NORTHWESTERN
UNIVERSITY

Electrical Engineering and Computer Science Department

Technical Report
NWU-EECS-06-03
April 21st, 2006

Creating Polite Agents: 5 Heuristics for User Experience Design

Sanjay Sood, Jay Budzik, Kristian J. Hammond

Abstract

Issues relating to agent/user interaction and usability are important in making agent research a reality. Although often discussed, interaction strategies have rarely been deployed with general users. We present the results of such deployments: a set of heuristics generated by listening to feedback from users of an autonomous information retrieval agent. The lessons learned from this deployment experience can be used to optimize the user experience for systems that require agent-application integration to perform information access or service invocation. Our hope is that these heuristics can be used to accelerate the transition from promising research prototype to usable, polite, and deployable systems.

Keywords: Agents, agent/application interaction, human-computer interaction, information retrieval, mixed-initiative interaction

Creating Polite Agents: 5 Heuristics for User Experience Design

Sanjay Sood, Jay Budzik, Kristian J. Hammond
Intelligent Information Laboratory
Northwestern University
1890 Maple Ave.
Evanston, IL 60201 USA
+1 847 467 1265
{sood, budzik, hammond}@infolab.northwestern.edu

ABSTRACT
Issues relating to agent/user interaction and usability are important in making agent research a reality. Although often discussed, interaction strategies have rarely been deployed with general users. We present the results of such deployments: a set of heuristics generated by listening to feedback from users of an autonomous information retrieval agent. The lessons learned from this deployment experience can be used to optimize the user experience for systems that require agent-application integration to perform information access or service invocation. Our hope is that these heuristics can be used to accelerate the transition from promising research prototype to usable, polite, and deployable systems.

Category and Subject Descriptors
H.1.2 [Information Systems]: User/Machine Systems, H.5.2 User Interfaces [Information Interfaces and Presentation]: User-centered design.

Keywords
Agents, agent/application interaction, human-computer interaction, information retrieval, mixed-initiative interaction.

INTRODUCTION
Numerous researchers in Computer Science have studied how intelligent services can be embedded into existing applications [4, 6, 10]. A system can perform work on the user's behalf to create more efficient work environments. Mundane, complex, and large tasks can be automated by an intelligent system to create these new efficiencies. Such systems, often referred to as *agents*, perform a wide variety of tasks, from information retrieval [4, 17], to automatic appointment scheduling [6] and music recommendation [9].

Some agents require explicit user input in order to provide service. For example, a comparison shopping agent would require the desired product or product category to begin gathering information for the user. This explicit invocation strategy requires the user (1) determine they want to perform the activity the agent supports (2) determine they want help from the agent and (3) understand enough about the problem domain (the products they are comparing) in order to express their needs to the agent. Although this may seem like a trivial amount of effort, it can still reduce the effectiveness of an agent deployment [3] because it requires too much from the user. For this reason, other agents take a different approach. By integrating with existing work environments, these agents support tasks for which the necessary context to perform the agent's work can be extracted from a user's software environment and actions they perform using existing software tools. Access to this information about the user's task is most easily provided by integrating with the applications that control the type of content the agent requires to operate. For example, an agent that recommends new music based on your past listening habits is likely to be informed by how you use the music player on your computer. This kind of integration reduces the amount of effort required by the user in order to benefit from the services the agent provides.

The integration of agents and applications, while attractive in theory, can pose real problems to the final user experience, in practice. Violated user expectations, high latency, lack of agent politeness and confusing interaction metaphors are a subset of the pitfalls that agent researchers may face when moving their integrated systems beyond the prototype phase of development.

While there has been much discussion on the roles that agents can perform in the context of supporting users in everyday tasks (e.g., [12]), there have been relatively few discussions on the types of requirements for integrating agents into existing applications [6, 11, 21]. These requirements end up being more than simple "engineering issues" in that they directly influence user experience design.

The opportunity is clear for delivering new levels of functionality to users by deploying intelligent agents embedded in existing applications. Our goal with this paper is to accelerate our community's ability to deploy interesting agent systems into the real world by providing practical guidelines for integrating such agents into existing

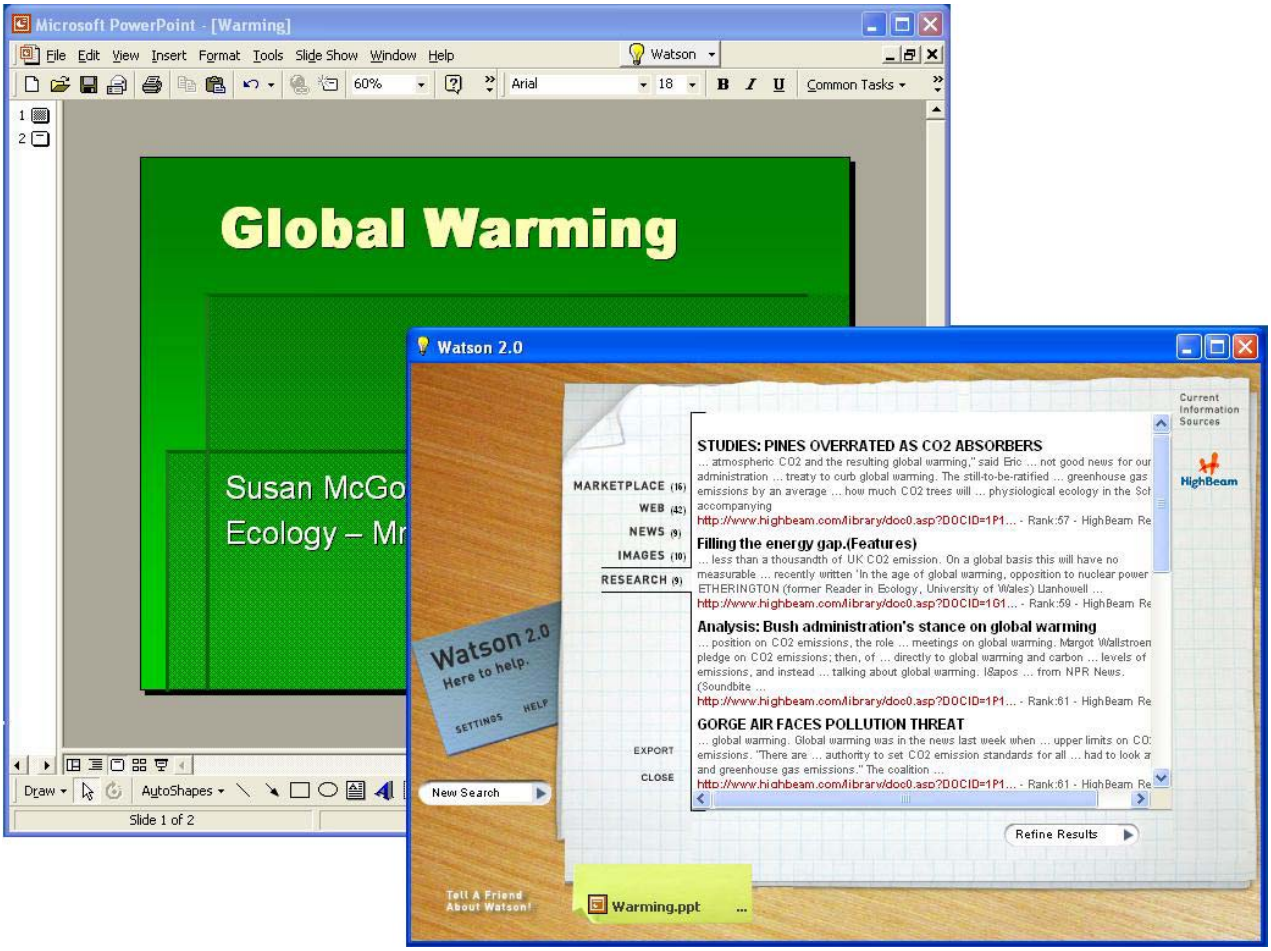

Figure 1: Watson has retrieved relevant information in the context of a user preparing a presentation on Global Warming.

environments. These guidelines are grounded in our experience deploying an agent with hundreds of beta users, eliciting feedback, and iterating on our design.

A TESTBED FOR INTERACTION DESIGN

Watson is an information access agent that automatically retrieves useful information in the context of an ongoing task [3]. Watson integrates with existing applications in order to gain access to documents users are manipulating. The system analyzes the user's active document in order to compute a representation of her current work context. This representation is then used to form queries to multiple online information repositories. The system collects the results of the queries, sorts and filters them, and presents them to the user in a separate window. Watson provides users with an awareness of online resources that may be useful to her in the context of her current task.

For example, say a user is building a presentation about global warming. As the user works, Watson retrieves relevant information she can use to enrich her presentation. In this case, Watson has retrieved news articles about global warming and current US and international policy, Web logs on the environment, and Web sites with tutorial explanations describing the greenhouse effect (see Figure 1). From the user's perspective, Watson delivers useful information without requiring any effort on their part. Users need not know that the information exists, or where to find it, or how to use search systems to retrieve it. Instead relevant and useful information is simply available to them.

Watson is an example of an agent that requires direct integration with existing applications to extract information necessary to perform a certain set of tasks. Watson integrates with many software products that are widely used by both novice and expert computer users. The current version of Watson is integrated with Microsoft Word, PowerPoint, Outlook and Internet Explorer.

There is a significant effort underway to develop Watson for general public and enterprise use. Working on a new version of Watson provides an opportunity to revisit the design of user interaction with the system, with a focus on making the application more usable. The integration requirements mean special care must be taken so as not to annoy the user by disrupting the activities they normally perform in existing applications. The focus of the guidelines described below is on striking a balance between the user's desire to have the information the agent

gathers and their expectations of the behavior of their existing applications and desktop environments.

Watson 2.0: Always On

One of the main interaction concerns in our latest iteration of research and development focused on having Watson "always on." In this version, our goal was to have Watson make information immediately available to the user with a single click, in the context of all open documents. This was a departure from the model of manual invocation in previous versions of Watson. In previous versions (Watson 1.0) [4] users were required to click on a button in their application in order to cause Watson to start finding related information. In the current version (Watson 2.0), users simply interact with their applications as they normally would, and Watson continually searches on their behalf. For the user, this means that information is delivered as a mater of course, instead of by an explicit decision on their part.

Through detailed analysis of the use of the initial version of the system, interviews with user groups, and internal discussion and modeling, we determined "always on" functionality would significantly increase the utilization of information resources and thereby increase the utility of the software by making relevant, on-point information instantly available. Interviews with users made clear that integration and usability concerns were paramount to ensuring "always on" functionality did not interfere with the user's ability to continue to use applications that support the core of their activity and work process.

Along with new concerns about having "always on" fully functional, the development of production quality software required a level of integration quality, stability, robustness, and usability not often expected of regular research software development.

LESSONS LEARNED

The process of developing and deploying Watson 2.0 provided us with the opportunity to critically examine the methods for integrating agent software into existing applications and the work that they support. Putting the software in the hands of regular users helped expose a variety of problems with Watson's performance as measured by overall usability and interaction ease. During our beta period, early versions of Watson 2.0 were distributed to 214 users from a variety of groups, including college students, businesspeople, "general consumers" in a variety of age groups, lawyers, and engineers. The lessons learned from repeated feedback from our users are represented here as a set of guidelines that directed our development and interaction design, and, we think, can be generally applied to any user experience design involving agent-application integration.

(I) Be Helpful: Maximize the Potential Utility of Agent Invocation

People often perform very specific tasks while using their computers. Whether it is searching the web for movie reviews, balancing ledgers, or modeling organic chemicals, a user employs various tools and techniques to accomplish their goals. Applications are the main tools that people use to do their work.

The ability to interact with application data provides the necessary information for an agent perform valuable work on behalf of a user. With the ability to leverage application-specific information streams, comes the responsibility of coherently and reliably integrating with the various applications the agent can operate within. Choosing the program(s) that an agent will integrate with is the easy part. The difficult part is dealing with the varied states that an application may operate in, and coordinating agent behavior in response to those states and transitions among them to ensure the potential utility of agent invocation is maximized. In other words, agent designers should think hard about when the agent should be invoked in order to ensure the user will be delighted by the result of the invocation.

Finding Operating Boundaries

In most cases, agents rely on the information being manipulated by an application to invoke services or access information. The actions an agent performs may be dependant on the interaction a user has had with the application. With this in mind, a necessary step to successfully integrating an agent is understanding and identifying the various states that an application may operate in. These states, often numerous, pose specific problems to the overall user experience of interacting with an agent while using an application. An application's states of operation are very important to ensuring the agent does not unnecessarily interrupt or annoy the user while they are performing work.

For example, Watson should do nothing when Microsoft Word is in its initial state of a blank document or when no document is open at all. Returning results for a blank Word document does not make sense in the context of creating a document (what would the results contain?). It also creates the risk of annoying the user by disrupting their current workflow by presenting incoherent results. States such as wizards and tutorials pose additional opportunities for tailoring agent/application interaction.

A basic understanding of various application states was critical to successfully implementing the "always on" feature in Watson. Since Watson was to continually returning results for open documents, we could not rely on the user to determine when it was appropriate to invoke the agent. Our goal was to ensure users would not be distracted by results that were incoherent given the application's current state.

Tracking and Responding to State Transitions

Many of the boundary states we identified have clear transition points to other states that Watson could function normally in. Identifying and acting reasonably in these situations was another important step of our developing an

effective system. Microsoft PowerPoint, for example, provides a straightforward model for document composition. For our purposes, reflecting this model in our agent did not require heavy lifting (unlike in, e.g., [5]). In the initial case of a blank slide presentation, Watson waits until the user has engaged in the task of writing their presentation to activate and analyze the slide set. As soon as the user finishes creating a new slide, Watson begins retrieving relevant information. Watson continually monitors an application's state until it has found an opportunity to act, and then engages in accessing information.

The importance of understanding an application's states is critical when integrating an agent with an application. The risk of behaving incoherently or unnecessarily can reduce the agent's "credibility" in the eyes of users. This can be complex when considering the plurality of tasks a user may engage in when using existing applications.

(II) Be Polite and Get Out of the Way

The complexity of the work that agents perform on behalf of a user depends on the type of task the agent is carrying out and the frequency with which the service is invoked. Certain types of agents will only be activated in occasional and opportunistic circumstances that require special assistance to deal with a certain task. Other times, an agent will be continuously processing real time information to provide a solution or assistance [7]. In other scenarios, an agent may be performing a set of actions concurrently in multiple instances or invocations. In any of these cases, it is imperative to keep a low resource profile computationally, and to maintain low bandwidth of attention requirements along interaction dimensions, to ensure the agent does not interfere with the task it is trying to support.

Be Invisible, Unless Attention is Warranted

Previous work in integrating agents with existing applications has focused on methods for attaching agents' controls to external applications [11, 21]. Under these approaches, agents can use an application's native user interface to control an application as the user would, and they can sense application state by analyzing bitmaps of the user interface or other readily-apparent representations. While working at the interface level brings interesting research questions to light, it poses significant problems in developing usable and interactive agent-application systems that can be deployed. A lack of shared control in these frameworks does not allow users to work on existing tasks without relinquishing control of their applications to an agent.

With the goal of invisibility, except at the point of notification, we ensure that all application-agent control is kept at a level that does not interfere with a user's current work. Significant effort was aimed at ensuring Watson seamlessly integrated into users' applications. The main goal of the software was to provide relevant, on-point information as users interact with their applications the way they normally would. This mandates that Watson never take control away from the user or delay execution of normal tasks in any way and also to ensure user applications are continually interactive. This requires keeping most of the agent-application interaction behind the scenes. Any interruptions created by Watson's integration with an application would interfere with a user's normal interaction, and, therefore, their ability to efficiently perform normal activities within these applications. This violates user expectations and disrupts their working environment.

Keeping a Low Profile

One of the major hurdles in implementing Watson's "always on" functionality was making the software usable in a high volume and/or low resource scenario. By design, Watson integrates into multiple instances of multiple programs on users' computer. Depending on how users manage their applications, there were cases in which users had numerous application instances running, which resulted in an equal number of Watson process threads being executed in parallel. In an initial version of our software, this resulted in "bringing Watson to its knees…with lots of lag… and unresponsive interface" [Watson beta feedback]. The software's behavior with respect to resource usage rendered it entirely ineffective in certain usage scenarios. Users would uninstall the application when they discovered it slowed down their systems.

There are varied views on the effects of poor performance and long system response times on the ability of people to accomplish tasks. From those who argue that system performance has a negative effect on a user's ability to accomplish a task [1, 2], to those that feel that interruptions allow for more planning time for problem solving and decision making tasks [15, 19, 20, 23].

Practical use, however, would suggest that software that reduces or eliminates the responsiveness and performance of another application would be deemed annoying, flawed, or simply unusable. Interfering with external applications and not meeting a user's expectations for application responsiveness reduces interferes with a user's ability to complete the tasks at hand. Keeping this in mind, it is important that agent developers ensure that users are always in control of their applications, making certain that the agent does not interfere with the user's interaction with their native software programs.

New "always on" functionality was a key feature in driving utilization of information repositories, opportunities for information discovery, and overall software utility. Removing this functionality to reduce the processing load was undesirable. Accommodating the use case of having scores of windows open was the only option.

We attempted, instead, to retain "always on" functionality as a feasible Watson feature. The solution was the

following changes to user experience design that resulted in changes to the way the software interacted with a user's applications:

- Watson will only process documents within the user's focus of attention. Watson will not work on documents the user is not actively interacting with. This significantly impacts the overall resource requirements of the system, even in desktop environments with multiple open windows.
- Watson delays initiating a search until a user has "settled in" to read a document or "paused for reflection" while writing. For example, in Internet Explorer, Watson waits until the user has been reading a page for several seconds before it begins processing. This ensures the user has full control of system resources while they browse the Web. Once they have settled in on a page (and the response time of the application doesn't matter as much) Watson will begin processing.
- Operating System scheduling priority is given to existing applications by reducing the priority of Watson processes.

The changes above have the effect of reducing Watson's impact on user applications and therefore lowering the cost perceived by users for enjoying Watson's services. From the user's perspective, the cost associated with having Watson run continually on their machine is negligible, so any improvement over existing services is appreciated.

This is a departure from other approaches, which require the agent reason about service invocation and interruption in order to minimize perceived costs [8]. Instead, we propose a simpler approach that combines the rules of invocation described above, along with heuristics to ensure the user is always in control. The agent need not have a complex model of user behavior to predict when it will have an opportunity to operate or interrupt. Instead, the agent can react to user behavior as it occurs, and politely "step aside," when appropriate.

(III) Be Accommodating: Offer Multiple Interaction Methods

The long term vision in many research circles is developing agents that can reliably interpret user interactions within tasks and among applications to automatically invoke services and information access [4, 10, 12]. Conversely, there are those who believe that the realities of ambiguity in the interpretation of user interactions, and weak task recognition machinery, in general, should move researchers to more seriously consider designs that allow users to have direct control over the services that are activated [18].

Allowing multiple interaction methods for agent software helps quell some of this tension (as represented in, e.g., [13]). A mixed scenario offers the benefits of both methods.

Let the user choose, but help them decide

Watson, by default, installs with the "always on" feature enabled. The system, however, allows users to choose an exclusively manual mode on an application by application or window by window basis (reverting to the kind of interaction model supported by Watson 1.0). For example, a user can access Watson's setting panel and chose to have Watson automatically search in all Microsoft Word instances, but only search in Internet Explorer when the agent's button is manually clicked. This particular feature was requested by beta users that were concerned with automatically initiating searches on sensitive or classified documents. In addition, users can tell Watson not to search automatically on a given Window, so that sensitive material is never touched by Watson.

There are times, however, when a choice between on and off is not effective. This offers an opportunity to insert intelligence in the form of exceptions to the rules.

Feedback from our beta users helped in developing some simple exceptions to automatic invocation. The main goal was to increase the acceptance of "always on" by reducing concerns around privacy.

The resulting solution introduced an exception to "always on" functionality which automatically enables manual mode for certain types of documents and resources: Watson will not automatically search based on Web documents that are retrieved through secure channels (HTTPS). The assumption (derived from user feedback) is that documents viewed through an encrypted connection are generally more sensitive and should not be inspected by Watson, due to privacy concerns. Allowing for maximum flexibility, Watson can be manually invoked on a secure page by simply clicking the agent button in the menu bar.

In general, allowing special, user-defined rules to block agent invocation provides users control over situations where agent services are unwanted or unnecessary. Easy-to-use interfaces that support this kind of "exception management" are currently being investigated.

Let the User Drive, but Help Keep them on the Road

Watson automatically retrieves information on behalf of the user. Watson considers the user's entire document as a basis for retrieving information. While this generally results in on-point information [4], the user may want to influence the results retrieved by specifying additional criteria or by focusing Watson on a specific section of their document.

This is achieved by allowing the user to select regions of the document Watson should emphasize. They can then request Watson update its results, given this new focus, and Watson will retrieve documents that are related to their selection in the context of the entire document. This keeps the locus of interaction within the application, which requires less learning and allows for transfer of existing knowledge of application operation. It affords direct

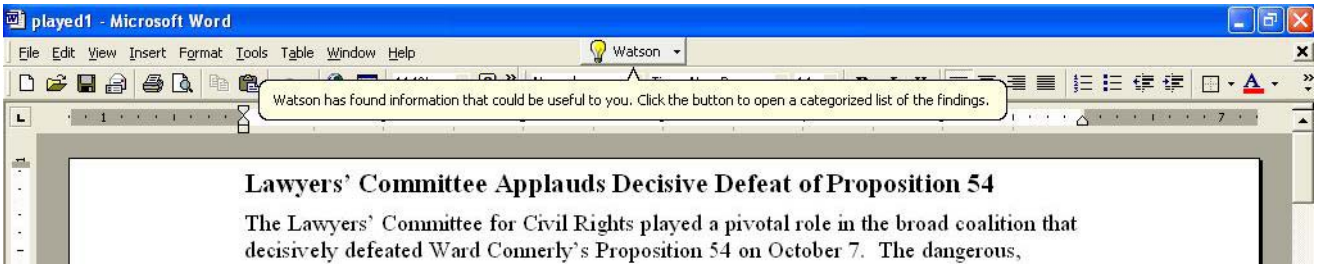

Figure 2: Watson explains its notifications to users the first few times they use the system. Here, Watson has found relevant information while a user is authoring a document in Microsoft Word.

manipulation of the parameters Watson uses to arrive at its representation of the user's context, and allows users a level of control over the results retrieved not previously offered.

In addition, users can enter terms into Watson directly, which are combined with Watson's existing representation of the user's document, in order to move the search results in a particular direction. This has the effect of disambiguating the meaning of the entered term, using the context of the document the user is manipulating. For example, our user working on the global warming presentation (presented earlier in Figure 1), could enter "China" into Watson, and receive information about the environmental impact of the industrialization of the East Asian country, instead of information on where to buy translucent cups and plates.

(IV) Be Unobtrusive: Inform, Don't Interrupt

Previous studies have considered the effects of distraction on the ability of people to get work done. In the context of software, many researchers agree that interrupting a user in the middle of a task reduces their performance while increasing the memory load required by the task at the time of interruption [1, 16]. In reaction to these findings, many researchers have explored ways of building systems that can monitor and predict opportune moments for interrupting a user, based on identifying task boundaries and external environmental sensors [2, 8].

These solutions, however, require relatively complex reasoning to identify when to interrupt a user. We propose a simpler approach that combines heuristics for interruption and service invocation, and offers a user interaction design that is intended to convey state with minimal demands on user attention. This design has worked well for us in creating an interaction experience that is both beneficial and non-disruptive.

Watson Notifications

In order for Watson users to benefit from the information retrieved on their behalf, its availability must be communicated. In addition, novice users might not understand a minimal visual cue, when initially introduced. Thereby we needed an interface that both afforded rapid learning for novice Watson users, and one that would not annoy users once they got used to using the system.

Watson communicates it has retrieved results through a button integrated with the application's toolbar. The button contains a light bulb that lights up when Watson has retrieved results.

Watson also incorporates self-explaining notifications to instruct users on how to use the software. After installation, Watson provides a series of message balloons that are displayed alongside user interface elements in order to ensure users are familiar with the system's functionality. Messages make users aware of the agent's button in applications by drawing attention to the controls in the user's desktop and explaining what the change from "light off" to "light on" means. This feature was aimed at making the software self-explanatory so users could benefit from it immediately.

An initial version of Watson 2.0 included notification balloons on the button that instructed users that information was available. Many of our beta users found these notifications incredibly disruptive and annoying. The main cause for user angst towards these notifications was the placement and frequency of the messages; since Watson was "always on" the balloons were "always there" (Figure 2).

People were interrupted by notifications that appeared without them doing anything. In addition, several users reported that the notification obscured their documents, directly interfering with their work.

The more aggressive notification scheme was a reaction to the user feedback we received during an earlier release of Watson 2.0. In this version, Watson simply changed the color of the light bulb. This more peripheral notification, however, resulted in users being unaware that Watson had returned search results for their document, thereby reducing the overall utility of the agent.

The feedback between the two versions of notification led us to find the middle ground between annoying (too many notifications) and invisible (notification that the user didn't notice). The solution we implemented was modifying the light bulb icon to flash for a couple seconds when Watson had finished retrieval. In addition, the notification balloons were retained, but only for the first two times the user worked within an application when Watson was running. Further beta user feedback verified the flashing icon was

sufficient to notify users, and that the introduction and eventual removal of notification balloons was sufficient both (1) to teach the user what Watson was and what the icon flashing meant and (2) to minimize user annoyance.

Striking a balance between over and under notification was necessary to maximize Watson's utility. It was through user feedback that we were able to identify the type of notification that was informative and non-disruptive.

(V) Be Consistent, Not Confusing

A general rule in user interface design stresses the importance of keeping a user interface consistent visually and functionally [14]. Consistent interfaces allow users to predict the way an application will react to interaction, significantly reducing learning curve for performing work within the software [22]. Meeting a user's expectations in interaction is an indicator of a predicable and usable interface design.

Agents that are embedded into existing applications often also require direct user interfaces. Interfaces for such agents must abide by the same rules of consistency that govern the applications they are integrated with. Providing a seamless and predicable user interaction will allow for easier adoption of agent technology.

The Watson Button: Consistently Controlling the Agent from within Applications

Watson is an agent that requires interaction with the user. Invoking new searches, updating existing searches, and accessing the setting panel are just a few of the functionalities Watson offers that require user input. Since Watson's primary function is to return relevant information inline with a user's regular workflow within an application, it was necessary to provide users the ability to access Watson controls from within the application they are using. Watson also must notify users when it has new information for them to consider, without interrupting or annoying them.

Watson, unlike many agents, integrates into multiple applications. Having multiple interface points increases the complexity of keeping the interaction and user interface consistent among all instances of the agent and application.

To achieve this, Watson draws a button in the upper right corner of the toolbar in Internet Explorer, Microsoft Word, and PowerPoint. Functionally and visually, the button is identical in all the applications with which Watson works. In addition, the button is always placed in the same location to keep with user expectations of consistency.

Interaction with the agent's interface is also consistent with interaction with the interface of the native application. Drop-down menus are used to organize Watson's controls and consistent coloring and shading are used to help the controls blend into the existing user interface. User notifications are presented as changes in the button's icon or as message balloons, mimicking the appearance of alerts in the existing applications.

Allowing Watson to integrate seamlessly into a user's workflow was a main goal in developing the system. Consistency in presenting user controls was a key factor in achieving this effect by allowing us to create consistent user expectations, manage them, and meet them effectively.

FUTURE WORK

As more and more people download, install and use Watson, the amount of feedback from real-world users will only increase. The heuristics for building polite and usable agents that we have laid out in this paper will help guide the next iteration of development of this and other systems, and will be refined and expanded with more user comments.

As discussed, understanding and identifying the various states in which an application may operate is important in avoiding unnecessarily interrupting users. Our future work includes a focus on understanding and detecting the various classes of documents and how they should affect agent behavior. In addition, the kind of tasks the user is engaged in, and the type of document they are manipulating, should be used to guide the retrieval of information. Relevance is parameterized by the goals the user is pursuing. For example, Watson should behave differently when analyzing a photo album versus a term paper, because the user is pursuing entirely different goals, even though she may be using the same application to achieve them. Building mechanisms to detect these various classes of goal, task, and document artifact, and developing the proper analysis techniques in order to achieve this, is currently under investigation.

SUMMARY AND CONCLUSION

Through Watson's continued development, we were given the opportunity to critically explore how our system performed before hundreds of real-world users. This helped expose a set of issues that, when addressed, resulted in the heuristics presented here. These heuristics can aide in the development and design of any system that requires interaction among agents, applications, and users.

Acknowledgements

Thanks to the Northwestern Center for Technology Commercialization (DevLab), Open Road Technologies, and the InfoLab for helping us further develop Watson into a deployable system. Contributors to this effort included Larry Birnbaum, Andy Crossen, Jace Frey, Mike Lee, and Rob Daugherty.

REFERENCES

1. Bailey, B. P., Konstan, J. A., & Carlis, J. V. (2000). Measuring the effects of interruptions on task performance in the user interface. In *IEEE Conference on Systems, Man, and Cybernetics 2000* (SMC 2000), IEEE, 757-762.
2. Bailey, B. P., Konstan, J. A., & Carlis, J. V. (2001). The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In

Human-Computer Interaction – INTERACT 2001 Conference Proceedings. IOS Press, IFIP, 593-601.

3. Budzik, J., Hammond, K., & L., Birnbaum. (2001). Information access in context. *Knowledge-based systems*. 14:37-53.
4. Budzik, J., and Hammond, K. J. (2000). User Interactions with Everyday Applications as Context for Just-in-time Information Access. In *Proceedings of Intelligent User Interfaces*. ACM Press.
5. Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. (1998) The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of UAI 1998*, AAAI Press.
6. Horvitz, E. (1999). Principles of Mixed-Initiative User Interfaces. In *Proceedings of CHI '99, ACM SIGCHI Conference on Human Factors in Computing Systems*, ACM Press.
7. Horvitz, E.. (2001). Principles and Applications of Continual Computation. *Artificial Intelligence Journal*, 126:159-196, Elsevier Science.
8. Horvitz, E & Apacible, J. (2003). Learning and reasoning about interruption, In *Proceedings of the 5th International Conference on Multimodal Interfaces*, ACM Press.
9. Lapat, L., Dunne, J., Flury, M., Shabib, M., Warner, T., Budzik, J., Hammond, K., & Birnbaum, L. (2002). mpme!: Music recommendation and exploration. In *Proceedings of Intelligent User Interfaces.*
10. Lieberman, H. (1995). Letizia: An Agent That Assists Web Browsing. *IJCAI* (1), 924-929
11. Lieberman, H. (1998). Integrating user interface agents with conventional applications. In *Proceedings of Intelligent User Interface*, ACM Press.
12. Maes, P. (1994). Agents that Reduce Work and Information Overload, *Communications of the ACM*, Vol. 37, No. 7.
13. Maes, P., and Schneiderman, B., (1997). Direct Manipulation vs. Interface Agents: A Debate. *Interactions*, 4(6), ACM Press.
14. Nielson, J. (1989). *Coordinating User Interfaces for Consistency*. Academic Press, Boston, MA.
15. O'Hara, K & Payne, S. J. (1999). Planning and the User Interface: The Effects of Lockout time and Error Recovery Cost. *International Journal of Human-Computer Studies*, 50, 41-59.
16. Oulasvirta, A, & Saariluoma, P. (2004). Long-term working memory and interrupting messages in human-computer interaction. *Behavior & Information Technology*, 23 (1), 53-64
17. Rhodes, B. J. (1996). .Remembrance Agent: A continuously running automated information retrieval system, In *Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology*.
18. Schneiderman, B. (1992). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, ACM Press.
19. Speier, C., Valacich, J. S., & Vessey, I. (1997). The effects of task interruption and information presentation on individual decision making. In *Proceedings of the XVIII International Conference on Information Systems*. Atlanta: Association for Information Systems, 21-36.
20. Speier, C., Valacich, J. S., & Vessey, I. (1999). The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences*, 30 (2), 337-360.
21. St. Amant, R. and Zettlemoyer, L. S., (2000). The User Interface as an Agent Environment, In Proc. *Fourth International Conference on Autonomous Agents*, Barcelona, Spain, 483-490.
22. Thimbleby, H. (1990) *User Interface Design*. ACM Press.
23. Zijlstra, F. R. H., Roe, R. A., Leonova, A. B. & Krediet, I. (1999). Temporal factors in mental work: Effects of interrupted activities. *Journal of Occupational and Organizational Psychology*, 72, 163-185.