VIO: a mixed-initiative approach to learning and automating procedural update tasks

John Zimmerman, Anthony Tomasic, Isaac Simmons,
Ian Hargraves, Ken Mohnkern, Jason Cornwell, Robert Martin McGuire
Carnegie Mellon University
{johnz, tomasic}@cs.cmu.edu

ABSTRACT
Today many workers spend too much of their time translating their co-workers' requests into structures that information systems can understand. This paper presents the novel interaction design and evaluation of VIO, an agent that helps workers translate request. VIO monitors requests and makes suggestions to speed up the translation. VIO allows users to quickly correct agent errors. These corrections are used to improve agent performance as it learns to automate work. Our evaluations demonstrate that this type of agent can significantly reduce task completion time, freeing workers from mundane tasks.

Author Keywords
agents, interaction design, mixed initiative.

ACM Classification Keywords
H.5.2 User Interfaces: interaction design

INTRODUCTION
Today many workers in companies spend time translating requests into language and structures that information systems can understand. Consider the task of transferring a student from a waitlist to a course. The *requester*, a professor, has an intent that matches a common work task. The professor expresses her intent in an email to the department coordinator with relevant information such as the student's and course's names. The coordinator then logs in to the appropriate systems and makes the changes, translating the request into information the system can understand.

Organizations address translation tasks by assigning a *human-service-agent*, such as administrative assistants, webmasters, network administrators, purchasers, etc., who perform procedural translation tasks on behalf of coworkers or customers. Procedural translation tasks are good candidates for automation because the input is easily captured, the output is structured, and the tasks are repeatedly executed. In order to study this opportunity we have developed a machine-learning-based agent and mixed-initiative interface. Called VIO, our agent takes on the role of a webmaster's assistant (Figure 1). Requesters email requests (i.e., updates for a website) to the webmaster using natural language. VIO preprocesses the requests and pre-fills website update forms with suggestions. These pre-filled forms are presented to the webmaster for approval. The forms are an augmentation of a traditional direct-manipulation interface that allow the webmaster to quickly recognize the task and repair mistakes made by VIO. Our interaction design focuses on making repairs easy because (i) we accept that agents make errors, and (ii) having an interface that lets webmasters correct errors by doing their regular work and without generating additional work allows VIO to be deployed with little or no training. Through the process of repairing and approving forms, webmasters provide training data, allowing VIO to "learn in the wild," that is, directly from use. This frees the webmaster to then focus on non-procedural tasks that require more human skill.

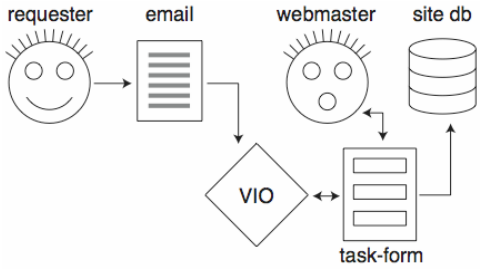

Figure 1. Webmaster repairs and approves the task form causing the web database to update and VIO to learn from the addition of a new training example.

Casting VIO as a webmaster's assistant is a first step to concretely test our ideas. However, the design principles of VIO generalize to a much larger set of procedural tasks found within organizations.

The design of VIO raises several fundamental HCI research questions including: (1) How effective is a human-service-agent collaborating with an agent that has had little training compared to a traditional direct manipulation interface? (2) How effective is VIO if it performs perfectly? (3) How do VIO's errors impact overall performance? In this paper we address all these questions. We begin by describing our novel interaction method that combines natural language interaction—in this case the preprocessing of incoming email requests—with existing direct manipulation tools, and a feedback loop to the machine-learning algorithms. In

CHI 2007, April 28–May 3, 2007, San Jose, California, USA.

addition, the paper describes two evaluations. The first is an empirical study where collaboration with VIO reduced task completion time by 17%. The second is a Keystroke-Level Model [8] analysis that provides more detail on the benefits provided by VIO. In addition, these evaluations provide insights into the cost of different types of agent errors.

RELATED WORK

Some researchers in the Human-Computer Interaction (HCI) community have championed the benefits of automation while others focus on the power of direct manipulation [16]. While the work on embodied agents has yet to demonstrate significant effects, the implementation of automation via machine learning in underlying systems has become core to both HCI research and development. Our system tries to find a happy medium in the tension between full automation and direct manipulation.

Our presentation of the request in terms of a form builds on Malone et al.'s work on Information Lens, where they found that semi-structured messages improve communication and coordination activities [13]. In discussing future work they predict a graceful degradation between automatically and manually handled tasks. Our interaction design follows this principle, allowing users to experience the graceful *improvement* of VIO.

Previous research has explored the use of natural language processing to automatically fill out forms [14, 17]. Additionally research prototypes have been built that convert email requests into website updates [12]. However, these systems have not presented agents that perform perfectly, nor have they addressed how users handle agent errors. In fact, previous work in this area presents no evidence of increased productivity through collaboration with an agent.

Out of the tension between automation and direct manipulation, the mixed-initiative community has arisen. They focus on exploring how humans and AI systems can more effectively collaborate to solve complex tasks. Hearst captures the essence of this when she notes that AI researchers want to model humans in order to make computers that act intelligently, while HCI researchers want to leverage computing to make it easier for users to make intelligent decisions [6]. The goal of the mixed-initiative community is to develop intelligent systems that allow for collaborative interaction to solve problems [6]. While most mixed-initiative systems have focused on solving complex planning tasks such as generating train schedules [4] we focus on how agents can free users from mundane tasks.

One mixed-initiative project that has focused on mundane tasks is the LookOut [7] system, which assists users with adding meetings to calendars. Like VIO, the system processes incoming email and attempts to fill a form, in this case a calendar event. While LookOut at a high level is quite similar, our design offers some advances. Our system has been designed to be domain independent, where LookOut only addresses meetings. Our system handles add, delete, and modify tasks, while LookOut only addresses add tasks. LookOut interrupts the user and focuses on assessing whether a message warrants an interruption [5]. Our design instead follows an interaction model set by Letizia [10], where the assistance falls to the side of the locus of attention, allowing the user to easily ignore the assistance but to also benefit from it by having it close. Finally, our system uses a deeper learning model that provides more effective help over time.

The Eager system [2], an early example of programming by demonstration, detected simple tasks that a user repeated twice during an interaction. The system then used highlighting (as we do) to draw the user's attention to extracted information. Eager would use the extracted information to construct a repetitive task for the user. Eager, however, has no feedback mechanism for repairing errors.

Previous research on email use claims that email is a "habitat" where much project-management work is done [3]. Bellotti et al. extended this, designing a system that assists users with completing tasks that arrive via email [1]. They developed a "thrask" (tasks and threads) interface to help users complete tasks that involve multiple email exchanges. Their system does not employ machine learning, but instead addresses the presentation of emails. We view our system as complementary in that VIO provides automation of mundane tasks so users can address more complex threaded tasks.

Finally, many commercial products exist to help organizations manage and automate their workflows. Business process automation software such as Remedy [15] provides an abstraction layer between the application logic and the business practice to be automated. This layer allows businesses to connect different systems together, but also creates environments where many workers must perform the form filling, translation tasks.

In summary, our system (i) integrates theories from mixed-initiative computing and natural-language research into a working system that has been demonstrated to improve human performance, (ii) advances the design of human-agent interfaces by addressing the issues of agent error and learning, and (iii) extends the research done on blending email applications with tasks and project-management.

DESIGN

The interaction design of VIO *embraces* the fact that agents make errors. Instead of investing huge engineering efforts in an attempt to build perfect agents, our design allows an agent with little or no training to observe a task and begin making suggestions. The interaction design allows users to repair agent errors without increasing the work they would have done without VIO. As VIO learns, the interaction style allows its suggestions to significantly reduce task completion time.

Figure 2 illustrates VIO's functional architecture. A requester initiates a task by sending an email request to the

webmaster that gets routed to the analysis module. VIO modifies the incoming email by adding a ranked list of likely tasks followed by a structured list of all possible tasks (Figure 3). The webmaster reviews the email and selects the appropriate task-form link. This transitions the webmaster to a task-form showing all of the elements available for update with as many fields as possible completed (Figure 4). The webmaster then adds any missing information and repairs any information VIO extracted incorrectly. She then submits the form, causing the execution module to execute a database transaction, and by extension, update the website. The results of the interaction are then forwarded to the learning module, which analyzes the entire interaction and improves VIO's performance.

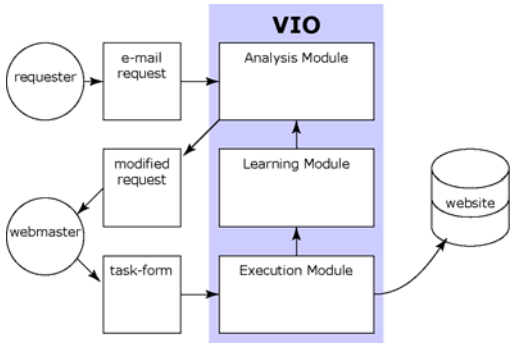

Figure 2. Functional architecture of VIO.

Figure 3 shows an incoming email request VIO has modified with a ranked and structured list of tasks. An earlier design presented a completed form based on VIO's best guess instead of a list of tasks. However, in pilot testing participants had difficulty recognizing when VIO had selected the wrong form. To address this problem we borrowed a technique from information retrieval and used a ranked list. This approach is a form of "soft fail" [11]: an agent error that still moves the user closer to their goal.

In an earlier prototype the ranked list showed only the top three items; however, pilot testing revealed problems with this interaction. When participants encountered a list where the appropriate task did not appear as one of the top three items, they hesitated before making a selection from the structured list. Participants spent time re-examining the incoming email, apparently worried that they had not understood the requester's intent. The current design addresses this hesitation by using a threshold value to determine the length of the ranked list. When the VIO has high confidence in a single form, it lists only one item like in the example above (Figure 3). When VIO has high confidence in several forms, it lists several forms. When VIO has low confidence for all forms, it makes no suggestion and only the structured list appears.

Figure 4 displays a Modify Person task-form. An earlier design placed the message text at the top of the task-form to communicate that users should start at the top and work down to the update button. In piloting, however, participants spent a great deal of time scrolling so they could copy and paste text between the email and the form elements. The current task-form layout addresses this issue by placing the source message on the left. This arrangement reduces scrolling and makes comparisons between the email and the form elements much easier. The form elements use background color to help communicate actions. Elements updated by VIO have an orange background, elements updated by the human-service-agent have a blue background, and elements that have not changed have a white background. Additionally, an orange highlight in the email helps users see items that have been extracted. In pilot studies VIO would occasionally make incomplete extractions that participants failed to see. In one example, the incoming email asked the webmaster to add "School of Information Sciences, University of Portlandia" as an affiliated organization. However, VIO extracted "Information Sciences University" as the organization. Highlighting the extracted text in the email allows users to more easily notice partial extractions.

VIO MACHINE LEARNING

In this section we briefly summarize the machine learning of VIO. For more on the machine learning of VIO, please see [19]. VIO addresses three interaction problems via machine learning: form ranking, entity-instance selection (i.e., selecting the record for a form), and field completion.

For form ranking, VIO logs the form selected for each message. The history of selections is used as a label to the log of messages. Messages are represented as a bag of words. A k-way boosted decision tree algorithm is trained to generate the model for form ranking. For the experiment described in the evaluation background section, this model has a mean reciprocal rank above .90.

For entity-instance selection, VIO logs the records selected for forms. The record selected is the label for (conceptually) a data set represented as the *difference* between the records and the words in the message. A Naïve Bayes classifier is trained to generate the model for entity instance selection. This model has the mean reciprocal rank above .85.

For form-field completion, each field value v completed by the user in a form is logged. A domestication algorithm searches the message for a string s similar to v and declares s a label for the field. The labels are used for a corpus of messages represented as strings. A conditional random field algorithm is trained to generate an extraction model. Extraction model entity F1 performance ranges from 0.0 to 1.0. Extraction performance depends most strongly on the number of labels for training and the length of v.

To summarize, during use the user is assumed to select the appropriate form, to select the appropriate target instance of the form, and to complete the appropriate fields for a form. VIO used the log of this interaction to label processed messages and train machine-learning models for form selection, instance selection, and form-field pre-filling. The resulting models are accurate, but imperfect, predictions of user interactions.

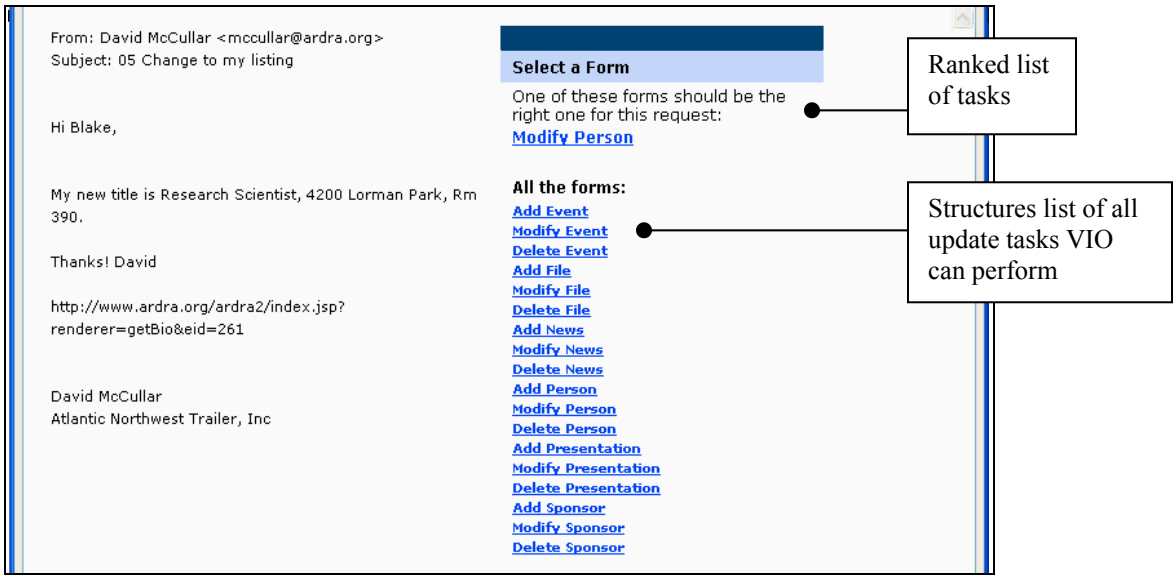

Figure 3. Email augmented with a ranked list of tasks and a structured list of all tasks. Clicking on the task links transitions the user from the augmented email to the task-form displayed in a web browser.

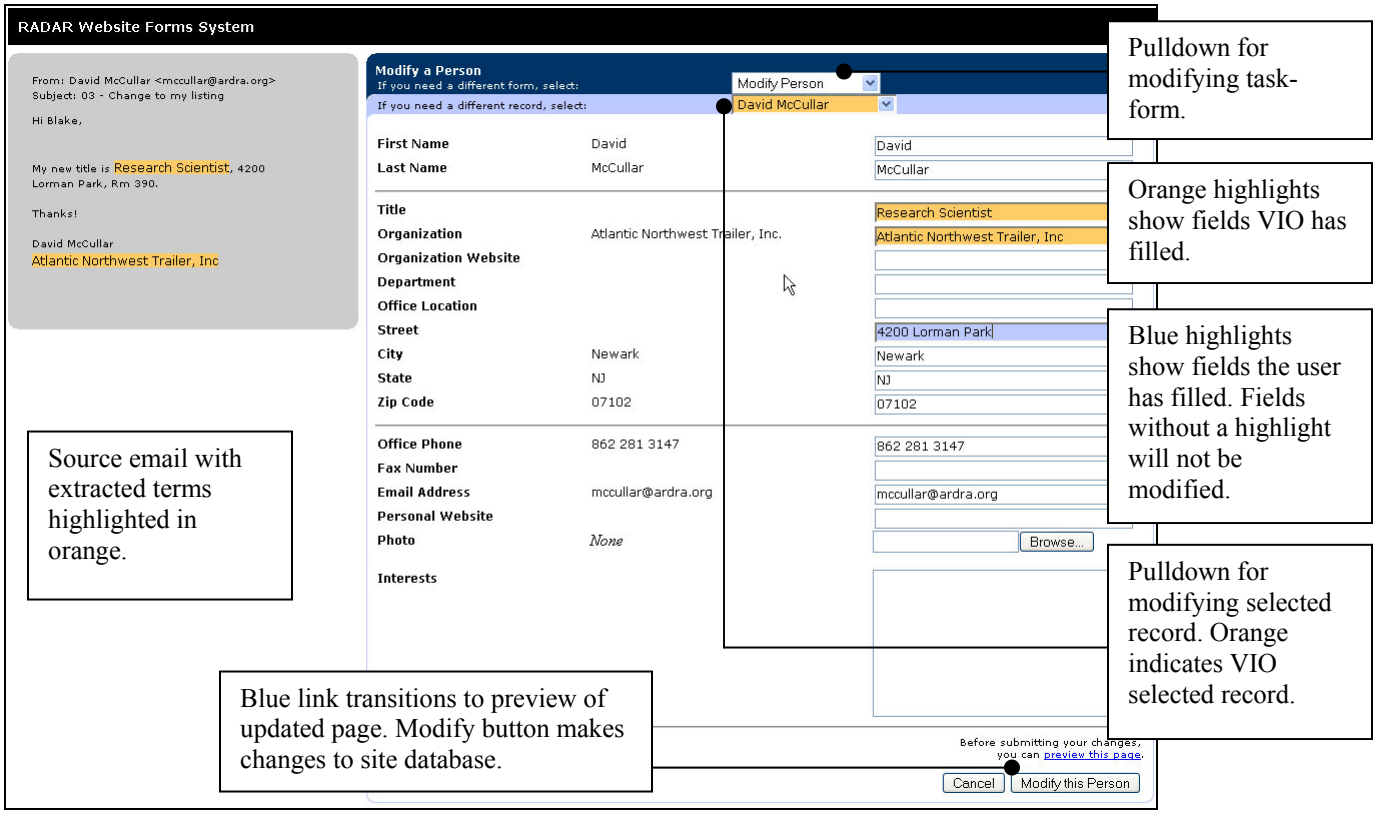

Figure 4. Task form where webmaster repairs any VIO errors and completes the request. Completed requests get added as training examples, improving VIO performance.

EVALUATION

In order to understand the impact of VIO we conducted two evaluations. An empirical study compared the performance of webmasters interacting with VIO against webmasters interacting with a traditional content-management system (CMS). A Keystroke-Level Modeling (KLM) analysis compared how skilled users would perform with VIO as it goes from knowing nothing to performing perfectly

Evaluation Background

In order to ground the study in the real world, we acquired a corpus of email messages from a professional webmaster. To preserve privacy, all email messages were anonymized

by replacing references to names, organizations, URLs, and personal data. In addition, we removed threaded discussions and split multiple requests into separate messages. During this procedure, we were careful to preserve the original language. This procedure produced a corpus of 228 messages sorted in roughly chronological order. The first 200 messages were used to train VIO. The remaining 28 messages were used for the evaluation.

During training, the appropriate form was selected, the appropriate target instance of the form was selected, and the appropriate fields were completed for each of the 200 messages. VIO used the log of this interaction to label processed messages and train machine-learning models for form selection, instance selection (reference resolution), and form field pre-filling. Note that in many domains human-service-agents process 200 requests every few days, so the VIO receives only a modest amount of training.

Empirical Evaluation

For the empirical evaluation of the interaction, we selected one hypothesis to test: Does interacting with VIO significantly reduce the amount of time needed to complete website update requests? In addition, we wanted to investigate (i) how errors committed by VIO impacted the participants' performance speed, (ii) how errors made by VIO impacted errors participants committed to the database, and (iii) general usability issues around the VIO interface.

Method

Participants played the role of the webmaster in one of two conditions: CMS and VIO. In the CMS condition participants completed tasks using a web interface designed to look like a traditional content management system (CMS). In the VIO condition participants completed tasks using the *same* interface, augmented with VIO's suggestions for form selection, instance selection, and pre-filled field values.

Participants first received instructions from the experimenter. The experimenter walked participants through an example task to quickly familiarize them with the interface. Next, participants completed 20 practice tasks. The goal of this extensive practice was to train the participants to work more like skilled users. While working on these tasks, participants could ask questions of the experimenter. Following the practice questions, participants completed the 8 evaluation tasks (Table 1). The number of tasks was kept small due to the amount of time spent on training. Following the completion of the final task, participants answered a survey on their perception of usability and were graded using a grading script that compared the final database state with the correct state.

Participants in the VIO condition first viewed an incoming email to the webmaster that had been modified with a prioritized and structured list of task-forms. Participants selected one of these, transitioning them from the email client to a web browser that displayed the task-form. In the task-form, participants repaired any agent errors and then clicked the update button, causing the database to update. A few participants chose to preview the results of an update before clicking the update button. Participants then returned to their email client to address the next task.

Task	Form	Task
T1	Add person	Create a new person record with first name, last name, email, organization, city, and state.
T2	Modify person	Add an image to an existing person record.
T3	Modify person	Add title, street address, and office location to an existing person record.
T4	Add news	Add a new news record with headline and body text.
T5	Delete person	Delete a person record.
T6	Modify person	Add image to an existing person record.
T7	Add news	Add a new news record with headline, body text and URL.
T8	Modify person	Fix a misspelled name in an existing person record.

Table 1. Evaluation tasks from the webmaster corpus.

Participants in the CMS condition followed a similar task flow. They first viewed the incoming request in the email client. Next they navigated to their browser, which displayed the "Task Picker" page, showing the same structured list of tasks that appeared in VIO's modified email (Figure 3). Here they selected the appropriate task from a structured list of links to all task-forms. Selecting a task caused the browser to transition to the task-form page. This form used the same layout as the VIO task-form (Figure 4) with the following exceptions. First, in the CMS condition, the task-form did not show the source email. Second, in the CMS condition there was no orange background color to indicate actions VIO had taken, since VIO had taken no actions. In the real world, software products do not make a connection between the requesting email and a website update task.

For incentives, participants were offered $30 to complete the experiment and penalized $5 for each mistake with a guaranteed minimum of $15. In addition, a $20 bonus was given to the top 50% of all participants based on a combination of their speed and accuracy. In this case accuracy entailed correctly making the requested update to the website. Our intent was to motivate them to work quickly and accurately, like a real webmaster.

As participants worked, a logging application and a screen-capture application ran in the background. The logging software captured the title of the window in focus. Participants opened each email message in its own window, performed the task, and closed the message window before moving on to the next one. Task completion time was defined as the amount of time between subsequent openings of new email message windows. The experimenter kept

track of the approximate time by hand for each session in order to check if the log analysis was working.

Forty people (31 men, 9 women) ranging in age from 18 to 35 with an average age of 22.62 participated. All subjects had previous experience with building or updating websites. Twenty were randomly assigned to each condition.

Measures

Task time. Task time was measured in milliseconds and is reported in seconds. Because task time was positively skewed, we truncated extreme values at the mean + 2.5 SD for each task.

Screen switches. The number of times an email message was in focus for greater than 1 second during the completion of a single task was recored.

Survey responses. To investigate participants' perception of the VIO and CMS interfaces, we asked them to complete a survey based on the instrument for measuring usability developed by van Schaik et al. [18]. Questions addressed the ease of use, usefulness, involvement, and flow control. Responses were made on a 7-point Likert scale (1 = "strongly agree," 4 = "neutral," 7 = "strongly disagree"). Prior to analysis, scores were inverted such that high scores indicated more positive assessments.

Survey responses were factor analyzed using Varimax rotation. The solution indicated the presence of five factors that accounted for 77% of the variance. Factor 1 (19% of the variance) was comprised of three questions about the *ease of use* of the system (e.g., "learning to use the system was easy"). The three factors formed a reliable scale (alpha = .90) and were averaged prior to analysis. Factor 2 (19% of the variance) was comprised of four questions about the perceived *usefulness of the software* (e.g., "using similar software would enhance my effectiveness"). These four questions formed a reliable scale (alpha = .88) and were averaged prior to analysis. Factor 3 (14% of the variance) was comprised of two *involvement* questions (e.g., "I made an effort to keep my mind on the activity"). These two questions were averaged prior to analysis (alpha = .72). Factor 4 (13% of the variance) was comprised of 4 questions about participant's feeling of *control* during the task (e.g., "I felt in control of myself", "I felt in harmony with the environment"). Responses to these four questions were averaged prior to analysis (alpha = .69).

Empirical Evaluation Results

We analyzed the results using a 2 (interface condition) by 8 (task) repeated measures ANOVA, in which interface was a between-subjects factor and task was a within-subjects factor. Table 2 details the average task completion times and standard deviation for these times for both conditions. As can be seen, subjects in the VIO condition performed significantly faster than those in the CMS condition ($F [1, 38] = 5.98$, $p < .05$). There was also a main effect of task ($F [7, 38] = 105.63$, $p < .001$), indicating that some procedures took longer than others, and a significant task by interface interaction ($F [7, 38] = 5.36$, $p < .001$).

The far right column details the decrease in time for VIO as a percentage of the total time from the CMS condition. The results show that across all questions, the VIO interface decreases the amount of time needed to complete a task by approximately 17%.

	CMS		VIO		
Task	Ave	Stdev	Ave	Stdev	decrease
T1*	89.22	27.70	69.29	27.64	22%
T2*	40.36	12.54	23.09	10.18	43%
T3	68.15	17.70	76.35	27.61	-12%
T4	43.12	11.97	44.26	18.80	-3%
T5*	25.33	5.46	18.05	12.38	29%
T6*	36.93	7.14	22.04	14.69	40%
T7*	42.81	9.89	34.80	9.90	19%
T8	41.25	9.56	32.66	18.31	21%
Ttl***	387.17	76.40	321.74	104.41	17%

Table 2. Mean completion time in seconds for each task by condition. * indicates p < .05, * indicates p < .001**

Agent Learning Errors

For each learning problem, the agent can make two general types of errors. A false negative error occurs when the agent fails to produce a valid suggestion. A false positive error occurs when the agent produces an incorrect suggestion. False negative errors represent a lost opportunity to help the user. False positive errors require the user first to recognize and then correct an error. Specifically, the following errors are possible:

WF: [Wrong form: false positive] selected wrong task-form as its top choice, but placed the correct task-form on the prioritized list.

MF: [Missed form: false positive] failed to list the correct form on the prioritized list.

WR: [Wrong record: false positive] selected the wrong record when displaying the task-form.

MR: [Missed record: false negative] failed to suggest a record for a form.

WX: [Wrong extraction: false positive] extracted the wrong information and added it to the task-form.

MX: [Missed extraction: false negative] failed to extract data included in the email needed in the task-form.

Table 3 lists errors VIO made. Note that VIO never made an MF, MR or WX error during the test. Column" VIO Ext" shows the number of items the VIO extracted from the email to the task-form. Task T3 requires additional explanation. T3's source email requests a modification to a person's title, address, and office location; however, VIO selected the wrong person. VIO selected "David Rodgerson" instead of "David McCullar." Additionally, VIO infers the requestor wishes to update his last name, and it extracts the last name "McCullar" of the actual requestor and overwrites "Rodgerson" with "McCullar."

Task	VIO Ext	Error	Error Summary
T1	2 of 5	WF	Suggests Modify Person (incorrect) followed by Add Person (correct).
		MX	misses: organization, city, and state.
T2	1 of 1		No errors made.
T3	1 of 3	WR	Extracts wrong person's record—matches first name but not last.
		MX	Misses: office location and street address.
T4	1 of 4	MX	Misses: headline and body text.
T5	n/a		No errors made.
T6	1 of 1		No errors made.
T7	0 of 3	MX	Misses: headline, body and URL.
T8	0 of 1	MX	Misses: last name.

Table 3. Errors made by VIO.

Participant Errors

Participants introduced few errors into the web site database. CMS participants introduced 12 errors and VIO participants introduced 15 errors. In looking more closely at the errors, we can see that the error VIO made to task T3 caused 13 of the participants in the VIO condition to introduce an error.

Usability survey

Participants' responses to our survey measures were averaged to create four factors: ease of use, usefulness of the software, feelings of involvement, and flow control. Means for each scale are shown in Table 4.

Responses were analyzed using one-way analyses of variance. For flow control, there was a borderline significant effect of condition ($F [1, 38] = 3.64$, $p = .06$), indicating that participants in the VIO condition perceived that they experienced less flow than participants in the CMS condition. ANOVAs on the other three scales showed no significant effects (all $F < 1$, *ns*.). VIO improved users' speed, yet it did not result in any loss in users' perceptions of ease of use, usefulness, or personal involvement.

Scale	CMS	VIO	p-value
Ease of Use	5.57	5.51	*ns.*
Usefulness	5.26	5.26	*ns.*
Involvement	5.23	5.39	*ns.*
Flow Control	4.79	4.29	0.06

Table 4. Average ratings from usability survey.

Discussion of Empirical Evaluation

The results in terms of performance support our hypothesis that VIO, even when it has had little training and makes errors, will significantly decrease the amount of time needed to perform a task. In this case participants experience approximately a 17% reduction in task time. This reduction comes from reduced navigation and from a reduced need to copy and paste.

Participants in the VIO condition reduced navigation in three ways. First, they had the list of tasks appear in the email, eliminating the need to navigate to the task picker page. Second, because they had the incoming email message displayed within the task-form, they did not need to toggle between the email and the task-form in order to copy and paste content. Third, the VIO selected the correct record automatically for the participant.

To further understand the impact of placing the email next to the form we examined the participants' logs. Table 5 shows the average number of times the task email window was in focus for greater than 1 second. The VIO and CMS columns show the number of times participants viewed the source email in each condition, and the "decrease" column shows the percentage decrease in time to complete a task for the VIO condition as compared to the CMS (from Table 1). Overall, the table indicates that the task email is viewed far less frequently in the VIO condition. Thus, users save the labor of switching between the email and the form.

Task	CMS	VIO	decrease
T1	8.30	2.50	22%
T2	2.50	1.60	43%
T3	5.90	1.85	-12%
T4	3.70	1.65	-3%
T5	2.05	1.75	29%
T6	2.45	1.90	40%
T7	3.50	1.70	19%
T8	2.70	1.70	21%
Average	3.89	1.83	17%

Table 5. Average number of times task email is in focus for greater than 1 second.

A 2 (interface condition) by 8 (task) repeated measures ANOVA, in which task was the repeated factor, indicated a significant main effect of condition ($F [1, 38] = 49.07$, $p < .001$), indicating that switches were much less common in the VIO condition. There was also a significant effect of task ($F [7, 38] = 37.96$, $p < .001$) and a significant task by condition interaction ($F [7, 38] = 23.39$, $p < .001$), indicating that the switches were more common in some tasks than others, and that the benefits of VIO over CMS for task switches were larger for some tasks than others. Interestingly, while tasks T2 and T6 showed the greatest reduction for the VIO condition, they both had very little difference in window toggling behavior. These tasks both involved adding a photo, and the time reduction seems to come from reducing the number of steps in saving the file from the email and linking it to the task-form.

With respect to task T3, VIO clearly generated an error that many participants allowed to enter the dataset. Even though the interface used an orange background to indicate that VIO had made a change to the last name field, 13 of the 20 participants failed to notice and correctly compensate for

the error. Two (2) of the 13 participants noticed the error but incorrectly compensated for it.

We propose three ways to address this problem of selecting the wrong record. First, the algorithm that selects the record could use machine learning to improve its accuracy. Second, VIO could be tuned to reduce the number of false positives at expense of reducing suggestions. This solution would mean users select the record for more tasks; however, the slight increase in effort would be offset by reduced errors. Third, when VIO has high confidence for more than one record, the record information could be included in the modified email suggestion. For example, the email list of tasks might include "Modify Person: David Rodgerson," "Modify Person: David McCullar," "Add Person: David Rodgerson," and "Add Person: David McCullar."

Participants gave good usability scores for both conditions. This result indicates that the experiment was unbiased with respect to the design of VIO and CMS forms, and that the time and performance differences were not due to qualitative usability differences. In addition, the high scores for VIO indicate that it would likely be accepted by users.

KLM evaluation

The empirical evaluation demonstrated that the agent with a little training can provide significant benefit. However, this evaluation raised three questions: (i) How much does VIO action reduce time as compared to the interface design? (ii) How does learning reduce task time? and (iii) How much total time reduction can a perfect VIO provide? In order to gain some insight, we performed a Keystroke-Level Modeling (KLM) analysis. This method predicts average performance for skilled users by assigning values to user and system actions.

We used CogTool [9], a software tool that allows interface designers to import their interface screens, to measure the sizes and map the locations of buttons and other clickable targets, and to provide a script of a user's actions such as button presses, reading time, pointing time, system processing time, etc. The automation the tool provides significantly reduces the time and effort required to perform a KLM.

In both the CMS and the VIO interfaces there are multiple ways to perform each task. For this KLM we chose to have the skilled user do what we observed most participants in the empirical study do: read the message, navigate to the task-picker page (CMS only), select the correct form, toggle between the message window and the task-form for each field's content and paste it in (CMS), or copy and paste from the embedded message (VIO). There are two exceptions:

- T1 requests an update to the person's city and state to "Alexandria, VA." In the KLM the expert user pastes "Alexandria" into the city field, but simply types "VA" into the state field.
- T8 requests a one-letter correction to a person's last name (change "Grainder" to "Grainger"). In the KLM the expert user simply selects the incorrect letter and types the correction.

When interacting with the pull-down menu, the skilled user types the first few letters to get to the correct item in the list, then types the enter key to select it.

For this KLM we used a reading time of 300 words per minute. To apply the reading rate we considered the number of words in each email message, excluding blocks of text that are generally not read. For example, add-news tasks sometimes included the text of the news story, which we excluded from the word count. Text such as URLs and email addresses each counted as a single word. Reading times for the tasks ranged from 4 to 10 seconds.

Design

For this analysis we used the same eight tasks from the empirical study, under four conditions:

- CMS: This KLM was based on our CMS interface. This condition works as a benchmark for revealing the performance gains from the other conditions.
- VIO-interface-only: This KLM was based on a VIO with no training (no VIO suggestions were provided). In this condition the incoming email had the structured list of tasks, but no prioritized list. Also, in the task-form, no fields were updated by the VIO. This condition measures interaction design effects without interference from VIO.
- VIO with our current agent: This KLM makes the same errors as VIO made during our empirical study. This condition provides a view of an agent with some training. Also, this condition helps us see how accurately the KLM matches human performance, allowing us to investigate the time difference needed to recognize a VIO error.
- VIO with a perfect agent: This KLM models a perfect VIO: the VIO always selects the correct form and correctly extracts all elements from the email to use to fill in the form. This condition shows the maximum benefit the VIO can provide.

Results

Table 6 shows the modeled task-completion times in seconds. Column CMS shows the completion time for the CMS interface; column VIO-interface shows the completion time for VIO interface with no learning; column current-VIO shows completion time the VIO with the same level of learning as the empirical study; and column perfect-VIO shows the completion time for VIO with no errors. The bottom row details the total time for all tasks in each condition and the percent decrease in task time for the different VIO conditions as compared to the CMS condition.

The VIO-interface-only condition reduces the task time by 15% compared to the CMS; the current-VIO condition reduces task time by 43% compared to the CMS; and the perfect-VIO condition reduces the task time by 71% compared to the CMS.

The KLM models the time needed to make corrections to agent errors, but it is not intended to model time taken for users to think about the message and examine the agent-

filled form for mistakes. This accounts for part of the time difference between the KLM-estimated reduction in time of 43% for the current-VIO and the 17% found in the empirical study. In this way the KLM provides an approximation of actual behavior.

Task	CMS	VIO-interface	current-VIO	perfect-VIO
T1	58.980	46.499	27.212	8.984
T2	33.047	31.958	13.435	13.435
T3	62.075	45.001	29.772	8.952
T4	32.044	27.228	27.206	9.200
T5	15.501	14.702	9.180	7.830
T6	29.988	27.984	10.580	10.58
T7	36.594	32.255	32.193	13.97
T8	23.592	22.754	15.483	10.761
Total	291.821	248.381	165.061	83.712
% decrease in time		15%	43%	71%

Table 6. Performance data for the four KLM conditions.

Discussion

The VIO-interface-only condition produced a 15% reduction in time over the CMS condition. This 15% benefit comes exclusively from the interface because in this condition VIO uses no learning to automate the task. The savings comes exclusively from eliminating the need to navigate to the task-picker page and select the correct form, and from eliminating the need to toggle between email and the browser by embedding the message in the task-form.

The perfect-VIO condition produced a 71% reduction in time compared to the CMS. Subtracting the 15% benefit provided by the interface reveals the maximum benefit from the VIO's learning at 56%. This number may be high, as users of the system would still need to check to see if an error occurred, something not modeled in the KLM.

The current-VIO condition produced a 43% reduction compared to the CMS, revealing a 28% reduction provided by VIO in addition to the interface. The KLM prediction of human behavior clearly shows that as the agent learns, the users reduce task time by having more and more of the task automated. However, this savings of 43% is much higher than the 17% found in the empirical study.

Two main factors help explain this difference: skilled performance and error recognition. The 20 practice tasks participants completed before the timed tasks may not have provided enough training for them to perform at the level of an expert, particularly for the VIO interface, which requires interaction methods that are unfamiliar to participants. In addition, KLM results do not take into account the time needed to recognize errors; they only take into account the amount of time to fix it.

Figure 5 provides some insight into the issue of skilled performance and error recognition. This chart shows a plot of ranked participants by task-completion times in the empirical study. Assuming that the completion of the 20 training tasks aided some participants more than others, the ranked list can be viewed as a list of most-skilled to least-skilled users. Surprisingly the chart shows a set of near parallel lines, revealing that VIO reduced task competition time by the same amount regardless of the skill level. While the total amount of time reduced remains the same, as skill increases, and the total completion time gets small, the percentage of time saved by the agent as a factor of total task time increases.

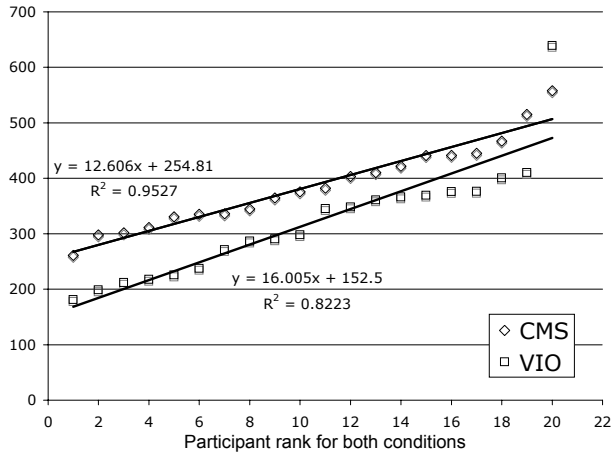

Figure 5. Ranked task completion time for VIO and CMS.

The five fastest (most-skilled) CMS participants had an average total task completion time of 300.6 seconds, just 3% slower than the KLM prediction of skilled performance. This data indicates that these participants may have been performing at skilled level after the 20 training tasks. The five fastest VIO participants had an average total task competition time of 207 seconds, 25% slower than the KLM predicted. From this data we can speculate that the additional time needed to recognize VIO errors is approximately 25% of the KLM prediction of total task time for skilled users. In considering reduction in task time, while VIO reduced task time by 17% for all users, it reduced task time by 31% for the top five performers when compared to the top five performers in the CMS condition.

CONCLUSION

This paper presents a new mixed-initiative interaction design where users train an agent to automate completion of mundane, procedural update tasks. The design embraces the idea that agents make mistakes through an interaction design that asks users to repair agent errors as a method of providing training data that improves the agent's learning. The design allows users to simply perform their work in order to train the agent.

Our empirical evaluation demonstrates that this interface coupled with an agent that has very little training can significantly reduce the amount of time that workers spend on mundane, procedural tasks. In addition our KLM analysis provides additional insights, such as the time

savings provided by the interaction design alone, and the time reduction provided by the assistance the agent offers.

One main insight gained during the design and evaluation of VIO is the need to develop the machine-learning system and the interface simultaneously. Co-development allowed for negotiation between the input requirements of the learning system and the actions required by the user to complete the work.

These studies provide evidence to support our research direction of using agents with little or no training to improve users' performance on procedural tasks, thus freeing workers to focus on tasks requiring more creative thinking.

Future Work

VIO has recently been deployed to assist a real webmaster in the maintenance of a large project website. In the current log of webmaster requests, approximately 50% of messages contain single tasks that can be executed with the existing system. As a next step, we plan to add machine learning and interaction capabilities to handle multiple tasks in a single message.

In addition, we are now developing a new mixed-initiative interface where end-users can construct their own workflows, allowing them to design the tasks they wish the agent to learn to automate.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010, Delivery Order No. D0300100003. The authors wish to thank professors Susan Fussell, Robert Kraut, and Bonnie John for help with study design and data analysis. We also thank the reviewers and Chris Scaffidi for detailed comments.

REFERENCES

1. Belotti, V., Ducheneaut, M., Howard, M., Smith, I. Taking Email to Task: The Design and Evaluation of a Task Management Centered E-Mail Tool. In *Proc. of CHI*, ACM Press (2003), 345-352.
2. Cypher, A., Eager: Programming Repetitive Tasks by Demonstration. in *Watch What I Do: Programming By Demonsration* Cypher, A. et. al. (eds), MIT Press, 1993.
3. Ducheneaut, N., Bellotti, V. E-mail as habitat: An exploration of embedded personal information management. *Interactions 8*, 5 (2001), 30-38.
4. Ferguson, G., Allen, J., Miller, B. TRAINS-95: Towards a Mixed-Initiative Planning Assistant. In *Proc. of AIPS,* AAAI Press (1996) 70–77.
5. Fogarty, J., Hudson, S.E., Atkeson, G.G., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J.C., Yang, J. Predicting human interruptibility with sensors. *ACM Transactions on Computer-Human Interaction 12*, 1 (2004), 119-146.
6. Hearst, M. Mixed Initiative Interaction. *IEEE Intelligent Systems 14*, 5 (1999), 14-23.
7. Horvitz, E. Principles of Mixed Initiative User Interfaces. *Proc. of CHI*, ACM Press (1999), 159-166.
8. John, B.E. Information processing and skilled behavior. In J. M. Carroll (Ed.) *Toward a multidisciplinary science of human computer interaction* Morgan Kaufman (2003), 55-101.
9. John, B.E., Salvucci. D.D., Multipurpose prototypes for assessing user interfaces in pervasive computing systems. *IEEE Pervasive Computing 4*, 4 (2005), 27-34.
10. Lieberman, H. Autonomous interface agents. *Proc. of CHI*, ACM Press (1997), 67-74.
11. Liebermann, H. Beating Some Common Sense into Interactive Applications. Seminar talk given at Carnegie Mellon's Human-Computer Interaction Institute. (February 23, 2005).
12. Lockerd, A., Pham, H., Sharon, T., Selker, T. Mr. Web: An Automated Interactive Webmaster. *Ext. Abstracts CHI*, ACM Press (2003), 812-813.
13. Malone, T., Grant, K., Lai, K., Rao, R., Rosenblitt, D. Semistructured Messages are Surprisingly Useful for Computer-Supported Coordination. *ACM Transactions on Information Systems 5*, 2 (1987), 115-131.
14. Meng, F. A Natural Language Interface for Information Retrieval from Forms on the World Wide Web. *Proc. of IS*, ACM Press (1999), 540-545.
15. Remedy: http://www.bmc.com/remedy/
16. Shneiderman, B., Maes, P. Direct Manipulation Versus Interface Agents. *Interactions 4*, 6, ACM Press (1997), 43-61.
17. Stylos, J, Myers, B.A., Faulring, A. Citrine: Providing Intelligent Copy-and-Paste. *Proc. of UIST*, ACM Press (2004), 185-188.
18. Van Schaik, P., Jing, J. Five Psychometric Scales for Online Measurement of the Quality of Human-Computer Interaction in Web Sites. *International Journal of Human-Computer Interaction 18*, 3 (2005), 309-322.
19. Tomasic, A., Simmons, I., Zimmerman, J., Learning Information Intent via Observation. *Proc. of the 16th International World Wide Web Conference (WWW),* 2007.