Usability through System-User Collaboration*

Deriving Design Principles for Greater ERP Usability

Tamara Babaian, Wendy Lucas, Jennifer Xu, and Heikki Topi

Bentley University, Waltham, MA 02452, USA
{tbabaian,wlucas,jxu,htopi}@bentley.edu

Abstract. Enterprise Resource Planning (ERP) systems have become essential in industry, yet the potential value created through system use can be illusive due to poor usability. Extensive interviews with users revealed that the underlying complexity of these systems manifests itself in unintuitive interfaces that are challenging to use. Given the lack of progress made with traditional design approaches, we propose a different tactic based on a system-user collaborative approach. This entails that the system acts as a collaborative partner by sharing knowledge, providing task-specific support, and adapting to user behaviors. Based on this collaborative view, we derive a set of principles for guiding the design of ERP systems and provide concrete examples demonstrating (1) how a lack of collaborativeness contributes to various usability problems, and (2) how our proposed design principles can be used to enhance the collaborativeness and, hence, the usability of ERP systems.

1 Introduction and Motivation

Enterprise Resource Planning (ERP) systems are widely employed in industry to integrate various business processes. While this integration has the potential to provide tremendous operational value, using these systems can be a challenge for novices and even experienced users. ERP interfaces are typically unintuitive, presenting an abundance of information reflecting the underlying complexity of the processes around which they are built. The poor usability of these systems has been noted in industry reports [13, 14] and field studies on usage [15, 25, 7].

The lack of progress in addressing the usability of ERP systems has motivated our interest in this topic. The prevailing theme in user interface design is the human-centered paradigm, with its emphasis on knowing the user. While user-based methods work well for uncovering usability problems [8], they typically focus on a narrow scope of specific features of the existing implementation. This tends to lead to localized fixes rather than system-wide alterations of the design [19]. This is particularly problematic for ERP systems, whose broad scope and

* This material is based in part upon work supported by the National Science Foundation under Grant No. 0819333. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

integrated functionality require the use of multiple system features by multiple users for achieving comprehensive goals.

We propose that viewing system-user interactions through a "collaboration lens" affords a novel perspective that is advantageous for improving usability. The human-computer collaboration paradigm specifies that the system must act as a partner to its users by supporting them in the increasingly complex environments of modern applications [11]. This changes the dynamic from the user being the only one with responsibilities and knowledge about the process to one in which the system is called upon to do its part. Note that this approach is different from Computer-Supported Cooperative Work (CSCW), which is concerned with computing technology that supports human collaboration.

The work presented here is part of a multi-method research project for addressing ERP system usability that uses collaboration theory as a unifying framework. Components of this project include conducting field studies for identifying usability issues, modeling usability based on collaboration theory [7], developing software artifacts for addressing usability limitations identified in field studies [2], and designing an infrastructure that supports input logging for use in evaluating proposed design interventions [1].

In this paper, we make the following contributions:

- demonstrate how to use the theory of collaboration to derive novel design principles for improving the usability of enterprise systems,
- highlight usability problems in ERP systems based on findings from our interviews and observations in the field, and
- link the collaborative properties of a system to usability using empirical data and theory.

The next section of this paper describes our theoretical framework and related work. We then examine the link between usability and collaboration as revealed by concrete examples from a field study of ERP users at three organizations. Next, we derive a set of design principles based on characteristic properties of ERP systems and illustrate how they can be applied for achieving greater usability. We conclude with a discussion and directions for future work.

2 Theoretical Framework and Related Work

2.1 Human-Computer Collaboration

The collaboration paradigm of human-computer interaction (HCI) [24] views the interaction between a system and its user as a process in which they work together to achieve shared goals. There are various philosophical accounts [23, 4] and computational frameworks (e.g. [12, 6, 17]) of collaboration involving humans and/or computer agents. Terveen's review article [24] summarizes several different approaches to modeling collaboration in interfaces. Terveen identifies the following key issues as being present in virtually all of these approaches:

1. An agreed-upon goal of collaboration (often referred to as the collaborative activity). The specification of the goal may not be complete at the onset. During the collaboration, the parties gradually explore and decide on the essential details.
2. Plans for performing the activity, division of tasks between the parties, and coordination. As with the goal, such plans may be only partially specified initially and evolve with time.
3. Shared context. The parties must be aware of the progress towards the goal.
4. Communication. The parties must share information and communicate to establish their goals, allocate tasks, etc. Observation of the other partners' activities and behaviors is also essential.
5. Adaptation and learning. Effective collaboration leads to partners learning about each other and adapting to each other in order to maximize the success of their joint efforts.

Terveen distinguishes human-complementary from human-emulation approaches for implementing system-user collaboration. The latter is focused on developing human-like abilities in the system's interface by, in particular, communicating via natural language and modeling and recognizing the mental state of the human user, including his beliefs, goals, and plans. The human-complementary approach, on the other hand, recognizes the fundamental difference in the natural strengths of humans and computers and aims to make "the computer a more intelligent partner" (page 73) by means that leverage the natural strengths of each party. Our work presented here falls into the class of human-complementary solutions.

Grosz [11] distinguishes two ways in which the formal theoretical frameworks of collaborating agents are applied in the design of software: (1) using the theoretical framework directly as a formal specification that prescribes the constraints on the system's behavior, and (2) as a design guide that provides an "insight" into relevant aspects of successful system-user collaboration at the design stage. Grosz argues that effective human-computer collaboration does not require human-like abilities in the interface but can be brought upon by different mechanisms. She calls for investigating approaches in design that strengthen the collaborative properties of system interfaces and solving computational research problems that arise in implementing such approaches.

The theory of collaboration guides our design approach by serving as a lens through which system-user interactions are viewed. In particular, we follow the philosophical view of Bratman [4] and the SharedPlans [12] mathematical model of collaborative action expressed in the form of a logic. Throughout this paper, when we refer to the theory of collaboration, we are referring to these two theories.

Grosz summarizes the SharedPlans model in [11, page 536]:

> "Translated into English, the definition states that for a group activity to be collaborative, the participants must have (1) intentions that the group perform the group activity; (2) mutual belief of a recipe; (3) individual or group plans for the constituent subactions of the recipe; and

(4) intentions that their collaborators (fellow group members) succeed in doing the constituent subactions."

A *recipe* refers to how to perform the activity, as agreed upon by all participants. The specification implicitly requires that participants communicate as necessary to share their knowledge in order to establish mutual belief in the overall recipe, form individual and group plans, or provide helpful information. Partners must also maintain some knowledge of the context of their interaction for that interaction to be efficient. Clause (4) above also implies that subtasks are assigned according to the collaborators' capabilities, and that partners must be committed to helping each other when the success of their joint activity requires it [12, 11]. Note that the collaboration we consider in this work involves only two partners, the system and its user, and no subgroups are involved.

The SharedPlans formulation is consistent with the principles Bratman identifies as required for a joint activity to be a collaboration. His requirement of *commitment to the joint activity* implies that the parties have intentions to successfully perform the activity together. It also captures intentions to refine the group and individual plans for the activity, share information whenever necessary and, overall, act in a way that leads to the success of the collaborative enterprise. The *mutual responsiveness* requirement states that the partners must adjust their own behaviors based on the actions and intentions of their collaborators in a way that facilitates achieving their joint goal. *Commitment to mutual support* further requires that all parties be ready to help a partner who is having difficulty with her portion of the activity if they can provide such help.

The key parameters summarized by Terveen are present in both SharedPlans and in Bratman's account, though "packaged" differently. This allows us to refer to Terveen's concepts in our exposition throughout this paper, while employing the more elaborate and nuanced specifications of SharedPlans and Bratman where necessary. We must note that those two theories also address important collaboration-related phenomena that extend beyond Terveen's list of five.

Software artifacts that embody collaborative behavior in some form have been implemented for a variety of domains (e.g.[21, 5, 3]). However, we are not aware of any applications of the collaboration paradigm to large-scale multiuser organizational systems, such as ERP systems, other than our own work.

2.2 Usability and Design Principles for Enhancing Usability

We rely on one of the most widely accepted definitions of usability, which is based on the ISO standard 9241-11. It defines usability as the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." [16, page 2]. While there are other definitions of usability (e.g. [9, page 4], [10, page 300], [22]), they are fundamentally consistent with the core elements of the ISO definition.

The three core terms are defined as follows: effectiveness is specified as "accuracy and completeness with which users achieve specified goals;" efficiency refers to "resources expended in relation to the accuracy and completeness with which

users achieve goals;" and satisfaction is "freedom from discomfort, and positive attitudes towards the use of the product." For example, if the user's goal in a specific ERP context is to complete purchase requisitions, effectiveness refers to the extent to which the finished purchase requisitions reflect the intended purchases accurately and are complete; efficiency refers to the number of purchase requisitions completed within a unit of time; and satisfaction refers to the extent to which the user is able to complete the task without discomfort and with positive attitudes regarding the process of using the system. Three other elements of critical importance to this definition of usability are "specified users," "specified goals," and a "specified context of use." Any concrete way to measure effectiveness, efficiency and satisfaction will produce different results depending on the user – goal – context of use combination.

A great deal of literature – both academic and practitioner – exists on design guidelines and principles for enhancing usability. This work can be divided into two streams: one focusing on characteristics of the artifact and the other on the design activity itself. Since our work concerns the artifact, we consider only the former stream here.

Polson and Lewis [20] present a set of design principles for applications that would allow "successful guessing." These highly regarded principles were based on their CE+ theory concerning the learnability of a system and are aimed at walk-up-and-use interfaces. While we are also deriving design principles from theory, a key difference is our focus on interfaces for handling complex processes.

The work that is most relevant to our own is Nielsen's [18] time-tested and well-known set of usability heuristics, which can be viewed as both design guidelines and a way to define systems in terms of concrete indicators of specific characteristics. Nielsen analyzed seven sets of well-known usability heuristics and used principle components factor analysis to extract nine factors, essentially integrating the heuristics into a set of design principles based on their ability to "explain" usability problems. Each heuristic is listed below and followed by a description:

1. Visibility of system status: provide users with feedback regarding the status and progress in task performance.
2. Match between the system and the real world: use the vocabulary of terms and follow the conventions with which users are familiar.
3. User control and freedom: allow users to redo or undo actions; do not limit the actions that users can take at a specific time.
4. Consistency and standards: define and present the same things in the same way across the system.
5. Error prevention: reduce the opportunities for users to make mistakes.
6. Recognition rather than recall: make objects and options available and visible for users.
7. Flexibility and efficiency of use: allow experienced users to use accelerators of action (such as shortcuts).
8. Aesthetic and minimalist design: include only relevant information in dialogs.
9. Help users recognize, diagnose, and recover from errors: be able to detect errors, identify their sources, and provide constructive solutions.

Nielsen's usability heuristics have been widely accepted in the field of HCI and frequently cited in usability evaluation studies. It has been reported that these heuristics could help find serious usability problems that are likely to cause "major delays or [prevent] the users from completing their task" [18, page 154], resulting in low effectiveness and efficiency.

Like Nielsen, our purpose in designing guidelines is to capture the most frequently occurring usability problems and to lay a foundation for designing more usable systems. Our principles, however, are derived theoretically from collaboration theory, which can be used to explain Nielsen's principles and also to provide another view of human-computer interactions that may reveal a wider range of usability issues.

3 Lessons From the Field

In this section, we present lessons learned from our field studies. We interviewed 33 employees at three organizations located in a northeastern U.S. state during fall 2008 and spring 2009. These organizations represented different industry sectors (IT, property management, and medical device manufacturing), used different ERP systems, and had varying levels of system experience. The interviews were semi-structured and conducted with a set of interview questions that were designed to learn users' perceptions of and experience with ERP systems through the lens of collaboration. Interviews were audio recorded and then transcribed and coded. Using examples from these interviews, we reveal the link between collaboration and usability by demonstrating how the absence of key properties of system-user collaboration [24] can be viewed as violations of several of Nielsen's heuristics [18].

Shared goal. Successful system-user collaboration requires that each party in the collaborative activity knows the shared goal. Since ERP systems are designed to assist users, the shared goal between the system and a user is often equivalent to the user's goal. Thus, it is important for the user to be clear about what the business goal is before performing any task. This is illustrated by the following comment from a superuser (an experienced user who has a strong understanding of how the data and processes are related and routinely uses different components of the system):

> *User 1: And the first thing is [to] forget about the system for a sec. Keep your keyboard away, discuss as a group, individually, collectively, whatever, what are you trying to do conceptually. Then execute.*

The plan to achieve the goal. Even though users know the business goal and the logical steps in the plan for achieving it, those steps may not be easily mapped to the functions provided by ERP systems. The systems' interfaces, which are designed for a wide range of business processes, contain various functions, menu items, instructions, clickable icons and buttons, etc. The intrinsic complexity

of the interrelated business processes coupled with complex interfaces makes it difficult for users to perform even a very simple action like locating a function. Moreover, many tasks involve a series of steps, screens, functions, forms, and data. The ERP systems in our study did not provide navigational or procedural guidance through these processes:

> *User 2: It [the system] doesn't tell you what steps to take next. You have to basically know what the next step is for your process, for what your job title is to do.*

As a result, users spend a significant amount of time in training, communicating with colleagues, and using trial and error approaches to learn the steps for their tasks. These steps are described by users as "unintuitive," as they often do not match the logical steps that users associate with the business processes. As a result, users frequently create "cheat sheets" that document the procedures and steps required for a task. With practice, users may no longer need these notes, but they continue to rely on them for performing non-routine tasks:

> *User 3: I have a little checklist, so when I do ACH payments, I just have screen charts and just little directions that I need to go back in and redo it. I have just directions on step-by-step with the screen chart. This was just so much easier.*

The need for memorization and notes is in violation of several of Nielsen's [18] usability heuristics. The match between ERP systems and the real world is not particularly good (violating heuristic #2), and, clearly, ERP systems require recall rather than supporting interactions based on recognition (in violation of heuristic #6). In addition, the lack of navigational and procedural guidance violates heuristic #1: system status is typically not available, leaving users unsure of their progress in performing a task.

Shared context. In an ERP system, the shared context includes the business context for the task. That contextual information may not be explicit and immediately clear to users:

> *User 4: If I had a new person in purchasing, I'd need to tell them what the company code was and how our GL chart of accounts worked and what our cost-center structure is – a lot more details in order for them to be able to enter just one invoice. And then it's spider webs off of that as far as whether it's a fixed asset or pre-paid, etc. So there's a lot more information that needs to be shared there if we had a new employee in any one of those areas.*

Actually, the system maintains the business contextual information and could easily present it to users if it had been designed to do so. This example suggests another violation of Nielsen's heuristic #2: the match between the system and the real world is not as good as it could be. This example also points to a contribution of the collaborative theory-based approach to viewing system interactions, i.e., the emphasis on sharing information between the system and its users, which is not part of Nielsen's heuristics.

Information on progress and feedback to the user are also important context that the system should provide in general and are essential when the user needs help. The system must first be able to detect and recognize that need for help, but ERP systems typically play a passive role:

> *User 5: Now, the thing is that in this case, the system is not reaching out to you saying that you obviously need help. It's me having to go find it there. Just to go back to that GL account scenario, rather than just telling me you had to put something in, if it knew automatically what that one was supposed to be, and once you failed, say three times, or X times putting in the wrong one and then at that point, it would query you – you obviously need help here. And then it would send you to a help desk function.*

Furthermore, ERP systems often fail to utilize the contextual information they possess regarding the organization, user, business process, and task. An error message may simply report that there is a problem without offering any diagnoses or suggestions, or even isolating where that problem exists. While some error messages provide possible solutions, users often find them to be too general to be helpful:

> *User 6: No, it doesn't tell me detail, but it tells me that it cannot be performed at this time.*
> *User 2: It's just the [dinging] sound, yes! Nothing comes up and you know that you're looking at the wrong order in the wrong [location]. It doesn't come up with a pop up screen that says this is the wrong order.*

Some users try to seek solutions by reading system-provided help documents, but this is usually not a productive use of time, as the documents are often not specific to the task at hand and do not consider the context of the activity. As a result, users typically ask someone else (coworkers, superusers, IT staff) for help:

> *User 5: I would just call someone because again, I have spent time trying to figure it out and go through the menu path, and I feel like I always get more lost and I'm just trying to save time, so I just usually pick up the phone and call someone.*

These examples illustrate violations of Nielsen's heuristics #1 and #9: in an error situation, the users find it impossible to use the system to identify needed status information, and the system is often unable to help users recover from errors.

Communication. A collaborative activity will likely fail if the parties do not maintain good communication. Communication requires sharing knowledge, which, in ERP systems, includes business data, the procedure for a task, status and progress reporting, and context. To communicate and share knowledge, the system should speak the users' language and use the vocabulary of terms with which they are familiar. However, the terminology used by some ERP systems

is drastically different from the users' and little or no explanation is provided about what terms mean:

> *User 5: I don't know how it's chosen that for vendors it's XK, and for purchase orders it's ME prior to the numbers. I do know, obviously, the numbering system as far as O1 is for creation, O2 is for change, and O3 is for display. But no idea what it means to the actual function!*
> *User 7: Sometimes when I get an error message and I don't know why I'm getting it then that's when it's questionable about what's going on. Because usually it's in codes, and I don't understand that.*

Incomprehensible terms and error messages are in direct violation of Nielsen's heuristic #2 (the match between the system and the real world), which includes components such as "Speak the user's language" and "Contains familiar terms and natural language."

Adaptation and learning. To maximize the long-term success of a collaboration, each party needs to learn from and adapt to the others. In interactions with ERP systems, users are the ones who must do the adapting. An alternative would be for the system to also adjust to its users' behaviors by taking into consideration their previous actions. This would enable the system to automatically populate previously entered data, list functions in the order of frequency of use, offer an option of repeating a frequently performed task, etc. However, such capability is currently lacking:

> *User 7: I don't think it does something to make it easier due to the replication of me doing something. So, if the system had enough intelligence that it noticed that I am always printing the details for all the items that are on the overall report, and then it would say let me offer you, do you want to print all of this? You seem to be doing this always.*

Nielsen's heuristics do not include any with a direct match to such behavior. He does identify "use of default values so that the user does not have to re-enter information" [18, page 153] as one of the heuristic candidates that was not included in the final set of nine. Use of default values is, of course, more narrowly defined than the broader consideration of previous actions, and this is clearly one of the areas where our approach extends those heuristics.

The above examples illustrate the lack of collaboration between ERP systems and their users, and the frequent violations of Nielsen's usability heuristics can be framed in terms of the non-collaborative behavior of these systems. A true collaboration aimed at enhancing usability requires a partnership, the absence of which is best summarized by a superuser in this vivid way:

> *User 8: So with the system, it's somebody that just smirks at you. And when you make mistakes, it looks at your with the same kind of dopey look on its face ... And it starts forcing you to kind of work around and work over and work under. And that's the frustrating part about it that again is the biggest pain in the backside. So, I would say, it's not a good partner, I feel like it's an impassive sometimes uncooperative coworker.*

4 Design Principles

In this section, we first analyze the central constructs of collaboration theory as they apply to system-user interaction in the ERP domain. We then derive design principles for making the system a better collaborative partner, based on our extensive field study of ERP usage and the theoretical framework outlined in section 2.1.

4.1 System-User Collaboration in the ERP Domain

A shared goal and intentions towards it are the prerequisites to any collaboration; thus, we review these concepts in the context of the ERP domain. The overall **goal** of this system-user collaboration is to automate the management of data related to the business processes within an organization for achieving greater organizational efficiency. This high-level goal can be decomposed into smaller, more specific activities (such as fulfilling a customer order) and even further down into transactions involving an individual user and the system working on a particular task. While the set of tasks an individual user is exposed to is limited, they are part of a chain of tasks that correspond to the components of a business process.

Intentions towards collaboration. We view the process of users collaborating with ERP systems as being similar to the collaboration that occurs between co-workers in an organization. While co-workers can choose not to participate in the organization's processes, this option is rarely taken due to the likely negative consequences. Similarly, employees are motivated to use the ERP system for both contractual reasons as well as for benefits derived from such use. A further simplification is that, unlike a human co-worker, the system does not have any competing intentions; all of its time and resources are devoted to its users.

Given the proper intentions towards the shared goal, it is important to consider the **knowledge** and **abilities** of the collaborators for the optimal division of labor according to each partners' strengths. In terms of **knowledge** and **abilities**, an ERP system is an embodiment of widely generalized organizational practices. It has superb capacity for storing, organizing, retrieving, and visualizing organizational data. An employee has partial knowledge of the organization's operations and business transactions, business practices, and associated data. It is important to realize that, while this knowledge depends on the employee's role, it is always incomplete.

4.2 Designing for Collaborative System-User Interactions

Knowing the plan and communication. To engage in a successful collaboration in performing an activity, the system and its users must be aware of the *overall recipe*. The system's *communication* to the user regarding the steps to be taken includes textual and pictorial labels on input fields and buttons; components used for navigation, such as menus and lists of transactions; and

instructions and other text provided in dialogs and error messages. When users are familiar with the steps involved in completing a task, they are quite efficient at using the system to do so. However, the learning process is lengthy and characterized by negative terms such as "brutal" and "intimidating." This is due to multiple factors, including the mismatch between the users' and the system's vocabularies and the generic nature of the interfaces, which do not reflect the practices with which users are familiar. From the collaborative standpoint, the user is the one forced to take on the burden of learning to speak the system's language, utilize the necessary functionality, and navigate to the appropriate interfaces.

The complexity of the learning process and the overall effort expended by the user would be greatly reduced if the system took part of this process on itself. For example, having labels on menu options, transaction names, input fields, etc. be consistent with the organizational vocabulary would greatly improve the user's understanding and confidence. Furthermore, the graphical interface could be customized to include only those input components that are essential for the organization. This latter kind of optimization is sometimes done in practice but is typically avoided because of initial costs and, more importantly, incompatibilities with later versions of the ERP software that will incur future costs. These considerations lead us to our first design principle (henceforth abbreviated DP):

DP1. The user interface should provide a mechanism for customizing the vocabulary of terms used by the system in its communication to the user, the composition of business transactions, and the content of the system's informational output to match the practices of the organization. There should be a mechanism for incorporating the customizations from an earlier version of the system to a later one.

This design principle does not prescribe a particular method of customization. For example, it can be done using machine learning techniques that draw on the history of system-user interactions, or can be performed manually, or can be achieved using some combination of the two. While we are working on developing effective methods (algorithms, representations) and design sketches for implementing our design principles, they are beyond the scope of this paper.

To further aid a novice or an infrequent user in understanding the steps required to complete a task, the system should provide navigational support and information on progress in completing a process. From the collaboration standpoint, this sort of explanatory guidance is required, since the system is the one with complete knowledge of the relationships between the data, the process, and the interface components, and must share any knowledge that the human partner needs to perform her part.

DP2. The system should provide navigational and progress guidance to a user performing a transaction, indicating the broader context of each interaction in terms of the related business process components and specifying the completed and remaining parts. A sufficiently competent user should be able to turn off this guidance if it becomes a distraction.

Commitment to helping a partner in need. When something goes wrong during an interaction and an error is signaled by the system, users often experience difficulties understanding and resolving the problem. The poor quality of the error message can be a factor, but even if the message is reasonably descriptive, the user's difficulties are often due to the following:

- Data-to-process and process-to-process relationships play a critical role in defining ERP system functions, but they are too numerous to be known in their entirety. Because of this complexity, the structures and relationships between processes and data are typically hidden from the users, leaving them unable to diagnose the causes of many even trivial problems, such as an incorrectly specified code.
- Even when a user is familiar with the business context and operation of the interfaces being used, another hurdle in diagnosing problems stems from the fact that ERP systems involve multiple users working on different parts of time-extended business processes. The processes affect different but related portions of business data, but individual users often lack an understanding of how their tasks relate to the broader process in which they are taking part. This greatly impedes their ability to diagnose and correct errors that resulted from the actions of other users in a related task interface.

As a result of the above plus the user's perception of the ERP systems as "intimidating," the most common error diagnosis and resolution strategy involves asking another person (colleague, superuser, consultant, etc.) for help. The system can be so obscure that even users who have encountered the same error before often cannot recall how to overcome it. The help function is regarded as a waste of time due to the lack of context of the information that is presented and the effort required by the user to "connect the dots." In our field studies, we have invariably encountered stories about errors that took days to diagnose.

Collaboration presupposes a commitment to completing the joint activity and helping a partner who is having a problem performing her part. When a system signals an error, it is a clear sign that it is aware of the fact that something has gone wrong. Typically, the ERP system's involvement in diagnosis and correction of errors stops at the reporting stage. In many cases, the system has access to contextual data that would explain the cause of the problem. Sometimes the solution or a set of possibly helpful actions are readily available and identifiable, but the system usually takes a passive role, leaving the burden of diagnostic discovery to the user. This behavioral pattern is primarily due to the lack of focus at the design stage to providing error diagnostics and recovery functions in the system interfaces.

Two examples illustrate our point. The first is a simple, real life case in which a user has entered a shipment date into a field. The system rejects the user's input and generates an error message stating that the date is in an incorrect format. The system waits passively for the user to correct the error. A more collaborative response would be for the system, when displaying the error message, to also bring up its calendar feature from which the date can be selected, or at least suggest the use of this feature.

An example with a less obvious solution is an error that resulted from a mismatch between the parameters of a business process entered by different users. This is the type of error that would typically be sent to a superuser to diagnose. Instead, the system could aid in the diagnosis by providing the broader context of the interaction within which the mismatch occurred: i.e., display a list of the related transactions and provide easy access to views of the related data.

Not all error situations are due to the actions of the user; for instance, a storage device failure may prevent the system from saving data. Commitment to the success of the collaborative activity requires that the system not give up until it has explored other avenues for problem resolution and consulted with the user when his agreement to a solution is required.

DP3. When the system detects a problem, it should identify the possible causes and ways of resolving it. If the fix is obvious, the system should inform the user and perform it. If it isn't obvious, the possible causes and resolution scenarios should be presented to the user and be readily executable. If the system is unable to identify resolution strategies, it should present the user with the relevant data and transactions.

Deciding whether to proceed with a fix to a problem with or without engaging the user depends on the nature of the problem and, sometimes, the particular user's preferences. These items should be carefully considered during the system design stage to make error resolution and diagnosis effective.

Other helpful behaviors. Helpful behaviors are those that increase the effectiveness and efficiency of the collaborative efforts of the parties and increase the likelihood of the success of the joint activity. As theory states, such behaviors stem from the partners' commitment to the success of the joint activity. Humans often do things that are helpful for a group effort without being explicitly asked. For example, when going to a business meeting, they take their calendar with them, knowing the group may need to schedule the next meeting. In doing so, people are using their knowledge of the task, the environment, their partners, and their commonsense reasoning abilities. There are many opportunities for designing helpful behaviors into the system based on both a priori analysis of the tasks, environment, and users, as well as the data collected during system use. An example is displaying those currencies most frequently selected by the user in prior interactions at the top of the list of world currencies.

A less straightforward example involves the wide variety of search interfaces common to ERP systems. To find a code for a specific material, for example, one can search through the entire material master, or by material by plant, or by material group. If a user invokes a search interface, instead of blindly offering a collection of tabs for all possible search options, the system can use the contextual information available to it to rank-order the options and to fill in any known details. If the user is working on an order form and has specified the plant for which the order is to be placed, a search for material by plant can be highlighted and the search interface should include the specified plant number.

DP4. In presenting selection choices, the system should utilize what it knows about the user, the organization, the task, and the context, and provide faster access to the more likely choices than the less likely ones. Where the choice of data or action is obvious, the system should have an option of not waiting for the user to enact it. The user should have an option to replace/cancel the system's provided choice of data/action.

5 Discussion

The above design principles for achieving greater usability of ERP systems by improving their collaborative strength were derived using the theory of collaboration and findings from our field studies. Quotations from those studies, presented in Section 3, highlight collaborative weaknesses of ERP systems. They also provide evidence of usability problems, many of which can be explained using Nielsen's usability heuristics.

As shown in Table 1, our proposed design principles encompass Nielsen's usability heuristics, thus supporting usability. However, the principles go far beyond merely restating and aggregating those heuristics: they provide a unified theory-based perspective that explains their utility in terms of human-computer collaboration. Furthermore, what sets our work apart from other design principles for usability is its emphasis on the system's role in using its capabilities and knowledge to maximize the effectiveness and efficiency of its use in service to the user's goals.

Table 1. Design principles, implied usability heuristics and underlying collaboration requirements.

Design principles	Nielsen's heuristics	Collaboration requirements
DP1	2,7,5	(Terveen) Communication; Adaptation and Learning (Bratman) Commitment to joint activity; Mutual responsiveness (SharedPlans) Mutual belief of recipe
DP2	1,6,7,5	(Terveen) Shared Context; Determining goals; Communication; Planning, allocation of responsibility and coordination (Bratman) Commitment to joint activity; Mutual responsiveness (SharedPlans) Mutual belief of recipe
DP3	5,9,6,7	(Bratman) Commitment to joint activity; Commitment to mutual support (SharedPlans) Intention that the collaborators succeed
DP4	6,7,8,3,5	(Bratman) Mutual responsiveness (SharedPlans) Intention that the collaborators succeed

To illustrate our claim, we consider DP4. Relative to Nielsen's heuristics, a system that implements DP4 would enable recognition rather than recall and

would have a minimalist design, in order to make likely choices easy to reach and de-emphasize or remove the irrelevant ones. The shortcut property of heuristic #7 is manifested by the system enacting the obvious choice, while the user freedom property of heuristic #8 is preserved by allowing the user to undo the system's choice and follow up with her own. Easy access to the most relevant choices reduces the chances of the user selecting the wrong one (heuristic #5). What collaboration theory *adds* to this formulation is the system's responsibility to bring to bear all its knowledge of the users, tasks, organizational practices, and context of the interaction in order to produce the most useful choices and present or enact them in an effective way.

The set of the principles presented here is not intended to be exhaustive. User interface design for usability involves considerations of varying granularity: from general design of the interaction sequences to minor details of layout and style. In our analysis, we deliberately focused on the "big picture," highlighting the aspects of the interaction that we found particularly problematic for the users. However, the principles have broad applicability and demonstrate how the theory of collaboration can be used as a design guide to address design issues at various levels of granularity.

6 Conclusions

The human-computer collaboration paradigm employed here has been applied in several domains, with the implicit goal of creating software that is more effective, efficient and pleasant to work with and that behaves like a user's partner. However, the link between the collaborative properties of such systems and usability has not been formally addressed before this work. Quotes from actual ERP users presented in this paper highlight the relationship between poor usability and the collaborative weaknesses of a system. We have presented design principles for improving the usability of ERP systems, derived from collaboration theory and field studies, and outlined how they address the usability shortcomings in terms of Nielsen's usability heuristics and our field observations.

While some practices that implement collaborative behaviors exist in commonly used interfaces, developing new, effective methods for the particular properties of the ERP domain is part of our on-going investigations. We are currently developing algorithms and representations to support the design principles described here and are implementing artifacts to demonstrate the application of those principles.

References

1. Babaian, T., Lucas, W., Topi, H.: A data-driven design for deriving usability metrics. In: Proc. of ICSOFT-07. pp. 154–159 (2007)
2. Babaian, T., Lucas, W., Topi, H.: Visualizing the process: A graph-based approach to enhancing system-user knowledge sharing. In: Proc. of ICEIS-05. pp. 122–128 (2007)

3. Babaian, T., Grosz, B.J., Shieber, S.M.: A writer's collaborative assistant. In: Proc. of IUI-02. pp. 7–14. ACM Press (Jan 2002)
4. Bratman, M.E.: Shared cooperative activity. Philosophical Review 101(2), 327–341 (1992)
5. Breazeal, C., Hoffman, G., Lockerd, A.: Teaching and working with robots as a collaboration. In: Proc. of AAMAS-04. pp. 1030–1037. IEEE Computer Society (2004)
6. Cohen, P., Levesque, H.: Teamwork. Nôus 25, 487–512 (1991)
7. Cooprider, J., Topi, H.and Xu, J., Dias, M., Babaian, T., Lucas, W.: A collaboration model for ERP user-system interaction. In: Proc. of HICSS-10 (2010)
8. Dumas, J.S.: User-based evaluations. The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications pp. 1093–1117 (2003)
9. Dumas, J.S., Redish, J.C.: A Practical Guide to Usability Testing. Intellect, Ltd, Bristol, UK (1999)
10. Gould, J.D., Lewis, C.: Design for usability: Key principles and what designers think. Commun. ACM 28(3), 300–311 (1985)
11. Grosz, B.G.: Beyond mice and menus. Proceedings of the American Philosophical Society 149(4), 529–543 (12 2005)
12. Grosz, B.G., Kraus, S.: Collaborative plans for complex group action. Artificial Intelligence 86(2), 269–357 (1996)
13. Herbert, L.: Put business applications to the usability test. Forrester Research (2006)
14. Hestermann, C.: Key issues for enterprise resource planning. Gartner (2009)
15. Iansiti, M.: ERP end-user business productivity: A field study of SAP & Microsoft: Keystone strategy. http://download.microsoft.com/download/4/2/7/427edce8-351e-4e60-83d6-28bbf2f80d0b/KeystoneERPAssessmentWhitepaper.pdf (downloaded 12/21/2009) (2007)
16. ISO 9241-11: Ergonomics requirements for office work with visual display terminals, part 11 -guidance on usability. International Standards Organization (1998)
17. Kinny, D., Ljungberg, M., Rao, A.S., Sonenberg, E., Tidhar, G., Werner, E.: Planned team activity. In: Castelfranchi, C., Werner, E. (eds.) Artificial Social Systems (LNAI-830). Springer Verlag, Amsterdam, The Netherlands (1994)
18. Nielsen, J.: Enhancing the explanatory power of usability heuristics. In: CHI. pp. 152–158 (1994)
19. Norman, D.A.: Human-centered design considered harmful. Interactions 12(4), 14–19 (2005)
20. Polson, P.G., Lewis, C.H.: Theory-based design for easily learned interfaces. Hum.-Comput. Interact. 5(2), 191–220 (1990)
21. Rich, C., Sidner, C.L., Lesh, N.: Collagen: applying collaborative discourse theory to human-computer interaction. AI Mag. 22(4), 15–25 (2001)
22. Rosson, M.B., Carroll, J.M.: Usability engineering: scenario-based development of human-computer interaction. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2002)
23. Searle, J.R.: Collective intentions and actions. In: Cohen, P.R., Morgan, J., Pollack, M.E. (eds.) Intentions in Communication, pp. 401–415. MIT Press, Cambridge, MA (1990)
24. Terveen, L.G.: Overview of human-computer collaboration. Knowledge-Based Systems 8(2-3), 67–81 (1995)
25. Topi, H., Lucas, W., Babaian, T.: Identifying usability issues with an ERP implementation. In: Proc. of ICEIS-05. pp. 128–133 (2005)