User-Context for Adaptive User Interfaces

Anil Shankar, Sushil J. Louis, Sergiu Dascalu
Dept. of Computer Science & Engineering.
University of Nevada, Reno
Reno, NV, 89557
{anilk, sushil, dascalus}@cse.unr.edu

Linda J. Hayes, Ramona Houmanfar
Dept. of Psychology.
University of Nevada, Reno
Reno, NV, 89557
{lhayes, ramonah}@unr.nevada.edu

ABSTRACT
We present results from an empirical user-study with ten users which investigates if information from a user's environment helps a user interface to personalize itself to individual users to better meet usability goals and improve user-experience. In our research we use a microphone and a web-camera to collect this information (user-context) from the vicinity of a subject's desktop computer. Sycophant, our context-aware calendaring application and research test-bed uses machine learning techniques to successfully predict a user-preferred alarm type. Discounting user identity and motion information significantly degrades Sycophant's performance on the alarm prediction task. Our user study emphasizes the need for user-context for personalizable user interfaces which can better meet effectiveness and utility usability goals. Results from our study further demonstrate that contextual information helps adaptive interfaces to improve user-experience.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Algorithms, Design, Experimentation.

Keywords: context, user-context, machine learning, learning classifier systems

INTRODUCTION
Consider the scenario of Jane setting a few appointments in her calendar. According to Preece et. al., Jane's calendaring interface is *effective* if it is doing what it is supposed to do, that is, her calendar should generate alarms as reminders for her appointments [12]. If Jane's calendaring interface generates suitable alarm types so that she does not miss any of her appointments, then the calendar provides Jane some *utility*, that is, it provides the right kind of functionality so that Jane can accomplish what she wants to do. We believe that a calendaring interface which is aware of Jane and learns her preferences for alarm-types can better meet these two important usability goals of effectiveness and utility. Such a context-aware interface also has the potential to improve Jane's user-experience.

In general, user preference for an application action varies depending on the context in which the application is used. For example, if Jane has an appointment for a meeting and is facing away from her computer, she might prefer her calendaring application to generate a voice alarm for her appointment. In a different scenario, if Jane is talking with someone in her office she might prefer her calender to generate a visual alarm for her meeting. In the same situation, Jack might prefer to have a visual alarm if he is facing away from his computer; a voice alarm if he is talking with someone in his office. Application action preferences in addition to varying upon the context of use also vary according to individual users. Wouldn't it be nice if Jane's calendar was context-aware and adaptively generated her preferred alarm types?

Current computer interfaces rely on the activity of an internal clock, keyboard and mouse to provide input or context to interact with a user. This reliance on meager contextual contextual information makes computer applications unaware of a user and her environment. Context unaware user interfaces can only make weak attempts to adapt their behavior to individual user needs. A user's environment is a rich source for other types of contextual information like motion and speech in addition to the activity of a keyboard or mouse or an internal clock. In our research, we use simple sensors to gather such contextual information from a user's environment and *mine* this data using machine learning techniques to generate a user model. Our goal is to incorporate this user-model into a generalized context-aware framework for enabling multiple user-interfaces that personalize themselves to individual users.

We present results from an empirical user study which investigates if contextual information from a user's environment is beneficial for user interfaces to adapt their behavior to individual idiosyncrasies to better meet usability goals and thereby improve user-experience. Our study subjects were ten graduate students in our department. We collected data for an alarm type preference while a subject read different articles on a computer in four different sessions, each lasting 45 minutes. Section gives more details about the design of our user study. *Sycophant*, our context-aware calendaring inter-

IUI'07, January 28–31, 2007, Honolulu, Hawaii, USA..

face and research test-bed learns a mapping from user-related contextual features to alarm types [11, 16]. Sycophant successfully learns to predict a user-preferred alarm type chosen from a set of four alarm types: no alarm, visual alarm, voice alarm and both (voice and visual). Sycophant's best accuracy for this alarm-type prediction is 88 percent using XCS, a learning classifier system [17].

We briefly describe what we mean by *context* in our research in the next section. Section gives our user study details. Our results in Section shows the beneficial effect of contextual information for Sycophant to adapt its alarm generation behavior to individual users. We summarize our results in learning user preferences based on user-context in the last section.

CONTEXT, USER-CONTEXT FOR LEARNING USER PREFERENCES

Context awareness is a widely researched topic in the area of ubiquitous computing [2, 3]. Researchers in this area have done a significant amount of work on standardizing a clear definition of *context*. Here is Dey's widely accepted definition of context [4]: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves"*.

Applying Dey's definition of context to a desktop PC, we further define *user-context* as [14]: *"Any information regarding a user's presence (or absence) in the vicinity of a computer"*. We differentiate between *external* and *internal* user-context in our research. External user-context is information that a computer senses from the external environment. This includes user-movements in the immediate vicinity of the computer and the presence or absence of speech. Internal user-context is any information that a computer senses from its internal environment. This information generally relates to keyboard activity, mouse usage, and the activity of different processes within a user's computer. We use Dey's definition of context to address the issue of learning user preferences. We believe that context-aware interfaces can better *personalize* themselves to a user. An interface personalizes itself to a user if it learns user-preferences and adapts its behavior accordingly.

In the area of context-aware interfaces and environments, Horvitz et al. built statistical methods and cost-benefit approaches to infer decisions about alerting users [7, 8]. Hudson, Fogarty, Atkeson et al. built predictive models to infer the state of interruptability of a user [5, 9], Bailey and Adamzyck quantitatively evaluated the effect of interruptions on the productivity of a user [1], and Kulkarni's *ReBa* used sensor devices to localize a user in context-aware environments [10]. Our research integrates and extends all these ideas in human-computer interaction. We design a sensor-based approach for gathering user-context for enabling interfaces to learn user preferences Like Fogarty, we use real sensors in our research to gather user-context. In addition to predicting the interruptibility of a user, we also predict the type of alarm to use for an individual user. In this paper, we provide describe our empirical user study to test the generalization of our context-learning approach across different subjects.

USER STUDY

We designed our user study to simulate an average work day in our research lab in the computer science department where one activity includes reading research papers while listening to music and being interrupted with conversations from neighbors. Our study goal was to investigate if contextual information gathered from a user's environment helps Sycophant's to accurately predict a user-preferred alarm type. Each subject (user) participated in four separate 45 minute sessions. During a session we instructed a subject to read an article within the first 30 minutes and answer questions pertaining to the article in the remaining 15 minutes. Sycophant generated different alarms for a subject while she read an article. We set the content of these alarms to help a subject participate in our study and answer questions related to the article. Sycophant's alarms were of four different types: visual alarm, voice alarm, visual and voice alarm, and no-alarm. Each alarm was a hint related to the article or the study. Sycophant's visual alarm is a pop-up window, and the voice alarm is an automated voice generated by a text-to-speech synthesizer. The subject participating in a session provided feedback specifying the type of alarm she preferred whenever Sycophant generated a alarm. We provided a notepad and a pen to our study subjects for writing down the information provided by the alarms (hints). Subjects studied an article for the first 30 minutes and provided feedback to Sycophant on their preferred alarm-type whenever Sycophant generated a alarm (hint). During the last 15 minutes of our 45 minute study, subjects answered questions pertaining to the article. Sycophant did not generate alarms for a subject while a subject answered questions.

Our study had two patterns of variation: cognitive load on a subject (four article reading tasks to be performed) and alarm types. We applied the following four treatments to our study subjects: Music with no talk, Talk with no music, Both music and talk, and No music or talk. An alarm type was the independent variable in our study and we measured it at these four levels: no-alarm (no hint is generated for the subject), visual alarm (a pop-up window displayed the hint), voice alarm (a hint is voiced out), and both (visual and voice alarms are generated). We used the same order of treatments for the sequence of alarm types across all subjects in our study.

Table 1 shows our experimental design for a subject reading a short article, long article, long article and a short article in different sessions. In our study, a self-report is the subject-preferred alarm type and the construct being measured is the alarm type which Sycophant generates. We ensure content-validity for this study by seeing to it that self-report and the construct being measured are the same. We control the confounding variables by seclusion of a subject while the study is being conducted. We use a randomized design to control the variation in the study. The alarms are generated in a random order for every session and this order is preserved for all the subjects in our study. Each unit in our study corresponds to an article reading task. A subject reads a short article in the first session, a longer article in the second session, an-

Table 1: Experimental Design for a Study Subject

Session	Task	Alarm Order	Treatment
1	short article	0, 2, 3, 1 3, 1, 2, 0 1, 0, 2, 3 2, 1, 0, 3	Talk, No-music Music, No-talk No-music, no-talk Music, Talk
2	long article	3, 1, 0, 2 1, 2, 0, 3 1, 3, 2, 0 3, 0, 2, 1	Talk, No-music Music, No-talk No-music, no-talk Music, Talk
3	long article	1, 3, 0, 2 2, 3, 1, 0 2, 0, 3, 1 3, 2, 0, 1	Talk, No-music Music, No-talk No-music, no-talk Music, Talk
4	short article	0, 3, 2, 1 1, 0, 2, 3 1, 0, 2, 3 3, 0, 1, 2	Talk, No-music Music, No-talk No-music, no-talk Music, Talk

other long article in the third session and finally a shorter article in the fourth session. A subsequent subject reads a long article in the first session, a shorter article in the second session, another short article in the third session and a longer article in the fourth session. We perform this variation of article reading lengths for every pair of subjects.

RESULTS

We evaluated the performance of four machine learning algorithms, Zero-R, One-R, J48, and XCS to learn user preferences. We use Zero-R's performance as the base rate for evaluating a machine learning algorithm's performance. Zero-R is a primitive learning scheme which predicts the majority class in categorical data or average class if the class is numeric. For example, if a user-preferred a visual alarm in 6 out of 10 cases, then Zero-R would predict that this subject always preferred a visual alarm. One-R generates a one level decision tree which tests only one particular attribute and constructs a set of rules based only on that attribute [6]. J48 is *Weka*'s implementation of Ross Quinlan's C4.5 decision tree [18, 13]. Wilson's XCS classifier system, a genetics-based machine learning scheme, is our fourth learning scheme. [17]. We use a two-sample t-test with a 95 percent confidence interval to compare different learning algorithms ($t_{p=0.025,N=9}$). We use the same statistical test to compare the performance of a learning algorithm across different data sets.

We merged the data collected from the 10 subjects in our study and ended up with a total of 582 exemplars. We provide more details about the construction of our user-context data in [15, 16]. We selected a subset of features in our user-context data after ranking these features using the *information gain ratio* filter in *Weka* machine learning toolkit and retained user-context features which had a non-zero gain ratio. Next, we created three-stratified folds ($2/3$ of the data for training the rest for testing) of this data set to gauge the training and test set performances. We consider Sycophant's task of deciding to generate an alarm-type from the set of four alarm-types as the *four-class alarm problem*; *two-class alarm problem* is the task of deciding whether or not to generate an alarm (interrupt) for a user.

Table 2: Prediction accuracy on the four-class alarm problem

I	II	III	IV
Machine Learner	Original Data	No User-Context	No External Context
Zero-R	48.62	48.62	48.62
One-R	63.23	48.62	63.23
J48	62.71	50.00	62.54
XCS	88.35	31.26	67.51

Table 3: Prediction accuracy on the two-class alarm problem

I	II	III	IV
Machine Learner	Original Data	No User-Context	No External Context
Zero-R	85.56	85.56	85.56
One-R	86.42	86.42	86.42
J48	87.11	86.42	86.59
XCS	85.91	71.13	74.39

Tables 2 and 3 show the test-set performance (averaged over ten runs) of Zero-R, One-R, J48 and XCS in learning user preferences across all these subjects on the four-class and two-class alarm problems respectively. We list the learning algorithm in column one. Column two shows the predictive accuracy, that is, the test-set performance of a learning algorithm on the four-class problem of predicting a alarm type from a set of four alarm types (visual, voice, both visual and voice, and none). We show the predictive accuracy of the four machine learning algorithms on a data set after removing the user-context features (user-id, motion and speech features) in column three. Column four shows the learning algorithms' performance on a data set which has no external context (no motion or speech features).

On the four-class alarm problem, all the learning schemes perform better than Zero-R (base-rate) and XCS significantly outperforms other machine learners. Removing user-context considerably degrades the performance of all machine learners except Zero-R (the majority voting algorithm). Removing external user-context degrades the performance of XCS and J48. We notice a similar behavior on the two-class alarm problem; removing user-context or external user-context degrades the performance of XCS and J48. The high predictive accuracy of the four machine learning schemes on the two-class alarm problem of deciding whether or not to interrupt a user bolsters Fogarty's work on predicting the state of interruptability of a user [5]. Examining One-R's tree showed that user-identity attribute was chosen as the most important

attribute on the data-set which had all the user-context features and mouse-activity attribute was chosen when the user-context features were removed. Our results clearly indicate the following: Knowing who the user is helps Sycophant to adapt its alarm-type to that particular user; User-context features like user-id and motion information are critical for Sycophant's ability to accurately predict alarm-type preferences; Removing external user-context (motion and speech) information degrades Sycophant's ability to predict alarm-type preferences. These statistically significant results show that an adaptive context-aware user interface can improve user-experience.

CONCLUSION AND FUTURE WORK

In this paper we investigated if context-aware user interfaces capable of adapting their behavior to individual users can better meet usability goals and improve user-experience. Our study showed that user-context information benefits our context-aware calendaring interface, Sycophant, to successfully personalize itself to individual subjects Next, we obtained results which indicated that Sycophant adapted (personalized) its behavior to individual subjects. Sycophant primarily used user-identity information along with information from a user's external environment (motion) to successfully predict a user-preferred alarm type.

Encouraged by the positive results from our user study, we are currently implementing a generalized user-context aware framework for enabling existing applications to personalize themselves to individual users. We plan to incorporate open-source applications like Sunbird (a calendaring application) and XMMS (media player) within our framework and enable these applications to be user-context aware. We plan to distribute this user-context software and gather long-term usage data to substantiate our claims regarding the importance of contextual information for adaptive user interfaces which can improve user-experience by learning their preferences.

ACKNOWLEDGMENTS

We thank the ten different subjects in our study for their time. This work was supported in part by contract number N00014-0301-0104 from the Office of Naval Research.

REFERENCES

1. Piotr D. Adamczyk and Brian P. Bailey. If not now, when?: the effects of interruption at different moments within task execution. *Proceedings of the 2004 conference on Human factors in computing systems*, pages 271–278, 2004.

2. P. J. Brown. The stick-e document: a framework for creating context-aware applications. In *Proceedings of EP'96, Palo Alto*, pages 259–272. also published in it EP–odd, January 1996.

3. P. J. Brown, J. D. Bovey, and X. Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64, October 1997.

4. Anind K. Dey, Gregory D. Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware appplications. *Human Computer Interaction*, 16, 2001.

5. James Fogarty, Scott E. Hudson, and Jennifer Lai. Examining the robustness of sensor-based statistical models of human interruptibility. *Proceedings of the 2004 conference on Human factors in computing systems*, pages 207–214, 2004.

6. Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Mach. Learn.*, 11(1):63–90, 1993.

7. Eric Horvitz and Johnson Apacible. Learning and reasoning about interruption. *Proceedings of the 5th international conference on Multimodal interfaces*, pages 20–27, 2003.

8. Eric Horvitz, Paul Koch, and Johnson Apacible. Busybody: creating and fielding personalized models of the cost of interruption. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 507–510, New York, NY, USA, 2004. ACM Press.

9. S. Hudson, J. Fogarty, C. Atkeson, J. Forlizzi, S. Kiesler, J. Lee, and J. Yang. Predicting human interruptibility with sensors: A wizard of oz feasibility study. *Proceedings of CHI 2003, ACM Press*, 2003.

10. A. Kulkarni. A reactive behavioral system for the intelligent room. *Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2002.*, 2002.

11. Sushil J. Louis and Anil Shankar. Context learning can improve user interaction. In *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration, IRI - 2004, November 8-10, 2004, las Vegas Hilton, Las Vegas, NV USA*, pages 115–120, 2004.

12. Jennifer Preece, Jenny Preece, Yvonne Rogers, and Helen Sharp. *Beyond Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

13. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.

14. Anil Shankar. Simple user-context for better application personalization. *Master's Thesis, University of Nevada, Reno, NV, 2006.*, 2006.

15. Anil Shankar and Sushil J. Louis. Better personalization using learning classifier systems. In *Proceedings of the 2005 Indian International Conference on Artificial Intelligence, December 20-22 2005, Poona, INDIA*, 2005.

16. Anil Shankar and Sushil J. Louis. Learning classifier systems for user context learning. In *2005 IEEE Congress on Evolutionary Computation,September 2-5 2005, Edinburgh, UK*, 2005.

17. Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

18. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, USA, 2000.