# Reasoning for Intelligent System-User Interactions with Enterprise Resource Planning Systems*

**Wendy Lucas and Tamara Babaian**
Computer Information Systems Department
Bentley University
{wlucas, tbabaian}@bentley.edu

## Abstract

Enterprise Resource Planning (ERP) systems are recognized as having unintuitive interfaces and being difficult to use. At least part of the reason for these problems comes from the complexity of the processes they support and the need for generic interfaces that can be adapted for use by a wide range of industries. We examine here the application of usage logs and AI planning to strengthen the ability of the system to act as a collaborative partner providing individualized support to its users, particularly in resolving errors. This work is part of a larger project investigating if the usability of ERP systems can be improved by adopting human-computer collaboration as a paradigm for system design.

## 1 Introduction

A system's ability to learn and reason about its users, the tasks they perform, and their system-related goals is essential for optimizing human-computer interactions. While it is generally difficult to effectively and efficiently enable this ability, introducing learning and reasoning to enterprise-wide systems presents its own unique challenges. These systems implement business processes that involve a vast number of users working on a multitude of interrelated tasks that can span extended time periods. Despite user training and ongoing usage, the system's logic remains in large part opaque to even the most experienced users due to the broad scope and inherent complexity of these processes. Even experienced users are typically unaware of the relationships and interdependencies between tasks and the semantics of individual input fields, forms, and interface options. This significantly impedes the ability of all users to make optimal use of the system, especially when it comes to making sense of and resolving errors. Our focus here is on applying learning and reasoning to support system-user interaction and, in particular, error handling.

*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.

The work we describe here is part of our on-going project to study if the usability of Enterprise Resource Planning (ERP) systems can be improved by adopting human-computer collaboration [Terveen, 1995; Grosz, 2005; Shieber, 1996] as a paradigm for system design. This perspective shifts the view of the system from being a mere repository of data and process interfaces towards being more of a knowledgeable partner, thus achieving a greater degree of collaboration between the system and its users. To provide stronger collaborative support, the system must be aware of the users and their actions, the tasks and processes, and its own capabilities so that it can reason about itself as well as its users. Our approach is to embed this knowledge into the system via a model that captures the existing structure and semantic relationships between interface components and usage history. Some initial results of this work have been reported in [Babaian *et al.*, 2007].

Existing studies of enterprise system users have shown the benefits of providing them with prerecorded recipes for different tasks (e.g. [Leshed *et al.*, 2008]). Such recordings can be created manually (e.g. [Eisenstein and Rich, 2002]) or in an automated fashion (e.g. [Little *et al.*, 2007]). Our aim is to provide recipe-based guidance for users within a single system. As we propose in this paper, some of these recipes can be derived by mining usage logs [Linton and Schaefer, 2000] while others can be created dynamically by the system by means of automated planning [Babaian *et al.*, 2002]. Usage log mining will leverage data on frequently performed, ordinary practices; planning will create recipes to support infrequent situations by employing explicit reasoning.

Usage data can be used for a variety of purposes, including workflow mining, estimating a user's experience level, and usability assessments of various system features. By embedding an automated usage logging component into an ERP system, that data can also be used for dynamically adjusting the interface to better meet the needs of the individual users. The model we employ links usage log data to particular users, interface components, tasks, and processes. This makes it possible for users to specify in a transparent way which usage histories are relevant for them.

Automated planning can be used when usage histories are either not available or not easily identifiable, as, for example, in error situations. The system can provide active support to its users by dynamically identifying a plan to invoke particu-

- *Domain data* stores the enterprise data routinely maintained by existing enterprise systems.
- *User* represents the user information.
- *UI Concept* includes entities representing the user interface components, their grouping on interface pages, and their relationships to the domain data.
- *Task Concept* represents tasks as named collections of interface pages.
- *Business Process* represents the system-defined composition of tasks into business processes.
- *Logging* contains the entities that record the dynamically collected data, capturing all occurring system-user interactions on the key-press (or mouse click) level along with the instances of enacted task interfaces. The usage logs specify which tasks and input controls were enacted and when, what data was entered, and which user entered it.

Figure 1: Components of the system data model used to support usage logging and subsequent analysis of logs.

lar interface components that are necessary for the diagnosis and/or resolution of an error. Planning can also leverage the information from the usage logs to provide contextual and historical information.

The next section of this paper describes usage logging and its application to supporting users in error situations and basic system usage. Section 3 presents examples of how planning can be applied in the context of automated preference elicitation and error handling. We conclude with a discussion of the benefits and challenges inherent to this work.

## 2 Usage logs

Usage logs provide fodder for analyzing user performance [John and Kieras, 1996], visualizing usage data [Al-Qaimar and McRostie, 2006], and workflow discovery [Shen *et al.*, 2009; Greco *et al.*, 2005], to name but a few applications. The granularity of the data that is captured determines how useful the logs can be for each of these purposes. As Ivory and Hearst (2001) point out, dealing with click-level interactions makes it difficult to relate the captured events with particular tasks and parts of an interface. Our approach overcomes this challenge by using our model of processes-tasks-interfaces-users-usage as a framework through which data can be viewed (see Figure 1). This model allows the system to analyze usage data at various levels of granularity: from a single user to a user group, from a single button in an interface to commonly accessed sequences of tasks or error-prone parts of a process.

Using this embedded model strengthens human-computer interaction by enabling the computer to reason about the context of each interaction. The system can relate the current interaction to prior relevant usage history. Relevance can be identified by the system, potentially with the help of the user, based on the recorded contextual information about tasks, users, and processes.

One way in which we have applied usage logs from our model is for automatically deriving usability data [Babaian *et al.*, 2007]. Measurements include the time it takes to complete a task, a series of tasks, or a business process; the time spent on a session and the number of tasks performed within that session; the sequence of actions the user follows for completing a task; and the number of corrections at the keystroke level a user makes when entering a data value.

While ERP systems are designed with a predetermined workflow in mind, actual usage often varies from the prescribed process. Knowledge gleaned from prior use of system interfaces about the order in which users typically perform tasks associated with a process, the number of times they have performed those tasks, which fields they fill in, and the data choices for those fields can also be used to guide and support the user, as described next.

### 2.1 Usage Logs as a Repository of Organizational and Individual Practices

ERP systems are designed to satisfy the needs of every kind of organization. As a result, the default interfaces are very generic: they present a multitude of forms, fields, and selection options within every page or screen; search interfaces for selecting possible values for input fields provide users with an overwhelming number of choices. A large proportion of input fields are in fact optional, but in the initial stages of system use, users are stymied trying to figure out which fields are required and which are not (based on their organization's practices). While ERP systems do typically include customization options for form design that allow the elimination of unused fields and the streamlining of interfaces, this customization is perceived as costly, and certainly too costly to be done for each user. Users also have customization options that they can set themselves, but even experienced users have trouble finding these options, let alone setting them. Thus, providing automated usage-based mechanisms for identifying required fields and reconstructing completed processes are promising opportunities for providing better support to users.

In field studies of ERP system users [Topi *et al.*, 2005], subjects noted that the sheer number of fields in each form is rather intimidating, especially for inexperienced users. Even those users who described themselves as "living" in the system admitted that, when performing tasks that are done on an infrequent basis, such as yearly reports, they typically need to consult in-house documentation detailing the sequence of interfaces and the particular fields that need to be filled in.

Data from the usage log can be used to remind a user about how a process was completed any of the prior times that she worked on it. It is possible to dynamically reconstruct the procedure that was followed (i.e., the recipe), the values that were entered, and the output that was produced. This reconstructed process can be "replayed" for the user.

The system can also determine how frequently the user accesses a particular interface and adjust the level of instructional/error support to the appropriate level. While this type of assessment could not be accomplished quickly with typical click-level data, ERP systems require their users to identify the tasks on which they are working via a selection mechanism; capturing this information with our model therefore

makes task recognition trivial.

For a new user in particular, it is helpful to see the fields that were filled in by others working on the same tasks. The usage logs can easily provide this information: those fields that have been used in the past can be highlighted and even color-coded based on the amount of usage. Allowing the user to control the list of individuals whose usage histories the mining should be based on personalizes the results to the user's own role and associations in the organization. This approach allows the system and the user to each contribute to this collaborative effort in ways most reflective of their respective knowledge.

# 3 Using AI planning to enhance interactions

Automated planning has been successfully used to embed reasoning about actions (e.g. [Babaian *et al.*, 2002; Barish *et al.*, 2000]), enabling the system's robust and autonomous operation in support of its users' goals. An example of this is the Writer's Aid system [Babaian *et al.*, 2002], which works in parallel with the user of a text editor by identifying and downloading bibliographic records and citations matching user-specified keywords. The use of reasoning and planning in Writer's Aid is isolated to just one type of activity: bibliographic search. A partial order planning algorithm is used, which interleaves planning with execution of partial plans for information gathering from the bibliographic data repositories and, sometimes, from the users themselves.

In this project, we are relying on the same planning techniques that were employed in Writer's Aid. However, in the ERP domain, we aim to expand the use of automated planning to reasoning about a much broader variety of the system's interface functions. By modeling various interface components as actions with preconditions and effects, we give the system the ability to reason about its own actions. This enables the planning capability to be used for enhancing and directing system-user interactions in a variety of ways, including automated preference elicitation and support in error situations.

The PSIPLAN formalism for planning with correct but incomplete information about the world, used in Writer's Aid, distinguishes sensing actions (i.e., those that result in previously unknown information being added to the state of knowledge) from domain actions (i.e., those that alter the world state). The ability of the planner to function with only incomplete information is essential because, in order to provide effective support with minimal overhead to the user, the planner should be able to operate with little to no initial information.

In the ERP-interface domain described here, the initial state of knowledge about the world may contain little or no knowledge at all. Information is collected as needed via execution of sensing actions: by querying the user or the usage logs. Information obtained from such querying is then added to the state of knowledge and recorded for future use. The planner's ability to avoid redundancy in sensing is also essential when the sensing actions involve querying the user to avoid repetitive questioning.

## 3.1 Automated preference elicitation

As described in [Babaian, 2002], by modeling system actions with preconditions requiring knowledge of user preferences, we can implement preference elicitation as a means for satisfying those preconditions.

For example, consider one of the real life issues brought to our attention by an ERP system user. He pointed out that the list of selections for country codes always includes a complete list of all countries, even though only five are relevant for his company. While limiting the list is definitely within the scope of customizable options, this was not something that the IT department had chosen to do. Given the tens of thousands of database tables supplying generic data for use by any company, this type of customization is just too costly. Automated preference elicitation is ideal for use in these cases.

Continuing with our example, suppose that the interface function of displaying a drop-down box with country codes was tagged with a precondition of knowing the subset of codes that a particular user had chosen to be displayed:

$$g = \forall x.KW(DisplayCountry(x, ?user))$$

where $KW(P)$ denotes *knowing whether* $P$ is true or false. Variables identified with a $?$ are assumed to be existentially quantified and instantiated at the time of execution. Thus, the precondition above represents knowing whether $x$ is a country code preferred by the specified user, for all values of $x$. (Here and throughout, for the sake of brevity, we outline the ideas and do not present the details of the planning representations and reasoning algorithms. For more detailed descriptions, see [Babaian *et al.*, 2002].)

Upon the user invoking the drop-down box for the country code, the reasoner establishes whether the precondition $g$ is satisfied, given it's state of knowledge. If the precondition is not satisfied, the reasoner will search for a plan to achieve it. Below are some plausible choices, all of which achieve goal $g$ and can be presented to the user (we assume that the interface and action descriptions presented below are available to the planner):

- An interface through which the user can specify those country codes to be displayed whenever a country code is requested.

  $Action$ : USelectCountries(?user)
  $Precondition$ : $none$
  *Effect* :$\forall x.KW(DisplayCountry(x, ?user))$

- An action that consists of adopting the list of preferred countries that have been selected by some other user. This action has a precondition that the preferred countries of the user-source for this information is known.

  $Action$ :UAdoptOthersPref(?user, ?source)
  $Precondition$ :$\forall x.KW(DisplayCountry(x, ?source))$
  *Effect* :$\forall x.KW(DisplayCountry(x, ?user))$

- A two-action sequence that utilizes the usage logs for finding those country codes that this particular user has chosen before (action UsageLogCountries(?user)) and

asking if he would like that set of countries to be displayed (UConfirmCountries(?user)).

$Action$ :UsageLogCountries(?user)
$Precondition$ :$none$
$Effect$ :$\forall x.KW(EnterCountry(x, ?user))$

$Action$ :UConfirmCountries(?user)
$Precondition$ :$\forall x.KW(EnterCountry(x, ?user))$
$Effect$ :$\forall x.KW(DisplayCountry(x, ?user))$

An alternative to the above would be the commonly used adaptation of basing the content of the drop-down box on the codes selected most recently by the user. The planning-based mechanism is more explicit and, therefore, more transparent to the user. Although it requires the user's direct involvement with the specification, such involvement, when occurring within the context of a relevant task, is more likely to lead to a positive outcome.

### 3.2 Support in error situations

ERP systems offer limited help to users in error situations, with generic error messages providing little if any information on why a value entered by the user is incorrect within the context of the current interaction. A system that can reason about its own actions, including those actions that require user input, can create a plan or plans for proactively dealing with potential error situations.

Consider the task of entering line items into a purchase requisition form. For simplification purposes, let's assume that each line in the form specifies the material to be purchased, the quantity, and the plant for which it is being procured. Before a line item can be recorded in the database, it must pass a validation condition that can be formulated as a conjunction of the following propositions, denoted by $G$:

$InputValue(MaterialField, ?x)$,
$InputValue(PlantField, ?y)$,
$Material(?x), Plant(?y), PlantUseMaterial(?y, ?x)$.

$InputValue(?f, ?v)$ designates that input field $?f$ contains value $?v$, $Material(?x)$ designates that $?x$ is a valid material code, $Plant(?y)$ denotes $?y$ is a valid plant code, and $PlantUseMaterial(?y, ?x)$ reflects that material $?x$ is used by plant $?y$.

The first two conditions can be verified by inspecting the input form, while the last three can be verified by checking the appropriate database tables.

Suppose the user has entered the code 55 into the material field and $PL1$ into the plant field, thus binding $?x$ to 55 and $?y$ to $PL1$. Suppose further that material code 55 does not match any records in the material master list. There are two possible explanations for this input: either this is a case of user error, or 55 is a new code for a new part that is not yet in the master list. The reasoning engine establishes that the two propositions of the above goal $G$ that are left unsatisfied are $Material(55)$ and $PlantUseMaterial(PL1, 55)$. In support of goal $G$, the planner identifies the following sequences of actions and offers them to the user, invoking the appropriate interfaces in response to the user's selection.

1. UAddMaterial(55), AddPlantUseMaterial(PL1, 55) - A two-action plan in which the user enters a new material record under code 55 and the system then (autonomously or with the help of the user) specifies plant $PL1$ as using material number 55. We do not present the detailed descriptions of preconditions and effects of these actions here, but note that the combined effect of this plan includes both propositions $Material(55)$ and $PlantUseMaterial(PL1, 55)$ being true. Thus, this plan will render the goal satisfied. Note, however, that this course of action is only appropriate if the user did indeed intend to use a new material that had not yet been added to the master list.
2. UPickMaterialForPlant(PL1, MaterialField) - An action of displaying a selection of all possible material codes identified as being used by plant PL1, letting the user select one such value (identified here as $?z$), and transferring the selected value into the MaterialField input field. This action will have the effect of $Material(?z)$, $PlantUseMaterial(PL1, ?z)$, and $InputValue(MaterialField, ?z)$ where the value of $?z$ will be identified at runtime. This plan, if executed, will provably establish the required goal.
3. UPickMaterial(MaterialField) - An action of displaying a selection of *all* possible material codes and their descriptions in a dialog box, letting the user select one of them, and transferring the selected value into the MaterialField input field. This action will have the effect of $Material(?z)$ and $InputValue(MaterialField, ?z)$, where the value of $?z$ will be identified at runtime. This plan does not provably establish $PlantUseMaterial(PL1, ?x)$, but it does not guarantee its negation either, so it can potentially achieve the goal. We call such plans *hypothetical*.
4. Lastly, the plan that describes the typical behavior of existing systems is UEdit(MaterialField), which is another hypothetical plan that involves the user editing the value of the $MaterialField$. This potentially achieves $Material(?z)$ and $InputValue(MaterialField, ?z)$.

Of the above plans, only 1 and 2 provably achieve the goal. The other two plans may or may not lead to goal achievement, so may cause replanning to occur. Taking a closer look, we note that only plan 1 does not require the user to change the value entered for the material code. However, it is more likely that the user made a mistake entering that code and would be better off pursuing plans 2-4. Rank-ordering the possible plans is a challenge, particularly in cases with many alternatives, and different ordering criteria need to be investigated.

## 4 Discussion

We have presented an approach to enhancing system-user interactions in ERP systems that is based on usage logs and automated planning. In choosing these methods, we aim to address the following overarching concerns regarding adding intelligence to interactions:

1. *The unobtrusive nature of these enhancements and the associated learning.* Usage logs are collected without

any overhead on the user's part. The structured nature of ERP interfaces presents the advantage of being able to easily identify the users and tasks involved in system-user interactions. This data can be leveraged for a variety purposes.

Furthermore, customization is automatically triggered within the context of the task being performed. That customization can draw on usage data from one user or a group of users.

2. *The accuracy of the system's reasoning and its transparency to the user.*

   Usage histories, actions, and plans are concepts that humans employ naturally. Thus, system enhancements that are based on the use of these tools can be more easily explained and controlled by the user than more obscure learning methodologies.

   In order to earn the user's trust, the methods that employ intelligence should provide effective assistance, which requires precision in reasoning and the wise use of the system's and the user's resources. The planning algorithm should be sound, reasonably complete, and non-redundant, especially when it comes to information gathering, in order to competently reason about the user's preferences, usage history, and solutions to error situations. PSIPLAN has these characteristics.

We have developed an experimental prototype that incorporates our data model for usage logging. This prototype implements an interface for the purchase requisition process and has been used for investigating how collaborative principles can be applied to ERP interface design. We are in the process of embedding the planner and the support mechanisms examined in this paper for resolving error situations and providing more guidance to users.

One of the challenges to this work is scalability, which is a challenge to both usage-log-based reasoning and planning with large action sets. However, both the data and action sets can be partitioned. For example, smaller subsets can be based on user roles or groups of related tasks.

While our prototype is useful for experimental studies of our design interventions, it can only be used as a proof-of-concept to a limited extent. We cannot replicate the scope and complexity of ERP systems, which arise largely from the vast number of processes and users being supported. To truly test the impact of our recommended approaches would require implementation at the system level.

## References


[Al-Qaimar and McRostie, 2006] Ghassan Al-Qaimar and Darren McRostie. Evaluating user performance using KALDI: A computer-aided usability engineering tool. In *Proceedings of the IASTED Conf. on Software Engineering*, pages 242–251, 2006.

[Babaian *et al.*, 2002] Tamara Babaian, Barbara J. Grosz, and Stuart M. Shieber. A writer's collaborative assistant. In *Proceedings of Intelligent User Interfaces Conference (IUI-02)*, pages 7–14. ACM Press, January 2002.

[Babaian *et al.*, 2007] Tamara Babaian, Wendy T. Lucas, and Heikki Topi. A data-driven design for deriving usability metrics. In *ICSOFT (ISDM/EHST/DC)*, pages 154–159, 2007.

[Babaian, 2002] T. Babaian. Learning user preferences by satisfying knowledge goals. In *Personalized Agents. Papers from 2002 AAAI Fall Symposium*, pages 1–5, North Falmouth MA, November 2002. AAAI Press.

[Barish *et al.*, 2000] Greg Barish, Craig A. Knoblock, Yi-Shin Chen, Steven Minton, Andrew Philpot, and Cyrus Shahabi. The theaterloc virtual application. In *AAAI/IAAI*, pages 980–987, 2000.

[Eisenstein and Rich, 2002] Jacob Eisenstein and Charles Rich. Agents and guis from task models. In *Proceedings of Intelligent User Interfaces Conference (IUI-02)*, pages 47–54, 2002.

[Greco *et al.*, 2005] Gianluigi Greco, Antonella Guzzo, Giuseppe Manco, and Domenico Saccà. Mining and reasoning on workflows. *IEEE Trans. Knowl. Data Eng.*, 17(4):519–534, 2005.

[Grosz, 2005] B. G. Grosz. Beyond mice and menus. *Proceedings of the American Philosophical Society*, 149(4):529–543, December 2005.

[Ivory and Hearst, 2001] Melody Y. Ivory and Marti A Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv.*, 33(4):470–516, 2001.

[John and Kieras, 1996] Bonnie E. John and David E. Kieras. The goms family of user interface analysis techniques: comparison and contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4):320–351, 1996.

[Leshed *et al.*, 2008] Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. Coscripter: automating & sharing how-to knowledge in the enterprise. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1719–1728, New York, NY, USA, 2008. ACM.

[Linton and Schaefer, 2000] Frank Linton and Hans-Peter Schaefer. Recommender systems for learning: Building user and expert models through long-term observation of application use. *User Modeling and User-Adapted Interaction*, 10(2-3):181–208, 2000.

[Little *et al.*, 2007] Greg Little, Tessa A. Lau, Allen Cypher, James Lin, Eben M. Haber, and Eser Kandogan. Koala: capture, share, automate, personalize business processes on the web. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 943–946, New York, NY, USA, 2007. ACM.

[Shen *et al.*, 2009] Jianqiang Shen, Erin Fitzhenry, and Thomas G. Dietterich. Discovering frequent work procedures from resource connections. In *IUI*, pages 277–286, 2009.

[Shieber, 1996] Stuart Shieber. A call for collaborative interfaces. *ACM Computing Surveys*, 28A (electronic) Available at

http://www.acm.org/pubs/citations/journals/surveys/1996-28-4es/a143-shieber/, 1996.

[Terveen, 1995] Loren G. Terveen. Overview of human-computer collaboration. *Knowl.-Based Syst.*, 8(2-3):67–81, 1995.

[Topi *et al.*, 2005] Heikki Topi, Wendy T. Lucas, and Tamara Babaian. Identifying usability issues with an erp implementation. In *ICEIS*, pages 128–133, 2005.