed in one or more ways: separate it from other code changes and assign it a unique iterative cycle; enable an easy back out of the update; or limit the introduction of the risky change to a manageable subset in the field. For instance, in the MoMA application, because of the modular nature of the implementation, it was trivial to alter the top menu to disable any particular function if it was found to be problematic.

While the general spirit of iterative development in the field is to learn from people doing their actual jobs, it is also prudent to introduce various kinds of "stress tests." The most obvious of these is load testing; that is, measuring the performance under maximum loads. Prior to the opening days of EXPO'92, we had members of our team take positions at each one of the seven stations in a kiosk and, at a precisely signaled moment, all of us began interacting

with the system rapidly, following an orchestrated scenario of actions.

In addition to these kinds of system stress tests, Bødker recommends developing and testing some "worst-case scenarios" that deal with the human aspects of the system.[39] For instance, what happens when the person is interrupted several times successively in the middle of an interaction with the system? Does the system allow the person to recover state quickly, or does it require him or her to keep track of the state information (where the interruption occurred)? In the MoMA project, for example, if the users stopped interacting with the system, after a pre-set interval they would be asked whether they wished to continue. The system would maintain their state of activity unless they explicitly indicated they did not care to continue. If they did not respond within a certain time, the system would return the application to the starting state. This preserved the state for the user who wished to continue after an interruption and reset the system appropriately for a new user.

Use the right development tools and processes. Safe and effective use of IDF requires appropriate development tools. Choosing the right tools can make the difference between success and failure; indeed the lack of appropriate tools may prevent one from attempting this methodology at all.

The pressure to respond rapidly to the users in the field can introduce problems as programmers hasten to implement changes in addition to new functionality already planned. Effective tools are needed to prevent code from deteriorating into hard-to-maintain "spaghetti" code as layers of "temporary" fixes are applied in succession. To support IDF, software development tools must enable rapid iterative changes (both large and small) while protecting the application from unintended side effects.

EXPO, ALX, and Touch Illinois were created using a tool base known as ITS (Interactive Transaction Systems).[40] ITS, which we developed explicitly to support rapid IDF, is based on separating the specification of application "content" from "style." The tools themselves served to isolate the impact of changes in content on style, and vice-versa, making iteration cycles relatively simple and safe. Function could be added quickly and easily integrated because the implementation of the user interface was rule-based. After the style rules became sufficiently rich, only minor tweaks were required for appropriate rendering.

Due in large part to the separation of content and style afforded by the ITS tools, we were able to continue to improve the EXPO application throughout the duration of the fair. For instance, at one point, the user logs revealed that a significant percentage of visitors who made restaurant reservations through the kiosk were not showing up at the restaurants. After investigating this problem, we discovered that many visitors mistakenly thought the reservation function was no more than some sort of interesting game. In a matter of weeks, we made several rounds of significant changes to the restaurant reservation interface—without jeopardizing underlying functionality—until the problem was resolved.

By the time we began working with MoMA, tools and architectures based on open standards that met many of our requirements had become widely available. Not surprisingly, a complete and ideal tool suite for supporting iterative development in the field does not yet exist at this time. Nonetheless, we have adopted many of these new tools by taking advantage of their wide availability and the ability of stakeholder teams to support and maintain them. The rapid progress in such supportive technologies, architectures, and tools are continuing to make IDF more feasible than it once was. A list of desirable tool attributes and some technologies we have found that possess them is discussed next.

Enabling rapid iteration while protecting the application: As highlighted in the discussion of ITS, designing an application with a modular structure helps isolate the implementation of different features, thus enabling changes in one section of code without negatively affecting another. Similar results were obtained in the MoMA project, by implementing the application in DHTML (Dynamic HTML) and running it within a Web browser. The UI (user interface) was marked up in XML (eXtensible Markup Language),[41] which enabled the use of abstraction for hiding the implementation details. We used XSLT (eXtensible Stylesheet Language Transformations)[42] to define style rules that transformed the abstract XML representation into HTML. CSS (Cascading Style Sheets)[43] provided fine styling control. The function underlying the application was implemented in JavaScript**.[44] Overall, the MoMA system has a modular design; new modules can be added with little or no impact on existing code. Changes to the MoMA user interface involved updating the XML represen-

tation or the CSS rules, frequently without the need to update the Javascript code. In general, Model-View-Controller[45] designs help to separate user interface code from functional code. Such modularity also provides the ability to quickly back out from the field code any changes found in user testing to be problematic.

Tracking code changes. These are more traditional tools (such as tools for software configuration management) that can track changes and, if necessary, help back them out. Bug and feature-tracking tools, like Bugzilla,[46] can manage software bugs, enhancement requests, and wish lists, allowing them to be prioritized and making sure they don't get overlooked. In addition, certain programming methods such as eXtreme Programming,[47] with its paired programming approach, can help prevent errors in design or implementation by keeping at least two pairs of eyes on changes as they are being designed and implemented.

Enabling remote upgrading and monitoring. In laboratory situations and controlled testing environments, it is relatively straightforward to ensure that systems perform as desired. However, when an application is deployed in the field, the only alert that a problem exists may be a user calling to complain. In the case of a public access system, even that channel may not be available. In IDF, because we continuously operate and maintain the evolving proto-application in the field, remote upgrading and monitoring is essential. To this end, we have created a suite of tools for remote monitoring that also provides remote upgrading and rebooting functions.

Maintaining duplicate systems. In addition to monitoring, we maintain duplicate systems in the laboratory that precisely mirror those installed in the field. With duplicate hardware, software, network configurations, and so forth, we are able to reproduce problems observed in the field or detected via remote monitoring. We can then determine the problem and implement and test a fix under field-like conditions. Having machines dedicated for this purpose reduces the lag between problem reporting and fix delivery. Teams of testers can use the duplicate systems to test new versions of code before they are deployed. These duplicate systems also serve as "hot" backups that can be immediately swapped with a field system that fails.

Applicability of IDF

We have used IDF in various application domains. All of these systems were created to solve actual business problems of IBM customers. All were deployed in environments where they supported significant business functions, and hence we believe it is reasonable to develop applications for entire lines of business using IDF. However, because there are many applications and environments we have not explored, we do not claim that IDF is appropriate for all types of system development.

A duplicate system that mirrors the one in the field can be used to reproduce and diagnose problems.

There are a number of development contexts that might reduce the efficacy or feasibility of IDF. One example might be a development effort that entails modifications to legacy systems. Unless sufficient compartmentalization can be achieved, changes to legacy system code may have a significant impact on the business, and hence the flexible application of rapid changes required by IDF might not be appropriate. In most of our work, we have avoided this problem because we have focused either on self-contained applications (such as EXPO and the MoMA project) or on applications that interfaced with legacy systems via well-defined mechanisms (such as ALX and Touch Illinois). In addition, because we have applied IDF to user-facing interactive applications, its applicability to the development of system software or middleware is likewise beyond the scope of our experience.

Size and scale issues might also pose challenges for IDF. We have refined IDF predominantly while developing small to midsized applications; most likely this approach would have to be modified to work smoothly in the development of extremely large applications. Even in such cases, it may be possible to decompose an application into parts and/or stages to which this approach may be applied.

Although our discussion contrasted the two opposing alternatives of field-based versus laboratory-based testing, we recognize a useful continuum be-

tween these two extremes. In those cases where a field situation cannot be found or reproduced, IDF might not be a viable approach. In life-critical systems, for example, it is clearly not desirable to engineer the life-threatening situation for testing purposes. In such cases, field-based iteration should be used to explore the real world context as much as possible (e.g., environmental, motivational, and social factors) and incorporate the knowledge gained in creating the simulated testing situations.

IDF may not be appropriate for applications that are used infrequently, because "camping out" in the field environment in order to observe the system being used becomes impractical. For instance, in recent work involving the search function on www.ibm.com, it was not practical to go to the workplace of potential users and wait until they had the need to use the Web site in order to observe their behavior. In such cases, a reasonable compromise is to ask users to reconstruct a recent visit to the site or ask them to perform some realistic search problems. An alternative would be to "instrument" the site in such a way that people may be contacted remotely when they are interacting with the system under development.

Beyond the characteristics of the application itself, other factors could also limit the feasibility of IDF. The team may not have access to an appropriate set of tools. The development team may be operating in an organization whose structure and procedures are not conducive to IDF. For instance, success in IDF depends to a large degree on having open access to a large variety of stakeholders. In some organizations, certain stakeholders "own" the access to other stakeholders and are able to effectively prevent their involvement in the project. Another situation that would limit the use of IDF occurs where the business plan calls for a "marketing coup" that premature public exposure might interfere with.

Despite the above issues that might make IDF less effective or even unfeasible, we believe there are many development efforts that can benefit from this approach. Our experience indicates IDF works very well for developing small to midsized, highly interactive, user-facing applications, especially those involving discretionary (optional) use. Researchers in the field of human-computer interaction often select users from the target population and have them perform tasks in the laboratory. The laboratory-testing environment is that of the researcher rather than that of the user. The social and institutional contexts in which such testing takes place are those of the researcher, and the functions tested are those that the researcher believes to be important. Although there are cases in which laboratory studies are an effective and efficient way to test and improve systems, there are other situations where iterative development in the field is a practical alternative that provides additional benefits to all parties involved.

Conclusion

In this paper we have described IDF (Iterative Development in the Field), an approach that places the development team squarely in the middle of the user's world. It employs frequent evaluations and updates to a proto-application—deployed early and left to remain functional in the field throughout development—as a means of obtaining maximal stakeholder feedback to fuel the evolution of the system. Such a process has risks, but we believe these risks are manageable and that the benefits outweigh them. Nielsen reports on a survey of usability experts regarding usability-engineering methods.[6] Each method was rated based on whether it was actually used in a development project and on the general impact of that method on usability. It is interesting to note that the attributes rated as having had the most impact roughly correspond to the defining attributes of IDF.

As demonstrated, IDF is not a theoretical approach, but one that we have successfully employed multiple times in a variety of real-world settings. We have used these experiences to illustrate guidelines for those eager to meet the challenges and achieve the benefits of this extreme form of User-Centered Design. We hope these guidelines will help to enlarge the community of IDF practitioners, which in turn will provide more experiences that will help to validate and improve the approach.

Acknowledgements

We thank all the participants and contributors to the projects we have discussed here and John Gould and Stephen Boies who laid the groundwork for IDF. We also thank the anonymous reviewers for their insightful comments on an earlier draft and the *Systems Journal* editorial staff for their help in preparing the manuscript for publication.

**Trademark or registered trademark of Sun Microsystems, Inc.

Cited references and notes

1. J. Earthy, B. Sherwood Jones, and N. Bevan, "The Improvement of Human-Centred Processes: Facing the Challenge and Reaping the Benefit of ISO 13407," *International Journal of Human-Computer Studies* **55**, No. 4, 553–585 (2001).
2. J. D. Gould and C. Lewis, "Designing for Usability: Key Principles and What Designers Think," *Communications of the ACM* **28**, No. 3, 300–311 (1985).
3. H. R. Hartson and D. Hix, "Toward Empirically Derived Methodologies and Tools for HCI Development," *International Journal of Man-Machine Studies* **31**, 477–494 (1989).
4. *Human-Centred Design Processes for Interactive Systems*, ISO/FDIS Technical Report 13407, Final Draft, International Standard, Technical Committee TC159/SC4, International Organization for Standardization (ISO), Geneve, Switzerland (1999), http://www.iso.ch/iso/en/ISOOnline.frontpage.
5. *Human-Centred Lifecycle Process Descriptions*, ISO/TR Technical Report 18529, Technical Committee TC159/SC4, International Organization for Standardization (ISO), Geneve, Switzerland (2000), http://www.iso.ch/iso/en/ISOOnline.frontpage.
6. J. Nielsen, "The Usability Engineering Life Cycle," *Computer* **25**, No. 3, 12–22 (1992).
7. D. A. Norman and S. W. Draper, Editors, *User-Centered System Design: New Perspectives on Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, N.J. (1986).
8. B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 3rd edition, Addison Wesley, Reading, MA (1998).
9. T. K. Landauer, *The Trouble with Computers*, MIT Press, Cambridge, MA (1995).
10. R. Bias and D. Mayhew, *Cost-Justifying Usability*, Academic Press, New York (1994).
11. M. Maguire, "Methods to Support Human-Centred Design," *International Journal of Human-Computer Studies* **55**, 587–634 (2001).
12. K. Vredenburg, S. Isensee, and C. Righi, *User-Centered Design: An Integrated Approach*, Prentice Hall, Upper Saddle River, NJ (2002).
13. R. R. Hall, "Prototyping for Usability of New Technology," *International Journal of Human-Computer Studies* **55**, 485–501 (2001).
14. G. M. Donahue, "Usability and the Bottom Line," *IEEE Software* **18**, No. 1, 31–38 (Jan/Feb 2001).
15. J. F. Kelley, S. L. Spraragen, L. Jones, S. L. Greene, and S. Boies, "Extending User-Centered Methods Beyond Interface Design to Functional Definition," *Proceedings of the Human Factors and Ergonomics Society 40th Annual Meeting*, September 1996, Philadelphia, PA, Human Factors and Ergonomics Society, Santa Monica, CA (1996), pp. 343–347.
16. http://www.usability.serco.com/trump/index.htm
17. N. A. M. Maiden and G. Rugg, "ACRE: Selecting Methods for Requirements Acquisition," *Software Engineering Journal* **11**, No. 3, 183–192 (1996).
18. J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey, *Human-Computer Interaction*, Addison-Wesley, Wokingham, England (1994).
19. A. Smith and L. Dunckley, "Prototype Evaluation and Redesign: Structuring the Design Space through Contextual Techniques," *Interacting with Computers* **14**, 821–843 (2002).
20. L. Suchman, *Plans and Situated Action*, Cambridge University Press, Cambridge, MA (1987).
21. H. Beyer and K. Holztblatt, *Contextual Design: Defining Customer-Centered Systems*, Morgan Kaufmann Publishers, San Francisco, CA (1998).
22. W. Sperschneider and K. Bagger, "Ethnographic Fieldwork under Industrial Constraints: Toward Design-in-Context," *International Journal of Human-Computer Interaction* **15**, No. 1, 41–51 (March 2003).
23. M. J. Muller, J. H. Haslwanter, and T. Dayton, "Participatory Practices in the Software Lifecycle," *Handbook of Human-Computer Interaction* (*2nd Edition*), M. G. Helander, T. K. Landauer, and P. V. Prabhu, Editors, Elsevier Science, Amsterdam (1997), pp. 653–688.
24. H. R. Hartson and D. Hix, "Toward Empirically Derived Methodologies and Tools for HCI Development," *International Journal of Man-Machine Studies* **31**, 477–494 (1989).
25. *Handbook of User-Centred Design*, NECTAR, http://www.ejeisa.com/nectar/inuse/6.2/contents.htm.
26. *Cost-Effective User Centred Design*, http://www.usability.serco.com/trump/methods/recommended/.
27. R. Van Buskirk and B. W. Moroney, "Extending Prototyping," *IBM Systems Journal* **42**, No. 4, 613–623 (2003, this issue).
28. L. Liu and P. Khooshabeh, "Paper or Interactive? A Study of Prototyping Techniques for Ubiquitous Computing Environments," *Proceedings of CHI 2003*, Ft. Lauderdale, Florida, April 5–10, 2003, ACM, New York (2003).
29. G. Fischer and J. Ostwald, "Seeding, Evolutionary Growth, and Reseeding: Enriching Participatory Design with Informed Participation," *Proceedings of the Participatory Design Conference* (*PDC'02*), T. Binder, J. Gregory, I. Wagner, Editors, Malmö University, Sweden, June 2002, Computer Professionals for Social Responsibility, Palo Alto, CA (2002), pp. 135–143.
30. J. Nielsen, *Usability Engineering*, Academic Press, New York (1993).
31. D. Siegel and T. Rouchka, "Demo-Driven Design or Design-Driven Demos: Vaporware, Demos and Prototypes," *interactions* **9**, No. 4, 25–30 (July 2002).
32. Videotapes providing more detail of many aspects of the projects discussed here (including on-site system usage at EXPO'92) are available from the authors upon request.
33. L. Jones and S. L. Greene, "MoMA and the Three-Legged Stool: Fostering Creative Insight in Interactive System Design," *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, New York City, August 17–19, 2000, ACM, New York (2000), pp. 39–47.
34. The IBM Guest Services System for EXPO'92 won the Alexander C. Williams, Jr. Design Award from the Human Factors and Ergonomics Society for outstanding human factors contributions to the design of a major operational system, September 1996.
35. L. Jones, C. M. Danis, and S. J. Boies, "Avoiding the Mistake of Cloning: A Case for User-Centered Design Methods in Reengineering Documents," *Proceedings of the 32nd Hawaii International Conference on System Sciences*, Maui, HI, January 1999, IEEE, New York (1999).
36. B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer* **21**, No. 5, 61–72 (1988).
37. S. L. Greene, "Characteristics of Applications That Support Creativity," *Communications of the ACM* **45**, No. 10, 100–104 (2002).
38. J. Thomas, C. Danis, and A. Lee, *Who Speaks for Wolf?* Research Report RC22644, IBM Research Division, Yorktown Heights, NY (2002).
39. S. Bødker, "Scenarios in User-Centred Design: Setting the Stage for Reflection and Action," *Proceedings of the 32nd Ha-*

waii International Conference on System Sciences, Maui, HI, January 1999, IEEE, New York (1999).
40. C. Wiecha, W. Bennett, S. Boies, J. Gould, and S. Greene, "ITS: A Tool for Rapidly Developing Interactive Applications," *ACM Transactions on Information Systems* **8**, No. 3, 204–236 (1990).
41. *Extensible Markup Language* (*XML*) *1.0* (*Second Edition*), W3C Recommendation 6 October 2000, World Wide Web Consortium, http://www.w3.org/TR/REC-xml.
42. *XSL Transformations* (*XSLT*) *Version 1.0*, W3C Recommendation 16 November 1999, World Wide Web Consortium, http://www.w3.org/TR/xslt.
43. *Cascading Style Sheets, Level 2 CSS2 Specification*, W3C Recommendation 12 May 1998, World Wide Web Consortium, http://www.w3.org/TR/REC-CSS2/.
44. *ECMAScript Language Specification 3rd edition* (*December 1999*), Standard ECMA-262, ECMA, http://www.ecma-international.org/publications/standards/ECMA-262.HTM.
45. A. Goldberg and D. Robson, *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley, Reading, MA (1983).
46. *Bugzilla Bug Tracking System*, Mozilla.org, http://www.mozilla.org/projects/bugzilla/.
47. K. Beck, *Extreme Programming Explained: Embrace Change*, Addison Wesley, Reading, MA (2000).

Accepted for publication June 25, 2003.

Sharon L. Greene *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 (slg@us.ibm.com).* Dr. Greene is a research staff member at the Watson Research Center in Hawthorne, N.Y., where she is exploring pattern languages and tools to aid in pattern creation and utilization in the design and development of applications. Her recent activities in the areas of pattern languages include participation in the workshop on pattern languages at CHI 2002, co-organizing a workshop on socio-technical patterns at CSCW 2002, and co-organizing a workshop on Patterns and Pattern Tools for HCI at CHI 2003. Dr. Greene comes from a background in Experimental Psychology, having obtained her Ph.D. at Harvard University in 1981 doing research on concept formation in animals. She did a post-doctorate in cognitive development at Barnard College, Columbia University, followed by another in human-computer interaction at Bellcore. Dr. Greene joined IBM Research in 1987 where she has focused on the development of useful, usable, and fun applications, as well as on tools that enable development of such systems. Her work and publications have included studies of database query languages, tools for application development, uses for radio frequency identification technology, technology in support of creativity, and the design, development, and use of public access information and learning systems.

Lauretta Jones *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 (jonesla@us.ibm.com).* Lauretta Jones is a research staff member and manager at the Watson Research Center. Jones leads her multidisciplinary team in a never-ending journey to make computers fun, useful, and easy to use. Currently, the team is developing tools to help software developers create programs that are easier to use and that support learning-while-doing. She was co-organizer of a workshop on patterns and pattern tools for HCI at CHI 2003. Jones joined the Watson Research Center in 1990, becoming the first artist to work alongside IBM Research Division's computer scientists, cognitive psychologists, and programmers. Jones is the former Director of Undergraduate Computer Studies at the School of Visual Arts in New York, where she developed graduate courses in User-Centered Design and user interface development for art and design students. In 1988, she designed the interactive systems *Face to Face* and *Drawn to the Light*, which were purchased for exhibit by the Franklin Institute in Philadelphia and the Museum of Science and Industry in Chicago. Jones received a B.F.A. degree from the Cleveland Institute of Art in 1975. In addition to her position at IBM Research, Jones is an exhibiting artist and an instructor in botanical art at the New York Botanical Gardens.

Paul Matchen *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 (matchen@us.ibm.com).* Mr. Matchen is an advisory software engineer in the Next Generation HCI Components department at the Watson Research Center, where he has spent the past 10 years seeking to make computers more fun and easier to use in everyday tasks. His current work involves design and implementation of reusable HCI components using open-standards technologies (XML, ECMAScript, HTML, Web Services). He holds an M.S. degree in computer science from Polytechnic University, Brooklyn, N.Y. He has been working with a diverse group of professionals in the area of human factors in engineering since 1991, helping people work with computers in the real world.

John Thomas *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532 (jcthomas@us.ibm.com).* Dr. Thomas's current work is in the general area of learning objects and pattern languages. His prior work focused on developing new tools, techniques, and representations to support the capture, creation, analysis, organization, finding, and use of stories and scenarios in a business context. Dr. Thomas received a Ph.D. in psychology from the University of Michigan in 1971. Before joining the IBM Research Division in 1973, he managed, for two and a half years, a research project in the psychology of aging at Harvard Medical School. He spent the next 13 years at IBM doing research in various areas of human-computer interaction (HCI) including query languages, natural language processing, design problem solving, audio systems, and speech synthesis. In 1986, he began the Artificial Intelligence Laboratory at NYNEX Science and Technology, Inc. The laboratory did work in expert systems, machine vision, human-computer interaction, intelligent tutoring systems, robotics, speech recognition, and advanced tools for software design. Dr. Thomas was active in the formation of ACM's Special Interest Group in Computer Human Interaction and has served in various capacities including general co-chair of the CHI conference in 1991. Dr. Thomas has taught a variety of college-level courses, including courses in cognitive psychology, problem solving and creativity, the psychology of aging, and human factors in information systems. He is the author of over 130 publications and invited presentations in computer science and psychology.